ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เรื่อง การสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริงโดยใช้การประมวลผลสัญญาณเชิงเลขหลายอัตรา

โดย นายอภิชิต กระจ่างเย่า

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาอุปกรณ์การแพทย์

_____ คณบดีบัณฑิตวิทยาลัย

(อาจารย์ ดร.มงคล หวังสถิตย์วงษ์)

13 ตุลาคม 2549

คณะกรรมการสอบวิทยานิพนธ์

_____ ประธานกรรมการ

(รองศาสตราจารย์จิระศักดิ์ ชาญวุฒิธรรม)

_____ กรรมการ

(ผู้ช่วยศาสตราจารย์สุรพันธ์ ยิ้มมั่น)

_____ กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.พยุง มีสัจ)

_____ กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.พิพัฒน์ พรหมมี)

การสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยใช้การประมวลผลสัญญาณเชิงเลขหลายอัตรา

นายอภิชิต กระจ่างเย่า

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาอุปกรณ์การแพทย์ ภาควิชาฟิสิกส์อุตสาหกรรมและอุปกรณ์การแพทย์
บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2549

ISBN 974-19-0855-5

ชื่อ : นายอภิชิต กระจ่างเย่า
ชื่อวิทยานิพนธ์ : การสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง
โดยใช้การประมวลผลสัญญาณเชิงเลขหลายอัตรา
สาขาวิชา : อุปกรณ์การแพทย์
สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ที่ปรึกษาวิทยานิพนธ์ : รองศาสตราจารย์จิระศักดิ์ ชาญวุฒิธรรม
ผู้ช่วยศาสตราจารย์สุรพันธ์ ยิ้มมั่น
ปีการศึกษา : 2549

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอการสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยใช้หลักการของการประมวลผลสัญญาณเชิงเลขหลายอัตรา การสร้างจริงทำบนตัวประมวลผลสัญญาณดิจิตอล TMS320C31 ในการทดลองจะทำการลดขนาดสัญญาณคลื่นไฟฟ้าหัวใจ 2 เท่า 4 เท่า และ 8 เท่า ผลการทดลองพบว่าสัญญาณคลื่นไฟฟ้าหัวใจที่ได้รับการบีบอัดและขยาย จะมีค่าเปอร์เซ็นต์ความผิดพลาด (Percent Root-Mean Square Different : PRD) ต่ำ ซึ่งจากผลการทดลองสามารถนำเอาระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจ ไปสร้างเครื่องบันทึกคลื่นไฟฟ้าหัวใจ (ECG Recorder) และระบบการเฝ้าระวังดู (Monitor) ผู้ป่วยระยะไกลได้

(วิทยานิพนธ์มีจำนวนทั้งสิ้น 137 หน้า)

คำสำคัญ : คลื่นไฟฟ้าหัวใจ, การประมวลผลสัญญาณดิจิตอล

_____อาจารย์ที่ปรึกษาวิทยานิพนธ์

Name : Mr.Apichit Krajangyao
Thesis Title : Real Time ECG Compression/Decompression System with Multirate Digital Signal Processing
Major Field : Medical Instrumentation
King Mongkut's Institute of Technology North Bangkok
Thesis Advisors : Associate Professor Jirasak Chanwutitum
Assistant Professor Surapan Yimman
Academic Year : 2006

Abstract

This thesis introduces the real-time ECG compression/decompression system using multirate digital signal processing implemented on TMS320C31 DSP board. The designed ECG compress/decompress system is able to compress the ECG signals at ratio of 2 times, 4 times, and 8 times by decimation technique. The compressed signal is then recovered back by decompression and interpolation techniques. The consistency of this ECG compress/decompress system is shown by low PRDs (Percent Root Mean Square Differences) compared between the decompressed signal and the original signal. The reliable of estimated signal of the ECG compression/decompression system make it suitable for the application in telemedicine and home based patient monitoring.

(Total 137 pages)

Keywords : ECG (Electrocardiogram),DSP (Digital Signal Processing)

_____Advisor

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งจากบุคคลหลายท่าน ผู้วิจัยขอขอบพระคุณ รองศาสตราจารย์จิระศักดิ์ ชาญวุฒิธรรม ผู้ช่วยศาสตราจารย์สุรพันธ์ ยิ้มมั่น อาจารย์พยุง เดชอยู่ ที่ได้ให้แนวคิดและแนวปฏิบัติในการทำงานวิจัยนี้ ขอขอบพระคุณอาจารย์ภาควิชาฟิสิกส์อุตสาหกรรมและอุปกรณ์การแพทย์ทุกท่าน ที่ได้ให้ความรู้ความเข้าใจในระหว่างการศึกษา ขอขอบคุณ อาจารย์ชูสิทธิ์ ประดับเพ็ชร์ ที่ให้โอกาสในการเข้ามาศึกษาในระหว่างที่ทำงานที่มหาวิทยาลัยราชภัฏพระนครศรีอยุธยา และช่วยหาเอกสารประกอบในการทำงานวิจัยนี้

ขอขอบคุณ คุณศุภชัย ยิ่งเจริญ และ เพื่อนๆ อุปกรณ์การแพทย์รุ่นที่ 5 และพี่น้อง อุปกรณ์การแพทย์ในรุ่นอื่นๆ ที่มีส่วนช่วยให้งานวิจัยนี้สำเร็จลงได้

ขอขอบคุณบัณฑิตวิทยาลัยที่ช่วยในการดำเนินเรื่องต่างๆ และการให้คำแนะนำขั้นตอนของการทำงานวิจัยนี้

ขอบคุณ คุณชวลิต อินทรพงษ์ ที่เอื้อเฟื้อสถานที่พักและเครื่องคอมพิวเตอร์ ในช่วงระหว่างการทำวิทยานิพนธ์

ขอบคุณ คุณสมพิศ รักซ้อน ที่คอยให้กำลังใจและความรู้สึกที่ดีๆมาตลอด และเป็นแรงผลักดันให้งานวิจัยนี้สำเร็จไปได้ด้วยดี

ขอบคุณ พี่มนิสรา กระจ่างเย่า(พี่ปุ๋ย) และ หลวงพี่สุระ กระจ่างเย่า (พระโป้ง) ที่เป็นกำลังใจและคอยดูแลน้องชายคนนี้เสมอมา

สุดท้ายนี้ขอขอบพระคุณอย่างสูงสำหรับ คุณพ่อสุชีพ คุณแม่อัมรา กระจ่างเย่า ที่ให้ความรัก กำลังใจ ความเข้าใจ และ โอกาสดีๆ ในการศึกษา โอกาสดีๆในชีวิต และให้การสนับสนุนในทุกๆด้านตลอดมา

