สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เครื่องเล่น เอ็มพี 3

MP 3 Player Decoder

โดย

นางสาวธิดารัตน์ เดชะผล รหัส 47015874

นายเสกสรร ตะก้อง รหัส 47015887

ปกพ.
ร5820
2549

เลขหมู่.....
เลขทะเบียน 72747
วัน,เดือน,ปี 22 มิ.ย. 2550

b 11737190
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาภาคอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องเล่น เอ็มพี 3

MP 3 Player Decoder

โดย

นางสาวธิดารัตน์ เดชะผล รหัส 47015874

นายเสกสรร ตีะก่อง รหัส 47015887

อาจารย์ที่ปรึกษา

รศ.ดร. สุรพันธุ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์หรือรายงาน ปีการศึกษา 2549

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องเล่น เอ็มพี 3

ผู้จัดทำ 1. นางสาวธิดารัตน์ เดชะผล รหัส 47015874

2. นายเสกสรร ตั๊ะก๋อง รหัส 47015887

.....อาจารย์ปรึกษา

(รศ.ดร. สุรพันธุ์ เอื้อไพบูลย์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องเล่น เอ็มพี 3

นางสาวธิดารัตน์ เดชะผล รหัส 47015874
นายเสกสรร ต๊ะก๋อง รหัส 47015887
รศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

รูปแบบของไฟล์แบบเอ็มเป็กหนึ่งเลเยอร์สาม นั้นเป็นที่นิยมกันเป็นอย่างมาก ทั้งนี้เนื่องจากคุณภาพและขนาดของไฟล์เอ็มเป็กหนึ่งเลเยอร์สามมีขนาดเล็กกว่าไฟล์เวฟ

โดยวิทยานิพนธ์นี้จะเป็นการประยุกต์ให้ไฟล์เอ็มเป็กหนึ่งเลเยอร์สาม สามารถเล่นโดยไม่ใช้คอมพิวเตอร์ ซึ่งเครื่องเล่นเอ็มเป็ก-หนึ่งเลเยอร์สามนี้ทำงานร่วมกับกับแฟรชเมโมรี่ ซึ่งสามารถเก็บเพลงไว้ และใช้ PSoC ไมโครคอลโทลเลอร์เป็นตัวควบคุม

เครื่องเล่นเอ็มเป็กหนึ่งเลเยอร์สามที่สร้างขึ้นอาจพัฒนาให้มีน้ำหนักเบาและสามารถพกพาได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MP 3 Player Decoder

Miss.Thidarat dachapon ID.47015874

Mr.Saksun takang ID.47015887

Assist. Prof. Dr. Surapan Airphaiboon Advisor

Educational Year 2006

ABSTRACT

The MPEG-1 Layer 3 is very popular because of MPEG-1 Layer 3 file has high quality is smaller than WAVE format file

The projeet is connected with the design of MP3 PLAYER that help people can play MPEG-1 Layer 3 file without computer. MPEG-1 Layer 3 will work together with Flash Memory that store song and uses PSoC as the controller

The MPEG-1 Layer 3 player be developed to be a light and compactable object

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ รองศาสตร์ตราจารย์ดร.สุรพันธุ์ เอื้อไพบูลย์ อย่างสูงที่ให้คำแนะนำ ประสิทธิ์ประสาทวิชาความรู้ รวมไปถึงอาจารย์ ทุกท่านในภาควิชาอิเล็กทรอนิกส์ที่ได้ให้คำปรึกษาเพิ่มเติมในโครงงาน และคอยเป็นที่ปรึกษาในการทำโครงงานและปริญญานิพนธ์นี้จนสำเร็จไปได้ด้วยดี

ขอขอบคุณ บิดา-มารดา ที่คอยห่วงใยและให้การสนับสนุนในด้านการศึกษามาโดยตลอดจนข้าพเจ้ามีทุกวันนี้ รวมทั้งขอบคุณเพื่อนๆร่วมงาน ที่ได้ให้ความร่วมมือและให้คำปรึกษาพร้อมทั้งเป็นกำลังใจให้กันตลอดมา

นางสาวธิดารัตน์ เดชะผล

นายเสกสรร ด๊ะก๋อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
การเข้ารหัสแบบ MPEG

1.1 วิวัฒนาการของการเก็บข้อมูล

ในอดีตการเก็บข้อมูลประเภทเสียงจะเก็บในรูปแบบ Analog ซึ่งการเก็บในรูปแบบนี้ไม่สามารถเก็บข้อมูลของเสียงทั้งหมด และในการเก็บในระยะยาวทำให้การเก็บเสื่อมลงไป ทำให้คุณภาพเสียงที่ได้ต่ำลง ต่อมาได้มีการพัฒนาการเก็บข้อมูลเสียงในรูปแบบของ Digital ซึ่งการเก็บในรูปแบบนี้จำเป็นต้องมีการเปลี่ยนแปลงข้อมูลของเสียงในรูปแบบ Analog มาเป็นรูปแบบของ Digital ซึ่งเป็นหลักในการสุ่มตัวอย่าง โดยในการแปลงจะเก็บค่า Amplitude ของสัญญาณที่เข้าในขณะนั้นแล้วทำการแปลงเป็นเลขฐานสอง ซึ่งอุปกรณ์ที่เราใช้แปลงสัญญาณนี้ เรียกว่า Analog to Digital Converter หรือ ADC

ในการแปลงรูปแบบสัญญาณในรูปแบบ Digital มีปัญหาเรื่องของขนาดของข้อมูล ถ้าเราทำการสุ่มตัวอย่างจำนวนมากท่าไหร่ก็จะทำให้พื้นที่ในการเก็บข้อมูลมากเท่านั้น เช่น การสุ่มตัวอย่างในอัตราความถี่ (Sampling Frequency) 44.1 KHz ในการสุ่มตัวข้อมูลเสียงในรูปแบบ Digital ขนาด 16 bit โดยมีสัญญาณเข้าทางซ้ายขวา เพราะฉะนั้นใน 1 วินาที จะต้องทำการเก็บข้อมูลถึง 44,100 x 16 x 2=1,411,200 bit เลยทีเดียว ดังนั้นถ้าทั้งเพลงจะใช้เนื้อที่ในการเก็บมาก แต่ปัญหาทั้งหมดนี้สามารถแก้ไขได้ด้วยการเข้ารหัสแบบ MPEG

1.2 การเข้ารหัสข้อมูลแบบ MPEG

MPEG มีความหมายว่า Moving Picture Experts Group เป็นชื่อของกลุ่มให้ความร่วมมือกันสร้างมาตรฐานสากล (International Standard) เพื่อใช้ในการเข้ารหัสของข้อมูลภาพและเสียงที่อยู่ในรูปแบบสัญญาณ Digital ก่อตั้งเมื่อ ค.ศ.1988 ซึ่งถูกบรรจุไว้เป็นมาตรฐานสากล ISO/IEG

การจำแนกประเภทของการเข้ารหัสแบบ MPEG สามารถแบ่งได้เป็น

1.2.1 MPEG-1 ใช้ในการเข้ารหัสข้อมูลภาพและเสียง ในระบบเสียงในวีดีโอซีดี และเสียงเพลง

1.2.2 MPEG-2 ใช้ในการเข้ารหัสข้อมูลภาพและเสียง ในระบบ Digital Television และ DVD

1.2.3 MPEG-4 ใช้ในการเข้ารหัสข้อมูล Multimedia ที่ใช้กันในเว็บเพจ

1.2.4 MPEG-7 เป็นมาตรฐานที่ใช้ประโยชน์ในการเข้าสู่ Internal

1.2.5 MPEG-21 เป็นมาตรฐานเกี่ยวกับ Multimedia Same work

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 การเข้ารหัสแบบ MPEG-1 layer 3 หรือ MP3

การเข้ารหัสแบบ MPEG มีชั้นของ layer ต่างๆ มีความซับซ้อนที่แตกต่างกัน โดย layer ที่ 3 จะมีความซับซ้อนที่มากที่สุด ซึ่งจะสังเกตได้จากตารางที่ 1.1 จะเห็นได้ว่า layer สามารถลดขนาดของข้อมูลหรือบีบอัดข้อมูลได้มากถึง 1:10 หรือ 1:20 แต่การบีบอัดข้อมูลให้ขนาดของข้อมูลเล็กลงไม่ใช่ว่าจะไม่เสียอะไร เนื่องจากการบีบอัดข้อมูลบางส่วนแบบ MPEG นั้นเป็นการบีบอัดแบบมีการสูญเสีย
(Loss Compression) จึงทำให้การสูญเสียบางส่วน แต่การสูญเสียไม่มีผลต่อเสียงที่มนุษย์ได้ยิน โดยใช้คุณสมบัติการฟังของความถี่ของหูมนุษย์

ตารางที่ 1.1 อัตราการบีบอัดข้อมูลและความเร็วในการส่งข้อมูลและความเร็วในการส่งข้อมูลจากเครื่องอ่านดามมาตรา MPEG-1

1:4	By Layer 1 (corresponds with 384 kbps for a stereo signal)
1:6….1:8	By Layer 2 (corresponds with 256…192 kbps for a stereo signal)
1:10….1:12	By Layer 3 (corresponds with 128…112 kbps for a sterco signal)

จากกระบวนการเข้ารหัสที่ซับซ้อนทำให้การแปลงข้อมูลต้องใช้เวลาในช่วงหนึ่งในการทำงาน

1.4 รูปแบบการบีบอัดข้อมูลแบบมาตรฐาน MPEG-1

ซึ่งจะมีรูปแบบ 1 ช่วงสัญญาณและ 2 ช่วงสัญญาณแยกเป็นระบบเสียงได้ 4 ระบบ

1.4.1 ระบบโมโน จะทำให้ข้อมูลออกเพียง 1 ช่องสัญญาณ ซึ่งเป็นได้ทั้งสัญญาณซ้ายหรือขวาก็ได้

1.4.2 ระบบดูอัลโมโน (Dual Mono) ทำให้ข้อมูลผลลัพธ์ออกมา 2 ช่องสัญญาณ ซึ่งช่องหนึ่งเป็นเสียงจากลำโพงฝั่งซ้าย อีกช่องหนึ่งเป็นเสียงจากลำโพงฝั่งขวา

1.4.3 ระบบสเตอริโอ ข้อมูลที่ได้ออกมา 2 ช่องสัญญาณ แต่ช่องหนึ่งเป็นผลรวมของช่องสัญญาณซ้ายและขวา อีกช่องหนึ่งเป็นผลต่างของสัญญาณซ้ายและขวา

1.4.4 ระบบจอยท์-สเตอริโอ มีลักษณะคล้ายสเตอริโอ แต่จะมีการรวมเสียงความถี่ต่ำไว้ในช่องสัญญาณเดียวกัน และแยกความถี่สูงขึ้นมาเหมือนกับระบบสเตอริโอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางที่ 1.2 แสดงให้เห็นรูปแบบของการอัดข้อมูลแบบมาตรฐาน MPEG-1 ในการใช้งานต่างๆนอกจากนี้ยังแสดงอัตราส่วนในการบีบอัดข้อมูล

ตารางที่ 1.2 ความสัมพันธ์ระหว่างคุณภาพเสียงที่ต้องการกับขนาดของข้อมูลที่ถูกบีบอัด

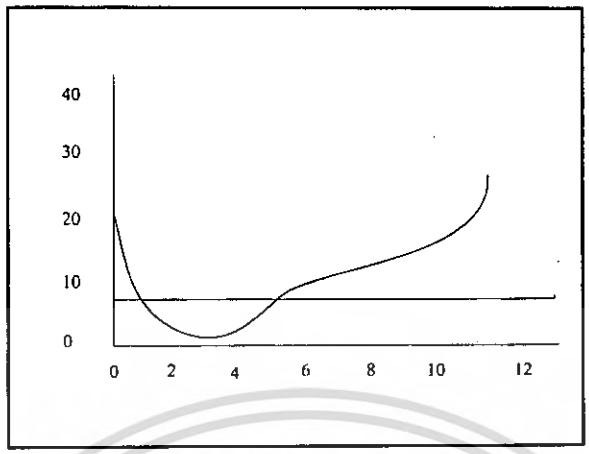
Sound quality	Bandwidth	Mode	Bit rate	Reduction ratio
Telephone sound	2.5 kHz	mono	8 kbps*	96:1
Better than short-wave	4.5	mono	16 kbps	48:1
Better than AM radio	7.5 kHz	mono	32	24:1
Simklar to FM radio	11 kHz	mono	56…64 kbps	26..24:1
Near-CD	15 kHz	mono	96 kbps	16:1
CD	>15 kHz	mono	112..128 kbps	14..12:1
***)Frequency uses a non-ISO extension of MPEG-3 for enhanced performance("MPEG 2.5")**				

1.5 การเข้ารหัสแบบ MPEG หรือการลดขนาดของข้อมูล

การลดขนาดของข้อมูลตามมาตรฐาน MPEG ใช้พฤติกรรมของมนุษย์มาเป็นเครื่องมือในการลดขนาดของข้อมูล จากการที่รู้กันว่ามนุษย์มีความสามารถในการรับฟังของมนุษย์จะได้ยินในช่วงความถี่ 20 Hz ถึง 20,000 kHz แต่ถ้าความถี่เสียงนอกจากนี้มนุษย์จะไม่สามารถได้ยินได้

และในกรณีของเสียงที่อยู่ในความถี่ 20 Hz ถึง 20,000 kHz มนุษย์ก็ไม่สามารถได้ยินเท่ากันทุกๆความถี่เท่าๆกัน ซึ่งสามารถทราบได้จากการทดลองโดยกการกำเนิดเสียงความถี่หนึ่งที่มนุษย์สามารถได้ยิน โดยค่อยๆเพิ่มความดังของเสียงจนหูมนุษย์ได้ยินและทำอย่างนี้ตลอดช่วงความถี่ที่มนุษย์ได้ยิน ซึ่งเป็นไปตามรูปที่ 1.1 จากรูปจะเห็นได้ว่า มนุษย์มีความไวต่อเสียงแตกต่างกัน ในความถี่ประมาณ 2-4 kHz หูของมนุษย์จะมีความไวมาก แต่ถ้าเป็นความถี่สูงหรือต่ำจะใช้ความดังมากจนกว่าที่หูมนุษย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.1 แสดงระดับเสียงเริ่มได้ยินที่ความถี่ต่างๆ

จากคุณสมบัติทั้งหมดรวมเรียกว่า ไซโคอคูสติกโมเดล (Psychoacoustic Model) ซึ่งเป็นวิธีการบีบอัดข้อมูลในมาตรฐาน MPEG

1.5.1 ข้อมูลเสียง Digital นำป้อนเข้าฟิลเตอร์เพื่อแยกเสียงความถี่ย่อย (Subband) ซึ่งมีความกว้างเท่ากับย่อนความถี่วิกฤติ จำนวน 32 ช่วงความถี่เรียกว่า Sub-band Filtering

1.5.2 ใช้ ไซโคอคูสติกโมเดล เป็นเครื่องมือในการวิเคราะห์ ข้อมูลส่วนที่ไม่มีผลต่อการได้ยินของมนุษย์ออกไป โดยการพิจารณาระหว่างความถี่ 2 ช่วงที่ติดกัน และพิจารณาย่อยลงไปในแต่ละช่วงความถี่ด้วย

1.5.3 ทำการวิเคราะห์ ถ้าวิเคราะห์แล้วพบว่าเสียงช่วงใดไม่มีผลต่อการได้ยิน ให้ตัดข้อมูลส่วนนั้นออกไป ไม่นำไปเข้ารหัสในส่วนถัดไป

1.5.4 ข้อมูลที่เหลือ นำข้อมูลที่เหลือมาเข้ารหัสซึ่งจะมีวิธีที่แตกต่างขึ้นอยู่กับแต่ละเลเยอร์

1.6 โครงสร้างข้องมูลของไฟล์ MP3

การบีบอัดข้อมูล MP3 มาตรฐาน สากลจะมีลักษณะเฟรมข้อมูลจะมีส่วนประกอบภายในอยู่ 4 ส่วน คือ

หัวข้อมูล	CRC	ข้อมูลข้างเคียง	ข้อมูลหลัก
32 บิท	0/16 บิท	136,256 บิท	

รูปที่ 1.2 แสดงข้อมูลแบบข้อมูล MPEG-1 layer 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.6.1 หัวข้อมูล (Header) เป็นข้อมูลขนาด 32 บิท แสดงลักษณะของไฟล์นั้นๆ

1.6.2 ส่วนตรวจสอบความผิดพลาด (CRC) เป็นข้อมูลขนาด 16 บิท ใช้ตรวจสอบข้อมูลของเฟรมว่าถูกต้องหรือไม่

1.6.3 ข้อมูลข้างเคียง (Side Information) มีขนาด 17 หรือ 32 บิท เป็นส่วนที่เก็บองค์ประกอบในการถอดรหัส

1.6.4 ข้อมูลหลัก (Main Data) มีความขึ้นอยู่กับอัตราการส่งข้อมูล (Baud rate) และอัตราการส่งข้อมูลในการแปลงกลับเป็นสัญญาณ Analog

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

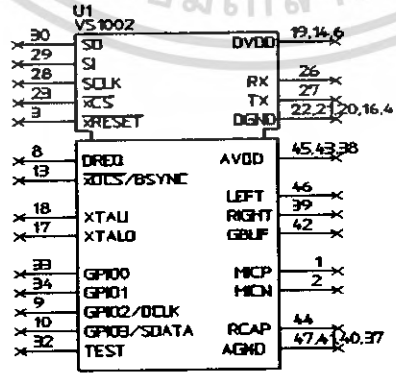
บทที่ 2

VS1011B ชิพถอดรหัสข้อมูลเอ็มเป็ก

2.1 คุณสมบัติ

- สามารถถอดรหัสข้อมูลเอ็มเป็กออดิโอ ได้ทั้งเลเยอร์ 1,2 และ 3
- สนับสนุนการถอดรหัสข้อมูลเอ็มเป็ก 1 และ 2 ทั้งเลอร์ ในเอ็มเป็ก 3 สามารถถอดรหัสได้ถึง เลเยอร์ 2.5 โดยการสนับสนุนทั้งข้อมูลแบบโมโนและสเตริโอ
- สามารถส่งข้อมูลด้วยความเร็วได้หลายระดับ
- ทำงานที่สัญญาณนาฬิกา 12.288 – 16 เมกะเฮิรตซ์ หรือ 24.576 – 26 เมกะเฮิรตซ์ สำหรับความเร็วในส่งต่ำ)
- ประหยัดพลังงาน
- ในชิพประกอบด้วย ดิจิตอลอะนาลอกคอนเวสเตอร์ (DAC) คุณภาพสูง โดยปราศจากเพสเอเรอระหว่างแชลแนล
- สำหรับสัญญาณอะนาลอกทำงานด้วยระดับแรงดัน 2.6-3.6 โวลท์
- สำหรับสัญญาณดิจิตอลทำงานด้วยระดับแรงดัน 2.1-3.6 โวลท์
- มีแรมชิพถึง 4 กิโลบิต สำหรับผู้ใช้
- มีฟังก์ชันใหม่เพิ่มเติม เช่น ข้อมูลเข้าเป็น PCM, ข้อมูลเข้าเป็นสตรีม (Streaming) เป็นต้น
- ประมวลผลใน 16 บิตเวิร์สของข้อมูล

2.2 ข้อมูลของขา VS1011B



รูปที่ 2.1 แสดงชิพ VS1011B แสดงขาต่างๆของชิพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.1 แสดงหน้าที่ของขา

ชื่อขา	ขาที่	ชนิดของขา	หน้าที่ของขา
DREQ	1	DO	ขาแสดงร้องขอข้อมูล
DCLK	2	DIO	ขาสัญญาณนาฬิกาสำหรับข้อมูลอินพุทแบบอนุกรม
SDATA	3	DI	ขาข้อมูลอินพุทแบบอนุกรม
BSYNC	4	DI	ขาสัญญาณแสดงข้อมูลซิงโครไนซ์
DVDD1	5	PWR	ขาไฟเลี้ยงดิจิตอล
DGND1	6	PWR	ขากราวด์ดิจิตอล
XTALO	7	CLK	ขาคริสตอลเอาท์พุท
XTALI	8	CLK	ขาคริสตอลอินพุท
DVDD2	9	PWR	ขาไฟเลี้ยงดิจิตอล
DGND2	10	PWR	ขากราวด์ดิจิตอล
XCS	11	DI	ขาเลือกอินพุทข้อมูลว่าเป็นข้อมูลหรือคอนโทรล
SCLK	12	DI	ขาสัญญาณนาฬิกาสำหรับข้อมูลอินพุทแบบอนุกรม
SI	13	DI	ขาอินพุทอนุกรม
SO	14	DO3	ขาเอาท์พุตอนุกรม
TEST0	15	DI	ขาสำรองไว้ทดสอบต่อ เข้ากับ DVDD
TEST1	16	DIO	ขาสำรองไว้ทดสอบต่อ ไม่ต้องต่อ
TEST2	17	DIO	ขาสำรองไว้ทดสอบต่อ ไม่ต้องต่อ
AGND1	18	PWR	ขากราวด์อนาลอก
AVDD1	19	PWR	ขาไฟเลี้ยงอนาลอก
RIGHT	20	AO	ขาเอาท์พุตแชลแนลขวา
AGND2	21	PWR	ขากราวด์อนาลอก
RCAP	22	AO	ขาเปรียบเทียบความจุไฟฟ้า
AVDD2	23	PWR	ขาไฟเลี้ยงอนาลอก
LEET	24	AO	ขาเอาท์พุตแชลแนลซ้าย
AGND3	25	PWR	ขากราวด์อนาลอก

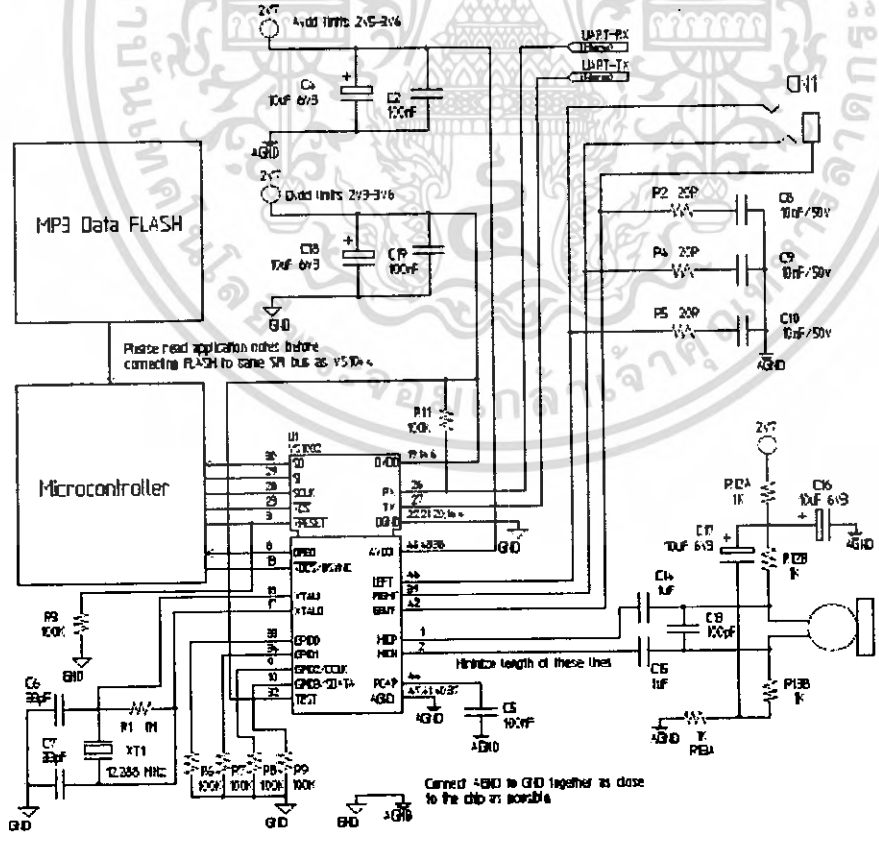
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อขา	ขาที่	ชนิดของขา	หน้าที่ของขา
XRESET	26	DI	ขารีเซตแบบอซิงโคนัส ทำงานที่ระดับ 0
DGND3	27	PWR	ขากราวด์ดิจิตอล
DVDD3	28	PWR	ขาไฟเลี้ยงดิจิตอล

2.3 ชนิดของขา

- DI ขาดิจิตอลอินพุท
- DO ขาดิจิดอลเอาท์พุท
- DIO ขาดิจิตอลอินพุท / เอาท์พุท
- AO ขาอนาล็อกเอาท์พุท
- CLK ขาสัญญาณนาฬิกา / ขาแสดงการต่อกับคริสตอล
- PWR ขาไฟเลี้ยงหรือกราด์

2.3.1 การนำชิพ VS1002D ไปใช้งาน



รูปที่ 2.2 แสดงการนำขา VS1011B ไปต่อใช้งานกับอุปกรณ์ไมโครคอลโทลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 ส่วนของการเชื่อมต่อข้อมูลอนุกรม SPI ของชิพ VS1011B สำหรับคาต้า (SDI : Serial Protocol Data Interface)

สามารถแบ่งได้เป็น 2 แบบคือ

- สามารถเชื่อมต่อข้อมูลอนุกรมสำหรับการส่งข้อมูล
- ส่วนเชื่อมต่อข้อมูลอนุกรมสำหรับการควบคุม

ตารางที่ 2.2 หน้าที่สำหรับขาส่งข้อมลอนุกรม SPI

ขา SDI	ขา SCI	หน้าที่
-	XCS	ถ้าขา XCS ถูกป้อนเข้าด้วยสัญญาณ "0" เป็นการเลือกว่าข้อมูลที่ส่งเป็นข้อมูลเอ็มเป็ก
DCLK	SCK	สัญญาณนาฬิกาของข้อมูลอนุกรม จะถูกใช้เหมือนสัญญาณนาฬิกาหลัก สำหรับอุปกรณ์การอินเตอร์เฟส ในกรณีที่ XCS เป็น "0" ที่ขอบขาขึ้นแรกจะทำให้บิตแรกถูกเขียนลงไปในขณะที่สัญญาณนาฬิกาก็กำลังให้สัญญาณต่อเนื่องไปเรื่อยๆ
SDATA	SI	ข้อมูลอินพุทแบบอนุกรม ในกรณีที่ XCS มีค่าเป็น "0" SI จะเริ่มสุ่มเอาข้อมูลที่ขอบขาขึ้นของสัญญาณ SCK
-	SO	ข้อมูลเอาท์พุท แบบอนุกรมในการอ่าน ข้อมูลจะถูกเลื่อนออกที่ขอบขาสัญญาณ SCK ที่ขอบขาลง ในการเขียน SO จะอยู่ในสถานะ High impedance

2.4.1 คุณลักษณะสำหรับการส่งแบบอนุกรมสำหรับการส่งข้อมูล (SDI)

สำหรับการส่งข้อมูลแบอนุกรมจะสามารถทำงานได้ 2 โหมด คือ มาสเตอร์ และ สเลฟ ในการทำงานในโหมดของมาสเตอร์ VS1011B จะทำการสร้างสัญญาณนาฬิกา DCLK ขึ้นมาเองโดยจะสามารถเลือกระดับความเร็วสัญญาณนาฬิกาได้ 2 ระดับ คือ 512 และ 1024 เมอกะเฮิรต์ ในโหมดสเลฟสัญญาณนาฬิกา DCLK จะถูกสร้างจากวงจรภายนอก

สำหรับข้อมูล (DATA signal) จะสามารถเปลี่ยนแปลงได้ทั้งที่ขอบขาขึ้น และ ขอบลาลงของ DCLK ได้ตามการโปรแกรมของเรา

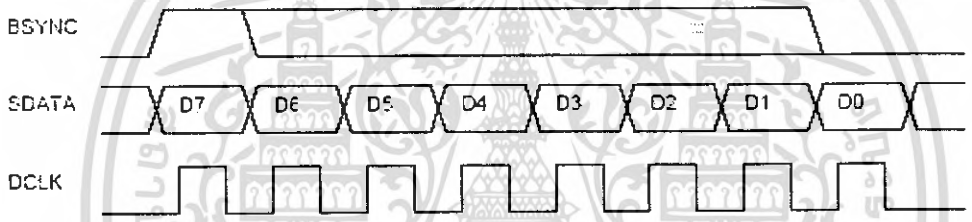
ชิพ VS1002D จะสมมุติเอาข้อมูลอินพุตของไบต์ซิงโครไนส์ เช่น การทำงานภานในตัวดีโค้ดเดอร์ ภายในชิพจะไม่หาไบต์ซิงโครไนส์สำหรับเฟรมจากข้อมูลที่เข้ามาเรื่อยๆ แต่จะสมมุติเอา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลที่เรียงเข้ามาอย่างถูกต้องเท่านั้น โดยไบต์ข้อมูลจะถูกส่งได้ทั้งแบบ MSB และ LSB ขึ้นอยู่กับการเซ็ทในโหมด

เพื่อความแน่ใจว่าข้อมูลที่เป็นไบต์จะถูกต้องจริงๆ การส่งข้อมูลแบบอนุกรมจะมีสัญญาณ BSYNC เข้ามา ระหว่างที่สัญญาณ BSYNC เป็น "1" สัญญาณนาฬิกาจะให้กำเนิดพัลส์แรกออกมา และสัญญาณ BSYNC จะตกลงเป็น "0" ก่อนที่ข้อมูลจะส่งครบ ถ้าสัญญาณ BSYNC ไม่ใช้ เราจะสามารถต่อเข้ากับไฟเลี้ยงภายนอก และอุปกรณ์ที่จะส่งข้อมูลเข้าจะต้องส่งได้อย่างถูกต้อง

สัญญาณ DREQ จะเป็นสัญญาณที่แสดงการร้องขอข้อมูลของ VS1002D ถ้า DREQ เป็น "0" ตัวส่งควรจะหยุดการส่งข้อมูลหม่ เพราะว่าจะเหลือเนื้อที่ 32 ไบต์ เป็นเนื้อที่ปลอดภัย ซึ่งจะง่ายสำหรับพวกอุปกรณ์ไมโครคอลโทลเลอร์ที่มีความเร็วต่ำ ถ้าหากไม่มีการตรวจสอบสัญญาณ DREQ ตัวส่งอาจจะมากเกิน 32 ไบต์ ทำให้ไม่อาจเกิดการดีโค้ดข้อมูลได้



รูปที่ 2.3 แสดงสัญญาณ BSYNC

2.5 ส่วนการเชื่อมต่อข้อมูลอนุกรม SPI ของชิพ VS1011B สำหรับคอลโทรเลอร์ (SCI : Serial Protocal for Serial Command Interface)

โดยปกติชุดคำสั่งของการส่งข้อมูลอนุกรมสำหรับไมโครคอลโทลเลอร์ จะประกอบด้วย ชุดคำสั่ง ,แอดเดรสไบต์ และ 16 บิต ดาต้าเวิรส์ (intrusions +address byte + 16 bit data word) โดยชุดคำสั่ง (intrusions) จะเป็นข้อมูล 8 บิด ซึ่งเป็นคำสั่งสำหรับการอ่านและเขียนดังตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.3 การแสดงชุดคำสั่งของชิพ VS1011B

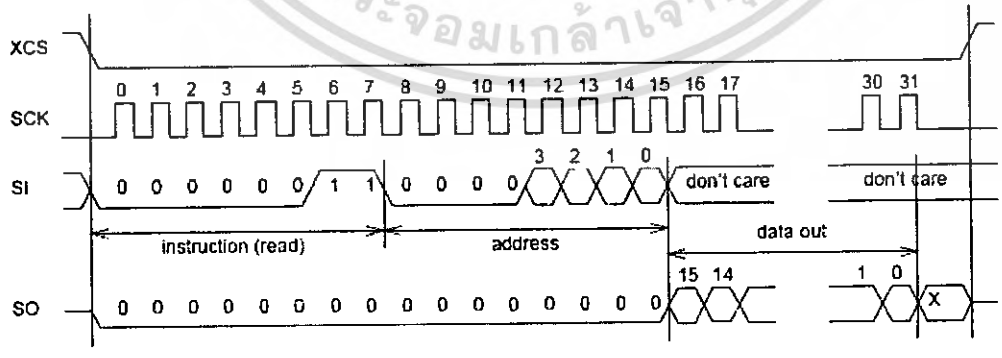
ชุดคำสั่ง		
ชนิด	**ออปโค้ด (Opcode)**	**ลักษณะการทำงาน**
READ	0000 0011	Read Data
WRITE	0000 0010	Write Data

*** หลังจากได้รับคำสั่งควบคุมแล้วตัวชิพ VS1011B จะไม่ยอมให้ส่ง SCI หรือ SDI ได้อีกเป็นระยะเวลา 5 ไมโครวินาที ****

2.6 การอ่านข้อมูลสำหรับ SCI (SCI Read)

VS1011B รีจิสเตอร์จะถูกอ่านตามลำดับดังนี้

- ให้สัญญาณ XCS เป็น "0" เพื่อเลือกให้ทำงานที่ SCI
- จากนั้นส่ง ออปโค้ดอ่าน (Read Opcode) 03H ผ่านสาย SI จากนั้นส่ง 8 บิต แอดเดรสตามลำดับ
- หลังจากที่ส่งแอดเดรสไปแล้ว หากมีข้อมูลใดส่งไปตามหลังชิพ VS1002D จะไม่สนใจแต่จะส่งข้อมูล 16 บิต ที่อยู่ในแอดเดรสที่ต้องการ โดยชิพ (Shifted) ออกไปทางสาย SO
- ให้สัญญาณ XCS เป็น "1" หลังจากชิพ (Shifted) ข้อมูลออกเรียบร้อยแล้ว เพื่อให้จบกระบวนการอ่านข้อมูล



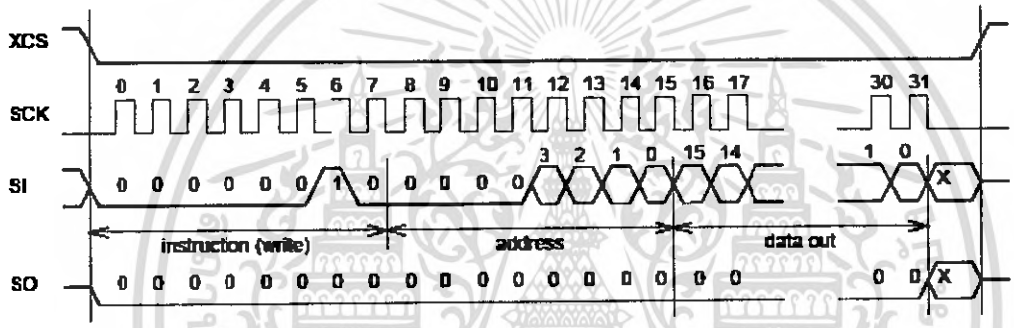
รูปที่ 2.4 แสดงสัญญาณการอ่านข้อมูลผ่านพอร์ตอนุกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 การเขียนข้อมูลสำหรับ SCI (SCI Write)

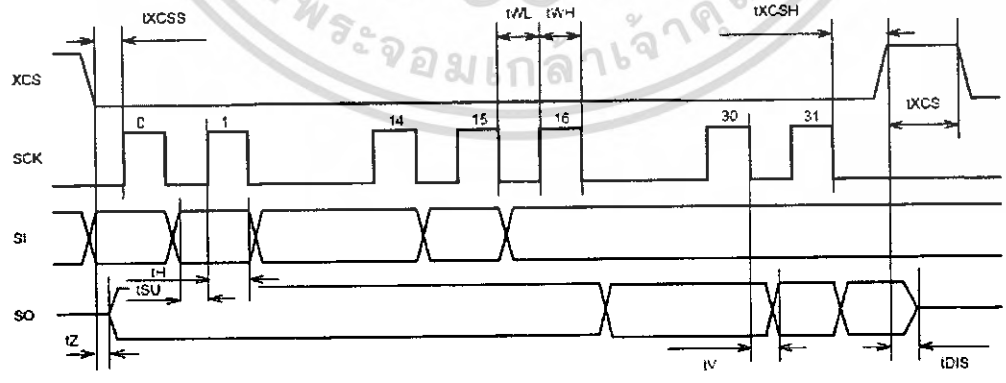
VS1011B รีจิสเตอร์จะถูกเขียนตามลำดับดังนี้