อภิชิต กระจ่างเย่า

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาของงานวิจัย

การประมวลผลสัญญาณดิจิตอลเจริญก้าวหน้าขึ้นเป็นอย่างมาก และนำไปใช้กับงานทางด้านต่างๆมากมาย งานทางด้านการแพทย์ก็เป็นงานหนึ่งที่ได้นำเอาระบบประมวลผลสัญญาณดิจิตอลมาประยุกต์ใช้งานตัวอย่างเช่น Bio-Signal Analysis, Medical Image Processing [1] เป็นต้น นอกจากการประยุกต์ใช้งานดังกล่าวแล้ว งานทางด้าน Data Recorder, Data Compression/ Decompression [1, 2, 3] เป็นงานที่มีประโยชน์และสามารถนำไปประยุกต์ใช้งานทางด้านการแพทย์ได้

สัญญาณคลื่นไฟฟ้าหัวใจ ECG เป็นสิ่งที่สำคัญในการวินิจฉัยอาการของผู้ป่วยจึงได้มีการสร้างเครื่องบันทึกคลื่นไฟฟ้าหัวใจขึ้นเพื่อใช้เก็บสัญญาณคลื่นไฟฟ้าหัวใจ เพื่อให้แพทย์ใช้วิเคราะห์อาการของผู้ป่วย และเป็นประโยชน์อย่างมากกับผู้ป่วยที่ต้องรับการผ่าตัดหรืออยู่ในภาวะวิกฤต ซึ่งแพทย์จำเป็นจะต้องสังเกตการเต้นของหัวใจอย่างต่อเนื่องตลอดเวลา แต่ปัญหาสำคัญอย่างหนึ่งของการสร้างเครื่องบันทึกคลื่นไฟฟ้าหัวใจ ก็คือหน่วยจำที่ใช้เก็บข้อมูลต้องมีจำนวนมากเพื่อรองรับการบันทึกสัญญาณคลื่นไฟฟ้าหัวใจจากผู้ป่วยอย่างต่อเนื่องตลอดเวลา ทำให้สิ้นเปลืองค่าใช้จ่ายในการเพิ่มหน่วยความจำ จึงได้มีการแก้ปัญหาโดยการลดขนาด (Compression) สัญญาณคลื่นไฟฟ้าหัวใจลง ซึ่งในการลดขนาดสัญญาณคลื่นไฟฟ้าหัวใจนี้กระทำได้หลายวิธี เช่น Wavelet transform, Long-term prediction [3] สำหรับวิทยานิพนธ์นี้จะนำเสนอการออกแบบและสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยหลักการประมวลผลสัญญาณเชิงเลขหลายอัตรา [4, 5, 6, 7] โดยจะออกแบบระบบบนโปรแกรม Matlab และจะนำระบบไปทำการสร้างจริงบนตัวประมวลผลสัญญาณดิจิตอล TMS320C31 [12, 13, 14] และหาค่าเปอร์เซ็นต์ความผิดพลาดเป็น (Percent Root mean square Different : PRD) [1, 2] ของสัญญาณ Output เปรียบเทียบกับสัญญาณ Input

1.2 วัตถุประสงค์

1.2.1 สร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยใช้การประมวลผลสัญญาณเชิงเลขหลายอัตรา เพื่อให้ได้เปอร์เซ็นต์ความผิดพลาดน้อยที่สุดและมีขนาดของข้อมูลน้อยที่สุด

1.2.2 นำระบบที่ได้ไปสร้างจริงบนตัวประมวลผลสัญญาณดิจิตอล TMS320C31

1.2.3 นำระบบที่ได้ไปเปรียบเทียบวิธีการบีบอัดข้อมูลวิธีอื่นเพื่อดูประสิทธิภาพของระบบ

1.3 ขอบเขตของการวิจัย

1.3.1 ออกแบบระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยการประมวลผลสัญญาณเชิงเลขหลายอัตรา

1.3.2 นำระบบที่ได้ไปสร้างบนตัวประมวลผลสัญญาณดิจิตอล TMS320C31

1.3.2.1 ทำการลดขนาดคลื่นไฟฟ้าหัวใจด้วยอัตรา 2 เท่า

1.3.2.2 ทำการลดขนาดคลื่นไฟฟ้าหัวใจด้วยอัตรา 4 เท่า

1.3.2.3 ทำการลดขนาดคลื่นไฟฟ้าหัวใจด้วยอัตรา 8 เท่า

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้ระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจ แบบเวลาจริงที่มีความผิดพลาดต่ำ

1.4.2 ได้ขนาดข้อมูลที่มีขนาดเล็กทำให้ใช้เนื้อที่ในการเก็บข้อมูลน้อย

1.4.3 สามารถนำระบบที่สร้างไปใช้ในการ Monitor ผู้ป่วยระยะไกลได้

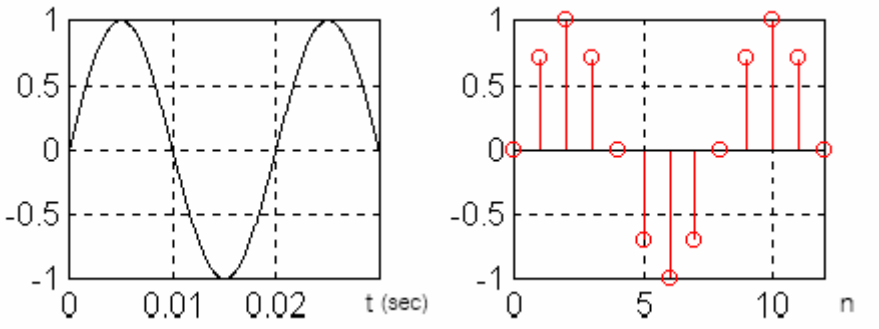
บทที่ 2
ทฤษฎีและหลักการ

2.1 การประมวลผลสัญญาณดิจิตอลเบื้องต้น [8]

2.1.1 สัญญาณต่อเนื่องกับสัญญาณไม่ต่อเนื่อง

สัญญาณต่อเนื่องทางเวลา คือสัญญาณที่พบเห็นในชีวิตประจำวันทั่วๆไป ยกตัวอย่างสัญญาณที่เห็นบนหน้าจอออสซิโลสโคป เช่น สัญญาณเสียง, สัญญาณไฟสลับ 50 Hz, และสัญญาณอื่นๆ ถ้าแทนสัญญาณด้วยสัญลักษณ์ x และแทนเวลาด้วยสัญลักษณ์ t จะกล่าวว่า x เป็นฟังก์ชั่นของ t หรือ x มีค่าที่เวลา t ใดๆ เขียนแทนสัญลักษณ์ได้ว่า $x(t)$ ซึ่งเป็นฟังก์ชั่นที่ต่อเนื่องสัญญาณต่อเนื่องนี้เรียกอีกอย่างหนึ่งได้ว่าสัญญาณแอนะลอก (Analog Signal)