- ให้สัญญาณ XCS เป็น "0" เพื่อเลือกให้ทำงานที่ SCI
- จากนั้นส่ง ออปโค้ดเขียน (Write Opcode) 02H ผ่านสาย SI จากนั้นส่ง 8 บิต แอดเดรสตามลำดับ
- จากนั้นส่งข้อมูลตามไป ข้อมูลจะถูกชิพ (Shifted) ออกไปตามสาย SI
- ให้สัญญาณ XCS เป็น "1" เพื่อจบกระบวนการเขียนข้อมูลลงใน SI
- ระดับสัญญาณ XCS เปลี่ยนไปจาก "0" ไป "1" จะเกิดหลังสัญญาณ SCLK เปลี่ยนจาก "1" ไป "0" การเปลี่ยนแปลงจะเกิดตรงกับบิด LSB ของข้อมูล



รูปที่ 2.5 แสดงสัญญาณการเขียนข้อมูลผ่านพอร์ตอนุกรม

7.6 SPI Timing Diagram



รูปที่ 2.6 การแสดงทามมิ่งไดอะแกรมของพอร์ตอนุกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 ลักษณะการทำงาน

VS1011B ใช้คุณสมบัติพื้นฐานการประมวลผลสัญญาณ VS_DSP ภายในบรรจุรหัสและหน่วยความจำที่จำเป็นสำหรับการถอดรหัสเอ็มเป็กทั้งหมด พร้อมด้วยพอร์ตอนุกรม กับตัวขยายควบคุมเอาท์พุตที่เป็นสัญญาณอนาล้อก สเตอริโอออดิโอ DAC หลายระบบ

ข้อมูลและบิตเรต ในส่วนเพิ่มเติทสามารถส่งข้อมูลด้วยความเร็วหลายระดับ (VBR) เป็นตัวสนับสนุน VBR ที่ใช้สำหรับเพลงจากใกล้เคียง VS1011B สามารถเล่นได้ทั้งไฟล์เอ็มเป็ก 1 และ 2 เลเยอร์ 1,2 และ 3 ส่วนประกอบไฟล์ทั้ง อัตราสุ่ม CD นั้นประมาณ 100 กิโลบิต/วินาที สำหรับตัวอย่างเพลงระบบสเตอริโอที่ความถี่ 44,100 เฮริตซ์ ด้วยเหตุที่การเข้ารหัสแบบเก่าต้องใช้ 128 กิโลบิต/วินาที สำหรับการทำงานเต็มที่ VBR การเข้ารหัสคุณภาพสูงได้ถูกนำมาใช้ประโยชน์อย่างกว้างขวาง

Serial Data อินเตอร์เฟส มีความหมายสำหรับการส่งข้อมูลที่ถูกบีบอัดแบบเอ็มเป็กออดิโอ

2.9 Serial Control (SCI)

Serial Control (SCI) สามารถเข้ากันได้กับ ลักษณะเฉพาะของ SPI-bus การส่งข้อมูลแบบ 16 บิต สามารถควบคุม VS1011B โดยการเขียนและอ่านรีจิสเตอร์ของอินเดอร์เฟส

2.9.1 การควบคุมหลักการเชื่อมต่อการควบคุม

- คำสั่งในการทำงาน
- การป้อนคำสั่งในการทำงาน
- การเข้าถึงข้อมูล
- สถานะของข้อมูล
- กระบวนการถอดรหัสข้อมูล
- การป้อนข้อมูล

2.9.2 รีจิสเตอร์สถานะ Status (RW)

รีจิสเตอร์สถานะบรรจุข้อมูลบน สถานะปัจจุบัน ของ VS1002D บิต 1 และ 0 ใช้ควบคุมอนาลอกเอาท์พุท ระดับเสียง 0 = -0 db , 1=6 db , 3 = -12 db บิต 2 คือ อนาลอก Powerdown บิต เมื่อเซ็ทเป็น 1 อนาลอก Powerdown

INT_FCNTLH(-)

INT_FCNTLH ไม่ใช่ user-accessible รีจิสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.4 แสดง SCI รีจิสเตอร์

Name	Type	Address	Function
โหมด	RW	0	โหมดคอลโทรล
STATUS	RW	1	สถานะของชิพ VS1011B
INT_FCTLH	-	2	รีจิสเตอร์ภายใน
CLOCKF	RW	3	ความเร็วสัญญาณนาฬิกา
SRATE	R	4	อัตราการสุ่มข้อมูล
AUDATA	R	5	ข้อมูลเสียงความดัง
WRAM	W	6	ส่วนการเขียนโปรแกรมลงหน่วยความจำ
WRAMADDR	W	7	แอดเดรสของหน่วยความจำที่ต้องการเขียน
HDAT0	R	8	จุดอ่านข้อมูล
HDAT1	R	9	จุดอ่านข้อมูล
A1ADDR	RW	10	แอดเดรสที่ทำการเริ่มโปรแกรม
VOL	RW	11	ควบคุมความดังของเสียง
A1CTRL[X]	RW	12+X	รีจิสเตอร์สำหรับการประยุกต์ใช้งานต่าง

ตารางที่ 2.5 แสดงโหมดที่ใช้ควบคุมการทำงานของ VS1011B

บิต	Function	สถานะ
โหมด(2)	โปรแกรมรีเซต	0ไม่รีเซต , 1รีเซต
โหมด(4)	ประหยัดพลังงาน	0 ไม่มี , 1 มี
โหมด(8)	สัญญาณ DCLK ทำหน้าที่ขอบขาขึ้น.ขาลง	0 ขอบขาขึ้น ,1ขอบขาลง
โหมด(9)	ส่งบิตไหนก่อนหน้าหรือหลัง	0 MSB ก่อน , 1 MSB หลัง
โหมด(10)	โหมดอินพุตข้อมูล	0 สเลพ , 1 มาสเตอร์
โหมด(11)	ถ้าเป็นมาสเตอร์จะมีสัญญาณนาฬิกา	0 (512KHz) , 1(1024KHz),

หมายเหตุ โหมด 0 , 1 , 3 , 5 , 6 ,7 ไม่ใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9.3 รีจิสเตอร์สัญญาณนาฬิกา CLOCKF (RW)

รีจิสเตอร์สัญญาณนาฬิกาซึ่งจะทำงานบางอย่างที่ความถี่มากกว่า 24,576 เมกะเฮริตซ์ ความเร็วของสัญญาณนาฬิกาจะเซ็ตในทุกๆ 2 กิโล เฮริตซ์ ดังนั้น สูตรการคำนวณค่าที่ถูกต้องสำหรับ รีจิสเตอร์นี้ คือ ClockIn Hz/2000 ค่านี้อยู่ระหว่าง 0.32767 ถึงแม้ว่าฮาร์ดแวร์จะมีขอบเขตความเร็วสูงสุด ที่ความเร็วต่ำกว่า 24.576 เมกะเฮริตซ์ ทั้งอัตราการสุ่มข้อมูล และบิตสตรีมจะทำงานไม่นานนัก ถ้ากำหนดให้ MSB ของ CLOCKF เป็น 1 จะทำงานภายใน clock-doubling clock ที่สูงถึง 15 เมกะเฮริตซ์ อาจจะเป็น doubled ถ้าทำงานเป็นClock doubled 15 LSBs ของรีจิสเตอร์ จะมีค่า Clock doubled ที่ใช้คำนวณ ดังนั้น รีจิสเตอร์นี้ต้องเซตค่าก่อนเริ่มการถอดรหัสข้อมูล MP3 มิฉะนั้นอัตราการสุ่มข้อมูล จะเซตไม่ถูกต้อง

ตัวอย่าง 1 : สำหรับความเร็วสัญญาณนาฬิกาที่ 26 เมกะเฮริตซ์ ค่าจะเป็น 13000

ตัวอย่าง 2 : สำหรับความเร็วสัญญาณนาฬิกาภายนอกและใช้ clock-doubling สำหรับ 27 เมกะเฮริตซ์ ค่าจะเป็น 0x8000+13500 = 46263

2.9.4 รีจิสเตอร์ SRATE (R)

เมื่อถอดรหัสข้อมูลถูกต้อง ออดิโอสุ่มข้อมูล สามารถพบใน SRATE ไม่เป็นจำนวนเต็ม

2.9.5 รีจิสเตอร์ AUDATA (R)

เมื่อถอดรหัสข้อมูลถูกต้องอัตราบิตเรตในปัจจุบันใน kบิตs/s สามารถพบใน 8.0 ของ **AUDATA** สำหรับบิตสตรีม Variable บิตเรท

2.9.6 รีจิสเตอร์ WRAM (W)

WRAM ใช้โหลดโปรแกรมประยุกต์เป็นโปรแกรมแอดเตรสของแรมเริ่มต้นเป็นตัวแรกโดยการเขียนเป็น WRAMDDR รีจิสเตอร์ ลำดับถัดไปเป็นการเรียกครั้งแรกของ WRAM ค่าที่ใช้ 16 บิตs ของข้อมูลสามารถส่งได้กับหนึ่งการเขียน WRAM และโปรแกรมเวิร์สเป็น 32 บิต สองการเขียนที่ตามมาจำเป็นสำหรับแต่ละโปรแกรมเวิร์สไบด์ ลำดับคือบิตที่มีค่าสูงสุดก่อน

2.9.7 รีจิสเตอร์ WRAMDDR (W)

WRAMDDR ใช้เซ็ทแอดเดรสของโปรแกรมสำหรับดิดตามการเขียน WRAM พื้นที่สำหรับให้ผู้ใช้เขียน ระหว่างแอดเดรส 4096...5119 (ที่แอดเดรส 4096...4111 จะถูกสำรองโดย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบ) แต่สำหรับการเขียน WRAM ทั้งหมดจะสนใจ แอดเดรสสูงกว่า 4096 ดังนั้นถ้าโปรแกรมต้องการเขียนที่ 4567 ต้องเขียนเป็น 4567+4096 = 8663 WRAMDDR

ตารางที่ 2.6 รีจิสเตอร์ HDATA and HDAT1 (R)

บิต	การทำงาน	ค่า	คำอธิบาย
HDAT[4 :3]	ID	3	MPG 2.5 (1/4-rate)
		2	MPG 2.5 (1/4-rate)
		1	ISO 11172-3 1.0
		0	MPG 2.0 (1/2-rate)
HDAT[2 :1]	LAYER	3	I
		2	II
		1	III
		0	สำรองไว้
HDAT [0]	โปรเท็กส์บิต	1	มีการโปรเท็กส์ด้วย CRC
		0	ไม่มี CRC
HDAT0 [15 :12]	บิตเรท		c.f. 11172-3
HDAT0 [11 :10]	อัตราการสุ่มข้อมูล	3	สำรองไว้
		2	32/16/8 KHz
		1	48/24/12 KHz
		0	44/22/11 KHz
HDAT0 [9]	Padrate	1	Addition slot
		0	Normal frame
HDAT0 [8]	Private bit		Not defined
HDAT0 [7:6]	Mode	3	โมโน
		2	ช่องสัญญาณคู่
		1	จอยส์สเตริโอ
		0	สเตริโอ
HDAT0 [5 :4]	ส่วนขยาย		c.f. 11172-3
HDAT0 [3]	ลิขสิทธิ์	1	มีลิขสิทธิ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

บิต	การทำงาน	ค่า	คำอธิบาย
		0	ฟรี
บิต	การทำงาน	ค่า	คำอธิบาย
HDAT0 [2]	ต้นฉบับ	1	ต้นฉบับ
		0	ก็อปปี้
HDAT0 [1:0]	Emphasis	3	CCITT J017
		2	สำรองไว้
		1	50/15 ไมโครนาที
		0	None

2.9.8 รีจิสเตอร์ A1ADDR (RW)

A1ADDR เป็นตัวแสดง แอดเดรส เริ่มต้นของการเขียนรหัสประยุกต์ ที่เร็วกว่า รีจิสเตอร์ WRAMDDR และ WRAM ถ้าไม่ใช่รหัสประยุกต์ รีจิสเตอร์นี้จะไม่เป็นตัวแรกหรือตัวแรกเป็น 0

2.9.9 รีจิสเตอร์ VOL (RW)

VOL ตัวควบคุมระดับเสียง สำหรับเครื่องเล่น สำหรับแต่ละช่องสัญญาณ ค่าในช่วงของ 0...255 อาจจะกำหนดให้มันเบาลงจากระดับสัญญาณเสียง สูงสุด (ระดับ 0.5 dB) ทางช่องซ้ายจะมีหลายค่าโดย 256 และค่าที่เพิ่มขึ้น ดังนั้นระดับสัญญาณเสียง สูงสุดเป็น 0 และ เงียบถ้าเป็น 65535 ตัวอย่างสำหรับระดับเสียง -2.0 dB ของช่องซ้ายและ -3.5 dB ของช่องขวา : (4*256)+7 = 1031 ดังนั้นที่ startup ระดับสัญญาณเสียง เซตเป็นเต็ม 1 ระดับสัญญาณเสียง การรีเซตซอฟต์แวร์ต้องไม่รีเซตระดับเสียง ดังนั้นการเซตระดับสัญญาณเสียง เป็น total silence (255 สำหรับทั้งช่องซ้ายและขวา) จะปิดอนาลอก power แต่ทำให้เกิดการสั่นในหูฟัง ถ้าคุณต้องการปิดระดับสัญญาณเสียง แต่ไม่ต้องการสั่นนี้ ต้องปรับระดับสัญญาณ เป็น 254 ทั้ง 2 ช่อง (0xFEFE)

2.9.10 รีจิสเตอร์ A1CTRL (RW)

A1CTRL –รีจิสเตอร์ x=[0.2] สามารถใช้เข้ากับโปรแกรมประยุกต์ของผู้ใช้ได้

72747

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมประยุกต์

มี 1 kWord (32 บิต) ของหน่วยความจำ RAM สำหรับผู้ใช้บนชิพ VS1011B นี้ใช้สำหรับเตรียมลักษณะตัวอย่างให้เหมือนกัน

- เบส,เสียงแหลม เป็นต้น
- แชลแนลผสมเสียง (สเตอริโอในโมโน)
- ดิจิตอลอีคอลไรเซอร์

การโหลดโปรแกรมประยุกต์ เป็นตัวแรกโดยการเขียนที่แอดเดรสเป็น WRAMDDR รีจิสเตอร์โปรแกรมตอนที่โหลดโดยการเขียนข้อมูลเป็นรีจิสเตอร์ WRAM

ทุกโปรแกรมจะมีการรีโหลดทุกครั้งที่ชิพ ขาดแหล่งจ่ายไฟประยุกต์แอดเดรส จะไม่เซตทุกครั้งตามเมื่อซอฟต์แวร์ระบบมีการรีเซต

ถ้า chan คือ อย่างใดอย่างหนึ่งใน 1 หรือ 2 ผู้ใช้อาจทำ in-place filtering ของ chan คือ จำนวนของช่องและ nSampl คือ จำนวนรวมของตัวอย่างที่ใช้ ที่สัญญาณ stereo ทั้งช่องและขวา ตัวอย่างเป็นตัวแทรก โปรแกรมอาจอาศัย nSampl นั้นแบ่งเป็น 4

ถ้า chan คือ 3 ระดับเสียง จะใช้การเซตแบบ SPI ในกรณีนี้ระดับเสียง จะเซตโดยอัตโนมัติ แต่ซอฟต์แวร์ยังคงเตรียมบางข้อมูลของการเซตระดับสัญญาณเสียง ใหม่ : nSampl ผู้ใช้จะเลือกเซตระดับสัญญาณเสียง และจุด d ที่ระดับสัญญาณเสียง multiplier รีจิสเตอร์ทางซ้าย ที่ทางขวา ระดับสัญญาณเสียง multiplier คือ d+1 ค่าระดับสัญญาณเสียง ที่ไม่ปรากฏ และ 32768 มีลักษณะกันgain 1.0 (e.g. 16384 -> 0.5)

ถ้า chan เป็น 4.6, AICTAL[chan-4] จะใช้เรียกและผู้ใช้ให้ค่า nSampl

ระดับสัญญาณเสียง คอนโทรล อยู่ที่ตำแหน่งตามหลังทุก ผู้ใช้โปรแกรม ดังนั้นโดยทั่วไปจะดีกว่าความคิดที่มีเพียง write filters นั้นแบ่งบางความถี่และไม่เน้นทั้งหมด ที่การชดเชยสำหับระดับสัญญาณเสียงที่ต่ำกว่า การเซตระดับสัญญาณเสียง หลัดอาจจะเซตอับน้อย dB เมื่อโทนคอนโทรล กำลังทำงาน

2.9.11 Stereo Audio DAC

การถอดรหัสข้อมูลดิจิตอลเป็นการเปลี่ยนรูปแบบ เป็นอนาลอก โดยการสุ่มตัวอย่าง 18 บิต multi-บิต sigma delta DA-converter เอาท์พุทการสุ่มตัวอย่างจะผ่านการกรองความถี่ต่ำโดยอนาลอกฟิวเตอร์ อัตราเอาท์พุทของ DA-converter จะเป็น1/4 ของอัตราสัญญาณนาฬิกา หรือ 128 เวลาสูงสุดอัตราการสุ่มตัวอย่างที่ใช้ สำหรับกรณีที่ใช้สัญญาณนาฬิกา 24.576 เมกะเฮิรตซ์ DA-converter จะทำงาน 128x48 กิโลเฮิรตซ์ เป็น 6.1444 เมกะเฮิรตซ์ ถ้าอัตราตัวอย่างอินพุทอยู่มากกว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

48 กิโลเฮิรตซ์ จะมีการเปลี่ยนแปลงภายใน 48 กิโลเฮิรตซ์ โดย DAC การถอดออกต้องการแบบแผนเฟสล็อกลูป ที่ซับซ้อนและยังคงยอมให้ใช้อัตราการตัวอย่างหลายค่ากับหนึ่งความถี่สัญญาณนาฬิกาหลักหลัก ที่ระบุ

เอาท์พุทสามารถแยกความเงียบโดยผู้ใช้ ถ้าเอาท์พุทของการถอดรหัสถูกยกเลิก หรือ ข้อมูลอินพุทไม่รองรับความเร็วเพียงพอ อนาลอกเอาท์พุทจะเงียบโดยอัตโนมัติ อนาลอกเอาท์พุทมีบัฟเฟอร์ที่สามารถกระตุ้นโหลด 30 Ω ค่าสูงสุดของตัวเก็บประจุ 50 nF

2.10. การทำงานส่วนต่างๆของชิพ

2.10.1. สัญญาณนาฬิกา

ชิพ VS1011B จะทำงานบน 24.576 เมกะเฮิรตซ์ ของสัญญาณนาฬิกาหลักนี้ สามารถกำหนดจากวงจรภายนอก (ต่อที่ขา XTAL) หรือ โดยภายในคริสตอลอินเตอร์เฟส (ขา XTAL1 และ XTAL0) นี้เป็นสัญญาณนาฬิกา ที่พอเพียงกับเอาท์พุท ออดิโอคุณภาพสูงสำหรับทุกมาตรฐานอัตราตัวอย่าง

2.10.2. โหมดประหยัดพลังงาน

ในโหมดประหยัดพลังงาน ชิพแสดงเพียงสายคอลโทรลตัวขับอนาลอกเอาท์พุทจะปิดและการประมวลผลตกค้างในสถานะคงค่า (Hold State)

การรีเซตตัวเครื่อง

เมื่อขา xRESET ป้อนพัลส์ “0” เข้าไป VS1011B จะรีเซตและทุกรีจิสเตอร์คอลโทรลและค่ากำหนดภายในต้องเซตเป็นค่าเริ่มต้น สัญญาณ xRESET เป็นอซิงโครนัสไม่ขึ้นกับสัญญาณนาฬิกาภายนอก รีเซตโหมดดับเบิลจะฟูลเพาเวอร์ดาวน์โหลด เมื่อทั้งส่วนสัญญาณดิจิตอลและอนาลอกของชิพ VS1011B เป็นเพาเวอร์น้อยสุด จะทำให้เมื่อสัญญาณนาฬิกาหยุด

หลังจากตัวเครื่องรีเซตแล้ว จะเซตให้ตัวโปรแกรมในรีจิสเตอร์เช่นรีจิสเตอร์ควบคุมเสียงให้พร้อมแล้วจึงทำการเริ่มถอดรหัสสัญญาณเสียง

การรีเซตโปรแกรม

ระหว่างข้อมูลข้อมูลเอ็มเป็ก หากมีการรีเซตซอฟต์แวร์จะมีการกระตุ้นให้บิตที่ 2 ในรีจิสเตอร์ SCI และต้องรออย่างน้อย 2 ไมโครวินาที จากนั้นขา DREQ จะเป็น “0” ประมาณ 6000 แมทชีนไซเคิล (ประมาณ 250 ไมโครวินาทีจะเป็นการหน่วงเวลาเอาไว้เพราะว่าชิพ VS1011B จะทำงานที่สัญญาณนาฬิกา 25.576 เมกะเฮิรตซ์ เมื่อสัญญาณ DREQ จะเป็น “1” ให้ส่งข้อมูลใดๆ เข้าสู่ส่วน SDI หลังจากนี้ตัวชิพก็จะทำงานต่อไปได้อย่างปกติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าต้องการให้แน่ใจว่าตัวชิพ VS1011B จะไม่หยุดกลางคันถ้าหากว่ามีการส่งข้อมูลมาช้ามากให้ส่ง "2048II" เข้าสู่ส่วน SDI ก่อนที่จะทำการรีเซตโปรแกรม

การถอดรหัสข้อมูลหลักเอ็มเป็ก

โดยการทำงานปกติส่วนข้อมูลเอ็มเป็กเมื่อถูกถอดรหัส ตัวถอดรหัสจะทำการสุ่มเอาข้อมูลเอ็มเป็กแล้วมาทำการถอดรหัสเป็นสัญญาณดิจิตอลภายในใจส่วน DAC ถ้าหากมีการผิดพลาดระหว่างการถอดรหัสแล้วจะมีการเซตบิตขึ้นในรีจิสเตอร์คอลโทรล(SCI) ที่บิต HDAT0 และบิต HDAT0 ในกรณีที่เกิดล้มเหลวขึ้นในการส่งข้อมูลตัวถอดรหัสจะยังคงทำงานแต่สัญญาณอนาลอกจะไม่มีสัญญาณออกมา

2.11 การตรวจสอบว่ามีการทำงาน

ถึงแม้ว่าตัวชิพเองจะมีการตรวจสอบอยู่แล้ว แต่ก็อาจมีเหตุให้เกิดการไม่ถอดรหัสข้อมูลบางตัวทำให้ตัวโปรแกรมทำงานผิดพลาดได้ โดยอาจเกิดในกรณีการเร่งสปีดในการถอดรหัส

ตัวไมโครคอลโทรลเลอร์เองจะเก็บค่าความเร็วที่ VS1011B ต้องการไว้ถ้าตัวข้อมูลเกิดการหยุดส่ง หรือ ตัวชิพต้องการข้อมูลมากกว่า 60 กิโลบิตในทุกๆวินาที มันจะเป็นส่งผลต่อตัวไมโครคอลโทรลเลอร์เวลาตัวโปรแกรมเกิดการรีเซต ถ้าหากไม่มีการตรวจสอบแล้วการรีเซตทางฮาร์ดแวร์จะยังคงทำงาน

2.11.1 การทดสอบ

มีการทดสอบหลายอย่างในการทดสอบอย่างในการทดสอบชิพ VS1011B โดยอาจมีการทำสอบหน่วยความจำ,ส่วนส่งสัญญาณอนุกรมและเอาท์พุตสัญญาณไซน์ที่มีความถี่ตั้งแต่250-1500Hz

การทำทดสอบสามารถได้หลายทาง ถ้าหากมีข้อมูลเอ็มเป็กถูกถอดรหัสตัวถอดรหัสจะจบการถอดรหัสโดยทำการส่งข้อมูล 1024H ออกมา ทำให้มั่นใจได้ว่าถอดรหัสมองหาข้อมูลตัวต่อไปและจะยังไม่ถอดรหัสข้อมูลโดยปราศจากข้อมูลเอ็มเป็ก ถ้าไม่มีข้อมูลเอ็มเป็กถูกถอดรหัส ตั้งแต่การรีเซตครั้งก่อนขั้นตอนนี้ก็จะไม่เกิดขึ้น การทดสอบแต่ละครั้งจะเริ่มโดยส่งคำสั่ง 4 ไบต์ตามลำดับ โดยการส่งแต่ละครั้งจะให้เอาท์พุตออกมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.11.2 อัตราเร็วสูงสุดของสัญญาณนาฬิกา

การใช้สัญญาณนาฬิกาอื่นที่ไม่ใช่ความเร็วของสัญญาณนาฬิกาที่ 24.576 เมกะเฮิรตซ์ นี้จะส่งผลต่อความเร็วของการสุ่มเอาข้อมูล ความเร็วในการถอดรหัสบิตข้อมูลและผลกระทบต่อตัว DSP ความเร็วสูงสุดในการสุ่มข้อมูล

ที่สัญญาณนาฬิกา 24.576 เมกะเฮิรตซ์ หรือสูงกว่า อัตราการสุ่มเอาข้อมูลสูงสุดอยู่ที่ 48000 Hz สำหรับที่ความเร็วต่ำว่าสามารถคำนวณได้จาก (4800 x ความเร็วสัญญาณนาฬิกา) / 24.576 จะเห็นว่าถ้าความเร็วของสัญญาณนาฬิกาลดลง จะทำให้ความเร็วในการสุ่มเอาข้อมูลลดลงด้วย

หากนำเอาสัญญาณนาฬิกาที่ 26 เมกะเฮิรตซ์ มาใส่จะเป็นความเร็วสูงสุดของสัญญาณนาฬิกาใส การสุ่มเอาข้อมูลโดยจะมีความเร็วในการสุ่มข้อมูลถึง 50781 Hz ดังนั้น ทุกๆข้อมูลเอ็มเป็กจะถูกถอดรหัสออกมาอย่างถูกต้องที่ความถี่นี้ หากให้สัญญาณนาฬิกา 24 เมกะเฮิรตซ์ ลงไป ความเร็วสูงสุดในการสุ่มเอาข้อมูลจะเป็นแค่ 46875 Hz ในกรณีนี้ความเร็วในการสุ่มเอาข้อมูลที่ต่ำกว่า 44100 Hz สามารถถอดรหัสข้อมูลเอ็มเป็กได้อย่างถูกต้องแต่ที่ 48000 Hz แล้วจะไม่สามารถถอดรหัสได้อย่างถูกต้อง

2.11.3 อัตราเร็วสูงสุดของบิตข้อมูลในการถอดรหัส

ที่สัญญาณนาฬิกา 24.756 เมกะเฮิรตซ์ อัตราในการถอดรหัสข้อมูลเอ็มเป็กได้ 256 กิโลบิต/วินาที ที่สัญญาณนาฬิกา 28.5 เมกะเฮิรตซ์ สามารถถอดได้ถึง 320 กิโลบิต/วินาที และหากความเร็วของสัญญาณนาฬิกา

รีจิสเตอร์ของชิพ VS1011B

รีจิสเตอร์ทั้งหมดจะอยู่ที่ X-memory

รีจิสเตอร์ SCI เริ่มที่แอดเดรส 4000H

รีจิสเตอร์ SCI ทั้งหมดจะอยู่ระหว่างแอดเดรส 4000 H ถึง 40FFH

รีจิสเตอร์อนุกรม อยู่ที่แอดเตรส 4100H

SER_DATA (4100H) จะมีข้อมูลสุดท้ายที่อ่านมาจากดาต้าที่รับส่ง

ส่วน LSB ของ RES_DREQ (4101H) จะบอกถึงสถานะของสัญญาณ DREQ

รีจิสเตอร์ DAC อยู่ที่แอดเดรส 4200 H

ข้อมูลที่แปลงจากดิจิตอลเป็นอนาลอกจะถูกบันทึกเป็นลักษณะออดิโออินเตอร์รัปต์ โดยจะบรรจุข้อมูลเป็นลักษณะเป็นลักษณะค่าไซต์ (signed values)ที่ DAC_LAFT (4200H) และ DAC_RIGTH (4101H) ส่วน INF_FCTLL (4202H) ไม่มีการใช้งาน

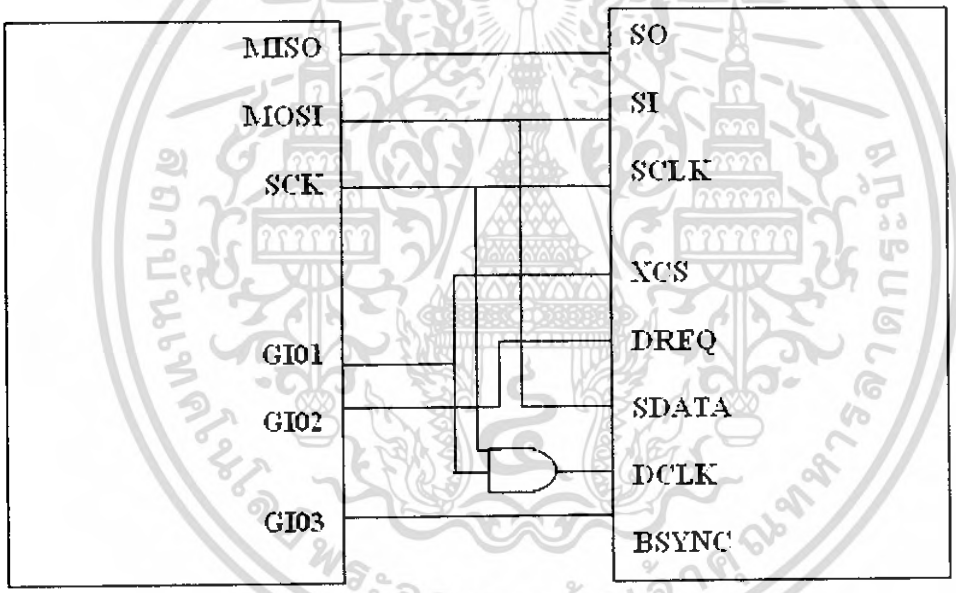
รีจิสเตอร์อินเตอร์รัปต์ที่ 4300 H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

INT_ENABLE(4300H) จะควบคุมการอินเตอร์รัปต์ โดยบิตที่ 0 หากต้องการให้มีการอินเตอร์รัปต์ ก็เซตให้มีค่าเป็น "1" หากไม่ต้องการให้มีการอินเตอร์รัปต์ก็เซตให้เป็น "0" บิตที่ 1 จะควบคุมอินเตอร์รัปต์ของ SPI และบิตที่ 2 จะควบคุมอินเตอร์รัปต์ของข้อมูล โดยการเกิดอินเตอร์-รัปต์แต่ละครั้งจะกินเวลา 6 Clock cycle ก่อนจะมีการเปลี่ยนแปลงรีจิสเตอร์นี้

2.12 การนำ VS1011B ไปใช้งาน

โดยการต่อใช้งาน VS1011B ใช้งานโดยปกติแล้วจะต้องต่อขาใช้งาน 8 ขา แต่เราสามารถลดการต่อลงให้เหลือ 7 หรือ 6 ขาได้ โดย 6 ขาถือเป็นต่อแบบประหยัดขาที่สุดแล้ว สามารถแสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 แสดงการต่อชิพ VS1011B แบบ 6 ขา

และหากแน่ใจว่าการส่งข้อมูลผ่าน SDI สามารถส่งได้อย่างถูกต้อง ขา BSYNC อาจจะไม่ต้องต่อกับไมโครคอลโทรลเลอร์ก็ได้ โดยนำไปต่อกับไฟเลี้ยง(VDD)

สิ่งที่ต้องเตรียมให้พร้อมสำหรับการทำงาน VS1011B

- MISO และ GIO2 ต้องเซตให้เป็นข้อมูลอินพุต ส่วนขาที่เหลือทั้งหมดให้เป็นอินพุต
- สัญญาณนาฬิกาของ SPI ต้องทำงานเมื่อ SPI มีการส่งข้อมูล และสัญญาณนาฬิกาจะต้องเป็น"0" เมื่อไม่มีการส่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้าตัวไมโครคอนโทรลเลอร์ไม่มีพอร์ต SPI แต่ไมโครคอนโทรลเลอร์ทำงานเร็วมากๆ ขา MISO MOSI และขา SCK สามารถสร้างจากขา I/O ได้

2.12.1 การทำงานเมื่อตอนรีเซต

เมื่อชิพ VS1011B ทำการรีเซต เมื่อเวลาผ่านไปประมาณ 4096 แมทชีนไซเคิล DREQ ควรจะเป็น "0" ถ้าหากไม่เป็น "0" แล้วหลังจากเวลาที่ผ่านไปประมาณ 6000 แมทชีนไซเคิล DREQ ควรจะเป็น "1" ถ้าขา DREQ ไม่เปลี่ยนแปลงตามนี้แล้วการทำงานของโปรแกรมจะไม่ถูกต้อง
วิธีการใช้ชิพ VS1011B สำหรับการต่อให้แบบประหยัดขา
วิธีการเลือกการส่งข้อมูล

จะมีการเลือกว่าให้ส่งข้อมูลสู่สาย SDI หรือส่งข้อมูลเข้าสาย SCI (ข้อมูลที่ควบคุมชิพ VS1011B)

ในการต่อแบบประหยัดขานี้ ขา GIO1 เป็นเสมือนขาที่เลือกเส้นทางการส่งข้อมูล ถ้า GIO1 เป็น "1" xCS จะไม่ทำงาน (เพราะแอกทีพที่ "0") และขา SCK ที่ต้อผ่านแอนเกต เข้าสู่ขา DCLK จะมีสัญญาณนาฬิกาออกมาทำให้ข้อมูลถูกส่งเข้าสู่สาย SDI แต่เมื่อขา GIO เป็นขา "0" xCS จะทำงานทำให้ไม่มีสัญญาณนาฬิกา ที่ขา DCLK เพราะต่อผ่านแอนเกต ดังนั้นข้อมูลจะเข้าสู่สาย SCI

2.12.2 สายส่งข้อมูลสู่สายสัญญาณ SCI

ถ้าเราต้องการส่งข้อมูลที่ทำให้เอาต์พุทออกมาเป็น -2dB ที่ช่องสัญญาณซ้าย และ -3.5 dB ที่ช่องสัญญาณขวา ดังนั้นเพื่อให้เป็นดังข้างต้นเราต้องส่งข้อมูล 0407H ไปรีจิสเตอร์ (VOL register)

- ให้ส่วน SCI ทำงานโดยให้ขา GIO1 เป็น "0"
- ส่งข้อมูล 4 ไบต์ 02H , 0BH , 04H , 07H ผ่านสาย SCI
- เมื่อส่งข้อมูลเสร็จสิ้นแล้วเซตให้ขา GIO1 เป็น "1"

2.12.3 การรับข้อมูลผ่านสาย SCI

ในกรณี สมมติเราอ่านข้อมูลผ่านรีจิสเตอร์เสียง (VOL register)

- ให้ส่วน SCI ทำงานโดยเซตให้ขา GIO1 เป็น "0"
- เขียนข้อมูล 2 ไบต์ (03H , 0BH) สู่ส่วน SCI
- อ่านข้อมูลเข้าสู่ส่วนรีจิสเตอร์ MISO ทีละ 8 บิต โดยจะเป็น MSB LSB ตามลำดับ
- ข้อมูลที่ได้จะเป็น 16 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.12.4 การส่งข้อมูลส่วน SDI (ข้อมูลเอ็มเป็ก)