สัญญาณไม่ต่อเนื่องเป็นสัญญาณที่มีค่าเพียงบางจุดของเวลา โดยทั่วไปเกิดจากการสุ่มสัญญาณต่อเนื่องด้วยคาบเวลาของการสุ่มคงที่ จะใช้สัญลักษณ์ n แทนเวลาแบบไม่ต่อเนื่อง โดย n เป็นตัวแปรที่มีค่าเป็นจำนวนเต็มเท่านั้น คือ $n = \ldots,-2, -1, 0, 1, 2, 3,\ldots$ และสัญญาณไม่ต่อเนื่องจะเป็นฟังก์ชันของ n ดังนั้นจะเขียนแทนสัญญาณนั้นได้ว่า $x(n)$



ก) สัญญาณต่อเนื่อง ข) สัญญาณไม่ต่อเนื่อง

ภาพที่ 2-1 สัญญาณต่อเนื่อง และสัญญาณไม่ต่อเนื่อง

ภาพของสัญญาณไม่ต่อเนื่องที่แสดงเปรียบเทียบกับสัญญาณต่อเนื่องดังแสดงในภาพที่ 2-1 เป็นภาพที่แสดงให้เห็นรูปร่างของสัญญาณ โดยในภาพที่ 2-1 (ก) เป็นการแสดงรูปร่างของสัญญาณต่อเนื่อง ซึ่งแสดงให้เห็นได้ตามหน้าจอของเครื่องออสซิโลสโคป เป็นต้น ส่วนในภาพที่ 2-1 (ข) เป็นการแสดงให้เห็นรูปร่างของสัญญาณไม่ต่อเนื่อง ซึ่งจะมีลักษณะเป็นลำดับค่าหรือลำดับ

ข้อมูลที่ถูกนำมาพอร์ตลงบนกราฟที่มีสเกลที่บอกตำแหน่งของเวลาและค่าของปริมาณข้อมูล โดยข้อมูลเหล่านี้ สามารถนำไปประมวลผลได้ด้วยคอมพิวเตอร์ หรือวงจรทางดิจิตอล ในอดีตมีผู้คิดว่าการประมวลผลสัญญาณดิจิตอลให้ผลลัพธ์เป็นเพียงการประมาณ ของการประมวลผลทางแอนะลอก แต่ด้วยทฤษฎีที่ถูกต้องที่ได้มีการคิดค้นมา ทำให้การประมวลผลทางดิจิตอลให้ผลลัพธ์ที่ถูกต้องและสามารถพิสูจน์ได้ว่าไม่ใช่ค่าประมาณของทางแอนะลอก

2.1.2 ส่วนประกอบในการประมวลผลสัญญาณดิจิตอล

ระบบประมวลผลสัญญาณโดยส่วนใหญ่ แสดงดังในภาพที่ 2-2 ซึ่งประกอบด้วยส่วนต่างๆ ดังนี้

2.1.2.1 วงจรแปลงสัญญาณแอนะลอกเป็นดิจิตอล

สามารถแบ่งได้เป็น 2 กระบวนการย่อย คือ

2.1.2.1.1 วงจรสุ่มสัญญาณ (Sample) สัญญาณขาเข้าของวงจรเป็นสัญญาณแบบแอนะลอก *x(t)* ส่วนสัญญาณขาออกเป็นสัญญาณไม่ต่อเนื่อง *x(n)* พารามิเตอร์วงจรสุ่มสัญญาณนี้คือ ค่าอัตราการสุ่ม (Sampling rate) หรือ ความถี่ในการสุ่ม ใช้สัญลักษณ์แทนว่า f_s ค่านี้เป็นตัวกำหนดว่าวงจรสุ่มจะสุ่มสัญญาณด้วยอัตรากี่ครั้งต่อวินาที หรือกี่เฮิรตซ์ (Hz)

2.1.2.1.2 วงจรแบ่งขั้นสัญญาณ (Quantizer) สัญญาณ *x(n)* ที่ได้จากวงจรสุ่มสัญญาณ ถือว่ามีความละเอียดเต็มที่ในทางขนาด ซึ่งในทางปฏิบัติเมื่อนำไปใช้งานจะต้องลดความละเอียดของ *x(n)* ลงให้สามารถแทนได้ด้วยสัญญาณดิจิตอลที่มีจำนวนบิตจำกัด กระบวนการลดความละเอียดนี้เรียกว่าการแบ่งขั้นของสัญญาณ (Quantization) ความละเอียดที่ได้จากการแบ่งขั้นสัญญาณขึ้นอยู่กับจำนวนบิตที่จะใช้ การแบ่งขั้นสัญญาณทำให้ค่าสัญญาณที่ได้คลาดเคลื่อนไปจาก *x(n)* จริง ซึ่งจะส่งผลเหมือนมีสัญญาณรบกวนเข้ามาในระบบ

วงจรสุ่มสัญญาณรวมกับวงจรแบ่งขั้นสัญญาณ ในทางปฏิบัติคือ ตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล (A/D Converter) ซึ่งจะรวมสองกระบวนการนี้อยู่ในวงจรเดียวกัน โดยทั่วไป จะใช้ตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอลในรูปของวงจรรวมสำเร็จรูป (IC)

ภาพที่ 2-2 ส่วนประกอบในระบบประมวลผลสัญญาณดิจิตอล

2.1.2.2 วงจรประมวลผลสัญญาณ

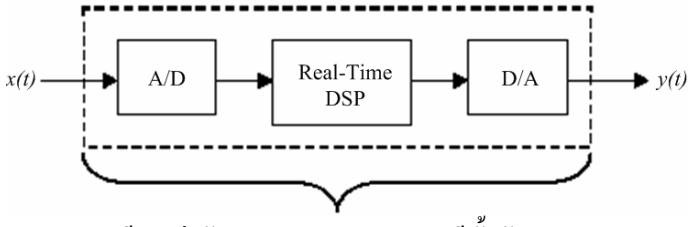
ในส่วนนี้ทำหน้าที่ประมวลผลสัญญาณ *x(n)* เพื่อกระทำผลบางอย่างกับสัญญาณ เช่น เป็นวงจรกรองความถี่บางย่านออกและให้ผลลัพธ์ของการประมวลผลเป็นสัญญาณขาออก *y(n)* วงจรประมวลผลสัญญาณนี้ ถ้าพิจารณาก็คือวงจรคำนวณ กล่าวได้ว่าจะกระทำการคำนวณหาสัญญาณขาออกจากสัญญาณขาเข้า โดยมองเห็นสัญญาณขาเข้าในลักษณะ “ลำดับค่า”