เราสามารถส่งข้อมูล 32 ไบต์ หรือ เล็กกว่าข้อมูลเอ็มเป็กสู่ VS1011B ได้

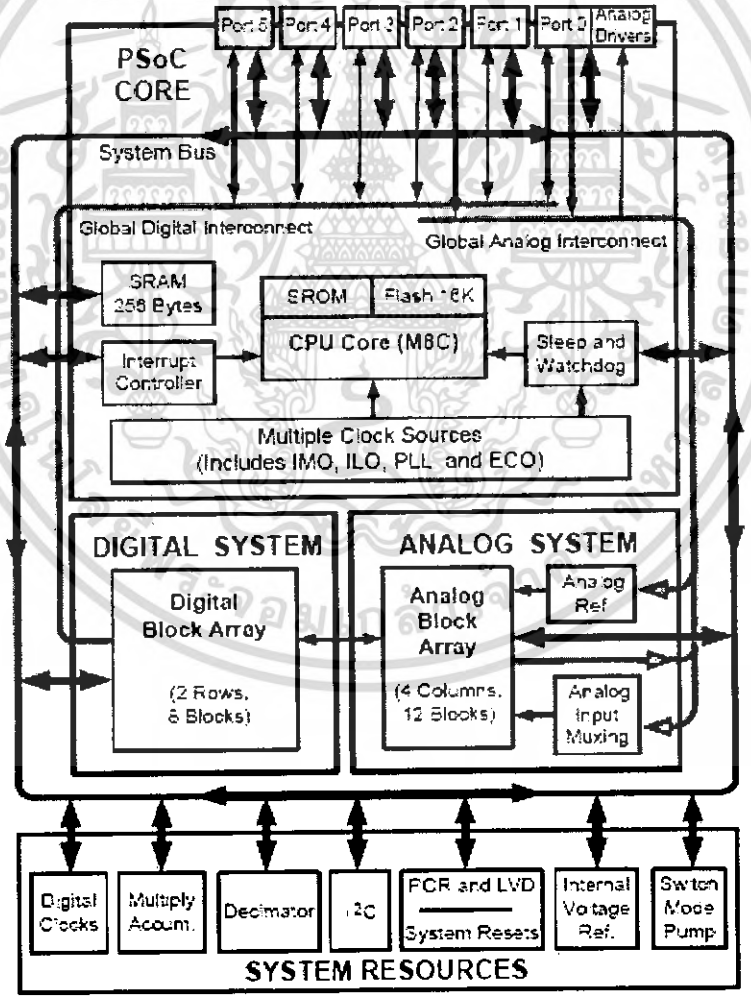
- จนกว่า DERQ เป็น "1"
- ให้ขา DCLK มีสัญญาณนาฬิกาโดยเซตให้ขา GIO เป็น "1"
- ในการส่งแต่ละไบต์ให้ทำตามขั้นตอนดังนี้
    1. ให้ขา BSYNC เป็น "1"
    2. ทำให้ SPI ทำงาน
    3. รอจนกว่าจะรู้ว่าข้อมูลบิตแรกถูกส่งออกไปแล้วแต่บิตสุดท้ายยังไม่ส่งออกไป
- เซตให้ขา BSYNC เป็น "0"
- รอจนกว่าไมโครคอลโทรลเลอร์ส่งข้อมูลผ่าน SPI จบจน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
โครงสร้างสถาปัตยกรรม
ไมโครคอลโทรลเลอร์ PSOC

3.1 โครงสร้างหลักของ Psoc MCU

ในไมโครคอลโทรลเลอร์ PSoC โครงสร้างภายในจะประกอบไปด้วยส่วนต่างๆมากมาย โดยแบ่งออกเป็น 4 ส่วนหลัก คือ PSoC Core ระบบดิจิตอล Digital System และระบบของอนาลอก Analog System และ System Resources โดยแต่ละส่วนจะเชื่อมต่อกันด้วยระบบบัส ดังรูป 3.1



รูป 3.1แสดงบล็อกไดอะแกรมของ PSoC

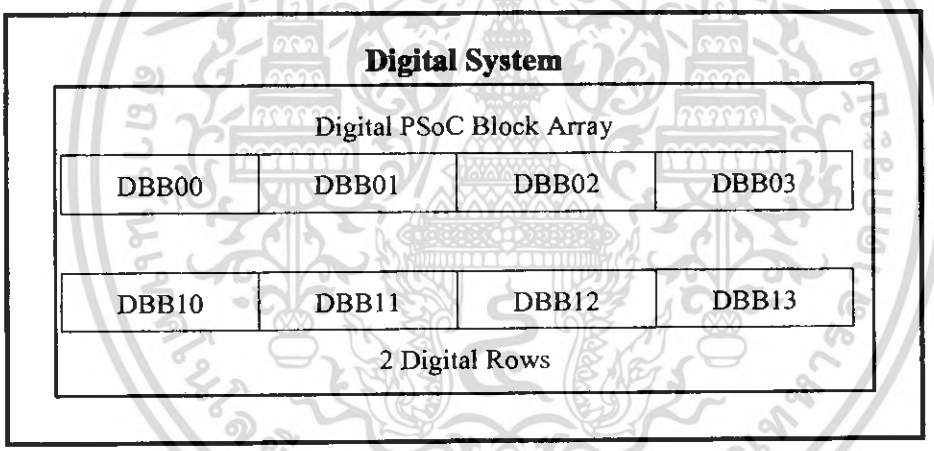
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.1 PSoC Core

เป็นส่วนหลักของการประมวลผล ซึ่งจะดูแลส่วนต่างๆ เช่น การประมวลผลคำสั่ง , การจัดการเก็บข้อมูลในหน่วยความจำ SRAM ,ควบคุมการอินเตอร์รัพท์, Sleep, Watchdog times และ การเลือกแหล่งสัญญาณนาฬิกา (Clock sources) เป็นต้น ซึ่งเราจะเรียกว่าส่วนที่จัดการต่างๆ เหล่านี้ว่า M8C ซึ่งเป็นสถาปัตยกรรมไมโครโปรเซสเซอร์ 8 บิต แบบ Harvard นอกจากนี้ภายใน PSoc Core ยังมีหน่วยความจำ SROM และ Flash memory สำหรับใช้เก็บคำสั่งอีกด้วย

3.1.2 Digital System

เป็นระบบของดิจิตอลบล็อกที่วางอยู่ที่รูปแบบของบล็อกอาเรย์ (array block) ในส่วนของ Digital System การวางบล็อกอาเรย์จะวางแถวละ 4 บล็อก จะมีอาเรย์บล็อกจำนวน 2 แถวรวมแล้วจะมีทั้งหมด 8 ดิจิตอลบล็อก ดังรูปที่ 3.2 ส่วน PSoC MCU เบอร์อื่นๆจะแตกต่างกันออกไป โดยบล็อกเหล่านี้จะรองรับการใช้งานในส่วนของโมดูลดิจิตอลต่างๆ



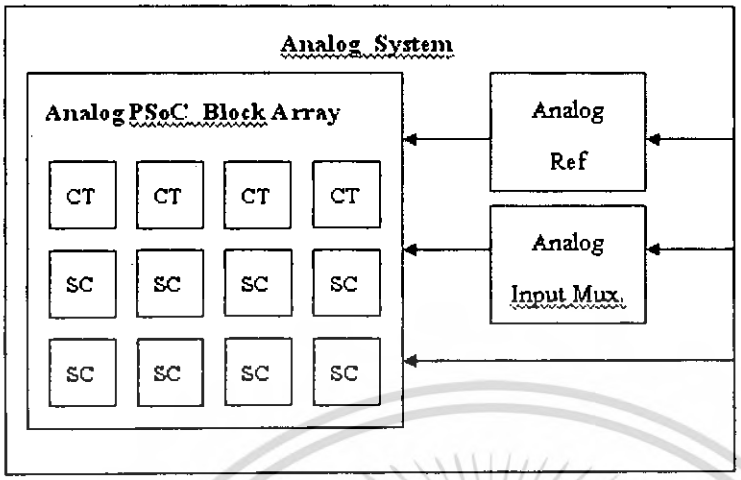
รูปที่ 3.2 แสดงโครงสร้างบล็อกของ Digital System

3.1.3 Analog System

ระบบของ Analog System จะประกอบไปด้วยส่วนต่างๆ คือ อนาลอกบล็อก(Analog block),แรงดันอ้างอิงอนาลอก(Analog Ref) และวงจรสัญญาณอินพุทอนาลอก(Analog Input Muxing) ดังรูปที่ 3.3

โดยอนาลอกบล็อกจะมีอยู่ 2 ประเภทด้วยกัน คือ CT (Continuous Time) และSC (Switched Capaciter) ซึ่งอนาลอกบล็อกจะถูกจัดเรียงเป็น 4 คอลัมน์ ในหนึ่ง คอลัมน์ก็จะประกอบด้วย CT หนึ่งบล็อก และ SC อีกสองบล็อก ซึ่งบล็อกเหล่านี้จะมีไว้สำหรับรองรับการทำงานของโมดูลลงในบล็อกต่างๆนั้นก็จะขึ้นอยู่กับความเหมาะสมของแต่ละโมดูล

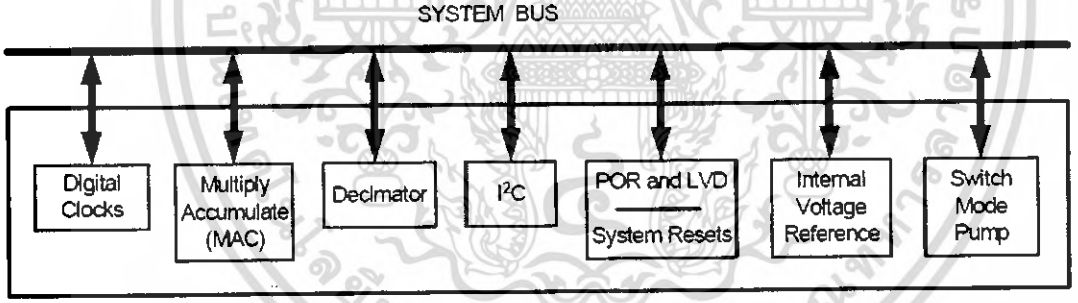
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 โครงสร้าง Analog System

3.1.4 System Resources

System Resources คือ ฮาร์ดแวร์ต่างๆของระบบ เช่น Digital clock , Multiply accumulate (MAC),Decimator,I2C,System resets , Internal voltage, Switch mode pump (SMP)เป็นต้น ดังรูปที่ 3.4

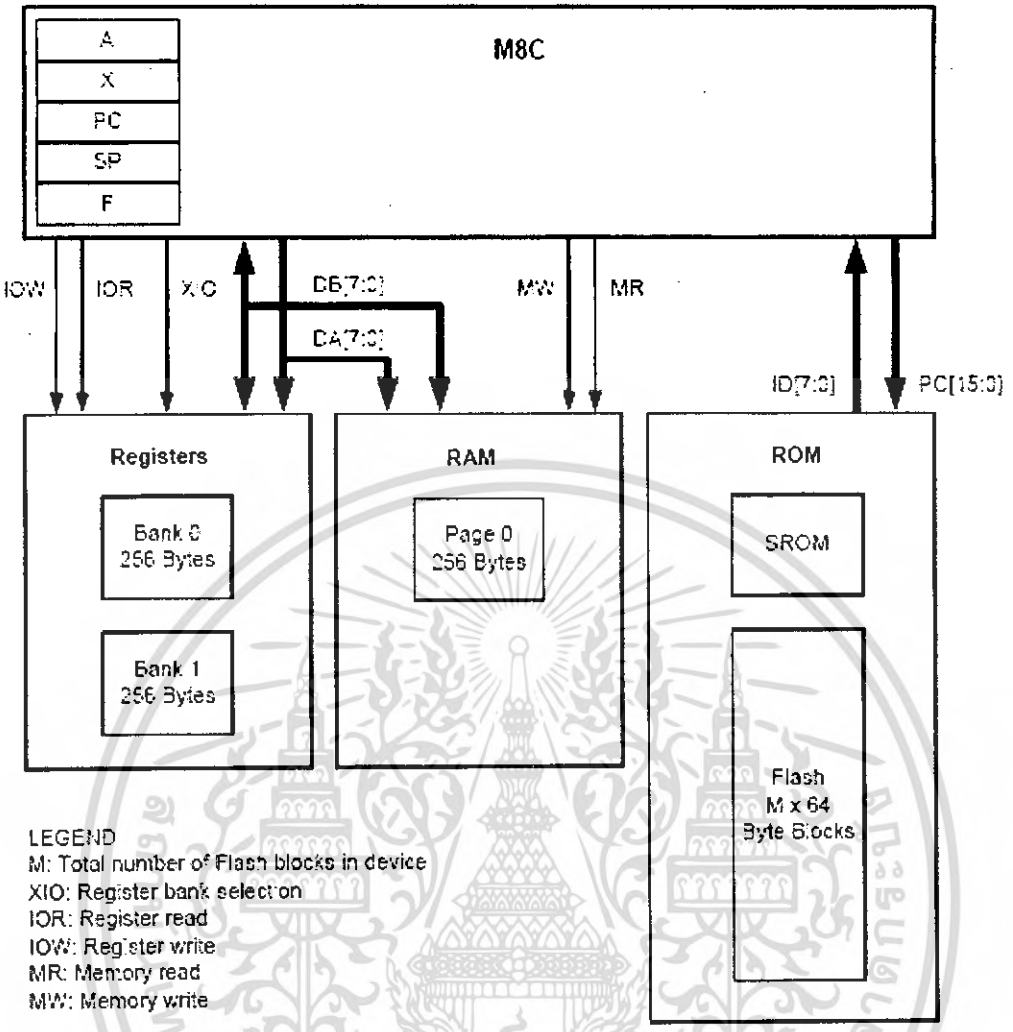


รูปที่ 3.4 ส่วนต่างๆ System Resources

3.2 หน่วยความจำ (Memory)

หน่วยความจำเป็นองค์ประกอบหนึ่งที่มีความจำเป็นต่อการทำงานของไมโครโปรเซสเซอร์ หรือ ไมโครคอลโทรลเลอร์ โดยจะมีการแบ่งหน่วยความจำออกเป็นประเภทต่างๆ ตามคุณสมบัติ และ การใช้งาน ซึ่งในส่วนของสถาปัตยกรรม M8C ก็จะแบ่งหน่วยความจำเป็น 3 ส่วน คือ ROM, RAM ,Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 แสดงลักษณะการเชื่อมต่อส่วนต่างๆของหน่วยความจำในสถาปัตยกรรม M8C

3.2.1 Rom จะมีการเข้าถึงโดยผ่านทาง Address Bus และ Data Bus โดยจากรูปภายใน Rom จะประกอบไปด้วย SRom (Supervisory Rom) และหน่วยความจำ Flash โดยหน่วยความจำ Flash จะแยกออกเป็นบล็อกๆ ละ 64 ไบต์ โดยผู้ใช้งานไม่จำเป็นต้องมาดูแลเรื่องของเขตแดนรอยต่อของหน่วยความจำในแต่ละบล็อก เพราะ M8C จะมีการจัดการให้เราโดยอัตโนมัติ รีจิสเตอร์ที่ทำหน้าที่เก็บตำแน่งของกาประมวลผล ก็คือ รีจิสเตอร์ PC จะมีการเพิ่มค่าขึ้นทุกๆ ไซเคิลคำสั่ง โดยจะเป็นรีจิสเตอร์ขนาด 16 บิต ซึ่งแบ่งออกเป็นสองส่วนคือ 8 บิตบน (PCH) และ 8 บิตล่าง(PCL) จะมีการเพิ่มค่าขึ้นหนึ่งค่า เมื่อ PC 8 บิตล่างนับถึงค่าที่ 256 (ยกเว้นคำสั่งที่กระโดด)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.1 Flash Program Memory Map

Address	description
0x000	Reset Vector
0x0004	Supply Monitor Interrupt Vector
0x0008	DBA 00 PSoC Block Interrupt Vector
0x000C	DBA 01 PSoC Block Interrupt Vector
0x0010	DBA 02 PSoC Block Interrupt Vector
0x0014	DBA 03 PSoC Block Interrupt Vector
0x0018	DBA 04 PSoC Block Interrupt Vector
0x001C	DBA 05 PSoC Block Interrupt Vector
0x0020	DBA 06 PSoC Block Interrupt Vector
0x0024	DBA 07 PSoC Block Interrupt Vector
0x0028	Analog Column 0 Interrupt Vector
0x002C	Analog Column 1 Interrupt Vector
0x0030	Analog Column 2 Interrupt Vector
0x0034	Analog Column 3 Interrupt Vector
0x0038	GPIO Interrupt Vector
0x003C	Sleep Timer Interrupt Vector
0x0040	On-Chip User Program Memory Status Here
…	***
…	***
…	***
0x3FFF	16K Flash Maximum Depending on Version

3.2.2 Register รีจิสเตอร์เป็นหน่วยความจำที่มีความสำคัญมากต่อการทำงานของระบบ ซึ่งการเขียนโปรแกรมจำเป็นต้องอาศัยรีจิสเตอร์เหล่านี้ไว้เก็บ หรือ พักข้อมูล รวมถึงการทำงานบางอย่างก็ต้องอาศัยรีจิสเตอร์ โดย M8C จะจัดพื้นที่รีจิสเตอร์เป็นแบบ แบงก์ (Bank) ซึ่ง M8C จะมีรีจิสเตอร์อยู่ 2 แบงก์ ด้วยกัน คือ พื้นที่แบงก์ของ User Space และ Configuration Space มีขนาดแบงก์ละ 256 ไบต์ เนื่องจากรีจิสเตอร์มีการจัดหน่วยความจำเป็นแบงก์แยกกันอยู่ดังนั้นในการเข้าถึงข้อมูลแต่ละ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนจึงต้องมีการเลือกแบงก์ข้อมูลที่เราจะเข้าถึง ซึ่งสามารถทำการเลือกแบงก์ได้โดยการกำหนดค่าให้บิต XIO ด้วยการเซต หรือเคลียร์

สถาปัตยกรรม M8C จะมีรีจิสเตอร์พิเศษภายใน 5 ตัวที่ใช้เกี่ยวกับการประมวลผลโปรแกรม ดังต่อไปนี้

- Accumulator(A) เป็นรีจิสเตอร์หลักที่ทำงานในส่วนของการกระทำทางคณิตศาสตร์และลอจิก
- Index (X) เป็นรีจิสเตอร์ชี้ทางอ้อม สำหรับการเข้าถึงข้อมูลแบบทางอ้อม
- Program Counter(PC) เป็นรีจิสเตอร์ที่เก็บตำแหน่งคำสั่ง CPU จะทำการตีความ หรือประมวลผล
- Stack Pointer(SP) เป็นรีจิสเตอร์ที่เก็บตำแหน่งของหน่วยความจำ Stack
- Flags(F) เป็นรีจิสเตอร์สถานะ

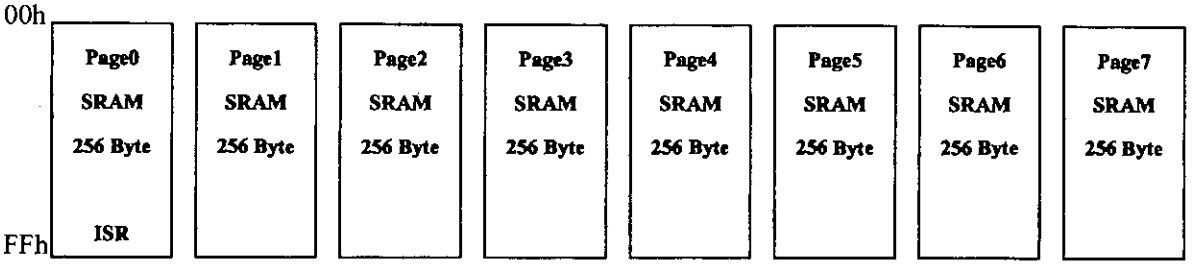
  รีจิสเตอร์ทั้งหมดจะเป็น 8 บิต ยกเว้นรีจิสเตอร์ PC ซึ่งจะมีขนาดเป็น 16 บิต เมื่อเกิดการรีเซตของระบบรีจิสเตอร์ A,X,PC SP จะมีสถานะเป็น 0x00 ส่วนรีจิสเตอร์ 0x02 ซึ่งจะเห็นได้ว่า Zero Flag มีการเซตเป็น 1

3.2.3 RAM (Random Access Memory) หน่วยความจำ RAM เป็นหน่วยความจำที่เก็บข้อมูลชั่วคราว ซึ่งข้อมูลไม่สามารถคงอยู่ได้หากไม่มีไฟเลี้ยงป้อนให้ โดยหน่วยความจำ RAM ของ PSoC MCU จะมีโครงสร้างแบบ Page ซึ่งจะมีขนาด Page ละ 256 ไบต์ ซึ่งสำหรับ PSoC ในตระกูลอื่นๆก็จะมีขนาดที่ต่างกัน ตามตารางที่ 3.2

ตารางที่ 3.2 ขนาดของหน่วยความจำ RAM ของ PSoC

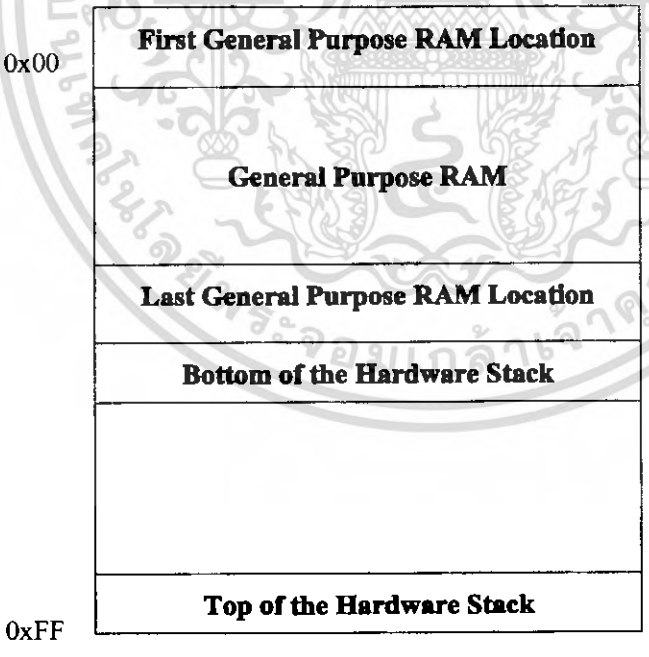
PSoC Device	**ขนาดของ RAM**	
CY8C29x66	2KB	8 Page
CY8C27x66	2KB	8 Page
CY8C27x43	256 Bytes	1 Page
CY8C24x23	256 Bytes	1 Page
CY8C22x13	256 Bytes	1 Page

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.6 แสดงลักษณะของหน่วยความจำ

หน่วยความจำส่วนหนึ่งภายนอกของ RAM จะถูกใช้งานเป็นหน่วยความจำสแต็ก (Stack) จากรูปที่ 3.7 จะเห็นได้ว่าตำแหน่งแรกของหน่วยความจำสแต็ก จะเริ่มต้นที่ตำแหน่งต่อจากตำแหน่งสุดท้ายของพื้นที่ General Purpose RAM และจะไปสิ้นสุดที่ตำแหน่งสุดท้ายคือ 0xFF แต่ถ้ามีการเพิ่มตำแหน่งของสแต็กอีก จาก 0xFF ตำแหน่งต่อไปจะเป็น 0x00 ซึ่งเหตุการณ์นี้เรียกว่า Stack Overflow โดยไม่ควรจะเกิดขึ้นเนื่องจาก อาจจะทำให้ตำแหน่งของสแต็กไปทับกับตำแหน่งของข้อมูลที่เรากำลังใช้งานอยู่ซึ่งอาจจะทำให้เกิดความผิดพลาดได้



รูปที่ 3.7 แสดงโครงสร้างหน่วยความจำ

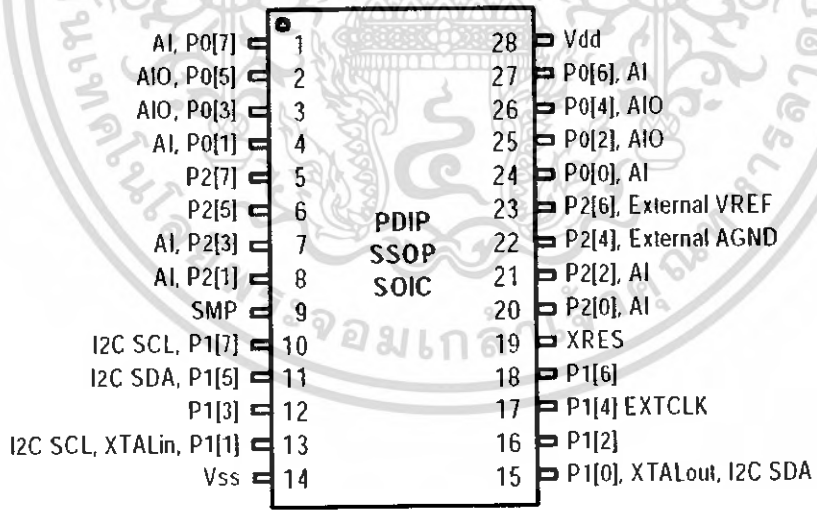
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 ขาสัญญาณอินพุต/เอาต์พุต (GPIO: General Purpose IO)

ขาสัญญาณ I/O (Input/Output) ซึ่งจะมีสัญญาณของ PSoC MCU นั้นสามารถทำงานได้ทั้งในส่วนของขาสัญญาณดิจิตอล และขาสัญญาณอนาลอก ซึ่งในการใช้งานจะผ่านทางรีจิสเตอร์ต่างๆ ดังตารางที่ 3.3

ตารางที่ 3.3 รีจิสเตอร์ที่เกี่ยวข้องกับการใช้งาน I/O (GPIO Register)

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Access
0,xxh	PRTxDR	Data Register								RW:00
0,xxh	PETxIE	Bit Interrupt Enable								RW:00
0,xxh	PETxGS	Global Select								RW:00
0,xxh	PETxDM2	Drive Mode2								RW:FF
1,xxh	PETxDM0	Drive Mode0								RW:00
1,xxh	PETxDMI	Drive Mode1								RW:FF
1,xxh	PETxIC0	Interrupt Control 0								RW:00
1,xxh	PETxIC1	Interrupt Control 1								RW:00



รูปที่ 3.8 แสดงตัวถังและการวางตำแหน่งขาสัญญาณต่างๆของไอซี

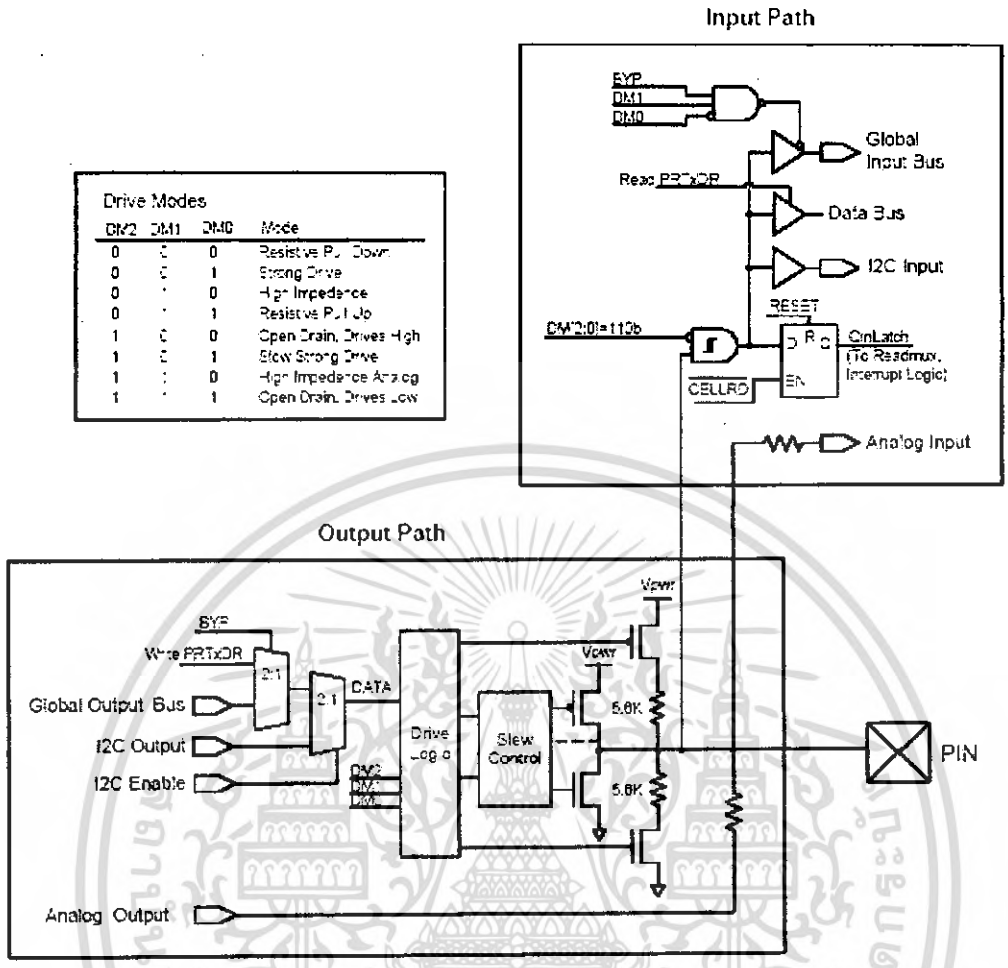
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขาสัญญาณอินพุต/เอาท์พุต หรือขาสัญญาณ GPIO ของไมโครคอลโทรลเลอร์ PSoc แต่ละขาสัญญาณนอกจากการทำงานในโหมด I/O ปกติแล้ว บางขาสัญญาณยังสามารถทำงานทำงานในหน้าที่อื่นๆ ได้อีกด้วย

ตารางที่ 3.4 แสดงขาสัญญาณต่างๆ

ชื่อขาสัญญาณ	คำอธิบาย	Input/ Output
SMP	Switch Mode Pump	Power
Vdd	Supply Voltage	Power
Vss	Ground	Power
XRES	External Reset(Active High)	Input
P0[0]-P0[1]	Port0[0],0[1], Analog Input	Input/ Output
P0[2]-P0[5]	Port0[2],0[3], 0[4],0[5], Analog Input/ Output	Input/ Output
P0[6]-P0[7]	Port0[6],0[7], Analog Input	Input/ Output
P1[0]	Port1[0],XTALOut / SDATA / I2C SDA	Input/ Output
P1[1]	Port1[1],XTALIn / SCLK / I2C SCL	Input/ Output
P1[2]	Port1[2]	Input/ Output
P1[3]	Port1[3]	Input/ Output
P1[4]	Port1[4],EXTCLK	Input/ Output
P1[5]	Port1[5],12C SCL	Input/ Output
P1[6]	Port1[6]	Input/ Output
P1[7]	Port1[7],12C SCL	Input/ Output
P2[0]- P2[3]	Port 2[0],2[1], 2[2],2[3],Non-Multiplexed Analog Input (Switched Capasitor)	Input/ Output
P2[4]	Port 2[4],External AGND	Input/ Output
P2[5]	Port 2[5]	Input/ Output
P2[6]	Port 2[6],External VREF	Input/ Output
P2[7]	Port 2[7]	Input/ Output
P3[0]- P3[7]	Port 3[0], 3[1], 3[2], 3[3], 3[4], 3[5], 3[6], 3[7]	Input/ Output
P4[0]- P4[7]	Port 4[0], 4[1], 4[2], 4[3], 4[4], 4[5], 4[6], 4[7]	Input/ Output
P5[0]- P5[3]	Port 5[0], 5[1], 5[2], 5[3], 5[4], 5[5], 5[6], 5[7]	Input/ Output

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.9 บล็อกไดอะแกรมของขาสัญญาณ GPIO

จากรูปที่ 3.9 จะเห็นได้ว่าโครงสร้างขาสัญญาณ GPIO มีความซับซ้อนมาก ทั้งนี้เพราะ MCU ได้ออกแบบโครงสร้างขาสัญญาณ ให้สามารถทำงานได้หลากหลายคุณสมบัติ โดยผู้ใช้งานสามารถกำหนดความต้องการในการใช้งานได้ โดยเราสามารถกำหนดคุณสมบัติการทำงานของขาสัญญาณในโหมดต่างๆ ได้ดังนี้

- Pull Down (Resistive Pull Down) โหมดนี้ขาสัญญาณ I/O จะถูกต่อตัวต้านทานลงกราวด์
- Strong (Strong Drive) โหมดนี้เหมาะสำหรับใช้งานเอาท์พุตดิจิตอล
- High Z (High Impedance)โหมดนี้ขาสัญญาณ I/O จะมีความต้านทานสูงเหมาะสำหรับใช้งานเป็นอินพุต
- Pull Up (Resistive Pull Up) โหมดนี้ขาสัญญาณ I/Oจะถูกต่อตัวต้านทานไป Vcc
- Open Drain High สถานะของสัญญาณ I/O เป็นแบบ Open Drain High

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Strong Slow (Slow Strong Drive)สถานะของสัญญาณ I/O เป็น Strong Slow
- High Z Analog Reset state เป็นสถานะความต้านทานสูงแบบอนาลอก ซึ่งจะเป็นค่าสภาวะเริ่มต้น (Default) หลังจากเกิดรีเซต (Reset state)

ตารางที่ 3.5 การกำหนด Drive Modes ของ GPIO โดยผ่านรีจิสเตอร์

DM2	DM1	DM0	Drive Mode	Data = 0	Data = 1
0	0	0	Resistive Pull Down	Resistive	Strong
0	0	1	Drive	Strong	Strong
0	1	0	High Impedance	Hi-Z	Hi-Z
0	1	1	Resistive Pull Up	Strong	Resistive
1	0	0	High	Hi-Z	Strong (Slow)
1	0	1	Slow Strong Drive	Strong (Slow)	Strong (Slow)
1	I	0	High Impedance Analog	Hi-Z	Hi-Z
1	1	I	Open Drain , Drain Low	Strong (Slow)	Hi-Z

การกำหนดคุณสมบัติของขาสัญญาณเหล่านี้ สามารถทำได้สองวิธี คือ การกำหนดที่ค่ารีจิสเตอร์ (PRTxDMx) ซึ่งวิธีนี้จะเป็นการเขียนโปรแกรมเข้าไปเซตค่ารีจิสเตอร์ ดังนั้นวิธีนี้ทำให้สามารถเปลี่ยนแปลงคุณสมบัติ Drive Mode ของขาสัญญาณได้ตลอดเวลาขณะที่ CPU กำลังประมวลผลอยู่ และอีกวิธี คือ การกำหนดโดยใช้โปรแกรม PSoc Designer Device Editor Configuration ซึ่งจะเรียกว่าการกำหนด Configuration I/O pins ซึ่งช่วยลดความยุ่งยากในการเขียนโปรแกรมลงได้อย่างดี

ขาสัญญาณ I/O ของ PSoc MCU จะประกอบไปด้วย บัฟเฟอร์อินพุต และวงจรขับด้านเอาท์พุตโดยสัญญาณ I/O เหล่านี้จะจัดไว้เป็นพอร์ต ซึ่งปกติ 1 พอร์ต จะมีทั้งหมด 8 บิต แต่จะมีบางกรณีที่พอร์ตนั้นๆ มีขาสัญญาณไม่ถึง 8 บิต ขาสัญญาณ I/O ต่างๆนี้สามารถทำงานในลักษณะต่างๆ ดังนี้

- Digital IO: เป็นขาสัญญาณดิจิตอล อินพุต/เอาท์พุต สามารถควบคุมการทำงานได้โดยผ่านค่าให้กับรีจิสเตอร์ PRTxDR
- Global IO : เป็นขาสัญญาณดิจิตอล อินพุต/เอาท์พุต ที่เชื่อมโยงระหว่าง Digital PSoC block
- Analog IO : เป็นขาสัญญาณ อินพุต/เอาท์พุต ที่เชื่อมโยงระหว่าง Analog PSoC block

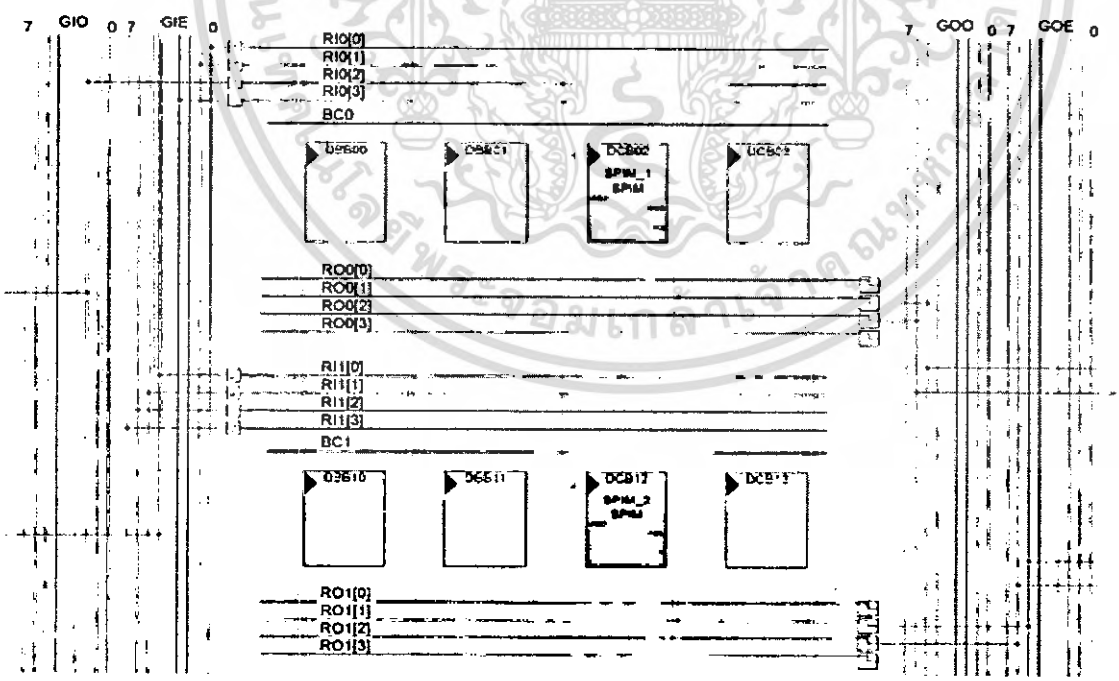
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.1 Digital IO

เป็นโหมดการทำงานหนึ่งของขาสัญญาณ อินพุต/เอาท์พุต ที่หน่วยประมวลผล M8C สามารถติดต่อเข้าถึงขาสัญญาณพอร์ตได้โดยตรง โดยไม่ต้องอาศัย PSoC Block ไม่ว่าจะเป็นการอ่านค่า หรือ การเขียนค่าไปที่พอร์ตซึ่ง M8C จะติดต่อผ่านรีจิสเตอร์ PRTxDR ซึ่งสถานะของรีจิสเตอร์นี้จะส่งผลโดยตรงกับขาสัญญาณ I/O Pin ของ PSoC MCU เช่น ในการเขียนข้อมูลลอกจิกไปยังพอร์ตผ่านรีจิสเตอร์ PRTxDR ขอข้อมูลดังกล่าวจะถูก เปลี่ยนให้เป็นสถานะทางไฟฟ้าไปปรากฏที่ขาสัญญาณต่างๆ ส่วนในกรณีการอ่านข้อมูลจากพอร์ตหน่วยประมวลผลสามารถอ่านค่าสถานะทางไฟฟ้าจากขาสัญญาณ I/O ได้จากรีจิสเตอร์ PRTxDR ด้วยเช่นกัน ซึ่งสถานะทางไฟฟ้าจะถูกเปลี่ยนเป็นลอจิกแล้วนำกลับมาเก็บที่รีจิสเตอร์

3.3.2 Global IO

Global IO จะถูกใช้เป็นสัญญาณเชื่อมต่อ (interconnect) ระหว่าง PSoC Block กับสัญญาณภายนอกโดยจะเชื่อมโยงผ่าน Global In และ Global Out ซึ่งสามารถเชื่อมโยงสัญญาณจาก GPIO เข้ามายัง Global In หรือ Global Out แล้วนำไปเชื่อมต่อกับสัญญาณใน PSoC Block ต่างๆได้ตามต้องการ ซึ่งในการออกแบบสามารถทำได้โดยใช้โปรแกรม PSoC Desingner ดังรูปที่ 3.11



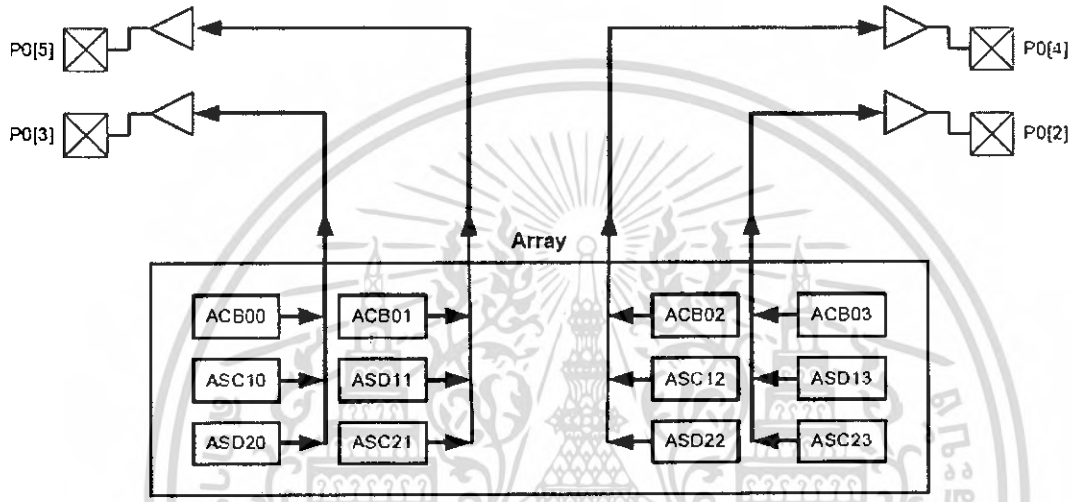
รูปที่ 3.10 แสดงหน้าต่าง interconnect ของ PSoC Desingner

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 Analog IO

เป็นขาสัญญาณ อินพุต/เอาท์พุต อนาลอก ซึ่งจะเชื่อมต่อสัญญาณภายนอกกับสัญญาณภายในอนาลอกบล็อกผ่านขาสัญญาณ AOUT โดยจะต่อตัวต้านทานภายในประมาณ 300 โอห์ม สำหรับโหมดของอนาลอกชนิด GPIO จะกำหนดเป็น High Impedance Analog Drive mode (Hi-z)

ขาสัญญาณ GPIO ที่ทำหน้าที่ขาสัญญาณ Analog Input/Output ซึ่งสามารถเชื่อมต่อกับ Analog Block ได้ จะมีทั้งหมด 4 ขาสัญญาณ คือ ขาสัญญาณ P0[2], P0[3], P0[4], P0[5] ดังรูปที่ 3.11



รูปที่ 3.11 แสดงขาสัญญาณ Analog I/O

3.4 ระบบสัญญาณนาฬิกา

ระบบสัญญาณนาฬิกาของ PSoc MCU เป็นระบบที่มีความหลากหลาย ค่อนข้างซับซ้อน และมีรายละเอียดการทำงานต่างๆ มากมาย โดยสามารถแบ่งแหล่งกำเนิดสัญญาณนาฬิกาเป็นส่วนๆ ได้ดังนี้

- Internal Main Oscillator (IMO)
- Internal Low Speed Oscillator (ILO)
- 32 kHz Crystal Oscillator (ECO)
- Phase Locked Loop (PLL)

3.4.1 สัญญาณนาฬิกาภายใน Internal Main Oscillator (IMO)

เป็นแหล่งกำเนิดสัญญาณนาฬิกาหลัก ซึ่งเป็นแหล่งกำเนิดสัญญาณภายใน PSoc MCU โดยสามารถผลิตสัญญาณนาฬิกาได้ ขนาด 24 MHz เรียกว่า SYSCLK และ ยังมีวงจรคูณความถี่เป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สองเท่า (Clock Double) ซึ่งจะทำได้ความถี่เป็น 48 MHz เรียกว่า SYSCLK2X โดยสามารถเลือกใช้งานได้ นอกจากนี้ยังสามารถปรับแต่งความถี่ได้อีกด้วย โดยรีจิสเตอร์ IMO_TR ที่ใช้สำหรับค่าความถี่ของสัญญาณนาฬิกาเพื่อให้มีความเที่ยงตรงมากยิ่งขึ้น

ในการผลิตสัญญาณความถี่นี้ได้สัญญาณอินพุตมาจากแหล่งกำเนิดสัญญาณออสซิลเลเตอร์ขนาดความถี่ 32 kHz จากภายในตัว PSoc MCU ซึ่งค่าความคลาดเคลื่อนของสัญญาณนาฬิกาทั้งจาก 24 MHz และ 48 MHz จะอยู่ที่ +/- 2.5 % ซึ่งมีผลมาจากสภาวะอุณหภูมิและ แรงดัน (3.3 V+/-0.3 และ 5 V +/-5%) โดยการใช้งานออสซิลเลเตอร์แบบภายในนี้มีข้อดีคือ ไม่จำเป็นต้องใช้อุปกรณ์ภายนอกใดๆต่อร่วมได้เลย แต่อาจจะมีปัญหาเรื่องของความคลาดเคลื่อนอยู่บ้าง ดังนั้นหากต้องการความเที่ยงตรงของสัญญาณความถี่สูงๆ ควรจะใช้แหล่งสัญญาณนาฬิกา คริสตอลออสซิลเลเตอร์จากภายนอก (ECO) ซึ่งโมดูลของ IMO จะสนับสนุนการใช้งานสัญญาณนาฬิกาจากภายนอกด้วย โดยอาศัยหลักการจากวงจร Phase Locked Loop (PLL) ซึ่งอินพุตของวงจร PLL นี้จะได้สัญญาณ 32 kHz ของคริสตอลออสซิลเลเตอร์จากภายนอก (ECO)

ความเร็วในการทำงานของ CPU นั้นสามารถเลือกได้หลายระดับโดยนำเอาสัญญาณนาฬิกาของระบบ 24 MHz SYSCLK มาหารด้วยอัตราส่วนต่างๆ ซึ่งกำหนดได้ด้วยรีจิสเตอร์ OSC_CRO ในบิต 2:0 ตามตารางที่ 3.6 หรือสามารถกำหนดได้จากหน้าต่าง Global Resources ของซอฟแวร์ PSoC Designs

ตารางที่ 3.6 การกำหนดความเร็วการทำงานของ CPU กำหนดโดยรีจิสเตอร์ OSC_CRO บิต 2:0

OSC_CRO[2:0]	Internal Main Oscillator	External Clock
000b	3 MHz	EXTCLK/8
001b	6 MHz	EXTCLK/4
010b	12 MHz	EXTCLK/2
011b	24 MHz	EXTCLK/1
100b	1.5 MHz	EXTCLK/16
101b	750 kHz	EXTCLK/32
110b	187.5 kHz	EXTCLK/128
111b	93.7 kHz	EXTCLK/256

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 สัญญาณนาฬิกาแบบความเร็วต่ำภายใน Internal Low Speed Oscillator (ILO)

เป็น สัญญาณนาฬิกาแบบความเร็วต่ำ(32 kHz) ที่อยู่ภายใน PSoc MCU โดยปกติแล้วจะให้กำเนิดสัญญาณให้กับการทำงานในส่วนของการประหยัดพลังงาน (Sleep) และ Watchdog นอกจากนี้ก็ยังสามารถนำเอาไปใช้เป็นสัญญาณนาฬิกาให้กับ Digital PSoC Block ได้อีกด้วย

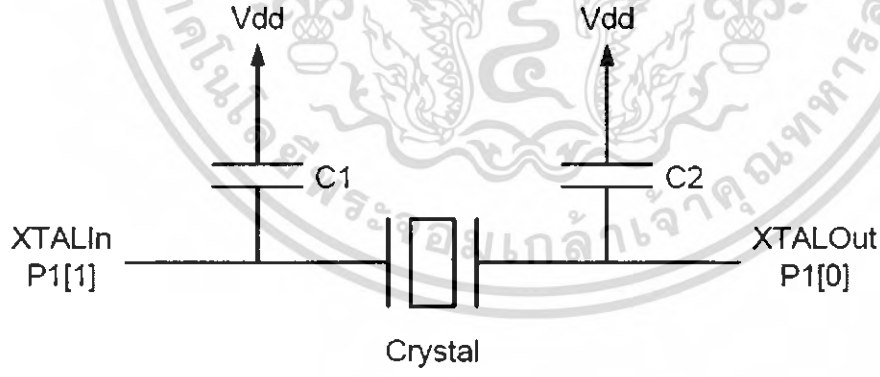
สามารถกำหนดระดับการใช้พลังงานของสัญญาณออสซิลเลเตอร์ ได้ 3 โหมดด้วยกัน โดยการกำหนดค่าให้กับรีจิสเตอร์ ILO_TR ในบิต Bias Trim[5:4] ดังนี้คือ

- โหมดปกติ (normal power mode)เหมาะสำหรับการใช้งานทั่วไปที่ต้องการความถูกต้องของสัญญาณสูง
- โหมดพลังงานต่ำ (low power mode) เหมาะสำหรับการทำงานในโหมดประหยัดพลังงาน Off ไม่ใช้งาน

นอกจากการกำหนดระดับพลังงานของออสซิลเลเตอร์แล้ว ยังสามารถทำการปรับค่าความถี่ของสัญญาณ(Freq Trim) ได้อีกด้วยโดยการปรับรีจิสเตอร์ ILO_TR ในบิต Trim[3:0]

3.4.3 สัญญาณนาฬิกา 32kHz จากคริสตอลภายนอก 32 kHz Crystal Oscillator (ECO)

เป็นแหล่งกำเนิดสัญญาณนาฬิกาที่ได้จากภายนอกโดยวงจรจากคริสตอลออสซิลเลเตอร์ 32 kHz โดยวงจรการต่อคริสตอลเข้ากับขาสัญญาณ P1[1](XTAL In) และP1[0] (XTAL Out) นอกจากนี้จะต้องต่อตัวเก็บประจุร่วมด้วยอีก 2 ตัว ดังรูปที่ 3.12



รูปที่ 3.12 วงจรคริสตอลออสซิลเลเตอร์ภายนอก

สัญญาณนาฬิกาจาก ECO นี้สามารถใช้เป็นฐานเวลาให้กับแหล่งสัญญาณนาฬิกาหลัก (IMO) ได้โดยการเลือกใช้งานในโหมด PLL ซึ่งจะได้สัญญาณนาฬิกาหลัก (IMO) ที่มีความเที่ยงตรงสูงขนาดความถี่ 24MHz ซึ่งจะเป็นสัญญาณนาฬิกาของระบบ นอกจากนี้การต่อสัญญาณนาฬิกาแบบ ECO ยังสนับสนุนการใช้งานคริสตอลความถี่ 32.768 kHz ซึ่งเป็นความถี่ที่ใช้กับ 1C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นาฬิกา (Real Time Clock (RTC)) เหมาะกับการนำไปประยุกต์การใช้งานเป็นนาฬิกา เพราะจะมีค่าเวลาที่เที่ยงตรงเช่นเดียวกับนาฬิกาทั่วๆไป ซึ่งการใช้งานออสซิลเลเตอร์ในโหมด ECO นี้บิต 7 ของรีจิสเตอร์ OSC_CRO จะต้องเซตเป็น "1" ด้วย

3.4.4 การใช้งานออสซิลเลเตอร์แบบECO

โดยการใช้งานสัญญาณนาฬิกาแบบ ECO นั้นแบ่งออกเป็น 2 ลักษณะ คือ

- ใช้เป็นแหล่งสัญญาณนาฬิกา CKL32K หรือใช้เป็นฐานเวลาให้กับ Sleep timer
- ใช้เป็นฐานเวลาให้กับสัญญาณนาฬิกาหลัก (IMO) ซึ่งกรณีนี้จะต้อมีการใช้งาน PLL โดยจะได้สัญญาณนาฬิกาของระบบ SYSCLK ขนาด 24 MHz ที่มีความเที่ยงตรงสูงมาก

จากการทำงานในสองลักษณะดังกล่าวจะมีการใช้ค่าตัวเก็บประจุที่แตกต่างกัน โดยกรณีของการใช้งาน ECO แบบไม่ใช้ PLL จะใช้ค่าตัวเก็บประจุ C1 ,C2 ที่มีค่าเท่ากัน ดังตารางที่ 3.7

ตารางที่ 3.7 ค่าตัวเก็บประจุที่ใช้กับวงจรคริสตอลภายนอก ECOแบบไม่ใช้ PLL

Package	C1(P1[1])	C2(P1[0])
8 PDIP	22 pf	22 pf
20 PDIP	22 pf	22 pf
20 SOIC	22 pf	22 pf
20 SSOP	22 pf	22 pf
28 PDIP	22 pf	22 pf
28 SOIC	22 pf	22 pf
28 SSOP	22 pf	22 pf
44 TQFP	22 pf	22 pf
48 PDIP	22 pf	22 pf
48 SSOP	22 pf	22 pf

ส่วนค่าตัวเก็บประจุของวงจรคริสตอลกรณีทีมีการใช้งาน PLL จะต้องใช้ตัวเก็บประจุ C1 ,C2 มีค่าแตกต่างกันดังตารางที่ 3.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.8ค่าตัวเก็บประจุที่ใช้กับวงจรคริสตอลภายนอก ECOแบบใช้ PLL

Package	C1(P1[1])	C2(P1[0])
8 PDIP	12pf	100pf
20 PDIP	12pf	100pf
20 SOIC	12pf	100pf
20 SSOP	12pf	100pf
28 PDIP	12pf	100pf
28 SOIC	12pf	100pf
28 SSOP	12pf	100pf
44 TQFP	12pf	100pf
48 PDIP	9pf	100pf
48 SSOP	12pf	100pf

3.4.5 การกำหนดการใช้งานสัญญาณนาฬิกาจาก ECO

การกำหนดการใช้งานสัญญาณนาฬิกาจาก ECO นั้นสามารถทำได้โดยการกำหนดค่าต่างๆ ในหน้าต่าง Device Editor ของซอฟแวร์ PSoC Designer โดยกำหนด 32K_Select เป็น External และหากต้องการให้สัญญาณนาฬิกาจาก ECO นี้เป็นฐานเวลาให้กับสัญญาณนาฬิกาหลัก IMO ก็สามารถทำได้โดยเลือก PLL_Mode เป็น Ext Lock นอกจากนี้จะต้องกำหนดให้สัญญาณ P1[0] เป็น XtalOut เป็น XtaIn อีกด้วย

เมื่อเรากำหนดการทำงานโดยการเลือกสัญญาณนาฬิกาจากภายนอก ECO เมื่อ CPU เริ่มต้นทำงาน หรือ หลังจากการเกิดรีเซต การทำงานของระบบจะต้องอาศัยสัญญาณนาฬิกาจากภายในไปก่อนระยะหนึ่งเพื่อรอให้ สัญญาณจากวงจรคริสตอลภายนอก ECO และ PLL มีเสถียรภาพพอ ก่อนถึงจะสวิตช์การทำงานมาใช้สัญญาณนาฬิกาจากคริสตอลออสซิลเลเตอร์ภายนอก แต่ถ้าหากพบว่าสัญญาณนาฬิกาจากภายนอกไม่ทำงาน มันจะทำให้สวิตช์สัญญาณนาฬิกากลับไปทำงานสัญญาณนาฬิกาภายใน

การใช้สัญญาณจากคริสตอลภายนอก จะให้ความถูกต้องเที่ยงตรงกว่าการใช้ การใช้สัญญาณนาฬิกาจากภายใน ดังนั้นการใช้สัญญาณจาก ECO จึงเหมาะกับการทำงานที่ต้องการใช้ความแม่นยำของสัญญาณนาฬิกาสูง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.6 Phase Locked Loop (PLL)

Phase Locked Loop (PLL) เป็นฟังก์ชันการทำงานในการสร้างสัญญาณนาฬิกาของระบบ โดยอาศัยอินพุทสัญญาณจากวงจรคริสตอลออสซิลเลเตอร์ภายนอก ซึ่งจะให้ความเที่ยงตรงกว่าสัญญาณนาฬิกาภายใน โดยเอาท์พุตสัญญาณของ PLL จะมีค่าความถี่เท่ากับ 23.986 MHz เมื่อใช้ความถี่คริสตอลภายนอกเท่ากับ 32.768 kHz โดยสัญญาณความถี่เอาท์พุตของ PLL จะถูกใช้เป็นสัญญาณนาฬิกาของ (SYSCLK) โดยการเลือกใช้งาน PLL สามารถ ได้โดยใช้ซอฟแวร์ PSoC Designer

- เลือกสัญญาณนาฬิกา 32_KSelect เป็น External
- เลือก PLL_Mode เป็น Ext Lock
- กำหนดให้ขาสัญญาณ P1[0] เป็น XtalOut เป็น P1[1] XtalIn

3.5 การทำงานในโหมด Sleep และ Watchdog

Sleep Mode หรือ โหมดการประหยัดพลังงาน จุดประสงค์ของการทำงานในโหมดนี้ก็เพื่อที่จะประหยัดพลังงานของ CPU ให้มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งเราสามารถควบคุมการทำงานของ Sleep Mode ได้โดยการเขียนโปรแกรมคำสั่งควบคุม ซึ่งเมื่อเราสั่งให้ CPU เข้าสู้การทำงานในโหมด Sleep จะมีผลทำให้ CPU หยุดทำงาน และทำให้ แหล่งกำเนิดสัญญาณนาฬิกา 24/48 MHz

Watchdog Timer คือ ส่วนที่ทำหน้าที่ในการเฝ้าระวังการทำงานของ CPU โดยจุดประสงค์ในการทำงานของ Watchdog Timer ก็เพื่อป้องกันการทำงานที่ผิดพลาดของ CPU ในที่นี้หมายถึง CPU ไม่สามารถทำงานในสิ่งที่ควรจะเป็นได้ หรือ CPU เกิดค้าง ไม่สามารถทำงานต่อได้โดย Watchdog Timer จะทำการรีเซต CPU ในทันทีที่ค่าการนับของมันสิ้นสุดลง เพื่อให้ CPU กลับไปตั้งต้นทำงานใหม่ได้

3.5.1 Sleep Timer

Sleep Timer คือ ค่าเวลาของการทำงานใน Sleep Mode ซึ่งจะเป็นไทเมอร์ขนาด 15 bit แบบนับขึ้น ซึ่งจะนับสัญญาณจากแหล่งกำเนิดสัญญาณนาฬิกา 32 kHz โดยสามารถเลือกได้ว่าจะใช้ภายใน (ILO) หรือ จากภายนอก โดยสามารถเลือกช่วงเวลา Sleep Timer ได้ดังตารางที่ 3.9

ตารางที่ 3.9 คาบเวลาของ Sleep Timer และ Watchdog Timer

Sleep Interval	Sleep Period	Watchdog Period
(Default) 00	1.95 ms(512 Hz)	6 ms
01	15.6 ms(64 Hz)	47 ms
10	125 ms(8Hz)	375 ms
11	1 s(1Hz)	3 sec

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางที่ 3.9 จะเห็นได้ว่าคาบเวลา Watchdog Period จะมีค่าเป็น 3 เท่าของค่าเวลา Sleep Timer โดยการเลือกคาบเวลาดังกล่าว สามารถทำได้โดยการกำหนดค่าที่หน้าต่าง Global Resource ในโปรแกรม PSoC Desiner

การเปิดการทำงาน Sleep Timer สามารถทำได้โดยการเซตที่ [3] ของรีจิสเตอร์ CPU_SCR0 โดยตรง หรือ เซตค่าผ่านมาโครฟังก์ชัน M8C_Sleep ของ M8C ดังตารางที่ 3.10

ตารางที่ 3.10 ตัวอย่างฟังก์ชันการใช้งานของ M8C ที่เกี่ยวข้องกับ Sleep / Watchdog Timer

ฟังก์ชัน M8C	**ความหมาย**
M8C_EnableWatchDog	;Enable การทำงานของ Watchdog
M8C_ClearWDTAndSleep	; เคลียร์ค่าการนับของ Watchdog Timer และ Sleep Timer
M8C_ClearWDT	; เคลียร์ค่า Watchdog Timer
M8C_EnableGInt	; Enable Global Interrupt
M8C_DisableGInt	; Disable Global Interrupt
M8C_Sleep	; เข้าสู่ Sleep Mode

3.5.2 การออกจาก Sleep Mode

เมื่อเข้าสู่การทำงานในสภาวะ Sleep ระบบต่างๆจะทำงานด้วยพลังงานต่ำ และ ในการออกจากการทำงานในสภาวะ Sleep Mode จะเรียกว่า Wake Up โดยสามารถเกิดขึ้นจากการเกิดอินเตอร์รัพท์ และจากการรีเซตต่างๆ เช่น Sleep timer interrupt, Global Interrupt ,supply monitor Interrupt , timer clocked externally เป็นต้น

3.6 การรีเซตของระบบ (System Resets)

การรีเซตของระบบ จะมีผลทำให้ CPU กลับไปทำงานในสถานะเริ่มต้น และทำให้รีจิสเตอร์ต่างๆมีค่าสถานะ กลับไปตามค่าของ Default states โดยการเกิดรีเซตสามารถเกิดขึ้นได้จากสาเหตุต่างๆได้ดังนี้

3.6.1 Power-on Reset (POR) เป็นการรีเซตที่เกิดจาก แหล่งจ่ายแรงดันของ CPU มีระดับแรงดันต่ำกว่าที่กำหนด

3.6.2 External Reset (XRES) เป็นการรีเซตจากขาสัญญาณ XRES ของPSoc MCU โดยจะเกิดการรีเซตเมื่อขาสัญญาณ XRES ได้รับสภาวะลอจิกสูง “1”

3.6.3 Watchdog Reset (WDR) เป็นการรีเซตที่เกิดจากการทำงานของ Watchdog Timer

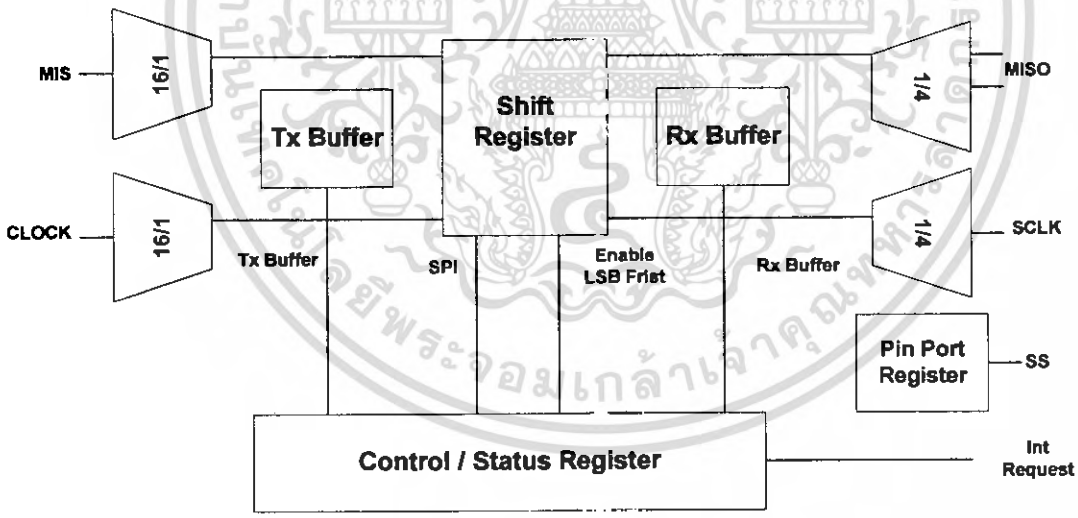
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7 การใช้งาน SPIM (Serial Peripheral Interconnect Master)

โมดูล SPIM คือ โมดูลการสื่อสารอนุกรมที่มีโปรโตรคอลการสื่อสารแบบ SPI โดยจะเป็นการส่งแบบเข้าจังหวะ 8 บิตข้อมูล ฟูลดูเพล็กซ์ (Full-duplex) โมดูลนี้จะทำงานเป็นมาสเตอร์ซึ่งสามารถจะนำไปต่อกับอุปกรณ์ SPI ที่เป็นสลาฟมากกว่าหนึ่งตัว ขึ้นอยู่กะกับการใช้งาน

3.7.1 คุณสมบัติของโมดูล

1. สนับสนุนการเชื่อมต่อในโหมดมาสเตอร์ ด้วยโปรโตรคอลการสื่อสารแบบ SPI
2. สนับสนุนโหมดสัญญาณนาฬิกาของ SPI คือ โหมด 1,2,3
3. สามารถกำหนดแหล่งสัญญาณนาฬิกาได้
4. สามารถเลือกการเชื่อมต่อเอาต์พุตของสัญญาณได้ MOSI และ SCLK wfh
5. สามารถสร้างสัญญาณอินเตอร์รัพท์ได้
6. ความเร็วการรับการส่งข้อมูลสูงสุด 12 MHz



รูป 3.13 แสดงบล็อกไดอะแกรมภายในโมดูล SPIM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยภายในโมดูล SPIM จะประกอบไปด้วย Tx Buffer ,Rx Buffer,Control Shitf register การใช้งานโมดูลจะเรียกใช้งานผ่านฟังก์ชัน API โดยค่าเริ่มต้นของโมดูล SPIM จะถูกเซตมีการส่งข้อมูลแบบส่งบิตนัยสำคัญต่ำสุดก่อน (LSB First) และ จะสนับสนุนการทำงานของสัญญาณในโหมดต่างๆ ตั้งแต่โหมด 0,1,2 และ 3 ซึ่งควรจะมีการเซตโหมดสัญญาณนาฬิการะหว่างอุปกรณ์ SPI Master และอุปกรณ์ SPI Slave ให้มีนาฬิกาเป็นโหมดเดียวกัน โดยโหมดการทำงานของ SPI จะเป็นดังตารางที่ 3.11

ตารางที่ 3.11 โหมดการทำงานของ SPI

SPI Mode			
Mode	**SCLK Edge Performing Data Latch**	**Clock Polarrity**	**Notes**
0	Leading	Non-inverting	Leading edge latches data. Data changes on trailing edge of clock.
1	Leading	Inverting	
2	Trailing	Non-inverting	Trailing edge latches data. Data changes on leading edge .
3	Trailing	Inverting	

3.7.2 ขาสัญญาณของการเชื่อมต่อ

3 SCLK คือ ขาสัญญาณนาฬิกาที่ให้จังหวะในการรับส่งข้อมูลของ SPI

- MOSI (Master-Out-Slave-In-Data) คือ ขาสัญญาณของตัวอุปกรณ์มาสเตอร์ ซึ่งจะไปเป็นขาสัญญาณอินพุตของอุปกรณ์สลาฟ

3.7.3 การใช้งานฟังก์ชัน API ของโมดูล SPIM

การใช้งานโมดูลจะกระทำผ่านการเรียกใช้ฟังก์ชัน API ซึ่งการใช้งานฟังก์ชันนี้อาจมีผลทำให้ค่าของรีจิสเตอร์ A และ X มีการเปลี่ยนแปลงไปด้วยโดยจะมีฟังก์ชันต่างๆให้ใช้งานดังต่อไปนี้

SPIM_Start ฟังก์ชันการกำหนดโหมดการทำงานของ SPI พร้อมทั้งเปิดการทำงานของโมดูล โดยกำหนดโหมดการทำงานจะต้องผ่านทางรีจิสเตอร์ A ซึ่งจะมีโหมดต่างๆ ดามตารางที่ 3.12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตาราง 3.12 โหมดการทำงานต่างๆของ SPI

Symbolic Name	Value	ความหมาย
SPIM_MODE_0	0x00	การทำงานของ SPI ในโหมด 0
SPIM_MODE_1	0x02	การทำงานของ SPI ในโหมด 1
SPIM_MODE_2	0x04	การทำงานของ SPI ในโหมด 2
SPIM_MODE_3	0x06	การทำงานของ SPI ในโหมด 3
Symbolic Name	**Value**	**ความหมาย**
SPIM_LSB_FIRST	0x80	ส่งข้อมูลทางด้านบิดต่ำก่อน
SPIM_MSB_FIRST	0x00	ส่งข้อมูลทางด้านบิดสูงก่อน

SPIM_Stop ฟังก์ชันการปิดการทำงานของโมดูล SPIM ฟังก์ชันนี้จะไปเคลียร์ค่าในบิต Enable ของรีจิสเตอร์ Control ทำให้โมดูล SPIM หยุดการทำงาน

SPIM_Enable ฟังก์ชันการเปิดการอินเตอร์รัพท์ของ SPIM ตามเงื่อนไขของการอินเตอร์รัพท์

SPIM_DisableIne ฟังก์ชันปิดการอินเตอร์รัพท์ของ SPIM

SPIM_SendTxData การส่งข้อมูลแบบ SPI ไปยังอุปกรณ์สลาฟที่ต่อร่วมด้วย

SPIM_bReadRxData ฟังก์ชันอ่านรัยค่าไบต์ข้อมูลจากอุปกรณ์สลาฟ โดยควรทำการเช็คสถานะของบัฟเฟอร์ (RxBuffer) ก่อนว่าว่างหรือไม่ ก่อนที่จะทำการเรียกใช้ฟังก์ชันนี้

SPIM_bReadStatus เป็นคำสั่งอ่านสถานะจากรีจิสเตอร์ควบคุม ของโมดูล SPIM ซึ่งภายในรีจิสเตอร์ จะประกอบไปด้วยสถานการณ์ทำงานต่างๆ ของโมดูลโดยค่าสถานะต่างๆ จะรีเทิร์นค่าออกมาผ่านทางรีจิสเตอร์ A ซึ่งค่าต่างๆจะเป็นดังตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ชุดสำหรับเก็บข้อมูล

SPI DATA FLASH เป็นชุดสำหรับเก็บข้อมูล โดยจะ Interface Read/Write ข้อมูล แบบ Serial SPI โดย Module

จะมีคุณสมบัติดังนี้

1. ใช้กับ Power Supply 2.5V-3.6V เท่านั้น
2. ขาสัญญาณ Input SI , SCK , CS , RESET และ WP สามารถรองรับแรงดัน 5.0V ได้
3. ใช้การ Interface แบบ SPI ที่รองรับความถี่ Clock ได้สูงสุด 20 Mhz
4. สามารถเก็บข้อมูลได้ถึง 2 MB
5. มีขา สำหรับป้องกันการเขียนข้อมูล ทับลงในหน่วยความจำ
6. หน่วยความจำนี้ จะถูกแบ่งออกเป็น 4096 Page และในแต่ละ Page จะมีขนาด 528 Byte
7. สามารถลบข้อมูลเป็น Page(528 Byte) หรือเป็น Block(1Block= 8 Page)ได้
8. มี SRAM Data Buffer อยู่ภายใน 2 ชุด ซึ่งมีขนาด 528 Byte เพื่อทำหน้าที่รับข้อมูลไปพักไว้ ก่อนที่จะส่งไปยัง หน่วยความจำหลัก(พื้นที่ Flash) ซึ่งจะขึ้นอยู่กับ Command ที่ใช้ว่าจะให้ผ่านBuffer หรือจะเข้าถึง หน่วย ความจำหลักโดยตรง