2.1.2.3 วงจรสร้างสัญญาณคืน (Signal Reconstruction)

ในส่วนสุดท้ายของภาพที่ 2-2 วงจรสร้างสัญญาณคืน ใช้ในระบบที่มีสัญญาณขาออกสุดท้ายเป็นสัญญาณต่อเนื่อง (การประมวลผลสัญญาณ บางอย่าง ต้องการสัญญาณขาออกเป็นสัญญาณแบบไม่ต่อเนื่อง จึงไม่จำเป็นจะต้องมีในส่วนนี้) โดยทำหน้าที่แปลงสัญญาณไม่ต่อเนื่อง *y(n)* ให้กลับเป็นสัญญาณต่อเนื่อง *y(t)* ซึ่งเป็นสัญญาณขาออกสุดท้ายของระบบ วงจรประเภทนี้คือ วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก (D/A Converter) ซึ่งมีรูปแบบวงจรรวมสำเร็จรูปเช่นกัน

2.1.3 การประมวลผลแบบเวลาจริง กับการเลือกใช้ตัวประมวลผลสัญญาณ

การประมวลผลแบบเวลาจริง (Real-Time Signal Processing) หมายถึง การประมวลผลที่กระทำที่อัตราจริงของสัญญาณขาเข้า และให้สัญญาณขาออกทันกับสัญญาณขาเข้าที่เข้ามา เช่น ในระบบที่มีอัตราการสุ่มของสัญญาณขาเข้า และขาออกเท่ากัน เมื่อมีสัญญาณขาเข้า เข้ามา 1 ค่า ระบบจะต้องประมวลผลให้ได้สัญญาณขาออก 1 ค่าก่อนที่สัญญาณขาเข้าตัวถัดไปจะเข้ามา เป็นต้น การประมวลผลแบบเวลาจริงมีการประยุกต์ใช้งานอย่างมาก และเป็นตัวแทนที่แท้จริงของระบบที่เคยใช้เป็นแบบแอนะลอกดังแสดงในภาพที่ 2-3 อย่างไรก็ตาม ระบบที่มีการประมวลผลแบบเวลาจริง ไม่จำเป็นต้องมีสัญญาณขาเข้า และออกเป็นสัญญาณแอนะลอกทั้งคู่เสมอไป ยกตัวอย่างเช่น การถอดรหัสสัญญาณเสียงที่ถูกบีบอัดข้อมูลมา ในกรณีนี้สัญญาณขาเข้าเป็นสัญญาณดิจิตอล ซึ่งคือข้อมูลเสียงที่บีบอัดมาแล้ว ส่วนสัญญาณขาออก คือสัญญาณเสียงแอนะลอกที่ต้องต่อส่งออกลำโพง ดังนั้น การประมวลผลจะต้องเกิดที่อัตราสุ่มจริงของสัญญาณเสียงขาออก ถือว่าเป็นการประมวลผลแบบเวลาจริง



เทียบเท่ากับวงจรแอนะลอก เพราะมีทั้งสัญญาณขาเข้า และออกเป็นแอนะลอกอย่างทันทีทันใด

ภาพที่ 2-3 การประมวลผลแบบเวลาจริงทำให้ DSP ทำหน้าที่เหมือนเป็นวงจรแอนะลอกได้

ส่วนการประมวลผลแบบไม่เป็นเวลาจริงนั้น ไม่มีข้อบังคับทางด้านเวลาในการประมวลผล ยกตัวอย่างเช่น การจำลองระบบประมวลผลด้วยโปรแกรม Matlab ในคอมพิวเตอร์ ในที่นี้ถือว่าคอมพิวเตอร์เป็นตัวประมวลผล ซึ่งถ้าใช้คอมพิวเตอร์ที่เร็วก็จะได้ผลลัพธ์เร็ว แต่ถ้าใช้คอมพิวเตอร์ที่ช้าก็จะได้ผลลัพธ์ที่ช้า แต่ผลลัพธ์ที่ได้จะไม่มีความแตกต่างกัน ทั้งนี้เพราะการประมวลผลไม่ได้เกิดขึ้นที่อัตราการสุ่มจริงของสัญญาณขาเข้า หรือขาออก ตัวอย่างอีกอันหนึ่ง เช่น การใช้โปรแกรมตบแต่งรูปภาพ (ภาพนิ่ง) เช่นโปรแกรม PhotoShop ซึ่งภาพนิ่งนี้ถือเป็นสัญญาณไม่ต่อเนื่องสองมิติ และโปรแกรมนี้ถือเป็นโปรแกรมที่มีฟังก์ชัน ในการประมวลผลภาพ (Image Processing) เนื่องจากภาพนิ่งไม่มีอัตราการสุ่มของข้อมูลที่เทียบต่อเวลา ดังนั้น การประมวลผลภาพนิ่งจึงถือได้ว่าไม่มีข้อบังคับทางด้านเวลา จึงไม่เป็นการประมวลผลแบบเวลาจริง

การประมวลผลสัญญาณแบบเวลาจริง ทำให้เกิดข้อกำหนดที่สำคัญขึ้นมาต่อการเลือกใช้ตัวประมวลผลสัญญาณ นั้นคือ การที่ต้องมีตัวประมวลผลที่เร็วพอที่จะประมวลผลสัญญาณให้ทันได้ โดยเฉพาะ ถ้าสัญญาณที่ต้องการประมวลผลที่มีอัตราการสุ่มที่สูง หรืออัลกอริธึมที่ใช้มีความซับซ้อนในการคำนวณมาก ก็จำเป็นที่จะต้องใช้ตัวประมวลผลที่มีความเร็วสูงมากยิ่งขึ้น

การเลือกใช้ตัวประมวลผลสัญญาณมีอยู่ 3 ทางเลือกใหญ่ๆดังนี้

2.1.3.1 การเขียนซอฟต์แวร์เพื่อใช้กับคอมพิวเตอร์ หรือใช้กับชิพไมโครโปรเซสเซอร์

ถึงแม้ว่าคอมพิวเตอร์ หรือไมโครโปรเซสเซอร์ไม่ได้ออกแบบมาเฉพาะสำหรับการประมวลผลสัญญาณ แต่สามารถนำมาใช้ได้ในงานที่ต้องการอัตราการประมวลผลไม่มากนัก หรือในการประมวลผลแบบไม่เป็นเวลาจริง อย่างไรก็ตาม ปัจจุบันคอมพิวเตอร์ส่วนบุคคลมีความเร็วสูงมาก จนสามารถนำมาใช้ทำการประมวลผลแบบเวลาจริงหลายๆอย่างได้ ตัวอย่างที่เห็นได้ชัด เช่นการถอดรหัสของเสียงหรือวีดีโอที่ถูกบีบอัดข้อมูลมาด้วยมาตราฐาน MPEG ซึ่งแต่ก่อนต้องใช้ฮาร์ดแวร์พิเศษในการถอดรหัส แต่ปัจจุบันใช้เพียงซอฟท์แวร์ก็สามารถทำได้แล้ว โดยอาศัย CPU ที่มีความเร็วสูงขึ้น