4.1 หน้าที่ของขาสัญญาณ SPI DATA FLASH

4.1.1 CS (Chip select) : Data Flash จะถูกเลือกเมื่อขา CS นี้เป็น Low และถ้าขานี้เป็น High จะไม่มีการส่งข้อมูลออกไป ที่ขา SI ส่วนขา SO จะยังอยู่ในสถานะ High-impedance เมื่อมีการส่งสัญญาณ จาก High- to- Low ไปยังขา CS นั่นหมายถึงต้องการเริ่มต้นการทำงาน และถ้าส่งสัญญาณจาก Low-to-High ไปยังขา CS อีกครั้ง จะเป็นการจบการทำงาน

4.1.2. SCK (Serial Clock) : ขานี้จะทำหน้าที่เป็น Input เพื่อรับสัญญาณ Clock จากภายนอกเข้ามาควบคุมจังหวะการไหลเข้า ออกของข้อมูลจาก Data Flash

4.1.3. SI (Serial Input) : ขานี้จะทำหน้าที่เป็น Input และถูกใช้เพื่อเลื่อนข้อมูลเข้าไปยัง Chip หน่วยความจำ

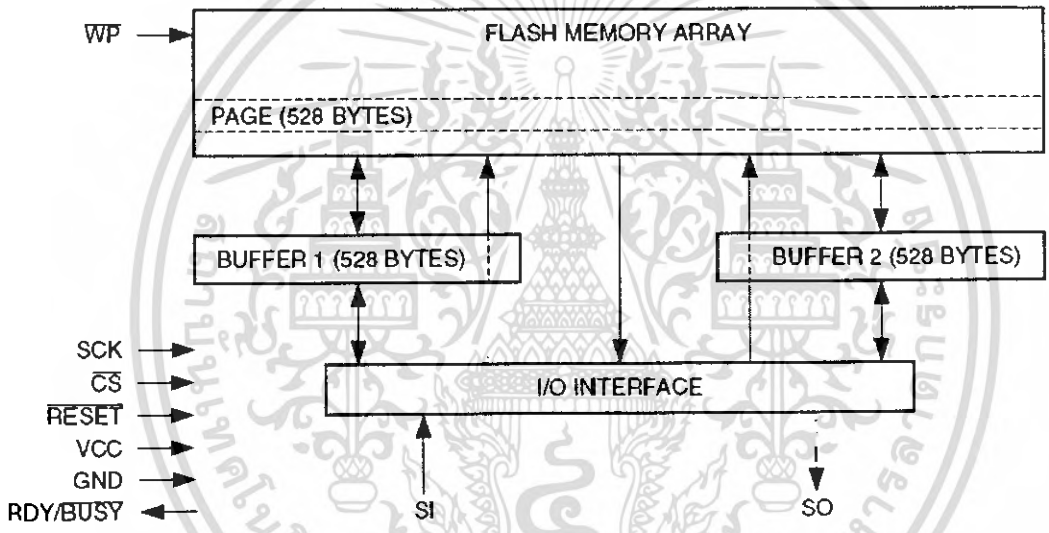
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.4. SO (Serial Output) : ขานี้จะทำหน้าที่เป็น Output ถูกใช้เพื่อเลื่อนข้อมูลออกจากตัว Chip หน่วยความจำ

4.1.5. WP (Write Protect) : ถ้าขานี้เป็น Low จะทำให้ในพื้นที่หน่วยความจำหลัก 256 Page แรก ไม่สามารถที่จะเขียนข้อมูล เข้าไปเก็บไว้ได้

4.1.6.RES (Reset) : เมื่อขานี้เป็น Low จะทำให้ Chip หน่วยความจำถูก Reset กระบวนการทำงานต่างๆก็ จะสิ้นสุด ลง และจะทำงานได้ใหม่เมื่อขานี้เป็น High

4.1.7. BUSY (Ready/Busy) : ขานี้จะทำหน้าที่เป็น Open drain Output ซึ่งจะให้สัญญาณเป็น Low เมื่อการทำงานภายในของ Chip หน่วยความจำยังไม่พร้อม ในสภาวะปกติขานี้จะต้องเป็น High



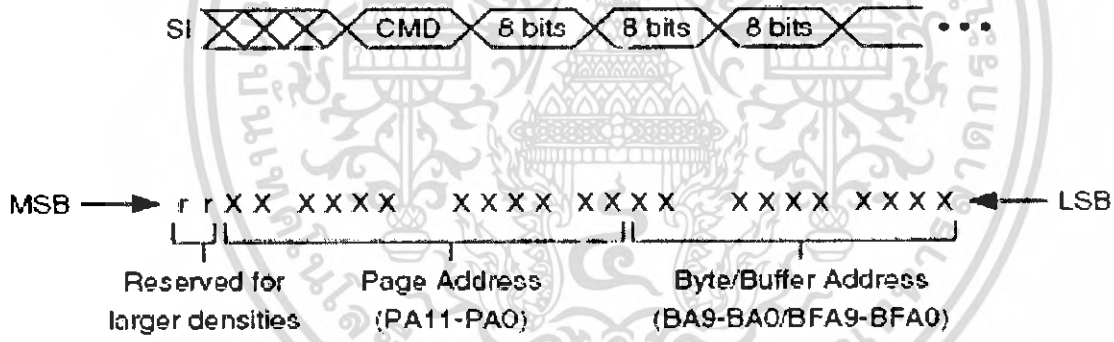
รูปที่ 4.1 บล็อกไดอะแกรม Memory Chip

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับพื้นที่หน่วยความจำหลักของ Chip หน่วยความจำ จะถูกแบ่งออกเป็น 3 ระดับ ซึ่งประกอบไปด้วย Sector(มี 32 Block/Sector) , Block(มี 8Page/Block) และ Page(มี528Byte/Page) การทำงานของโปรแกรม ในการ Read/Write data Flash นั้น จะปรากฏในลักษณะของ Page-by-Page คือ เวลาที่จะ Read/Write ข้อมูลแต่ละครั้งจะต้องทำครั้งละ 1 Page หรือ 528 Byte ส่วนในการลบข้อมูลนั้น สามารถลบได้ทั้งแบบ Block หรือแบบ Page

4.2 การทำงานของ Chip Memory Flash

การทำงานของอุปกรณ์นี้จะถูกควบคุมด้วยคำสั่ง ที่ส่งมาจาก MCU ผ่านทางการสื่อสารแบบ SPI โดย Opcode ของคำสั่งจะแสดงดังในตารางที่ 1-4 ด้านล่าง คำสั่งจะเป็น Opcode ขนาด 8 บิต จะถูก Start ที่ขอบขาลงของสัญญาณ CS ขณะที่ CS เป็น Low และมีสัญญาณ Clock เข้ามาก็จะทำให้เกิดการโหลด Opcode และตำแหน่ง Address ของ Buffer หรือ Address ของหน่วยความจำหลักออกไปยังขา SI โดยจะส่งบิต MSB ออกไปเป็นบิตแรก รูปแบบในการส่ง Command Read/Write จะแสดงดังรูปด้านล่าง



รูปที่4.2 แสดงรูปการส่ง Command แบบ Read/Write

ในการส่ง Command แต่ละครั้งผู้ใช้จะต้องเรียงลำดับการส่งให้เป็นไปตามรูปแบบด้านบนเสมอ โดย Byte แรกจะต้องส่ง Opcode ของ Command ออกไป จากนั้นก็ตามด้วย Address 3 Byte(24bit) ที่เหลือโดยแยกรายละเอียดให้เห็นในรูปด้านบน ซึ่งประกอบด้วย rr (2bit)2บิตนี้จะถูกสงวนไว้ใช้กับความจุของ Chip ที่มีขนาดมากขึ้น ดังนั้นจะต้องกำหนดให้เป็น '00' ส่วนอีก 12 บิต

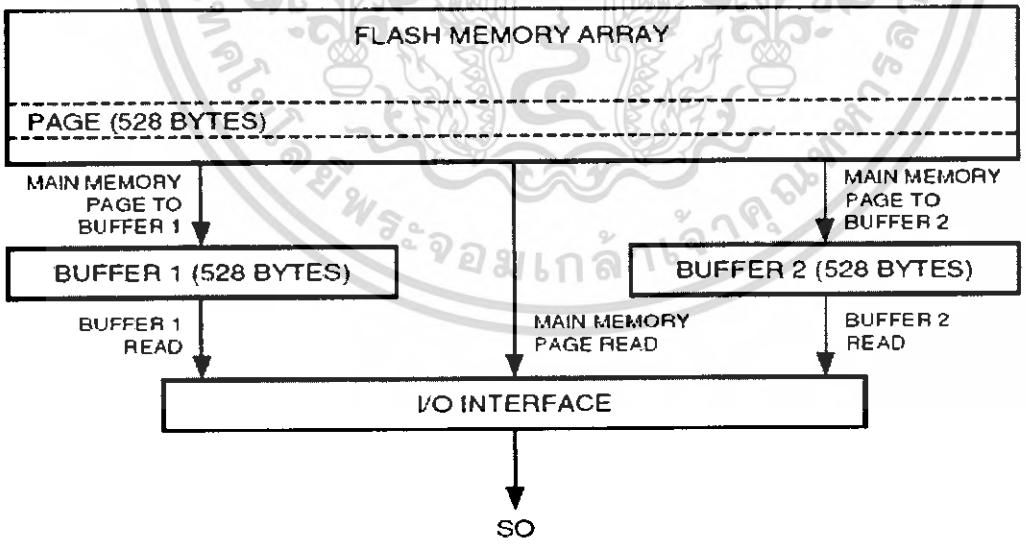
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อมา จะใช้กำหนดตำแหน่งของ Page Address(PA11-PA0)ซงจะสามารถอ้าง Address ได้ตั้งแต่ค่า 0-4095 และ 10 บิต สุดท้ายของ Command ที่จะต้องส่งออกไป คือ Byte/Buffer Address (BA9-BA0/BFA9-BFA0) ซึ่งก็คือ ตำแหน่ง Address ภายใน Page นั้นๆ หรือ ตำแหน่ง Address ใน Buffer1 หรือ Buffer2 ซึ่งจะสามารถอ้าง Address ได้ตั้งแต่ค่า 0-527

ในการใช้งานแต่ละคำสั่งนั้น บางคำสั่งจะอ้างเพียง Page Address หรือ อ้างเพียง Byte/ Buffer Address อย่างใดอย่างหนึ่งเท่านั้น ในส่วนที่ไม่ได้อ้างถึงนั้นผู้ใช้จะต้องใส่ค่า Don't Care (0 หรือ 1 ก็ได้) แทนในส่วน ที่ไม่ได้อ้างถึง เพื่อให้ Command ที่ส่งออกไปครบ 4 Byte ซึ่งสามารถดูรายละเอียดการส่ง Command แต่ละบิตได้ในตารางที่ 4

4.3 การอ่านและคำสั่งที่ใช้ในการอ่านข้อมูล

การอ่านข้อมูลจาก Chip นั้นจะแสดงให้เห็นดังบล็อกไดอะแกรมด้านล่าง ซึ่งจะสอดคล้องกับคำสั่งที่ใช้ในการอ่านดังตารางที่1 จากบล็อกไดอะแกรมจะเห็นว่าการอ่านข้อมูลนั้นมีการอ่านได้หลายแบบ ซึ่งในแต่ละแบบคำสั่งที่ใช้ในการอ่านก็จะแตกต่างกันไปผู้ใช้จึงต้องทำความเข้าใจในแต่ละคำสั่งเสียก่อนเพื่อจะเลือกใช้คำสั่งได้ถูกต้อง คำสั่งที่ใช้ในการอ่านนั้นอาทิเช่น อ่านข้อมูลจาก Flash ไปเก็บไว้ที่ Buffer 1 หรือ 2 ภายในตัว Chip เอง จากนั้นจึงใช้คำสั่งอ่านข้อมูลจาก Buffer 1 หรือ 2 เพื่อส่งข้อมูลออกมาที่ขา SO หรืออีกแบบหนึ่งคือ จะอ่านข้อมูลจาก Main Memory Flash ออกไปที่ขา SO โดยตรงก็ได้ ไม่ต้องผ่าน Buffer 1 หรือ 2 ภายใน Chip เป็นต้น



รูปที่ 4.3 แสดง บล็อกไดอะแกรม การอ่านข้อมูลจาก Chip

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางที่ 1 ด้านล่างจะเป็นชุดคำสั่งอ่านข้อมูลจาก Chip หน่วยความจำ ซึ่งจะมีอยู่ด้วยกัน 5 คำสั่ง โดยมี Opcode ให้เลือกใช้ 2 ค่า ซึ่งการที่จะเลือกใช้ค่าไหนนั้นจะขึ้นอยู่กับลักษณะของ Clock SPI ที่ผู้ใช้ส่งเข้ามาทางขา SCK ว่าส่งมาในลักษณะใด โดยลักษณะของ Clock นั้นจะมีด้วยกัน 4 แบบ ซึ่งดูได้จาก Timming ที่แสดงในคำสั่งที่ใช้อ่านนั้นๆ

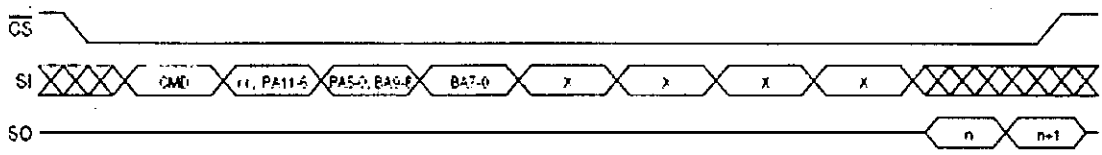
ตารางที่ 4.1 คำสั่งอ่าน (Read Command)

Command	SCK Mode	Opcode
Continuous Array Read	- Inactive Clock Polarity Low or High	68H
	- SPI Mode 0 or 1	E8H
Main Memory Page Read	- Inactive Clock Polarity Low or High	52H
	- SPI Mode 0 or 1	D2H
Buffer 1 Read	- Inactive Clock Polarity Low or High	54H
	- SPI Mode 0 or 1	D4H
Buffer 2 Read	- Inactive Clock Polarity Low or High	56H
	- SPI Mode 0 or 1	D6H
Status Register Read	- Inactive Clock Polarity Low or High	57H
	- SPI Mode 0 or 1	D7H

4.3.1 Continuous Array Read Command :

คำสั่งนี้จะเป็นการอ่านข้อมูลแบบต่อเนื่อง ซึ่งจะอ่านข้อมูลจาก Main Memory Flash ส่งออกมาที่ขา SO โดยไม่ผ่าน Buffer ภายใน เมื่ออ่านข้อมูลจนจบ Page(528Byte) ใดๆก็ตาม ในหน่วยความจำหลักแล้วอุปกรณ์ก็จะทำการอ่านข้อมูลต่อที่จุดเริ่มต้นของ Page ต่อไปและเมื่ออ่านข้อมูลมาถึงบิตสุดท้ายในหน่วยความจำหลัก อุปกรณ์ก็จะกลับไปอ่านต่อที่จุดเริ่มต้นของ Page แรก ของหน่วยความจำ โดย Opcode ของคำสั่งนี้คือ 68H หรือ E8H (ขึ้นอยู่กับลักษณะของ Clock) รูปแบบการส่ง Command จะแสดงดังรูปที่ 4.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



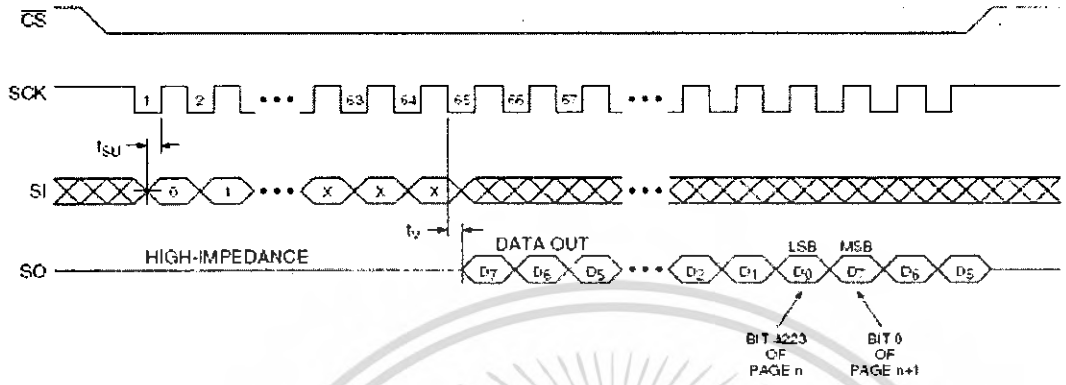
รูปที่ 4.4 แสดง Command Continuous Array Read

จากรูปหลังจากที่ส่ง Command 4 Byte แรกตามรูปแบบที่กำหนดไว้ออกไปแล้ว จะต้องส่งข้อมูล Don't Care (0 หรือ 1 ก็ได้) ตามออกไปอีก 4 Byte ในรูป ก็คือส่วนที่เป็น X จากนั้นข้อมูลที่อ่านถึงจะเริ่มถูกส่งออกมาทางขา SO ในขณะที่เริ่มต้นส่ง Opcode จนถึงกระบวนการอ่าน Data ขา CS จะต้องยังคงเป็น Low อยู่เสมอ และเมื่อมีการส่งสัญญาณจาก Low-to-High ไปยังขา CS ถึงจะเป็นการสิ้นสุดการอ่าน และขา SO ก็จะอยู่ในสภาวะ tri-state จะเห็นว่าในคำสั่งนี้จะมี Opcode 68H และ E8H ให้เลือก 2 ชุด ซึ่งจะเลือกใช้ Opcode ใดนั้นจะขึ้นอยู่กับรูปแบบของ Clock ที่ส่งมาให้ Chip ซึ่งจะมีอยู่ด้วยกัน 4 รูปแบบ โดยดูได้จาก Timing Diagram ที่แสดงรายละเอียดการทำงานในระดับบิต ตามรูปแบบ Clock ทั้ง 4 แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

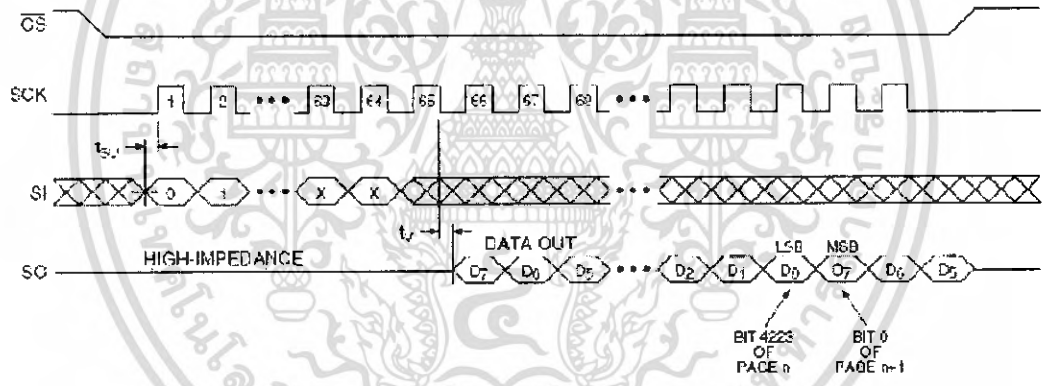
Detailed Bit-Level read Timing Inactive- Clock Polarity High

Continuous Array Read (Opcode :68H)



Detailed Bit-Level read Timing Inactive Clock -Polarity Low

Continuous Array Read (Opcode :68H)

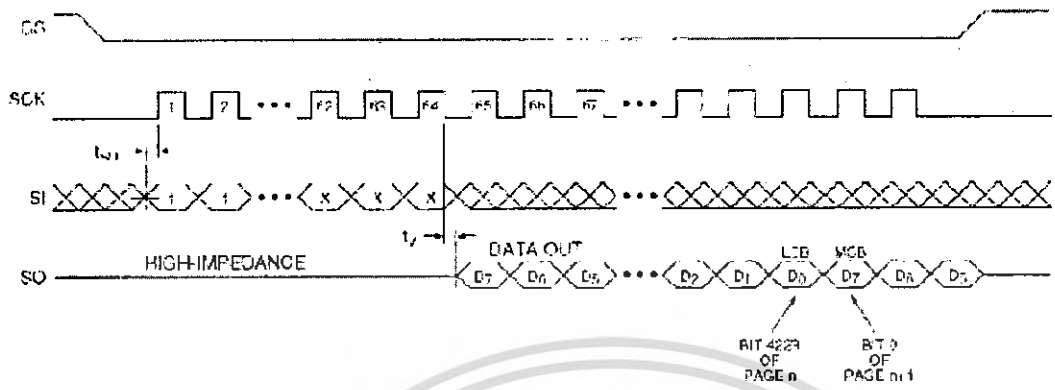


รูปที่ 4.5 แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode :68H (Inactive Clock Polarity High- Low)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

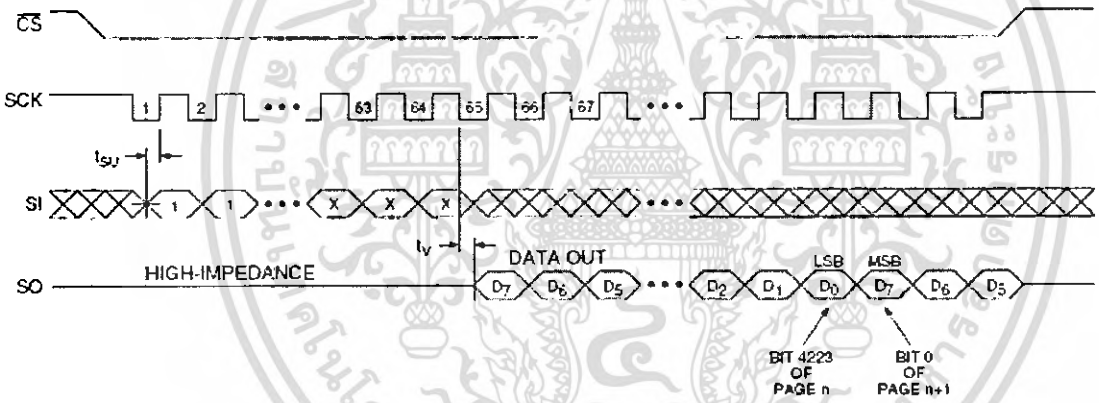
Detailed Bit-Level read Timing-SPI Mode 0

Continuous Array Read (Opcode :E8H)



Detailed Bit-Level read Timing-SPI Mode 3

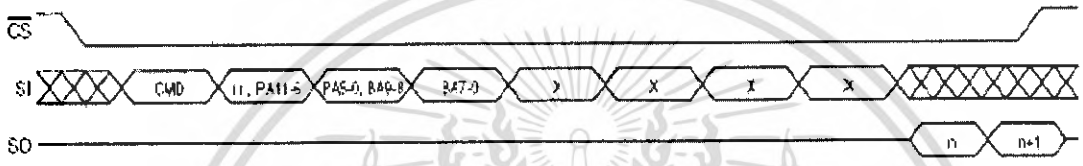
Continuous Array Read (Opcode :E8H)



รูปที่ 4.6 แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode :E8H (SPI Clock Mode 0 or mode 3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2 Main Memory Page Read Command : คำสั่งนี้จะให้ผู้ใช้อ่านข้อมูลโดยตรงจาก Page ใด Page หนึ่งใน 4096 Page ที่อยู่ใน Main Memory โดยไม่ผ่าน Buffer ภายใน ออกมายังขา SO เลย เริ่มต้นการอ่านนั้นจะต้องส่ง Opcode 52H หรือ D2H (ขึ้นอยู่กับรูปแบบของ Clock ที่ใช้งาน) ออกไปก่อน แล้วตามด้วย 24 บิตแอดเดรส ตามรูปแบบการส่ง Command ดังรูปที่8 สุดท้ายตามด้วยข้อมูล Don't Care (0 หรือ 1 ก็ได้) ตามออกไปอีก 4 Byte ในรูปก็คือส่วนที่เป็น X ขณะที่เริ่มต้นส่ง Opcode จนถึงกระบวนการอ่าน Data ขา CS จะต้องยังคงเป็น Low อยู่เสมอ และเมื่อมีการส่งสัญญาณจาก Low-to-High ไปยังขา CS ถึงจะเป็นการสิ้นสุดการอ่าน และขา SO ก็จะอยู่ในสภาวะ tri-state



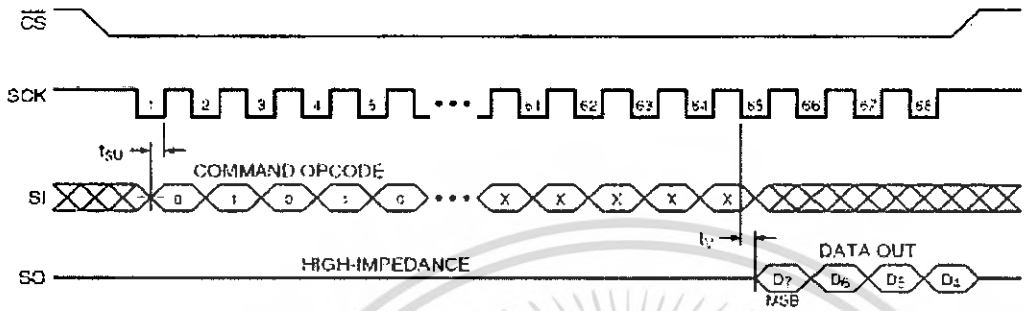
รูปที่ 4.7 แสดง การส่ง Command Main Memory Page Read

จะเห็นว่าในคำสั่งนี้จะมี Opcode ให้เลือก 2 ชุด คือ 52H และ D2H ซึ่งจะเลือกใช้ Opcode ใดนั้นจะขึ้นอยู่กับรูปแบบของ Clock ที่ส่งมาให้ Chip ซึ่งจะมีอยู่ด้วยกัน 4 รูปแบบ โดยดูได้จาก Timing Diagram ที่แสดงรายละเอียดการทำงานในระดับบิต ตามรูปแบบ Clock ทั้ง 4 แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

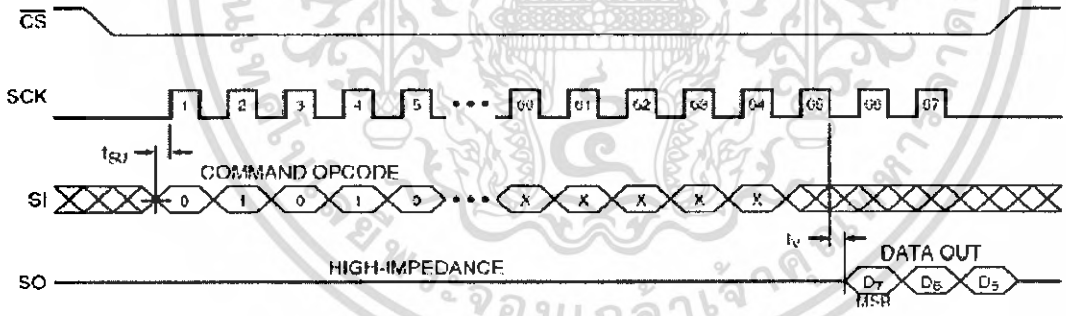
Detailed Bit-Level read Timing Inactive- Clock Polarity High

Main Memory Page Read (Opcode :52H)



Detailed Bit-Level read Timing- Inactive Clock Polarity Low

Main Memory Page Read (Opcode :52H)

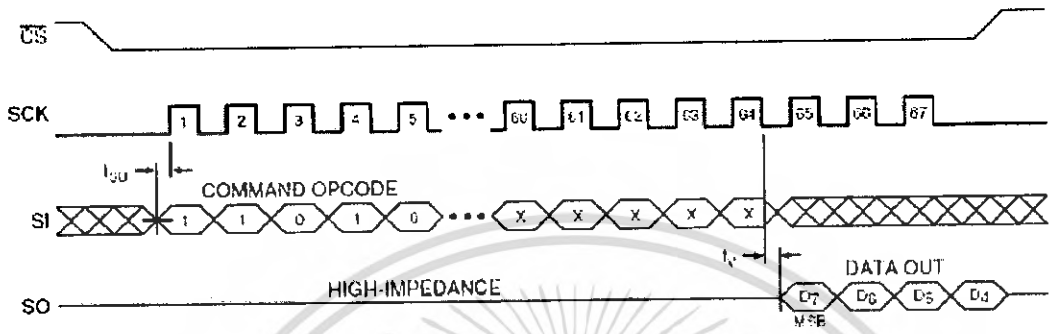


รูปที่ 4.8 แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode :52H (Inactive Clock Polarity High- Low)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

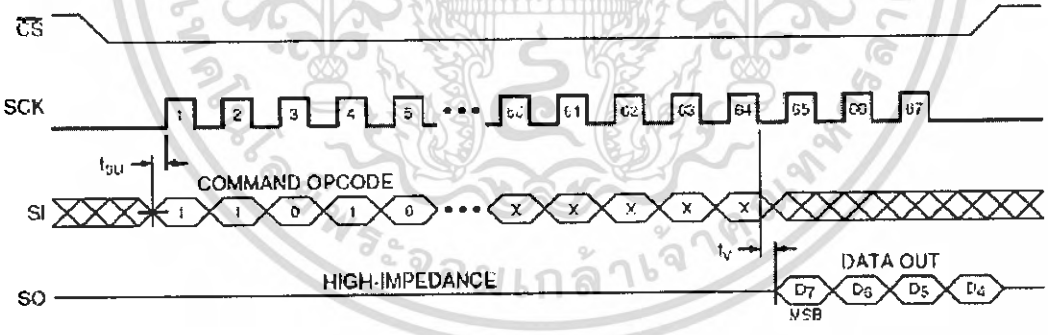
Detailed Bit-Level read Timing-SPI Mode 0

Continuous Array Read (Opcode :D2H)



Detailed Bit-Level read Timing-SPI Mode 3

Continuous Array Read (Opcode :D2H)

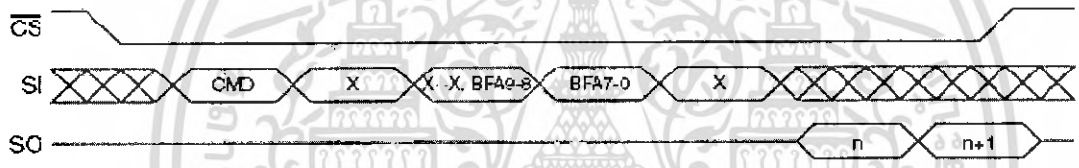


รูปที่ 4.9แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode :E8H (SPI Clock Mode 0 or mode 3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.3 Buffer1 Read and Buffer2 Read Command : คำสั่งนี้จะใช้สำหรับอ่านข้อมูลจาก Buffer1 หรือ Buffer2 ที่อยู่ภายใน Chip โดยถ้าใช้ Opcode 54H หรือ D4H(ขึ้นอยู่กับรูปแบบของ Clock ที่ใช้งาน) จะใช้เพื่ออ่านข้อมูลจาก Buffer1 และถ้าใช้ Opcode 56H หรือ D6H(ขึ้นอยู่กับรูปแบบของ Clock ที่ใช้งาน) จะใช้เพื่ออ่านข้อมูลจาก Buffer2

เริ่มต้นการอ่านนั้นจะต้องส่ง Opcode 8 บิต ออกไปก่อน แล้วตามด้วย 14 บิต Don't Care ตามด้วย 10 บิตแอดเดรส (BFA9-BFA0) ซึ่งเป็นตำแหน่งที่ตั้ง Byte แรกของข้อมูลที่จะอ่านจาก Buffer ซึ่ง Buffer นั้นมีขนาด 528 Byte ดังนั้นแอดเดรสจะอ้างได้ตั้งแต่ 0-527 สุดท้ายตามด้วยข้อมูล Don't Care (0 หรือ 1 ก็ได้) ตามออกไปอีก 1Byte ข้อมูลถึงจะเริ่มถูกส่งออกมาทางขา SO ดังแสดงในรูปด้านล่าง ขณะที่เริ่มต้นส่ง Opcode จนถึงกระบวนการอ่าน Data ขา CS จะต้องยังคงเป็น Low อยู่เสมอ และเมื่อมีการส่งสัญญาณจาก Low-to-High ไปยังขา CS ก็จะเป็นการสิ้นสุดการอ่าน และขา SO ก็จะอยู่ในสภาวะ tri-state



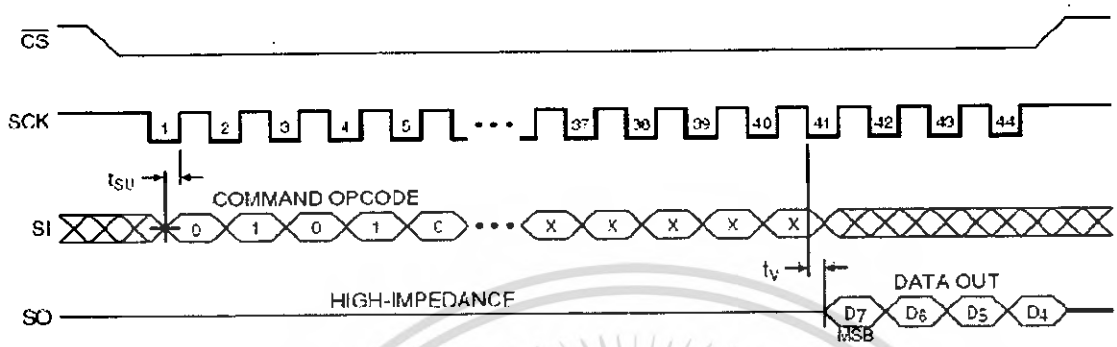
รูปที่ 4.10 แสดง การส่ง Command Buffer Read

จะเห็นว่าในคำสั่งอ่าน Buffer1 และ Buffer2 นี้จะมี Opcode ให้เลือก 2 ชุด คือ 54H หรือ D4H (Buffer1) และ 56H หรือ D6H(Buffer2) ซึ่งจะเลือกใช้ Opcode ใดนั้นจะขึ้นอยู่กับรูปแบบของ Clock ที่ส่งมาให้ Chip ซึ่งจะมีอยู่ด้วยกัน 4 รูปแบบ โดยดูได้จาก Timing Diagram ที่แสดงรายละเอียดการทำงานในระดับบิต ตามรูปแบบ Clock ทั้ง 4 แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

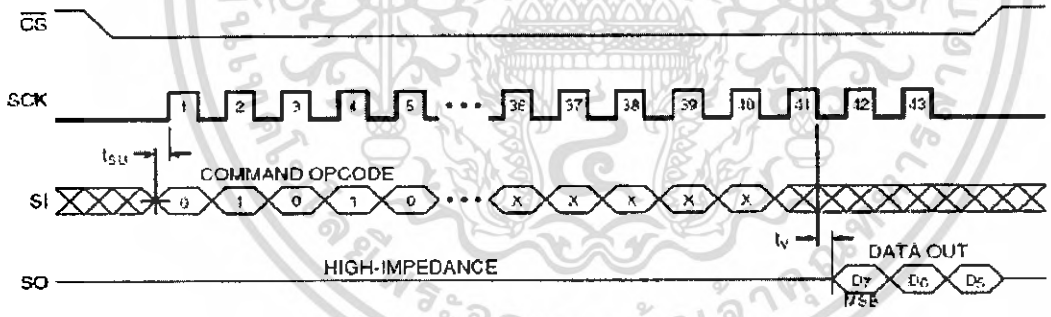
Detailed Bit-Level read Timing – Inactive Clock Polarity High

Beffer Read (Opcode :54 H or 56 H)



Detailed Bit-Level read Timing – Inactive Clock Polarity Low

Beffer Read (Opcode :54 H or 56 H)



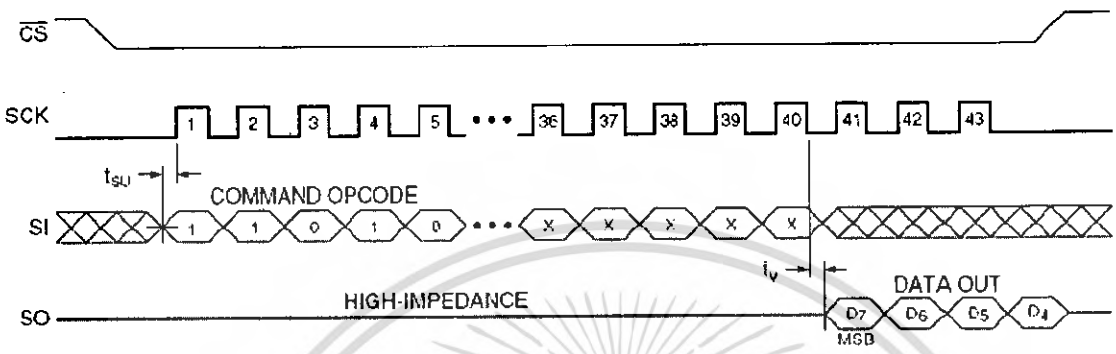
รูปที่ 4.11 แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode : 54 H or 56 H

(Inactive Clock Polarity High- Low)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

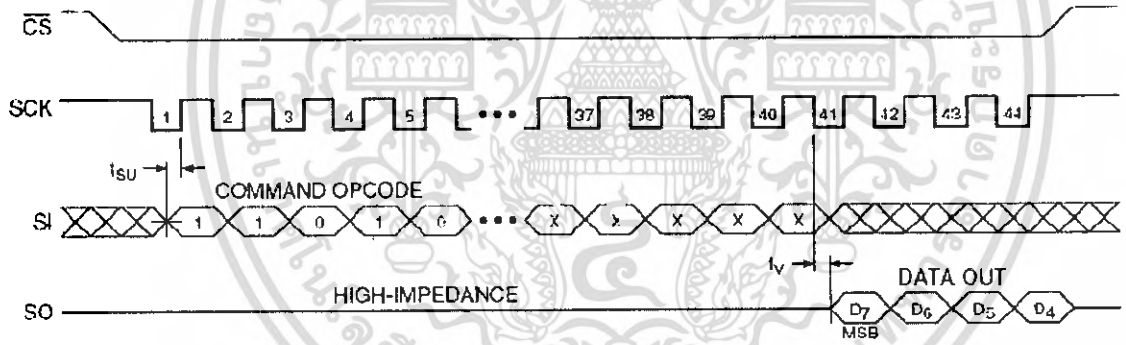
Detailed Bit-Level read Timing – SPI Mode 0

Buffer Read (Opcode : 04H or D6H)



Detailed Bit-Level read Timing – SPI Mode 3

Buffer Read (Opcode : 04H or D6H)



รูปที่ 4.12 แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode : 04H or D6H (SPI Clock Mode 0 Or Mode 3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.4 Status Register Read Command : คำสั่งนี้จะใช้สำหรับอ่าน Status Register เพื่อใช้ตรวจสอบความพร้อมในการรับคำสั่งอื่นต่อไป เริ่มต้นการอ่าน Status Register นั้นจะต้องส่ง Opcode 57H หรือ D7H (ขึ้นอยู่กับรูปแบบของ Clock ที่ใช้งาน) ไปยัง Chip หลังจาก Opcod บิตสุดท้ายถูกเลื่อนเข้าไปแล้ว ข้อมูล 8 บิตของ Status register ก็จะถูกส่งออกมายังขา SO โดยบิต MSB(bit7) จะถูกส่งออกมาเป็นบิตแรก เมื่อข้อมูลถูกเลื่อนออกมาจนครบ 8 บิต การทำงานก็จะวนกลับมาเริ่มส่งค่า Status Register ในบิตที่7 จนครบ 8 บิตซ้ำอีกไปเรื่อยๆ นั่นคือ Status Register ก็จะถูกส่งออกมา Update ไปเรื่อยๆตราบที่ขา CS ยังคงเป็น Low และ SCK ยังคงทำงานอยู่

ตารางที่ 4.2 Status Register Formet

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RDY/BUSY	COMP	1	0	1	1	X	X

จากตาราง Ready/Busy Status จะแสดงในบิตที่7 ของ Status Register โดยถ้าบิต7 นี้เป็น 1 นั่นแสดงว่าอุปกรณ์ พร้อม(Ready) ที่จะรับคำสั่งต่อไป แต่ถ้าบิต 7 นี้เป็น 0 แสดงว่าอุปกรณ์อยู่ในสถานะยังไม่พร้อม(Busy) ซึ่งจะมีอยู่ด้วยกัน 8 คำสั่งที่จะทำให้อุปกรณ์เกิดสภาวะ Busy ได้คือ คำสั่ง Main Memory Page to Buffer Transfer , Main Memory Page to Buffer Compare , Buffer to Main Memory Page Program with Built-in Erase , Buffer to Main Memory Page Program without Built-in Erase , Page Erase , Block Erase , Main Memory Page Program และ Auto Page Rewrite

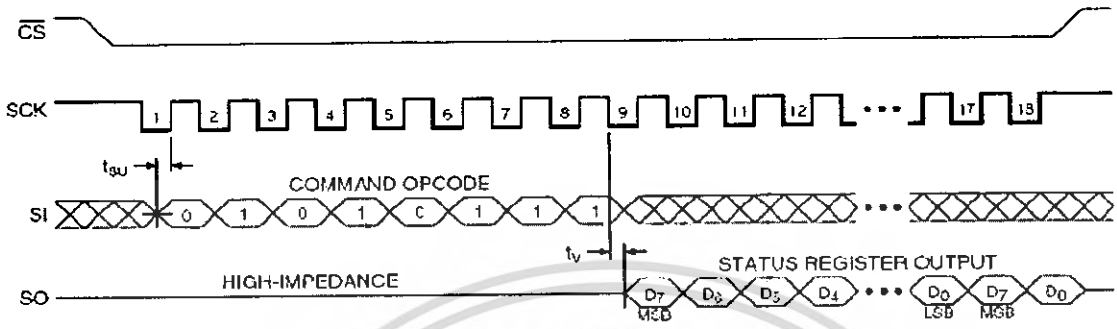
สำหรับบิตที่ 6 นั้นจะใช้แสดงสถานะการทำงานของคำสั่ง Main Memory Page to Buffer Compare โดยถ้าบิต 6 เป็น 0 แสดงว่าข้อมูลใน Memory Page เหมือนกับข้อมูลใน Buffer แต่ถ้าบิต 6 เป็น 1 แสดงว่าข้อมูลใน Main Memory Page อย่างน้อย 1 บิต ไม่เหมือนกับข้อมูลใน Buffer

จะเห็นว่าในคำสั่งอ่าน Status Register นี้จะมี Opcode ให้เลือก 2 ชุด คือ 57 หรือ D7H ซึ่งจะเลือกใช้ Opcode ใดนั้นจะขึ้นอยู่กับรูปแบบของ Clock ที่ส่งมาให้ Chip ซึ่งจะมีอยู่ด้วยกัน 4 รูปแบบ โดยดูได้จาก Timing Diagram ที่แสดงรายละเอียดการทำงานในระดับบิต ตามรูปแบบ Clock ทั้ง 4 แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

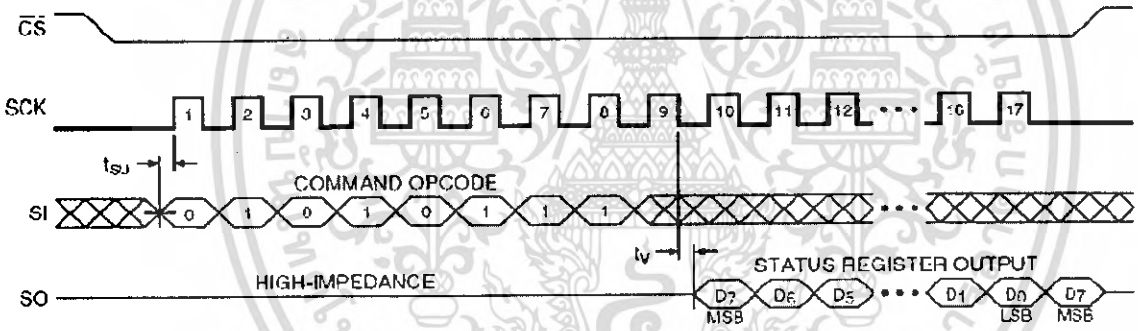
Detailed Bit-Level read Timing – Inactive Clock Polarity High

Status Register Read (Opcode :57 H)



Detailed Bit-Level read Timing – Inactive Clock Polarity Low

Status Register Read (Opcode :57 H)

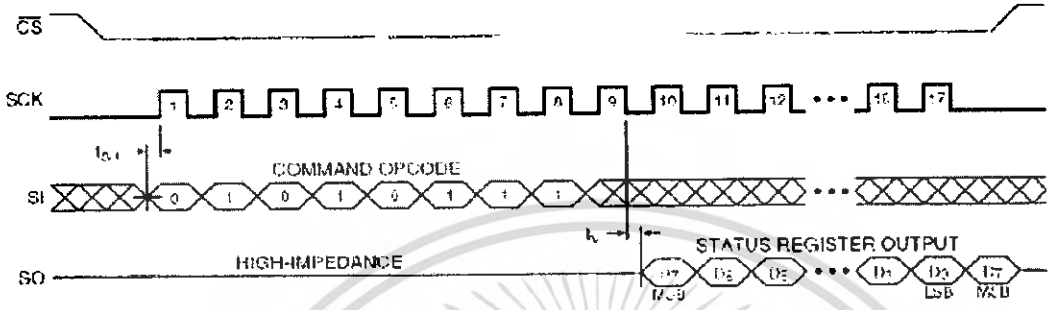


รูปที่ 4.13แสดงการอ่านข้อมูลเมื่อใช้คำสั่ง Opcode :57H (Inactive Clock Polarity High- Low)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

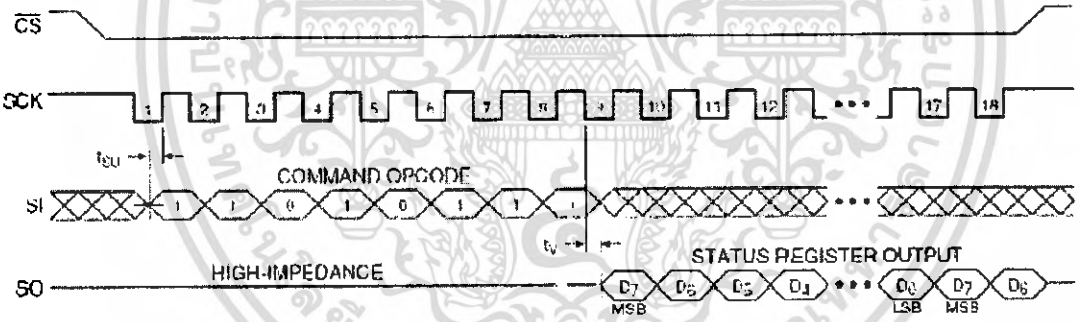
Detailed Bit-Level read Timing – SPI Mode 0

Status Register Read (Opcode : D7H)



Detailed Bit-Level read Timing – SPI Mode 0

Status Register Read (Opcode : D7H)



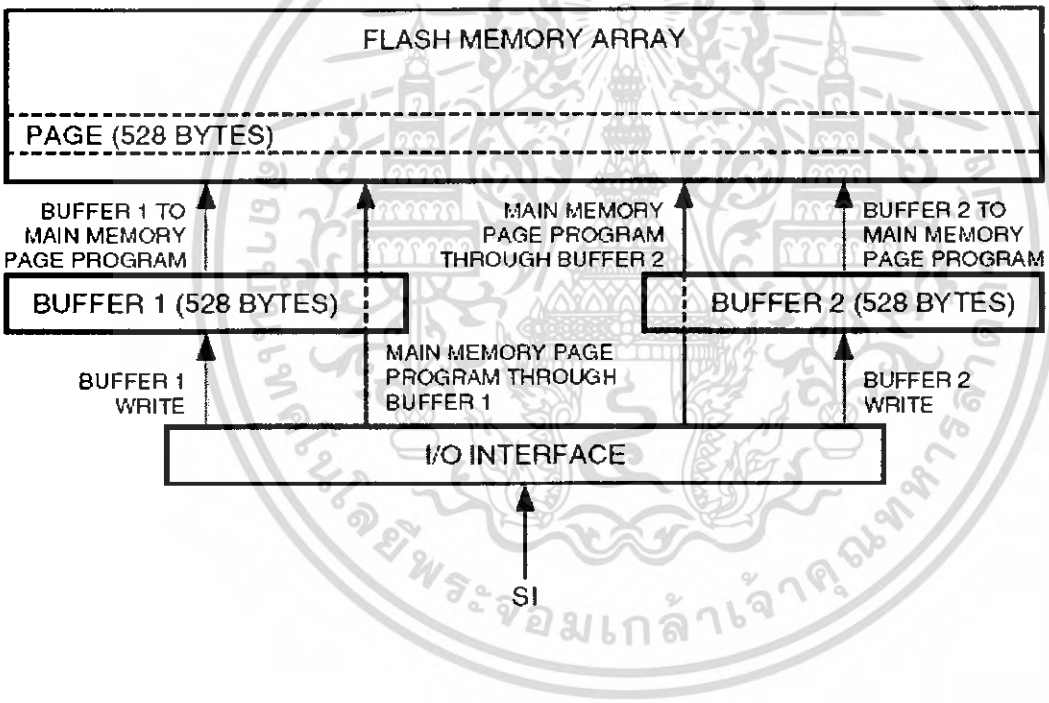
รูปที่ 4.14 แสดงการอ่าน Status Register Read เมื่อใช้คำสั่ง Opcode : D7H

(SPI Clock Mode 0 Or Mode 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 การเขียนและคำสั่งที่ใช้ในการเขียนข้อมูล

การเขียนข้อมูลลงใน Chip นั้นจะแสดงให้เห็นดังบล็อกไดอะแกรมด้านล่าง ซึ่งจะสอดคล้องกับคำสั่งที่ใช้ในการเขียนดังตารางที่2 จากบล็อกไดอะแกรมจะเห็นว่าการเขียนข้อมูลนั้นมีการเขียนได้หลายแบบ ซึ่งแต่ละแบบคำสั่งที่ใช้ในการเขียนก็จะแตกต่างกันไป ผู้ใช้จึงต้องทำความเข้าใจในแต่ละคำสั่งเสียก่อนเพื่อจะเลือกใช้คำสั่งได้ถูกต้อง จะสังเกตเห็นว่าคำสั่งที่ใช้สำหรับเขียน ในแต่ละคำสั่งนั้นจะมี Opcode เพียงชุดเดียว ดังนั้นจะเลือกใช้รูปแบบของสัญญาณ Clock แบบใดแบบหนึ่งใน 4 แบบที่ได้กล่าวไปข้างต้นในคำสั่งอ่านก็ได้ แต่ขอแนะนำให้ใช้รูปแบบของการส่งสัญญาณ Clock แบบเดียวกับที่ใช้กับคำสั่งอ่าน



รูปที่ 4.15 แสดงบล็อกไดอะแกรม การเขียนข้อมูลลง Chip

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตารางที่ 2 ด้านล่างจะเป็นชุดคำสั่งเขียนข้อมูลลง Chip และลบข้อมูลออกจาก Chip หน่วยความจำ ซึ่งจะมีอยู่ด้วยกัน 10 คำสั่ง ซึ่งรูปแบบการส่ง Command ก็จะเหมือนกับในรูปที่ 4

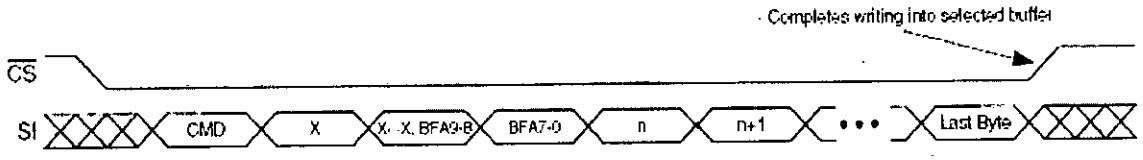
ตารางที่ 4.3 แสดงการส่ง Command

Command	SCK Mode	Opcode
Buffer 1 Write	Any	84H
Buffer 2 Write	Any	87H
Buffer 1 Main Memory Page Program with Built-in Erase	Any	83H
Buffer 2 Main Memory Page Program with Built-in Erase	Any	86H
Buffer 1 Main Memory Page Program without Built-in Erase	Any	88H
Buffer 2 Main Memory Page Program with Built-in Erase	Any	89H
Page Erase	Any	81H
Bloek Erase	Any	50H
Main Memory Page Program through Buffer 1	Any	82H
Main Memory Page Program through Buffer 2	Any	85H

Any = เลือกใช้ Clock แบบใดแบบหนึ่งใน 4 แบบ ได้แก่ Inactive Clock Polarity Low , Inactive Clock Polarity Low , SPI Mode 0 or SPI Mode 3

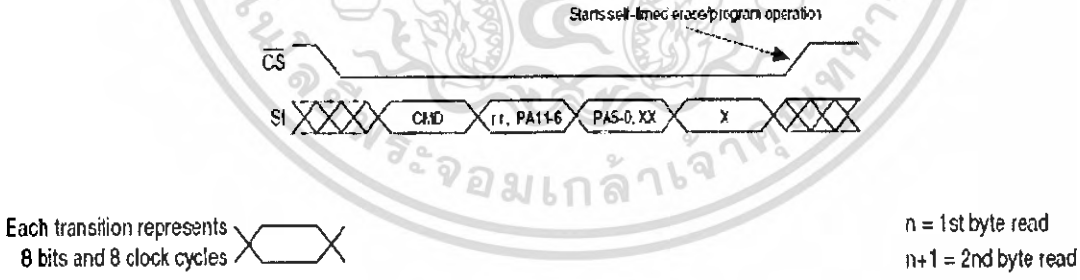
4.4.1Buffer 1 Write และ Buffer 2 Write : 2 คำสั่งนี้จะใช้สำหรับส่งข้อมูลจากภายนอกผ่านขา SI เข้าไปเก็บไว้ภายใน Buffer1 หรือ Buffer2 ของตัว Chip ซึ่งการโหลดข้อมูลเข้าไปยัง Buffer นั้น จะต้องส่ง Opcode 8 บิต คือ 84H สำหรับ Buffer1 หรือ 87H สำหรับ Buffer2 แล้วตามด้วย 14 บิต Don't Care และ 10 บิตแอดเดรส(BFA9-BFA0) ตามลำดับ ซึ่ง 10 บิตแอดเดรสนี้ก็คือ ตำแหน่งแอดเดรสเริ่มต้นใน Buffer ที่จะทำการเขียน สุดท้ายก็ตามด้วยข้อมูลที่จะเขียน 528 Byte ข้อมูลจะถูกโหลดเข้าไปยัง Buffer จนกระทั่ง ขา CS เปลี่ยนสถานะจาก Low เป็น High ถึงจะเป็นการหยุดโหลดข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.16 แสดง การส่ง Command Buffer Write

4.4.2 Buffer1 หรือ Buffer2 to Main Memory Page Program With Built-In Erase : 2 คำสั่งนี้จะใช้สำหรับเคลื่อนย้ายข้อมูล จาก Buffer 1 หรือ Buffer 2 เข้าไปเก็บไว้ยัง Main Memory ในการเริ่มต้นการทำงานจะต้องส่ง 8บิต opcode 83H สำหรับ Buffer1 หรือ 86H สำหรับ Buffer2 แล้วตามด้วย 2 บิต สงวน(π) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะทำการเขียนข้อมูลเข้าไป สุดท้ายตามด้วย 10 บิต Don't Care โดยจะเป็นไปตามรูปที่18 หลังจากส่ง Command ครบ 4 Byte แล้ว และทำให้ขา CS เปลี่ยนระดับจาก Low เป็น High จะทำให้ Page ที่ถูกเลือกใน Main Memory ถูกลบข้อมูลออกทั้งหมด จากนั้นก็จะเริ่มทำการย้ายข้อมูลจาก Buffer ไปเก็บไว้ยัง Page ใน Main Memory ที่ได้เลือกไว้ ระหว่างที่มีการเคลื่อนย้ายข้อมูลนี้ Status Register จะแสดงสภาวะ Busy



รูปที่ 4.17 Timming แสดง การส่ง Command Buffer to Main Memory Page Program

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

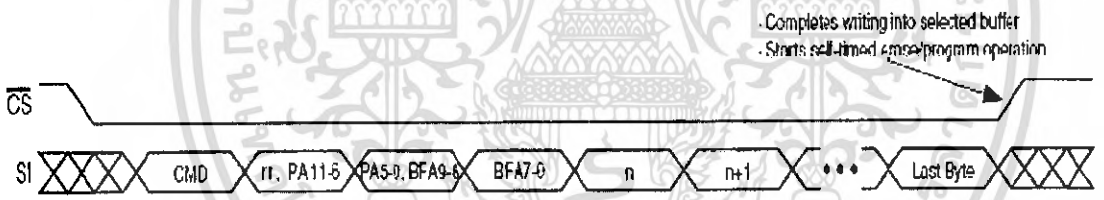
4.4.3. Buffer1 หรือ Buffer2 to Main Memory Page Program Without Built-In Erase : 2 คำสั่งนี้จะใช้สำหรับเคลื่อนย้ายข้อมูล จาก Buffer 1 หรือ Buffer 2 เข้าไปเก็บไว้ยัง Main Memory เช่นกัน แต่จะไม่มีการลบข้อมูล ที่อยู่ใน Page ออกก่อน ในการเริ่มต้นการทำงานจะต้องส่ง 8 บิต Opcode 88H สำหรับ Buffer1 หรือ 89H สำหรับ Buffer2 แล้วตามด้วย 2 บิต สงวน(rr) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะทำการเขียนข้อมูลเข้าไป สุดท้ายตามด้วย 10 บิต Don't Care หลังจากส่ง Command ครบ 4 Byte แล้ว และทำให้ขา CS เปลี่ยนระดับจาก Low เป็น High ก็จะเริ่มทำการย้ายข้อมูลจาก Buffer ไปเก็บไว้ยัง Page ใน Main Memory ที่ได้เลือกไว้ ระหว่างที่มีการเคลื่อนย้ายข้อมูลนี้ Status Register จะแสดงสภาวะ Busy

4.4.4.Page Erase : คำสั่งนี้จะถูกใช้สำหรับลบข้อมูลใน Page ของ Main Memory ซึ่งจะเป็นประโยชน์ต่อการใช้คำสั่ง Buffer to Main Memory Page Program Without Built-In Erase ตามออกมาหลังคำสั่ง Page Erase ในการเริ่มต้นการทำงานจะต้องส่ง 8บิต Opcode 81H แล้วตามด้วย 2 บิต สงวน(rr) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะทำการลบข้อมูลออกสุดท้ายตามด้วย 10 บิต Don't Care หลังจากส่ง Command ครบ 4 Byte แล้ว และทำให้ขา CS เปลี่ยนระดับจาก Low เป็น High ก็จะเริ่มทำการลบข้อมูลใน Page ที่ถูกเลือกออก ระหว่างที่มีการลบข้อมูลออกนี้ Status Register จะแสดงสภาวะ Busy

4.4.5.Block Erase : คำสั่งนี้จะใช้สำหรับลบข้อมูลเป็น Block ซึ่งก็คือการลบข้อมูล ใน Page ของ Main Memory ออกครั้งละ 8 Page ซึ่งจะเป็นประโยชน์ต่อการใช้คำสั่ง Buffer to Main Memory Page Program Without Built-In Erase เพื่อลดเวลาในการเขียนข้อมูลจำนวนมากลงไปยัง Main memory ในการเริ่มต้นการทำงานจะต้องส่ง 8บิต Opcode 50H แล้วตามด้วย 2 บิต สงวน(rr) และ 9 บิตแอดเดรส (PA11-PA3) ซึ่งเป็นตำแหน่ง Block ของ 8 Pageใน Main Memory ที่จะทำการลบข้อมูลออก สุดท้ายตามด้วย 13 บิต Don't Care หลังจากส่ง Command ครบ 4 Byte แล้ว และทำให้ขา CS เปลี่ยนระดับจาก Low เป็น High ก็จะเริ่มทำการลบข้อมูลใน Block ที่ถูกเลือกออก ระหว่างที่มีการลบข้อมูลออกนี้ Status Register จะแสดงสภาวะ Busy ในการส่ง Command (4Byte)ออกไปแต่ละครั้งนั้นผู้ใช้จะต้องทำให้ขา CS เปลี่ยนสถานะจาก High เป็น Low เสมอ และขา CS จะเปลี่ยนสถานะจาก Low มาเป็น High อีกครั้งเมื่อไหร่นั้นจะขึ้นอยู่กับรูปแบบของคำสั่งแต่ละคำสั่งที่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.6.Main Memory Page Program through Buffer1 หรือ Buffer2 : 2 คำสั่งนี้จะเป็นการรวมเอาการทำงานของคำสั่ง Buffer Write และ Buffer to Main Memory Page Program with Built-in Erase เข้าไว้ด้วยกัน นั่นคือเมื่อใช้คำสั่งนี้ ข้อมูลจะถูกเลื่อนจากขา SI เข้าไปยัง Buffer1 หรือ Buffer2 จากนั้นข้อมูลก็จะถูกเลื่อนต่อเข้าไปยัง Main Memory ให้โดยอัตโนมัติ ในการเริ่มต้นการทำงานจะต้องส่ง 8บิต Opcode 82H สำหรับ Buffer1 และ 85H สำหรับ Buffer2 แล้วตามด้วย 2 บิต สงวน(rr) และ 12 บิต แอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Pageใน Main Memory ที่จะทำการเขียนข้อมูลเข้าไป สุดท้ายตามด้วย 10 บิด แอดเดรส(BFA9-BFA0) ซึ่งจะเป็นตำแหน่งแอดเดรสเริ่มต้น ของ Buffer ที่จะใช้ทำการเขียนข้อมูลเข้าไป หลังจากส่ง Command ครบ 4 Byte แล้ว จะต้องทำการส่งข้อมูลตามไป ข้อมูลก็จะถูกเลื่อนเข้าทางขา SI และถูกเก็บไว้ยัง Buffer ในขณะที่เลื่อนข้อมูลเข้าไปนี้ ขา CS จะต้องยังคงเป็น Low อยู่ หลังจากโหลดข้อมูลครบ 528 byte แล้ว และทำให้ขา CS เปลี่ยนระดับจาก Low เป็น High ข้อมูลเดิมใน Page ที่ถูกเลือกไว้ก็จะถูกลบออกไป จากนั้นข้อมูลที่ถูกเก็บไว้ใน Buffer ที่โหลดเข้ามาในตอนแรกก็จะถูกเคลื่อนย้ายเข้าไปเก็บไว้ใน Page ของ Main Memory โดยอัตโนมัติ ในระหว่างการทำงานนี้



รูปที่ 4.18 Timing แสดงการส่ง Command Main Memory Page Program through Buffer

4.5 คำสั่งเพิ่มเติม

สำหรับคำสั่งเพิ่มเติมนี้จะแสดงดังในตารางที่ 3 ซึ่งจะเป็นคำสั่งในการเคลื่อนย้ายข้อมูลภายใน Chip ระหว่าง Buffer ไปยัง Main Memory หรือ จาก Main Memmory ไปยัง Buffer โดยรูปแบบ Clock SPI ที่จะใช้งานในคำสั่งชุดนี้จะเป็นรูปแบบใดรูปแบบหนึ่งใน 4 แบบ ที่กล่าวไปใน Mode ของคำสั่งอ่าน ก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.4 คำสั่งเพิ่มเติม

Command	SCK Mode	Opcode
Main Memory Page to Buffer 1 Transfer	Any	53H
Main Memory Page to Buffer 2 Transfer	Any	55H
Main Memory Page to Buffer 1 Compare	Any	60H
Main Memory Page to Buffer 2 Compare	Any	61H
Auto Page Rewrite through Buffer 1	Any	58H
Auto Page Rewrite through Buffer 2	Any	59H

Any = เลือกใช้ Clock แบบใดแบบหนึ่งใน 4 แบบ ได้แก่ Inactive Clock Polarity Low , Inactive Clock Polarity Low , SPI Mode 0 และ SPI Mode 3

4.5.1. Main Memory Page to Buffer 1 Transfer และ Main Memory Page to Buffer 2 Transfer : 2 คำสั่งนี้จะเป็นการเคลื่อนย้ายข้อมูลครั้งละ 1 Page(528 byte)จาก Main Memory ไปเก็บไว้ที่ Buffer1 หรือ Buffer2 การทำงานจะเริ่มต้นด้วยการ ส่ง 8 บิต Opcode 53H สำหรับ Buffer1 และ 55H สำหรับ Buffer2 ตามด้วย 2 บิตสงวน(π) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะทำการอ่าน สุดท้ายตามด้วย 10 บิต Don't Care ซึ่งจะเป็นไปตามรูปแบบการส่ง Command ในรูปที่ 3 ในขณะที่เริ่มทำการส่ง Opcode ออกไปที่ขา SI จนครบ 4 Byte ขา CS จะต้องเป็น Low และข้อมูลจะเริ่มถูกเคลื่อนย้ายจาก Main Memory ไปยัง Buffer ก็ต่อเมื่อ ขา CS ถูกเปลี่ยนสถานะจาก Low เป็น High ในระหว่างที่มีการเคลื่อนย้ายข้อมูลนี้ สามารถจะอ่าน Status Register ได้ว่าการเคลื่อนย้ายข้อมูลสมบูรณ์หรือยัง

4.5.2 Main Memory Page to Buffer 1 Compare และ Main Memory Page to Buffer 2 Compare : 2 คำสั่งนี้จะเป็นการเปรียบเทียบข้อมูลใน Page ของ Main Memory กับข้อมูลที่อยู่ใน Buffer1 หรือ Buffer2 การทำงานจะเริ่มต้นด้วยการ ส่ง 8 บิต Opcode 60H สำหรับ Buffer1 และ 61H สำหรับ Buffer2 ตามด้วย 2 บิตสงวน(π) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะใช้ทำการ Compare ข้อมูล กับ Buffer สุดท้ายตามด้วย 10 บิต Don't Care ซึ่งจะเป็นไปตามรูปแบบการส่ง Command ในรูปที่ 3 ในขณะที่เริ่มทำการส่ง Opcode ออกไปที่ขา SI จนครบ 4 Byte ขา CS จะต้องเป็น Low และเมื่อขา CS เปลี่ยนระดับสัญญาณจาก Low เป็น High ข้อมูล 528 Byte

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใน Main Memory Page ที่ถูกเลือก ก็จะถูกเปรียบเทียบกับข้อมูล 528 byte ที่อยู่ใน Buffer 1 หรือ Buffer 2 ในระหว่างที่ทำการเปรียบเทียบข้อมูลอยู่นี้ Status Register จะแสดงสภาวะ Busy และเมื่อการ Compare สมบูรณ์ ผลของการ Compare ก็จะถูก Update ในบิตที่6 ของ Register Status

4.5.3.Auto Page Rewrite through Buffer1 หรือ Buffer 2 : 2 คำสั่งนี้จะเป็นการรวมเอาการทำงานของคำสั่ง Main Memory Page to Buffer Transfer และ Buffer to Main Memory Page Program with Built-in Erase เข้าไว้ด้วยกัน โดยอันดับแรกข้อมูลใน Page จะถูกส่งออกจาก Main Memory ไปยัง Buffer 1 หรือ Buffer 2 จากนั้นข้อมูลเดียวกันนี้ จาก Buffer 1 หรือ Buffer 2 ก็จะถูกส่งกลับเข้าไปยัง Page เดิมของ Main Memory อีกครั้งหนึ่ง การใช้งานของคำสั่งจะเริ่มต้นด้วยการ ส่ง 8 บิต Opcode 58H สำหรับ Buffer1 หรือ 59H สำหรับ Buffer2 ตามด้วย 2 บิตสงวน(π) และ 12 บิตแอดเดรส (PA11-PA0) ซึ่งเป็นตำแหน่ง Page ใน Main Memory ที่จะใช้ทำการ Rewrite ข้อมูล สุดท้ายตามด้วย 10 บิต Don't Care ซึ่งจะเป็นไปตามรูปแบบการส่ง Command ในรูปที่ 3 ในขณะที่เริ่มทำการส่ง Opcode ออกไปที่ขา SI จนครบ 4 Byte ขา CS จะต้องเป็น Low และเมื่อขา CS เปลี่ยนระดับสัญญาณจาก Low เป็น High ข้อมูลใน Page ก็จะถูกส่งจาก Main Memory ไปยัง Buffer และข้อมูลเดียวกันนี้ที่อยู่ใน Buffer ก็จะถูกส่งกลับเข้ามายัง Page เดิมของ Main Memory อีกครั้ง ในระหว่าง ที่ทำงานนี้ StatusRegisterจะแสดงสภาวะBusy

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

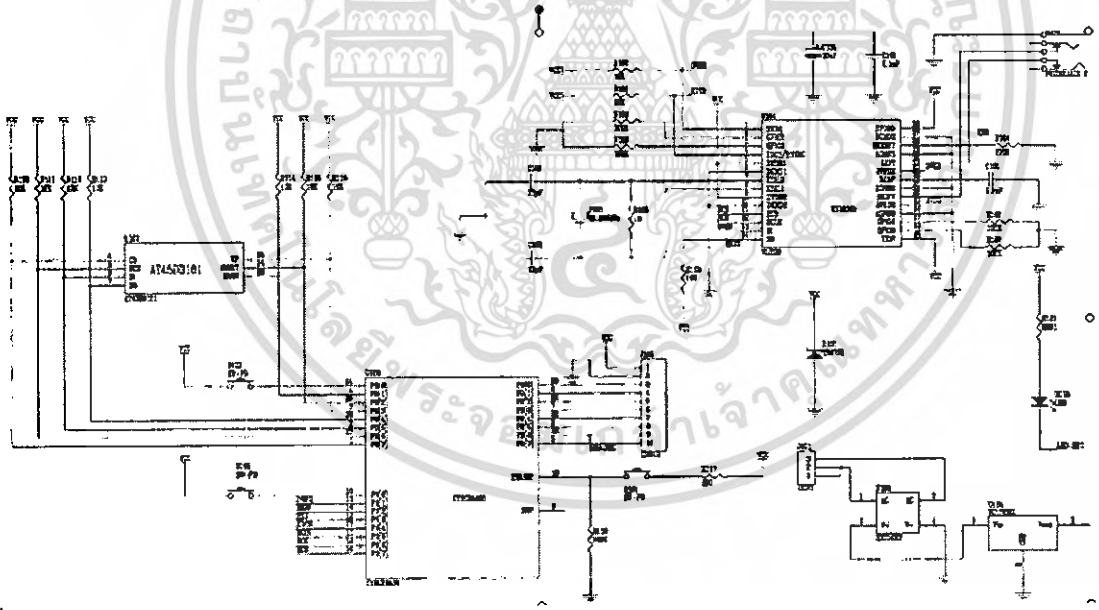
กระบวนการออกแบบ

5.1 กระบวนการออกแบบแบ่งออกเป็น 3 ส่วน

1. ส่วนควบคุมการทำงาน
2. ส่วนถอดรหัสข้อมูล
3. ส่วนเก็บข้อมูล

5.1.1 ส่วนที่ 1 ส่วนควบคุมการทำงาน

ส่วนควบคุมการทำงานทั้งหมด จะใช้ไมโครคอลโทรลเลอร์ PSoC เบอร์ CY8C466 ในการอ่านข้อมูลจากส่วนเก็บข้อมูลซึ่งก็ คือ data flash memory แล้วนำข้อมูลที่ได้ส่งไปยังส่งพอร์ตรหัสข้อมูล