2.1.3.2 การใช้ซอฟต์แวร์ร่วมกับชิพ DSP

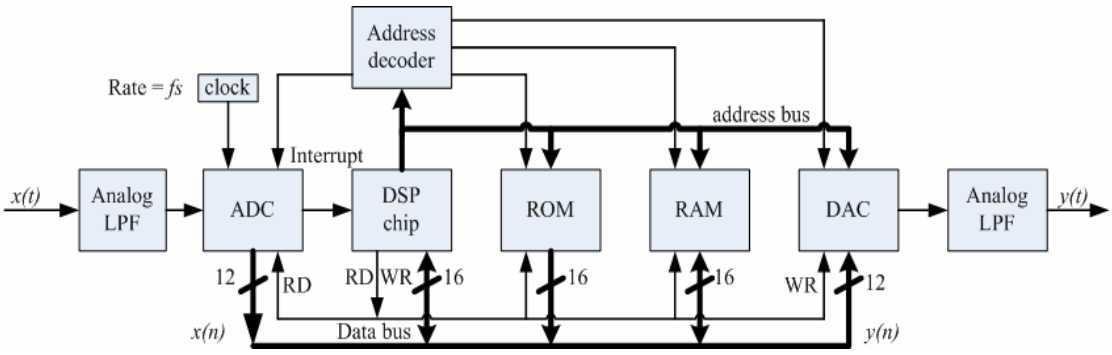
ชิพ DSP เป็นชื่อย่อของชิพประมวลผลสัญญาณ (Digital Signal Processor) คือไมโครโปรเซสเซอร์ที่ถูกออกแบบมาสำหรับงานประมวลผลสัญญาณแบบเวลาจริงโดยเฉพาะ โดยไมโครโปรเซสเซอร์ประเภทนี้จะมีสถาปัตยกรรมที่เอื้ออำนวยต่อการคำนวณ และการโอนถ่ายข้อมูลที่มีประสิทธิภาพ และความเร็วสูง เช่น การมีคำสั่งพิเศษในการคูณ, การบวกสะสม, หรือการอ้างข้อมูลแบบ Circular buffer เป็นต้น บางชนิดยังสามารถทำการประมวลผลหลายๆ ส่วนได้พร้อมกันในตัวเดียว (Multi-Processing) อีกด้วย บริษัทที่เป็นผู้นำด้านการผลิตชิพ DSP ได้แก่ Texas Instruments, Motorola, Analog Devices, และ AT&T เป็นต้น ซึ่งชิพ DSP นี้มีทั้ง

ประเภทที่เป็นการประมวลผลข้อมูลแบบจำนวนเต็ม (Fixed-Point)และประเภทที่ประมวลผลข้อมูลแบบเลขอิงดรรชนี (Floating-Point)

การใช้งานชิพ DSP นั้น ทำได้โดยเขียนเป็นโปรแกรมภาษาแอสแซมบลี หรือภาษาซีแล้วใช้คอมไพล์เลอร์แปลงเป็นภาษาแอสแซมบลี ข้อดีของการเขียนเป็นภาษาแอสแซมบลีโดยตรง คือ สามารถควบคุมการทำงานของชิพได้เต็มที่ ทำให้สามารถออกแบบโปรแกรมให้ทำงานได้เร็วกว่าและมีขนาดโปรแกรมเล็กกว่าการใช้โปรแกรมภาษาซี แต่ข้อเสียคือ ภาษาแอสแซมบลีเขียนยากกว่าและไม่สามารถโอนย้ายโปรแกรมไปทำงานได้ในชิพต่างตระกูลกัน หรือต่างผู้ผลิตกันได้

การต่อวงจรเพื่อใช้งานชิพ DSP ทำเช่นเดียวกับการต่อวงจรไมโครโปรเซสเซอร์ทั่วๆไป เพียงแต่มีตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล (ADC) และดิจิตอลเป็นแอนะลอก (DAC) เพิ่มขึ้นมาเท่านั้น ในภาพที่ 2-4 เป็นภาพทั่วไปของวงจร ซึ่งใช้ชิพ DSP แบบ Fixed-point 16 บิต เช่น TMS320C50 ของ Texas Instruments โดยมีบัสข้อมูลขนาด 16 บิต และมีตัวคูณ และตัวประมวลผลอื่นๆ ขนาด 16 บิต อยู่ภายใน ในภาพใช้ DAC และ ADC ขนาด 12 บิต มีขาข้อมูล 12 ขา เพื่อส่งข้อมูลแบบขนาน และต่อเข้ากับ 12 บิตล่างของบัสข้อมูล สังเกตว่ามีสัญญาณนาฬิกาซึ่งมีความถี่ f_s ป้อนให้กับ ADC เพื่อเป็นตัวกำหนดอัตราการสุ่มสัญญาณแอนะลอก ซึ่งคืออัตราของข้อมูลที่จะถูกอ่านเข้าชิพ DSP ไปประมวลผล

ชิพ DSP หลายยี่ห้อมีหน่วยความจำ ROM และ RAM บางส่วนอยู่ภายในตัวเอง ทำให้เพิ่มความเร็วในการทำงาน และสะดวกในการใช้งานมาก โดยงานที่ไม่ต้องการใช้ปริมาณ ROM และ RAM มากนัก อาจไม่จำเป็นต้องต่อหน่วยความจำภายนอกเลย



ภาพที่ 2-4 แผนภาพแสดงตัวอย่างวงจรที่ใช้งานชิพ DSP

2.1.3.3 การใช้ฮาร์ดแวร์ หรือ ไอซีที่ออกแบบเฉพาะงาน

ฮาร์แวร์ในที่นี้หมายถึง วงจรดิจิตอลซึ่งสามารถออกแบบให้ทำการประมวลผลข้อมูลได้เช่นกัน อัลกอริธึมที่เป็นที่นิยม เช่น FFT (Fast Fourier Transform) หรือ ตัวกรองดิจิตอล