รูป 5.1 ส่วนควบคุมการทำงานโดยใช้ไมโครคอลโทรเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนของการต่อใช้งานจะใช้พอร์ท 1 ตั้งแต่ P1[1]-[7] ต่อกับชุดถอดรหัสข้อมูล VS 1011B ในรูปแบบของ SPI ดังรูปด้านบน

ในส่วนของการต่อใช้งานจะใช้พอร์ท 2 ตั้งแต่ P0[1]-[7]ต่อกับชุดเก็บข้อมูล Data Flesh memory

เนื่องจาก PSoC มีคุณสมบัติเด่นที่แตกต่างไปจากไมโครคอลโทรลเลอร์ตระกูลอื่นๆ ก็คือ PSoC MCU จะรวมเอาการออกแบบทั้งทางด้าน ดิจิตอล และ อนาลอก มาไว้ด้วยกันทำให้การออกแบบที่ต้องมีความสัมพันธ์ระหว่าง อุปกรณ์ทางดิจิตอลและอนาลอก สามารถทำได้โดยง่ายและสะดวกสบายยิ่งขึ้น อีกทั้งยังทำให้ขนาดของการออกแบบเล็กลงอีกด้วย โดยเฉพาะวงจรทางด้านอนาลอก ซึ่งมักจะมีขนาดค่อนข้างใหญ่ อีกทั้งการทำงานของ PSoC ยังมีความคล่องตัวสูงทั้งเรื่องของแหล่งกำเนิดสัญญาณนาฬิกาที่หลากหลาย นอกจากนี้ยังมีฟังก์ชัน In-System Serial Programming (ISSP) ที่สามารถทำการโปรแกรมซอร์สโคสที่ได้ออกแบบลงไปในหน่วยความจำโปรแกรม(Flash Memory) ภายในตัวชิพได้ ซึ่งจะช่วยให้การพัฒนาโปรแกรมให้กับไมโครคอล-โทรลเลอร์สะดวกสบายยิ่งขึ้น

5.2 จุดเด่นของ PSoC MCU เมื่อเทียบกับไมโครคอลโทรลเลอร์อื่นๆ

5.2.1User Modules : สามารถเลือกใช้ทรัพยากรของระบบได้ตามต้องการทั้ง อนาลอกและดิจิตอล จะไม่ถูกจำกัดด้วยโครงสร้างฮาร์ดแวร์เหมือนกับไมโครคอลโทรลเลอร์อื่นๆ

5.2.2 API (Application Programming Interface): สนับสนุนการพัฒนาโปรแกรมด้วยฟังก์ชัน API ช่วยให้ผู้พัฒนาโปรแกรมสามารถเขียนการออกแบบโปรแกรมได้โดยง่าย

5.2.3 ISRs (Interrupt Service Routines) :สนับสนุน และ รองรับการทำงานแบบอินเตอร์รัพท์

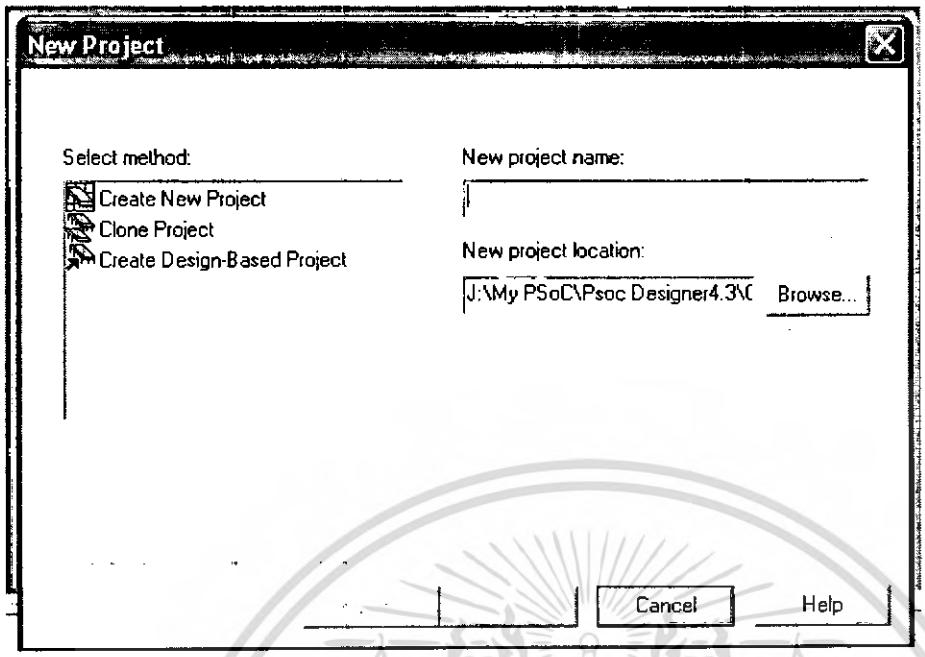
5.2.3 Interconnect device interface : สามารถทำการเชื่อมต่อสัญญาณต่างๆได้อย่างอิสระไม่ถูกกำหนดตายตัวตามฮาร์ดแวร์ เหมือนกับไมโครคอลโทรลเลอร์อื่นๆ

5.3 การเขียนโปรแกรม

5.3.1 การเขียนโปรแกรม จะใช้โปรแกรม PSoC designer 4.3 ในการจัดระบบการทำงานทำการสร้างโปรเจคขึ้นมาใหม่

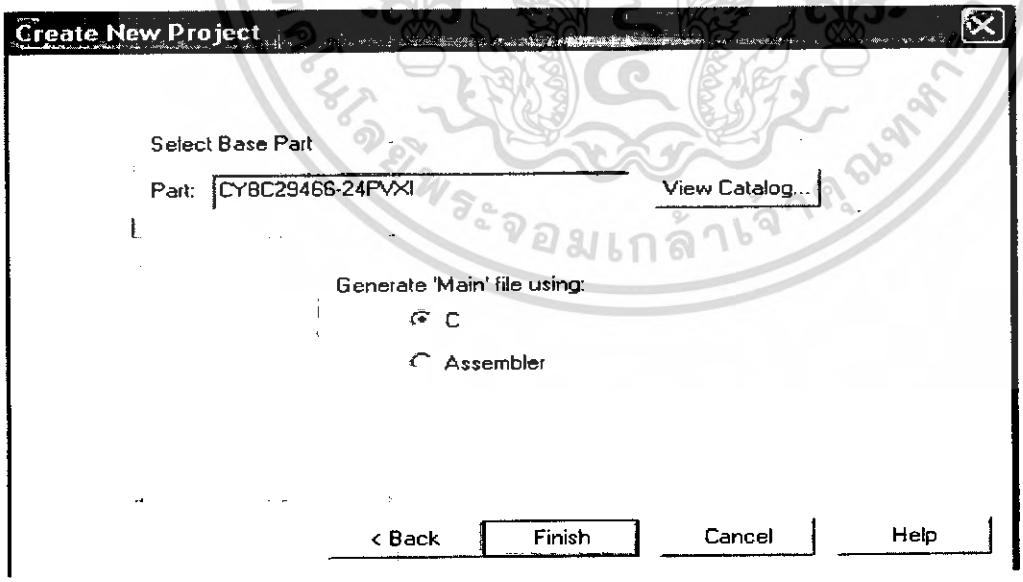
รูปที่ 5.2 แสดงโปรแกรม PSoC designer 4.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 5.3 แสดงหน้าต่างNew โปรเจค

5.3.2 การเลือก Part การเลือก Part ของอุปกรณ์เป็นเบอร์ CY8C29466-24PVXI และเลือกรูปแบบของภาษาที่จะพัฒนาเป็นภาษาซี



รูป 5.4 แสดงหน้าต่าง Create New Projec โดยการเลือกใช้ภาษาซี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.3 ทำการกำหนดค่าต่างๆให้กับ PSoC MCU

Global Resources	Value
Power Setting [Vcc / Sys	3.3V / 24MHz
CPU_Clock	SysClk/2
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	8
VC2= VC1/N	1
VC3 Source	SysClk/1
VC3 Divider	4
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

SPIM_1

User Module Parameters	Value
Clock	VC1
MISO	Row_0_Input_2
MOSI	Row_0_Output_1
SClk	Row_0_Output_2
Interrupt Mode	TXRegEmpty
ClockSync	Sync to SysClk
InvertMISO	Normal

รูปที่ 5.5 การกำหนดค่าต่างๆให้กับ PSoC MCU

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

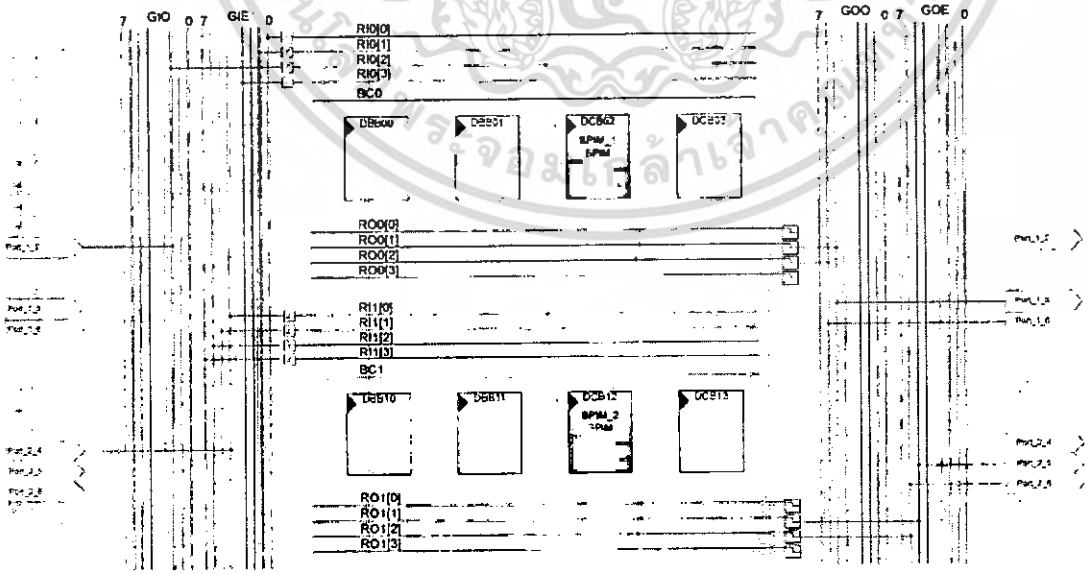
SPIM_2

User Module Parameters	Value
Clock	VC3
MISO	Row_1_Input_0
MOSI	Row_1_Output_1
SClk	Row_1_Output_2
Interrupt Mode	TXRegEmpty
ClockSync	Sync to SysClk
InvertMISO	Normal

Name	Port	Select	Drive	Interrupt
Port_0_4	P0[4]	StdCPU	High Z Ar	DisableInt
Port_0_5	P0[5]	StdCPU	High Z Ar	DisableInt
Port_0_6	P0[6]	StdCPU	High Z Ar	DisableInt
Port_0_7	P0[7]	StdCPU	High Z Ar	DisableInt
Port_1_0	P1[0]	StdCPU	Strong	DisableInt
Port_1_1	P1[1]	StdCPU	High Z	DisableInt
Port_1_2	P1[2]	GlobalInO	High Z	DisableInt
Port_1_3	P1[3]	StdCPU	Strong	DisableInt
Port_1_4	P1[4]	StdCPU	Strong	DisableInt
Port_1_5	P1[5]	GlobalOut	Strong	DisableInt
Port_1_6	P1[6]	GlobalOut	Strong	DisableInt
Port_1_7	P1[7]	StdCPU	Strong	DisableInt
Port_2_0	P2[0]	StdCPU	High Z	DisableInt
Port_2_1	P2[1]	StdCPU	High Z	DisableInt
Port_2_2	P2[2]	StdCPU	Strong	DisableInt
Port_2_3	P2[3]	StdCPU	Strong	DisableInt
Port_2_4	P2[4]	GlobalInE	High Z	DisableInt
Port_2_5	P2[5]	GlobalOut	Strong	DisableInt
Port_2_6	P2[6]	GlobalOut	Strong	DisableInt
Port_2_7	P2[7]	StdCPU	Strong	DisableInt

รูปที่ 5.6 การกำหนดค่าต่างๆให้กับ PSoC MCU (ต่อ)

5.3.3 ทำการเชื่อมต่อ Interconnect



รูปที่ 5.7 การเชื่อมต่อ Interconnect

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3.4 ทำการเขียนซอร์สโคส

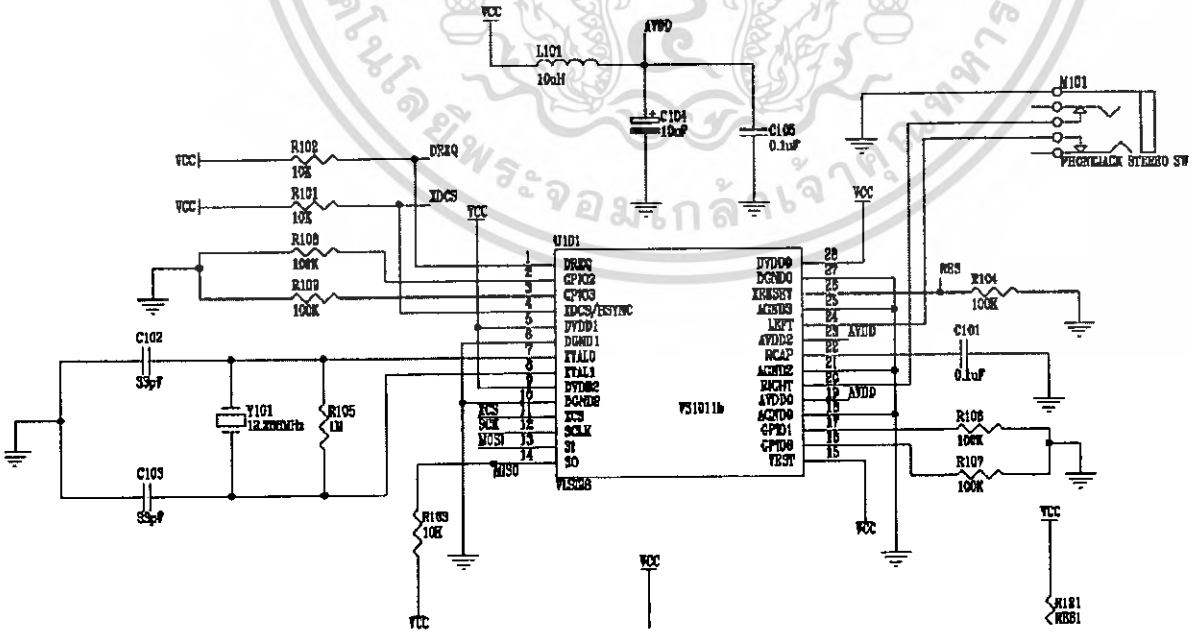
```
File  Edit  Format  View  Help
#define SCI_AICTRL3          0x0F              // Application Control Register. 3

//---------------------------------------------------------------------------
//
// Define For dataflash Command Register
//
//---------------------------------------------------------------------------
#define Continuous_Array_Read                    0xE8          // SPI Mode 3
#define Main_Memory_Page_Read                    0xD2     // SPI Mode 3
#define Buffer1_Read                             0xD4     // SPI Mode 3
#define Buffer2_Read                                      0xD6     // SPI Mode 3
#define Status_Register_Read          0xD7     // SPI Mode 3
#define Buffer_1_Write                                    0x84                    // Any SPI Mode
#define Buffer_2_Write                                    0x87                    // Any SPI Mode
#define Buffer1_to_Main_Memory_Erase          0x83              // Any SPI Mode
#define Buffer2_to_Main_Memory_Erase          0x86              // Any SPI Mode
```

รูปที่ 5.8 การเขียนซอร์สโคส

5.4 ส่วนที่ 2 ส่วนถอดรหัสข้อมูล

5.4.1 ส่วนถอดรหัสข้อมูลใช้ Chip VS1011b ในการถอดรหัสข้อมูล MP3 โดยการเชื่อมต่อใช้งานจะใช้การเชื่อมต่อในรูปแบบของ SPI

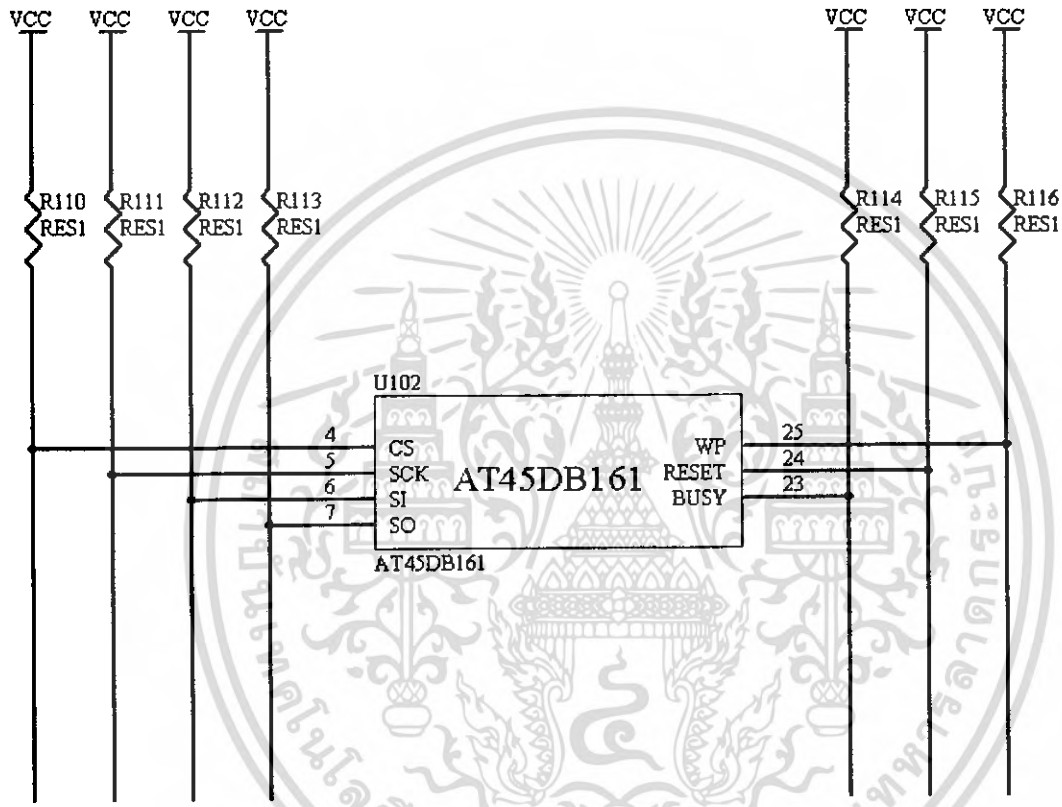


รูปที่ 5.9 ส่วนถอดรหัสข้อมูลใช้ Chip VS1011b

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.5 ส่วนที่ 3 ส่วนเก็บข้อมูล

5.5.1 ส่วนเก็บข้อมูลใช้ Chip AT45DB161 เป็น Chip Data Flash มีความจุ 2 Mbit ในการเก็บข้อมูลเพลง MP3



รูปที่ 5.10 ส่วนเก็บข้อมูลใช้ Chip AT45DB161

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

ผลการทดสอบ

6.1 ผลการทดสอบ

6.1.1 เครื่องมือที่ใช้ในการทดสอบ

คอมพิวเตอร์ ซีพียู AMD 1.4 MHz Memory 512 Byte ระบบปฏิบัติการวินโดวส์ XP โปรแกรมวินโดวส์ Media Player รุ่น 10

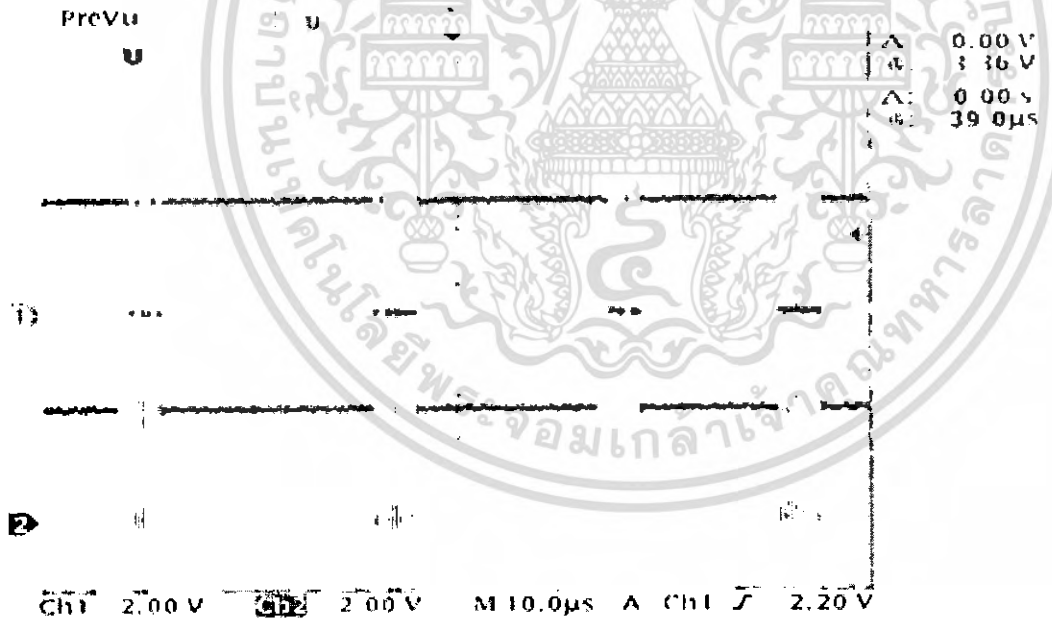
6.1.2 ออสซิลโลสโคป เป็นออสซิลโลสโคปดิจิตอล

6.1.3 ฟังก์ชันเจนเนอร์เรเตอร์

6.1.4 เครื่องเล่นเอ็มพีสามดีโค้ดเดอร์

6.2 การวัดสัญญาณ Read/Write

6.2.1 สัญญาณที่ใช้ในการ Write Data flash ได้ผลดังรูปที่ 6.1



รูปที่ 6.1 แสดงสัญญาณ Write Data flash Memory

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

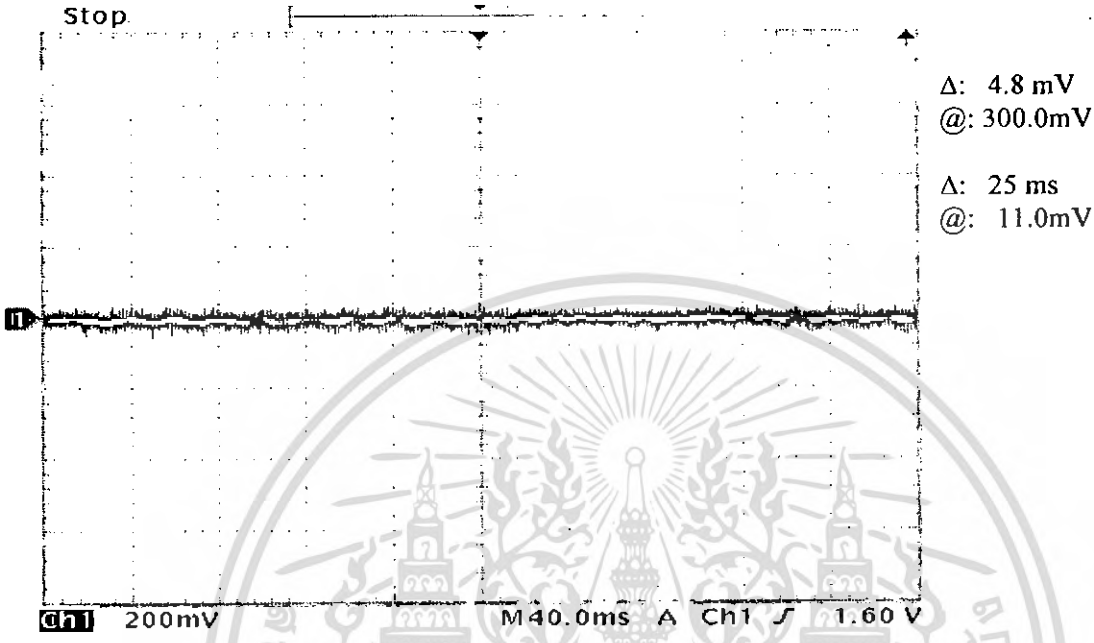
6.2.2 สัญญาณที่ใช้ในการ Read Data flash Memory ได้ผลดังรูปที่ 6.2

รูปที่ 6.2 แสดงสัญญาณ Read Data flash Memory

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

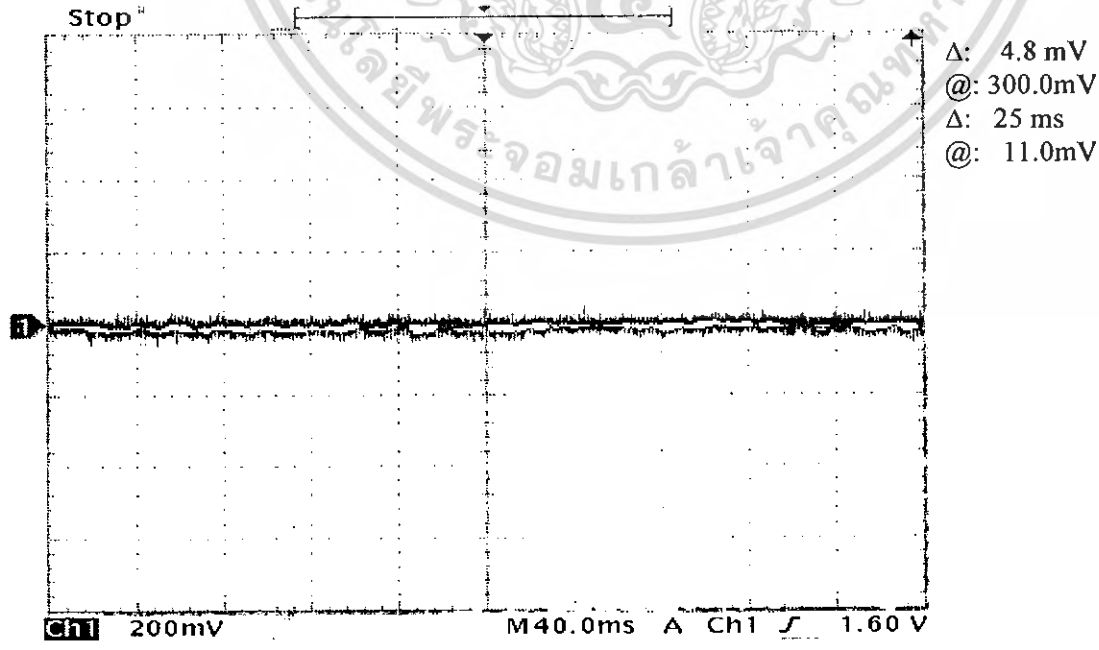
6.3 การทดสอบการตอบสนองต่อความถี่ 20 Hz - 20KHz

6.3.1 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 10 Hz ได้ผลดังรูปที่ 6.3



รูปที่ 6.3 แสดงสัญญาณ Sin 10 Hz

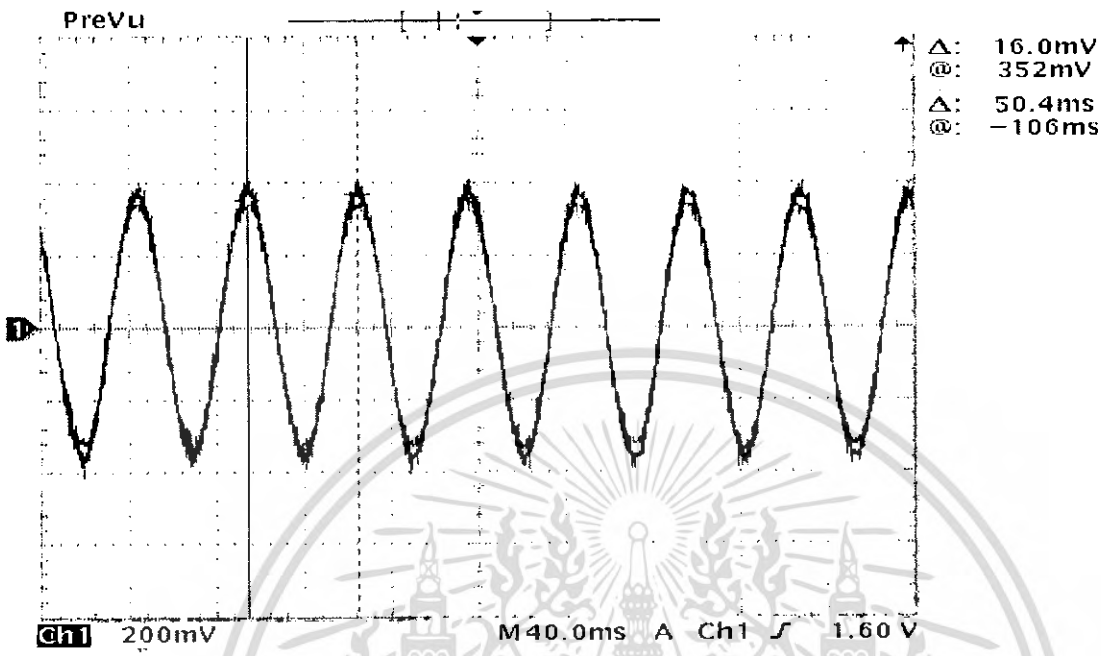
6.3.2 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 10 Hz ได้ผลดังรูปที่ 6.4



รูปที่ 6.4 สัญญาณ Sin 10 Hz เมื่อใช้เครื่องเล่น MP3 Decoder

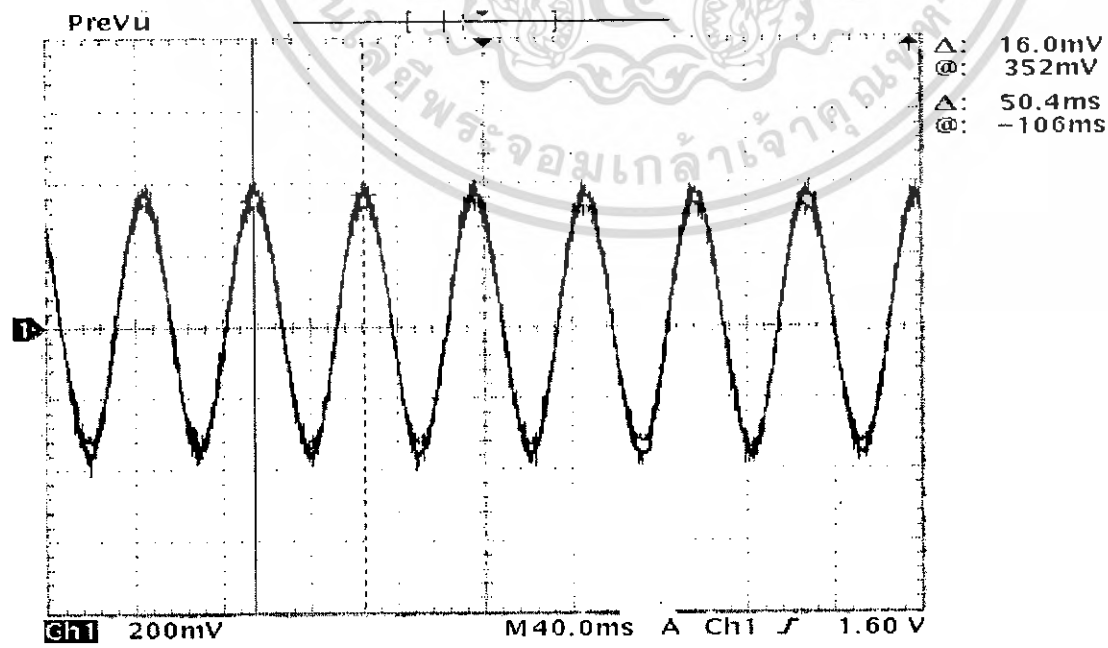
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.3 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 20 Hz ได้ผลดังรูปที่ 6.5.