สามารถหาซื้อได้ทั่วไป เป็นไอซีสำเร็จรูปที่ทำเฉพาะฟังก์ชันนั้นๆ แต่ถ้าต้องการอัลกอริธึมที่เฉพาะมากขึ้น ต้องทำการออกแบบเป็นไอซีเฉพาะงาน (Application Specific Integrated Circuits หรือ ASIC) ซึ่งต้นทุนในการออกแบบสำหรับทางเลือกนี้ค่อนข้างสูง ทางเลือกอีกทาง คือ การใช้ไอซีดิจิตอลประเภทโปรแกรมได้หรือ FPGA (Field Programmable Gate Array) ซึ่งปัจจุบันมีขนาดใหญ่มากพอที่จะนำมาใช้ทำการประมวลผลสัญญาณได้ การใช้ FEGA จะมีต้นทุนในการออกแบบที่ถูกกว่า ASIC

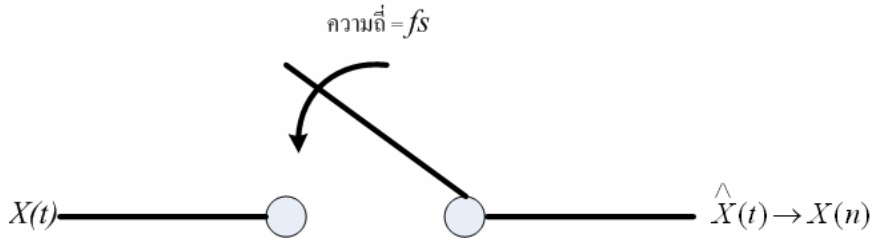
การเลือกใช้ตัวประมวลผลแต่ละแบบขึ้นอยู่กับลักษณะของงาน ความเร็วที่ต้องการ และต้นทุน ถ้าต้องการทำอุปกรณ์ที่มีการประมวลผลแบบเวลาจริง โดยทั่วไปการใช้ชิพ DSP จะดีที่สุด (ซึ่งชิพ DSP มีหลากหลายขนาด และความเร็วให้เลือกใช้) แต่ถ้าการประมวลผลไม่ซับซ้อนหรืออัตราข้อมูลไม่สูงมากจนไม่สามารถใช้ไมโครโปรเซสเซอร์ธรรมดาได้ การใช้ไมโครโปรเซสเซอร์จะทำให้ต้นทุนต่ำลงได้ ในกรณีที่ต้องการอัตราการประมวลผลสูงมากๆ อาจต้องใช้ฮาร์ดแวร์ในการประมวลผล โดยทั่วไปจะมีต้นทุนที่สูงขึ้น

2.1.4 การสุ่มสัญญาณและการสร้างสัญญาณคืน (Sampling and Reconstruction Signal)

การสุ่มสัญญาณคือกระบวนการแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล การสร้างสัญญาณคืน คือกระบวนการแปลงสัญญาณดิจิตอลกับเป็นสัญญาณแอนะลอก

2.1.4.1 การสุ่มสัญญาณ

การสุ่มสัญญาณคือ การนำเอาสัญญาณขาเข้าแบบต่อเนื่องมาผ่านสวิทซ์อุดมคติที่ต่อวงจรที่ตำแหน่งเวลาเท่ากับ …,*-2T, -T, 0, T, 2T, 3T,*…และเปิดวงจรที่เวลาอื่นๆ คือ มีความถี่ของการตัดต่อวงจรเท่ากับ f_s ครั้งต่อวินาที หรือมีคาบของการสุ่มเท่ากับ $T=1/f_s$ ดังแสดงในภาพที่ 2-5 เรียกสัญญาณขาออกว่า $\hat{x}(t)$



ภาพที่ 2-5 การสุ่มด้วยสวิตช์อุดมคติ

ถ้ามีสวิตช์อุดมคติจริง จะได้สัญญาณขาออก $\hat{x}(t)$ ซึ่งเป็นสัญญาณแอนะลอกที่มีลักษณะเป็นอุดมคติ คือมีค่าที่เฉพาะเวลา …,*-2T, -T, 0, T, 2T, 3T,*…เท่านั้น ส่วนเวลาอื่น มีค่าเป็นศูนย์ สัญญาณนี้มีลักษณะเหมือนสัญญาณไม่ต่อเนื่อง เพียงแต่มองเป็นสัญญาณแอนะลอกในแกนเวลา *t*

ถึงแม้จะไม่ใช้สัญญาณนี้ในลักษณะเป็นสัญญาณแอนะลอกแต่การมองนี้เพื่อการวิเคราะห์ในเชิงของความถี่ของสัญญาณ จะมีประโยชน์ในการบอกขีดจำกัดของกระบวนการสุ่มได้

วิเคราะห์หาสเปกตรัมของสัญญาณ $\hat{x}(t)$ ว่าสอดคล้องกับสเปกตรัมของ $x(t)$ โดยพิสูจน์ในแง่คณิตศาสตร์ ถ้านิยามว่ามีสัญญาณอิมพัลส์ (Impulse Signal) หรือเขียนแทนด้วยสัญลักษณ์ว่า $\delta(t)$ เป็นสัญลักษณ์ที่มีค่าเท่ากับ 1 ที่เวลา *t=0* และเป็น 0 ที่เวลาอื่นๆดังนี้

$$\delta(t) = \begin{cases} 1, & \text{t = 0} \\ 0, & \text{t = ค่าอื่นๆ} \end{cases} \qquad (2\text{-}1)$$

อาจมองว่าการสุ่ม คือ การนำเอาสัญญาณขาเข้ามาคูณเข้ากับสัญญาณอิมพัลส์หลายๆลูกที่มีคาบเท่ากับ T (ระยะห่างระหว่างอิมพัลส์แต่ละลูก) นิยามสัญญาณอิมพัลส์หลายๆลูกนี้เป็นสัญญาณ *s(t)* ซึ่งรูปร่างจะแสดงในภาพที่ 2-6 ในทางคณิตศาสตร์ สามารถเขียน *s(t)* ได้ว่าเป็นผลรวมของสัญญาณอิมพัลส์ที่ตำแหน่งเวลาต่างๆ *…,-2T, -T, 0, T, 2T, 3T,…*ดังนี้

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad (2\text{-}2)$$

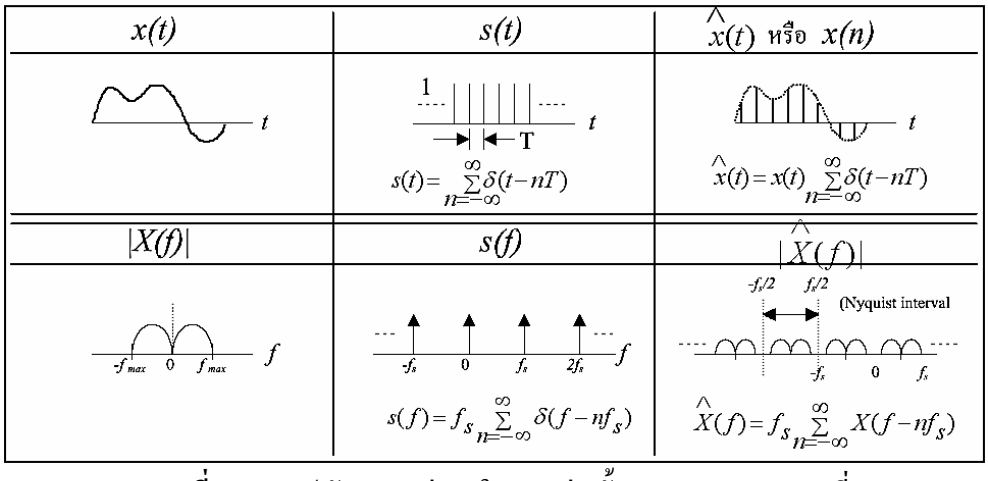
เมื่อสัญญาณ *s(t)* คูณเข้ากับสัญญาณขาเข้า *x(t)* จะได้สัญญาณที่ขาออกของตัวสุ่มสัญญาณคือ

$$\hat{x}(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \qquad (2\text{-}3)$$

ตัวอย่างของสัญญาณ $\hat{x}(t)$ มีลักษณะดังแสดงในภาพที่ 2-6 สัญญาณนี้สามารถนิยามให้เป็นสัญญาณไม่ต่อเนื่อง หรือเป็น “ลำดับของค่า” ซึ่งจะใช้สัญลักษณ์เป็น *x(n)* โดยใช้ค่า *n* เป็นตัวชี้แทน t จะได้ว่า *x(n)* มีความสัมพันธ์กับ *x(t)* ดังนี้

$$x(n) = x(t)\,|_{t=nT} \qquad (2\text{-}4)$$

สัญญาณ $\hat{x}(t)$ กับ *x(n)* หมายถึงสัญญาณตัวเดียวกัน เพียงแต่สัญญาณ $\hat{x}(t)$ เป็นการมองสัญญาณนี้ในลักษณะเป็นสัญญาณแอนะลอกในแกนของเวลา t ซึ่งจะเห็นว่าสัญญาณมีค่าเวลา *T = …,-2T, -T, 0, T, 2T,…* แต่สัญญาณ *x(n)* เป็นการมองสัญญาณเป็นสัญญาณไม่ต่อเนื่องหรือเป็นลำดับคือ มีค่าที่ *n = …,-2, -1, 0, 1, 2,…* สัญญาณ *x(n)* จะไม่อยู่ในรูปแบบของสัญญาณแอนะลอก แต่จะอยู่ในรูปแบบของลำดับข้อมูล



ภาพที่ 2-6 สรุปสัญญาณต่างๆในการสุ่มทั้งภาคเวลาและความถี่

พิจารณาในภาคความถี่ สมมติว่าสัญญาณ *x(t)* เมื่อใช้การแปลงฟูริเยร์ได้สเปกตรัมเป็น *X(f)* สมมุติว่ามีแถบความถี่(Bandwidth) จำกัด และมีองค์ประกอบความถี่สูงสุดอยู่ที่ f_{max} ดังแสดงในภาพที่ 2-6 ส่วนสัญญาณ *s(t)* จะสามารถพิสูจน์โดยใช้การแปลงฟูริเยร์(ขอละไม่พิสูจน์ให้ดู) ได้ว่ามีสเปกตรัมเป็นสัญญาณอิมพัลส์หลายลูกที่มีคาบคงที่เช่นกัน โดยคาบในที่นี้เท่ากับ f_s และมีขนาดคงที่เท่ากับ f_s เขียนเป็นสมการได้ดังสมการที่ 2-5

$$S(f) = f_s \sum_{n=-\infty}^{\infty} \delta(f - nf_s) \tag{2-5}$$

สำหรับสเปกตรัมของสัญญาณขาออก คือ $\hat{X}(f)$ หาได้โดยกฎที่ว่า การคูณในเชิงเวลาเท่ากับการคอนโวลูชัน(Convolution) ในเชิงความถี่ นั้นคือ เมื่อ $\hat{x}(t)$ เป็นผลคูณระหว่างสัญญาณ *s(t)* กับ *x(t)* จะได้ว่า สเปกตรัมของ $\hat{x}(t)$ คือ $\hat{X}(f)$ จะเท่ากับผลคูณคอนโวลูชันระหว่าง $X(f)$ กับ $S(f)$ ดังนี้(ขอนิยามสัญลักษณ์*แทนการคอนโวลูชัน)

$$\hat{X}(f) = X(f) * S(f)$$

เมื่อแทนค่า $S(f)$ ลงไปแล้วจัดให้เป็นรูปอย่างง่ายจะได้

$$\hat{X}(f) = X(f) * f_s \sum_{n=-\infty}^{\infty} \delta(f - nf_s)$$

$$= f_s \sum_{n=-\infty}^{\infty} X(f) * \delta(f - nf_s)$$

$$\hat{X}(f) = f_s \sum_{n=-\infty}^{\infty} X(f - nf_s) \tag{2-6}$$

สมการนี้ประกอบด้วยผลบวกของเทอม $X(f-nf_s)$ ซึ่งเทอมนี้คือ สเปกตรัมของสัญญาณขาเข้าที่เลื่อนจุดศูนย์กลางไปอยู่ที่ตำแหน่งความถี่ nf_s โดย n เป็นจำนวนเต็มตั้งแต่ $-\infty$ จนถึง $+\infty$ หมายถึงว่าสเปกตรัมของสัญญาณหลังการสุ่มคือ $\hat{X}(f)$ ประกอบด้วยสำเนาของสเปกตรัมของสัญญาณก่อนการสุ่ม คือ $X(f)$ อยู่มาก โดยสำเนาเหล่านี้ จะเกิดความถี่ …,$-2f_s$, $-f_s$, 0, f_s, $2f_s$,… เป็นศูนย์กลาง สำเนาแต่ละตัว มีชื่อเรียกอีกอย่างหนึ่งว่าเป็นภาพฉาย(Image) ของสัญญาณ รูปร่างของ $\hat{X}(f)$ ที่ได้แสดงดังในภาพ 2-6