รูปที่ 6.5 แสดงสัญญาณ Sin 20 Hz

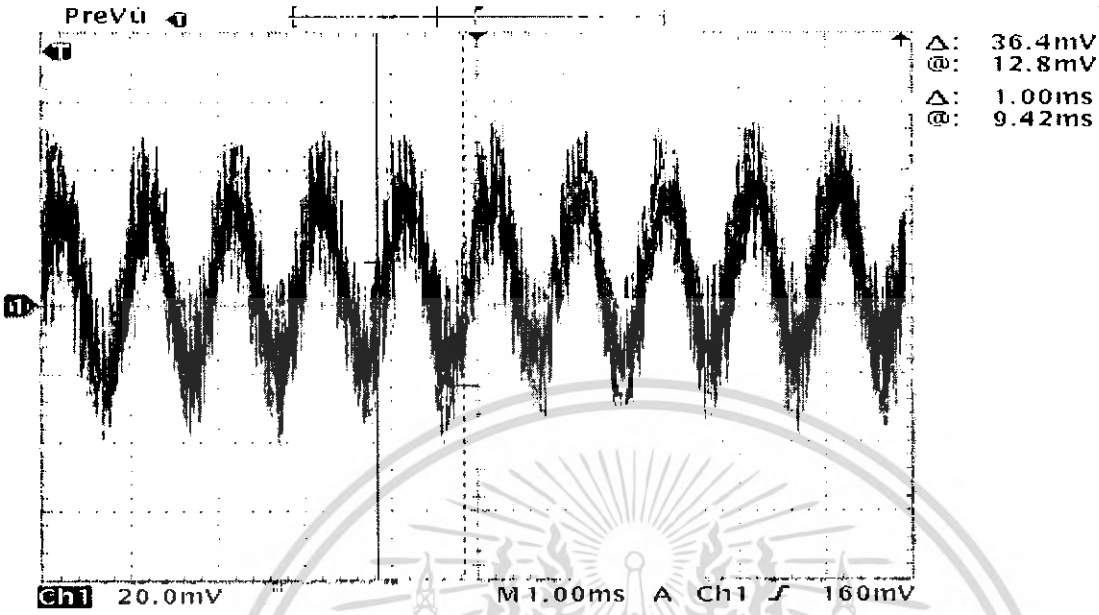
6.3.4 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin20 Hz ได้ผลดังรูปที่ 6.6



รูปที่ 6.6 สัญญาณ Sin 20 Hz เมื่อใช้เครื่องเล่น MP3 Decoder

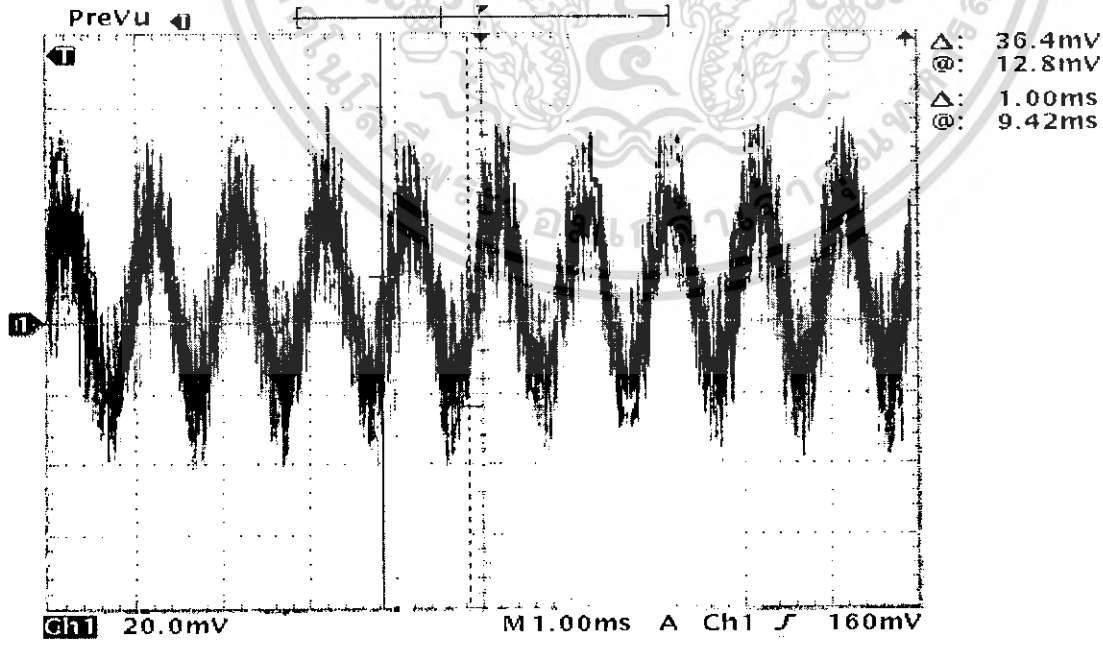
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.5 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 1KHz ได้ผลดังรูปที่ 6.7



รูปที่ 6.7 แสดงสัญญาณ Sin 1KHz

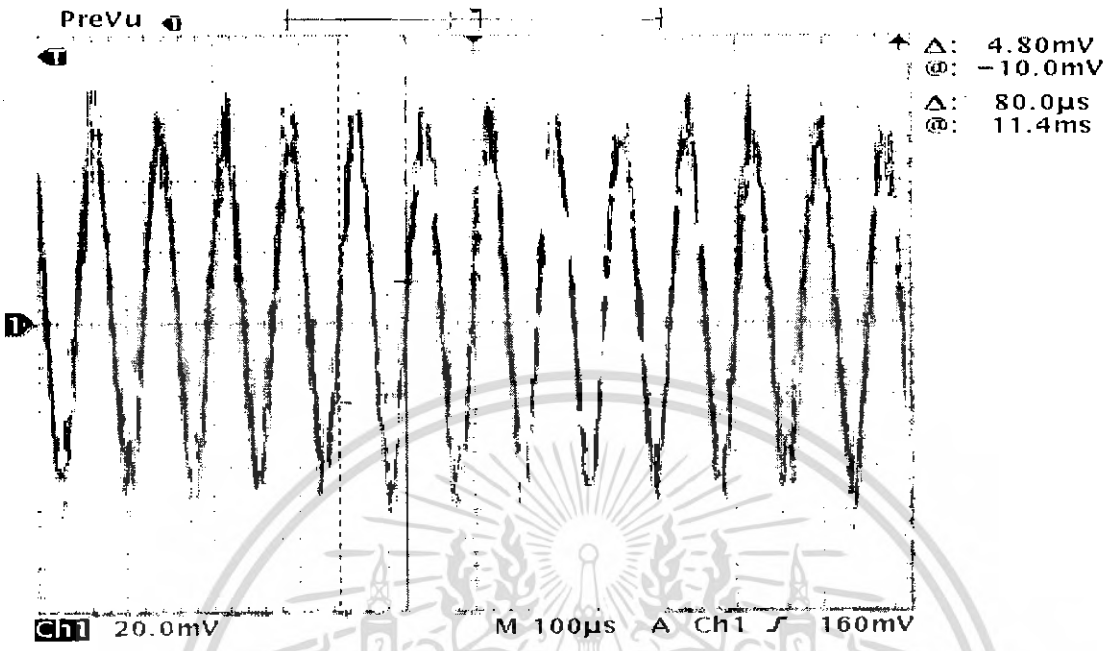
6.3.6 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 1KHz ได้ผลดังรูปที่ 6.8



รูปที่ 6.8 สัญญาณ Sin 1KHz เมื่อใช้เครื่องเล่น MP3 Decoder

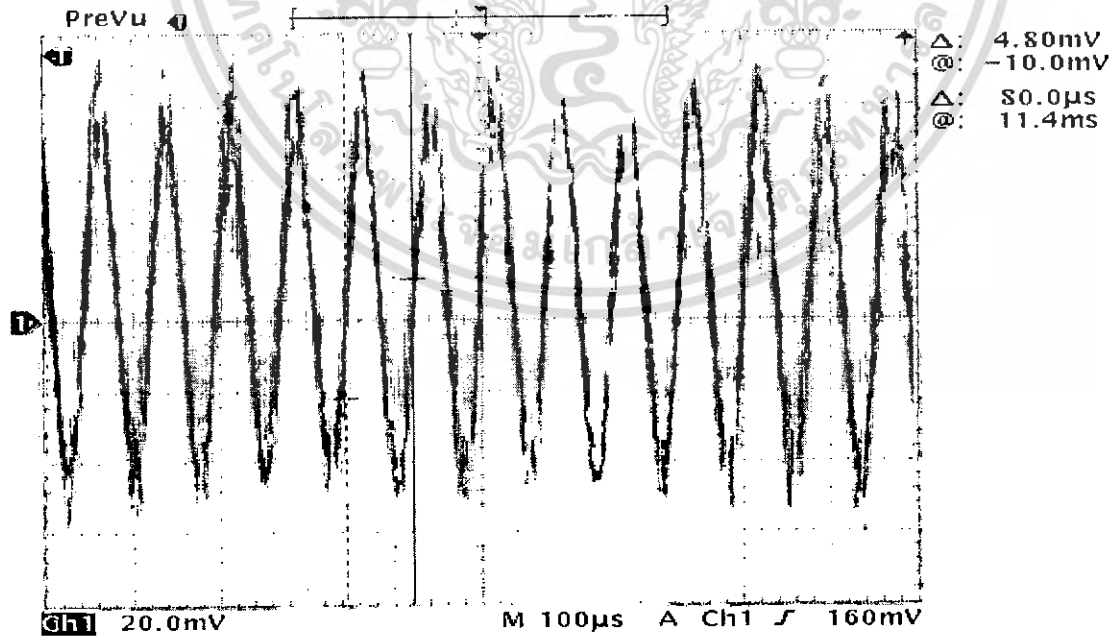
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.7 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 12 KHz ได้ผลดังรูปที่ 6.9



รูปที่ 6.9 แสดงสัญญาณ Sin 12 KHz

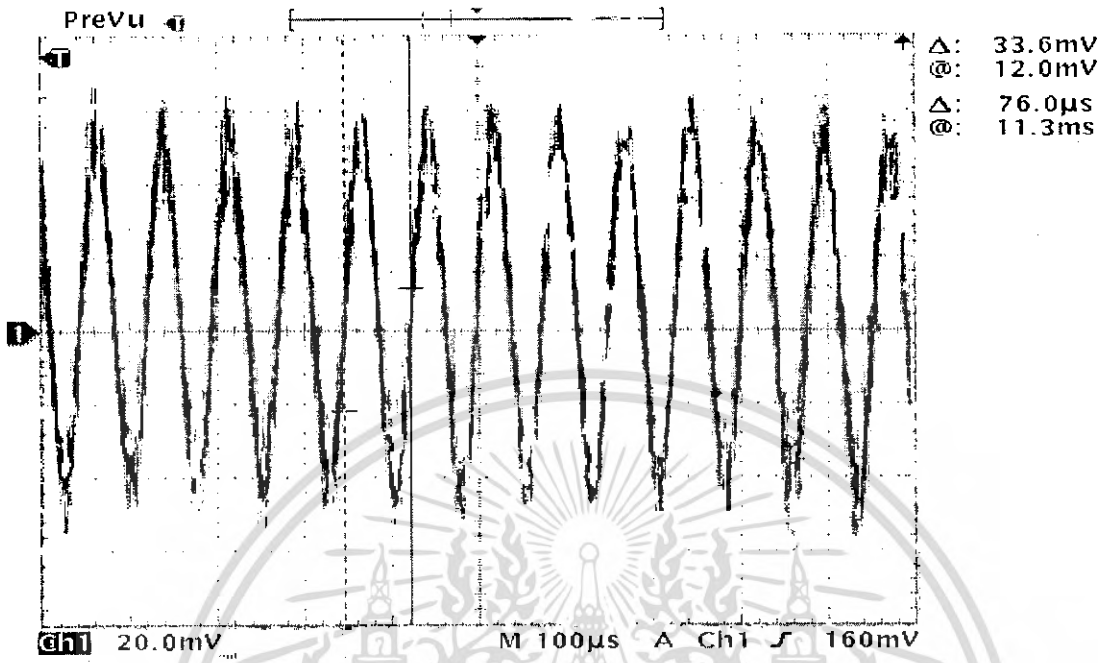
6.3.8 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 12 KHz ได้ผลดังรูปที่ 6.10



รูปที่ 6.10 สัญญาณ Sin 12 KHz เมื่อใช้เครื่องเล่น MP3 Decoder

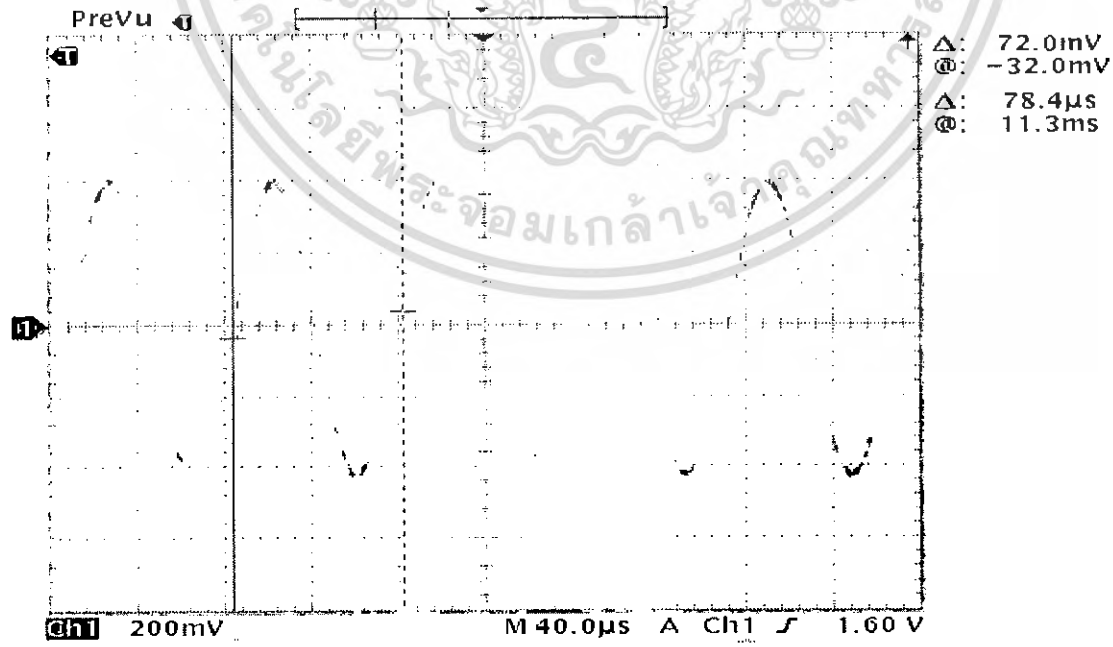
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.9 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin13KHz ได้ผลดังรูปที่ 6.11



รูปที่ 6.11 แสดงสัญญาณ Sin 13 KHz

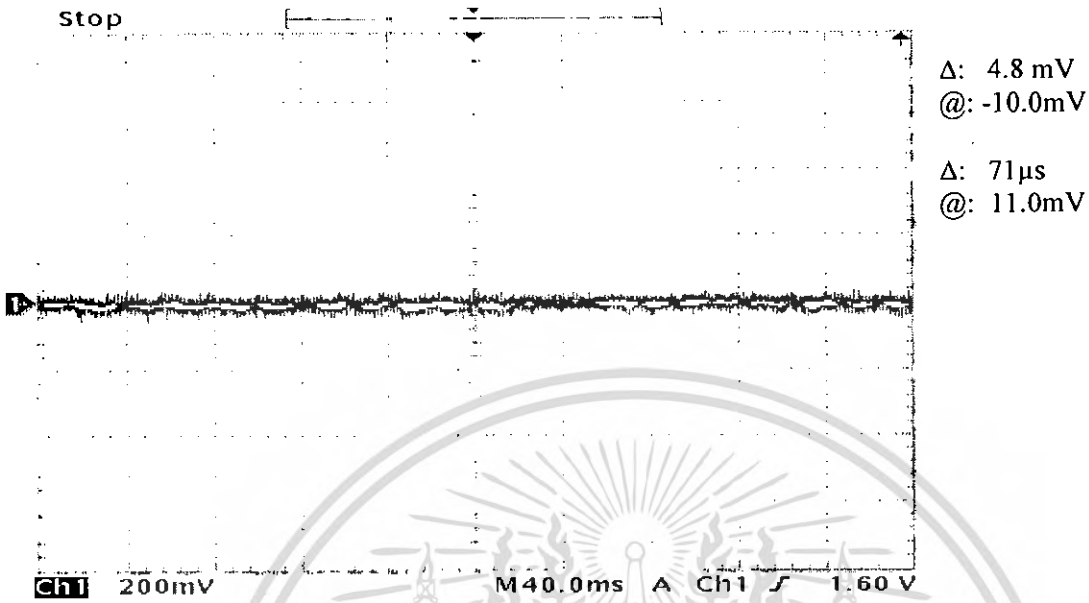
6.3.10 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 13KHz ได้ผลดังรูปที่ 6.12



รูปที่ 6.12 สัญญาณ Sin 13 KHz เมื่อใช้เครื่องเล่น MP3 Decoder

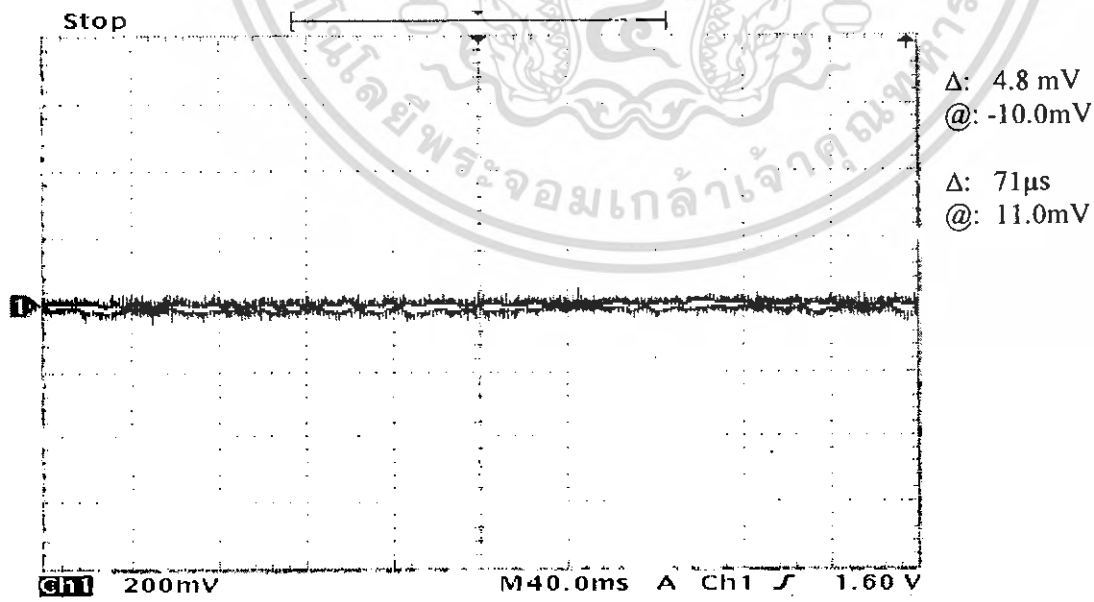
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.11 ใช้โปรแกรมวินโดวส์ Media Player 10 เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin14KHz ได้ผลดังรูปที่ 6.13



รูปที่ 6.13 แสดงสัญญาณ Sin 14 KHz

6.3.12 ใช้เครื่องเล่น MP3 Decoder เล่นไฟล์เอ็มพีสามที่เป็นไฟล์สัญญาณ Sin 14 KHz ได้ผลดังรูปที่ 6.14



รูปที่ 6.14 สัญญาณ Sin 14 KHz เมื่อใช้เครื่องเล่น MP3 Decoder

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่7

สรุปผลการทดสอบ

จากผลการทดสอบ โดยการวัดสัญญาณ Sine ในรูปแบบเอ็มพีสาม ตั้งแต่ 10 Hz – 20 KHz โดยการใช้โปรแกรมวินโดวส์Media Player ในการเล่นไฟล์ เอ็มพีสาม ตั้งแต่ 10 Hz – 20 KHz โดยวัดที่สัญญาณที่เอาท์พุตของคอมพิวเตอร์ เปรียบเทียบกับ การใช้เครื่องเล่นเอ็มพีสามดีโค้ดเดอร์ในการเล่นไฟล์เอ็มพีสาม ตั้งแต่ 10 Hz – 20 KHz จะพบว่าเครื่องเล่นเอ็มพีสามสามารถเล่นไฟล์สัญญาณ Sine ตั้งแต่ 20 Hz – 13 KHz สัญญาณ Sine ที่ต่ำกว่า 20 hz และสูงกว่า 13 KHz เครื่องเล่นไม่สามารถถอดรหัสได้ ทำให้รู้ว่าเครื่องเล่นเอ็มพีสามดีโค้ดเดอร์ สามารถถอดรหัสที่ความถี่ 20 Hz – 13 KHz

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

Souse code ควบคุมการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//=============================================================================
// File name      : Play Sound in dataflash
// CPU            : PSoC CY8C29466 (CYPRESS)
// Chip  Flash    : AT45DB161B (ATMEL)
// Chip MP3       : VS1002D (VLSI)
// SUPPLY         : LM1117T 3.3V
// Sofeware       : PSoC Designer 4.3 (CYPRESS)
// Download HEX : ET_PSOC_ISSP_V1.0 (ETT)
// Programer      : Mr.SEKSARN  TARKONG
// Class          : 3R/2,3
// Department     : Electronic Faculty of Engineering
// University     : King Mongkut's Institute of Technology Ladkrdbang KMITL
//
// PIN I/O
// PSoC          AT45DB161B    PSoC        SW
// P0[1]  <==    BUSY          P0[0]  ==>  PLAY
// P0[2]  ==>    RES#          P1[2]  <==  TX
// P0[3]  ==>    WP#
// P0[4]  <==    SDO
// P0[5]  ==>    SDI
// P0[6]  ==>    SCLK
// P0[7]  ==>    CS#
//
// SPI(PsoC) Mode3
// Clock SPI(PSoc) Max=6MHz Min=1KHz
// Uart (PSoC) VC3 =156 )Buard 19200
// Hyperterminal Moniter
//
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
// Define For VS1002D SCI Register
#define  SCI_MODE                   0x00        // Mode Control
#define  SCI_STATUS                 0x01        // Status
#define  SCI_BASS                   0x02        // Built-In Bass Enhancer
#define  SCI_CLOCKF                 0x03        // Clock Frequency + Double
#define  SCI_DECODE_TIME            0x04        // Decode Time in Second
#define  SCI_AUDATA                 0x05        // Misc. Audio Data
#define  SCI_WRAM                   0x06        // RAM Write
#define  SCI_WRAMADDR               0x07        // Base Address For RAM Write
#define  SCI_HDAT0                  0x08        // Stream Header Data0
#define  SCI_HDAT1                  0x09        // Stream Header Data1
#define  SCI_AIADDR                 0x0A        // Start Address of Application.
#define  SCI_VOL                    0x0B        // Volume Control
#define  SCI_AICTRL0                0x0C        // Application Control Register. 0
#define  SCI_AICTRL1        0x0D                // Application Control Register. 1
#define  SCI_AICTRL2        0x0E                // Application Control Register. 2
#define  SCI_AICTRL3        0x0F                // Application Control Register. 3
//
// Define For dataflash Command Register
#define  Continuous_Array_Read              0xE8        // SPI Mode 3
#define  Main_Memory_Page_Read              0xD2        // SPI Mode 3
#define  Buffer1_Read                       0xD4        // SPI Mode 3
#define  Buffer2_Read                       0xD6        // SPI Mode 3
#define  Status_Register_Read               0xD7        // SPI Mode 3
#define  Buffer_1_Write                     0x84        // Any SPI Mode
#define  Buffer_2_Write                     0x87        // Any SPI Mode
#define  Buffer1_to_Main_Memory_Erase       0x83        // Any SPI Mode
#define  Buffer2_to_Main_Memory_Erase       0x86        // Any SPI Mode
#define  Buffer1_to_Main_Memory_No_Erase    0x88        // Any SPI Mode
#define  Buffer2_to_Main_Memory_No_Erase    0x89        // Any SPI Mode
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define  Page_Erase                    0x81              // Any SPI Mode
#define  Block_Erase                   0x50              // Any SPI Mode
#define  Main_Memory_Page_Buffer1      0x82              // Any SPI Mode
#define  Main_Memory_Page_Buffer2      0x85              // Any SPI Mode
//
// Define VS1002D PIN IO
#define  VS1002D_DREQ_MASK             0x02          // DREQ  = P1[1] (00000010)
#define  VS1002D_XCS_HIGH()            PRT1DR |= 0b10000000        // XCS#(P1[7]) = '1'
#define  VS1002D_XCS_LOW()             PRT1DR &= 0b01111111        // XCS#(P1[7]) = '0'
#define  VS1002D_RES_HIGH()            PRT1DR |= 0b00001000        // RES#(P1[3]) = '1'
#define  VS1002D_RES_LOW()             PRT1DR &= 0b11110111        // RES#(P1[3]) = '0'
#define  VS1002D_XDCS_HIGH()           PRT1DR |= 0b00010000        // XDCS(P1[4]) = '1'
#define  VS1002D_XDCS_LOW()            PRT1DR &= 0b11101111        // XDCS(P1[4]) = '0'
#define  VS1002D_BSYNC_HIGH()          PRT1DR |= 0b00010000        // BSYNC(P1[4]) = '1'
#define  VS1002D_BSYNC_LOW()           PRT1DR &= 0b11101111        // BSYNC(P1[4]) = '0'
// Define AT45DB16 PIN IO
#define  dataflash_busy_MASK  0x02                          // busy  = P0[1](00000010)
#define  AT45DB16_CS_low               PRT0DR &= 0b01111111   // CS#low(P0[7]) = '0'
#define  AT45DB16_CS_hight             PRT0DR |= 0b10000000   // CS#hight(P0[7]) = '1'
#define  AT45DB16_reset_low            PRT0DR &= 0b11111011   // CS#low(P0[2]) = '0'
#define  AT45DB16_reset_hight          PRT0DR |= 0b00000100   // CS#hight(P0[2]) = '1'
#define  AT45DB16_wp_low               PRT0DR &= 0b11110111   // CS#low(P0[3]) = '0'
#define  AT45DB16_wp_hight             PRT0DR |= 0b00001000   // CS#hight(P0[3]) = '1'
// Define AT45DB16 PIN IO
#define  swlow                         PRT0DR &= 0b1111111    // CS#low(P2[7]) = '0'
// Define VS1002D Funetion
void VS1002D_Initial(void);
void VS1002D_HW_Reset(void);
void VS1002D_SW_Reset(void);
void VS1002D_Setup_Volume(unsigned char Left,unsigned char Right);
void VS1002D_Write_Zero(unsigned int count);
void VS1002D_Write_SCI(unsigned char SCI_Reg,unsigned int SCI_Data);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void SPI_WriteByte(unsigned char DataByte);
char VS1002D_Wait_DREQ_Ready(void);
void VS1002D_Write_SDI(unsigned char SDI_Data);
void delay(unsigned long int);                          // Delay Function
// Define DataFlash Function
void dataflash_Initial(void);
void play(void) ;
void SPI_WriteByte1(unsigned char DataByte);
unsigned char SPI_ReadByte(void);
char dataflash_Wait_busy_Ready(void);
void main()
{
  // Insert your main routine code here.
        long Sound_Pointer;
        int i;
        VS1002D_Initial();                              // Initial MP3 Player
        dataflash_Initial();
        swlow;
        while(1)
  { if(PRT0DR & 0x01)
        {   // Sound
              play();
              delay(10000);                             // Delay Before Next Sound
              delay(10000);
              } } }
//
// Initial VS1002D Function
void VS1002D_Initial(void)                              // Initial VS1002D
{
 unsigned char dummy;                                   // Initial GPIO Signal Interface VS1002D
 PRT0DR  = 0xFF;
 VS1002D_RES_HIGH();                                    // RES# = High
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 VS1002D_XCS_HIGH();                                    // XCS# = High
 VS1002D_XDCS_HIGH();                                   // XDCS = High
  SPIM_1_Start( SPIM_1_SPIM_MODE_0 | SPIM_1_SPIM_MSB_FIRST );
// Initial VS1002D Function
 VS1002D_HW_Reset();                                    // VS1002D Hardware Reset
 VS1002D_SW_Reset();                                    // VS1002D Software Reset
 delay(1000);   }
//
// VS1002D Hardware Reset
//
void VS1002D_HW_Reset(void)                             // Active VS1002D Hardware Reset
{ VS1002D_RES_LOW();                                    // Active  VS1002D RES# Pin
 delay(10000);
 VS1002D_RES_HIGH();                                    // Release VS1002D RES# Pin
 delay(10000);
}
// VS1002D Hardware Reset
//
void VS1002D_SW_Reset(void)                             // VS1002D Software Reset
{
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Write_SCI(SCI_MODE,0x0804);                    // Active Software Reset + SPI New
Mode                                                    // Wait 100mS After Reset Complete
 delay(10000);                                          // 100mS Delay
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Write_SCI(SCI_CLOCKF,0x9800);                  // VS1002 Clock = 12.288 MHz +
Double Clock
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Write_SCI(SCI_AUDATA,48000);                   // Set Sampling Rate 8KHz
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Write_Zero(2048);                              // Reset All Memory (2048)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Setup_Volume(0,0);                    // Set Volume = Maximum
}
//
// VS1002D Setup Volume
// Left,Right = 0...255
void VS1002D_Setup_Volume(unsigned char Left,unsigned char Right)
{
 unsigned int Regval;
 Regval  = Right;                              // Get Right Volume
 Regval  += (unsigned int)Left<<8 ;            // Get Left Volume
 while(VS1002D_Wait_DREQ_Ready());
 VS1002D_Write_SCI(SCI_VOL,Regval);            // Setup VS1002D Volume
}
//
// Send Zero to VS1002D
//
void VS1002D_Write_Zero(unsigned int count)    // Write Zero to VS1002D
{
 VS1002D_XDCS_LOW();
 do { VS1002D_Write_SDI(0x00);
    count--;
        }
   while(count);
 VS1002D_XDCS_HIGH();
}
//
// VS1002 Write Command
//
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void VS1002D_Write_SCI(unsigned char SCI_Reg,unsigned int SCI_Data)
{
 VS1002D_XCS_LOW();                    // Enable VS1002D SPI
 SPI_WriteByte(0x02);                  // SCI Write Command Code
 SPI_WriteByte(SCI_Reg);               // SCI Address For Write
 SPI_WriteByte((SCI_Data >> 8)& 0xFF); // Data Byte High
 SPI_WriteByte(SCI_Data & 0xFF);       // Data Byte Low
 VS1002D_XCS_HIGH();                   // Disable VS1002D SPI
}
//
// Write 1 Byte to SPI0
//
void SPI_WriteByte(unsigned char DataByte)      // Write Byte to SPI0
{
   unsigned char Dummy;
   SPIM_1_SendTxData( DataByte );
   while( ! (SPIM_1_bReadStatus() & SPIM_1_SPIM_SPI_COMPLETE) ) ;
   Dummy = SPIM_1_bReadRxData();
}
//
// Wait VS1002 DREQ Ready
// Wait DREQ = 1(FIFO OK)
//
char VS1002D_Wait_DREQ_Ready(void)              // Get VS1002D DREQ Pin Status
{
 unsigned char DREQ_Status;                     // DREQ Status Read
 DREQ_Status = PRT1DR;                          // Read DREQ Signal Status
 DREQ_Status &= VS1002D_DREQ_MASK;              // Verify DREQ Pin Status
 if(DREQ_Status == VS1002D_DREQ_MASK)           // Check DREQ Pin Status
 {
  return 0;                                     // VS1002D Busy Status
 }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 else
 {
  return 1;                                   // VS1002D Ready Status
 }
}
//
// VS1002D Write Data
//
void VS1002D_Write_SDI(unsigned char SDI_Data)
{
 SPI_WriteByte(SDI_Data);                     // Send 1 Byte to SDI(VS1002D)
}
// Long Delay Time Function(1..4294967295)
void delay(unsigned long int i)
{
 while(i > 0) {i--;}                          // Loop Decrease Counter
 return;
}
// Initial System Function
//
void dataflash_Initial(void)                  // Initial dataflash
{
 unsigned char dummy;
 SPIM_2_Start( SPIM_2_SPIM_MODE_3 | SPIM_2_SPIM_MSB_FIRST );
  // Initial Start Sugnal
 AT45DB16_CS_hight;                           // Disable Chips Select
 AT45DB16_reset_hight;
 AT45DB16_wp_hight;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// Function Read_Page
void play(void)
{
        unsigned int j;
        unsigned int k;
        unsigned int page;
        long Sound_Pointer;
        Sound_Pointer=0;
        page=0;
        while(Sound_Pointer<2050000)                    // Repeat Write "Sound0.c"
File
    {
        while(dataflash_Wait_busy_Ready());
        AT45DB16_CS_low;
        SPI_WriteByte1(0xD2);
        SPI_WriteByte1(page>>6);                        // 00+PA[11..6]
        SPI_WriteByte1(page<<2);                        // PA[5..0]+BA[8]
        SPI_WriteByte1(0x00);                           // BA[7..0]
        SPI_WriteByte1(0xFF);
        // 32 Bit Dummy Data
        SPI_WriteByte1(0xFF);
        SPI_WriteByte1(0xFF);
        SPI_WriteByte1(0xFF);
        VS1002D_XDCS_LOW();
        for(j=0;j<16;j++)
    {
        while(VS1002D_Wait_DREQ_Ready());               // Wait VS1002D Ready

        {
            for (k=0; k<32; k++)
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                VS1002D_Write_SDI(SPI_ReadByte());          // Write Data to
SPI0(VS1002D)
                Sound_Pointer++;
                if(Sound_Pointer==2050000)
                break;

                }
                }
                }
                VS1002D_XDCS_HIGH();
                AT45DB16_CS_hight;
                page++;  }
}
// Write 1 Byte to SPI0
void SPI_WriteByte1(unsigned char DataByte)                    // Write Byte to SPI0
{
   unsigned char Dummy;
   SPIM_2_SendTxData( DataByte );
   while( ! (SPIM_2_bReadStatus() & SPIM_2_SPIM_SPI_COMPLETE) ) ;
   Dummy = SPIM_2_bReadRxData();
}                                                              // Read 1 Byte From SPI0
unsigned char SPI_ReadByte(void)                               // Read Byte From SPI
{                                                              //unsigned char DataByte;
SPIM_2_SendTxData(0xff);
while( ! (SPIM_2_bReadStatus() & SPIM_2_SPIM_RX_BUFFER_FULL) ) ;
 return SPIM_2_bReadRxData();
}
// Wait dataflash Busy Ready
// Wait busy = 1(FIFO OK)
char dataflash_Wait_busy_Ready(void)                           // Get VS1002D DREQ Pin
Status
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

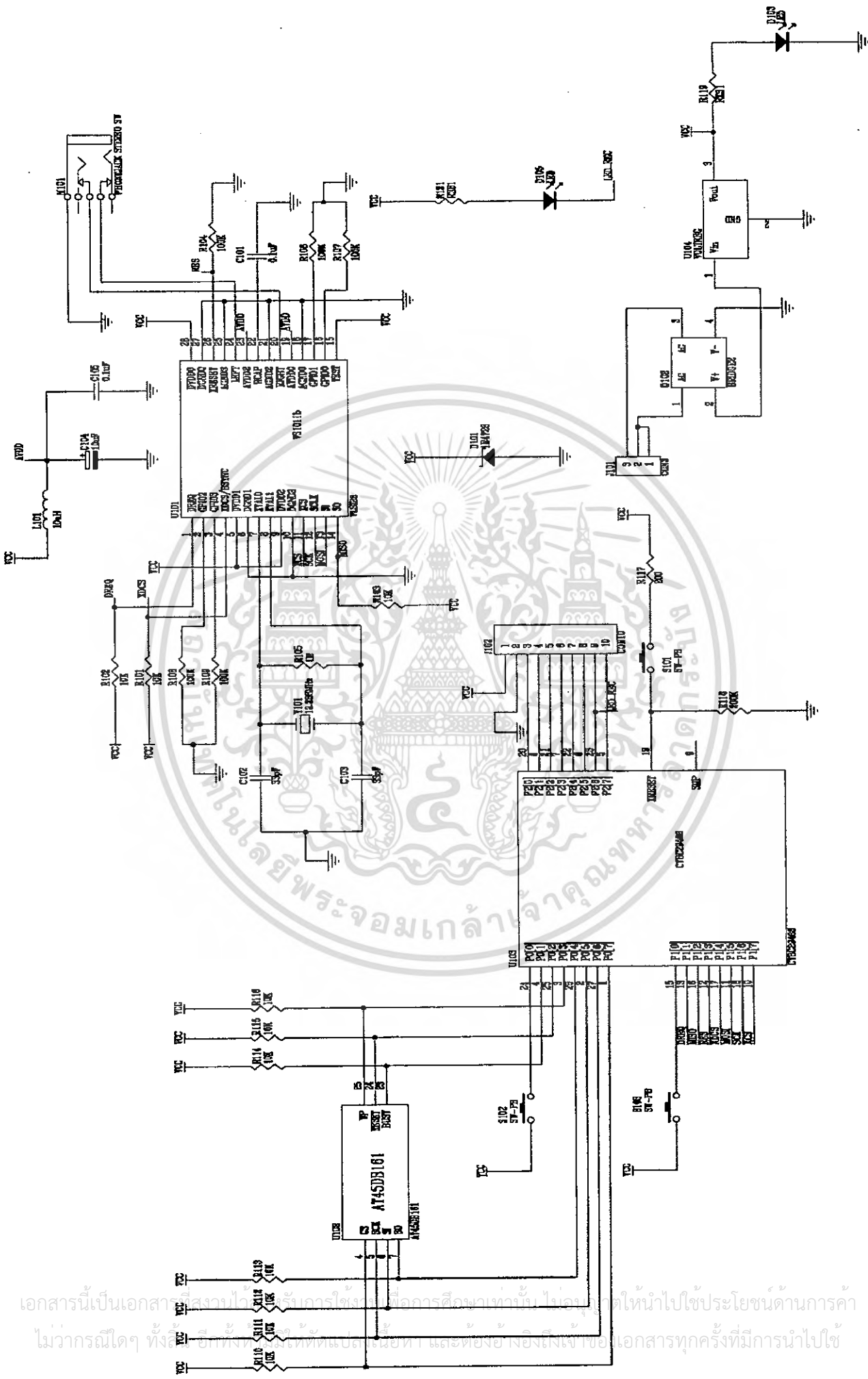
```
 unsigned char busy_Status;                          // DREQ Status Read
 busy_Status = PRT0DR;                               // Read DREQ Signal Status
 busy_Status &= dataflash_busy_MASK;                 // Verify DREQ Pin Status
 if(busy_Status == dataflash_busy_MASK)              // Check DREQ Pin Status
 {
  return 0;                                          // VS1002D Busy Status
 }
 else
 {
  return 1;                                          // VS1002D Ready Status
 }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

วงจรการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้