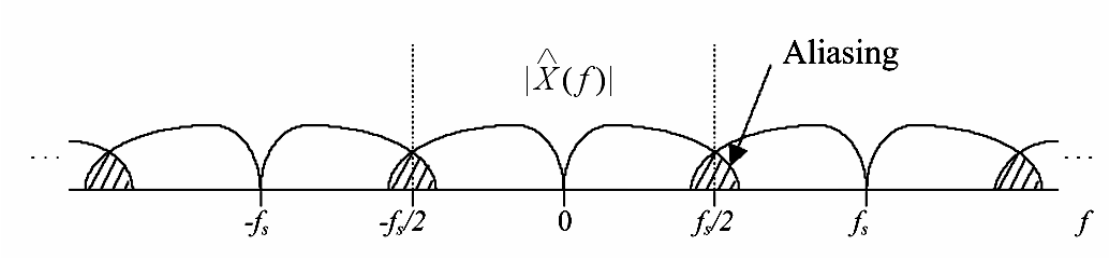
สังเกตว่า สเปกตรัม $\hat{X}(f)$ ที่หามาได้มีลักษณะที่เป็นอุดมคติในเชิงสัญญาณแอนะลอก คือมีองค์ประกอบของสัญญาณเรื่อยๆไปจนถึงความถี่อนันต์ แสดงว่ามีพลังงานไม่จำกัดซึ่งเป็นไปไม่ได้ ดังนั้นทั้งสัญญาณ *x(t)* และ $\hat{X}(f)$ (ถ้ามองในเชิงแอนะลอก) เป็นสัญญาณที่สร้างขึ้นในทางคณิตศาสตร์ เพื่อประโยชน์ในการวิเคราะห์ขีดจำกัดของกระบวนการสุ่ม

2.1.4.2 ทฤษฎีของการสุ่ม(Sampling Theorem)

ทฤษฎีการสุ่มสัญญาณ ระบุว่าถ้าสัญญาณที่ต้องการสุ่มมีความถี่สูงสุดที่ f_{max} เพื่อให้ได้สัญญาณที่สุ่มแล้วเป็นตัวแทนที่ถูกต้องของสัญญาณนี้ ความถี่ในการสุ่มจะต้องมีค่ามากกว่าสองเท่าของความถี่สูงสุดในสัญญาณหรือ

$$f_s > 2f_{max} \tag{2-7}$$

ซึ่งความถี่ที่ $2f_{max}$ นี้ มีชื่อเรียกว่า ความถี่ไนควิสท์ (Nyquist Frequency) หรืออัตราไนควิสท์ พิจารณาว่าถ้าใช้ f_s ต่ำกว่าค่าความถี่ไนควิสท์ ผลที่เกิดขึ้นจะสังเกตได้ชัดเจนในภาคความถี่ หลังการสุ่มจะเกิดสเปกตรัมของสัญญาณก่อนการสุ่มอยู่ที่ความถี่…,$-2f_s$, $-f_s$, 0, f_s, $2f_s$,… เป็นศูนย์กลาง เมื่อวาดรูปออกมาจะแสดงได้ดังภาพที่ 2-7



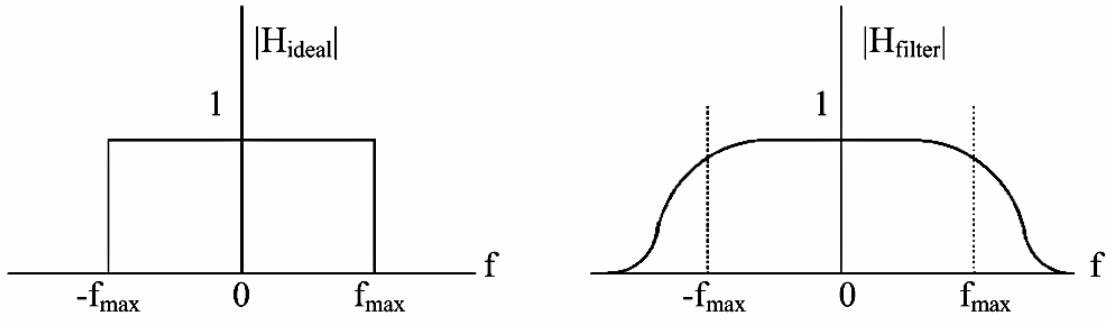
ภาพที่ 2-7 สัญญาณในเชิงความถี่ที่เกิดขึ้นเมื่อใช้ f_s ต่ำกว่า $2f_{max}$

เมื่อ f_s ต่ำกว่า $2f_{max}$ จะเห็นว่า สำเนาของ $X(f)$ ที่เกิดขึ้นมีช่วงของความถี่ส่วนปลายที่ซ้อนทับกัน เรียกองค์ประกอบความถี่ส่วนที่ซ้อนทับกันนี้ว่า Aliasing ซึ่ง Aliasing เป็นผลร้าย

กับระบบประมวลผลสัญญาณ เนื่องจาก ในระบบแบบไม่ต่อเนื่องจะไม่สนใจความถี่ในช่วง $-f_s/2$ จนถึง $f_s/2$ ซึ่งเรียกว่า ช่วงไนควิสท์ หรือย่านไนควิสท์ (Nyquist Interval) ซึ่งถ้าเกิด Aliasing ซ้อนทับในช่วงไนควิสท์นี้จะถือว่า สัญญาณขาเข้าที่สุ่มมาเกิดความผิดเพี้ยนก่อนที่จะเข้าไปในส่วนประมวลผลสัญญาณ จึงจำเป็นอย่างยิ่งที่ต้องพยายามไม่ให้เกิด Aliasing หรือเกิดน้อยที่สุดเท่าที่จะทำได้ซึ่งกระทำได้สองวิธี คือ

2.1.4.2.1 การใช้ตัวกรองเพื่อป้องกัน Aliasing (Anti-Aliasing Filter)

ในกรณีที่ไม่มั่นใจว่าสัญญาณที่จะทำการสุ่มไม่มีความถี่จำกัดอยู่ที่ f_{max} เช่น อาจมีสัญญาณรบกวนความถี่สูง หรือสัญญาณอื่น ปนอยู่ด้วย วิธีแก้ทำได้โดยการใช้ตัวกรองแอนะลอกแบบผ่านความถี่ต่ำ เพื่อจำกัดความถี่ของสัญญาณให้อยู่ในช่วงที่สนใจเท่านั้น (ต่ำกว่า f_{max}) คือ ต้องการตัวกรองผ่านความถี่ต่ำที่มีความถี่ตัดที่ f_{max} ดังแสดงผลตอบสนองเชิงความถี่แบบอุดมคติของตัวกรองนี้ ดังแสดงในภาพที่ 2-8 (ก)



ก) กรณีอุดมคติ ข) ความเป็นจริง

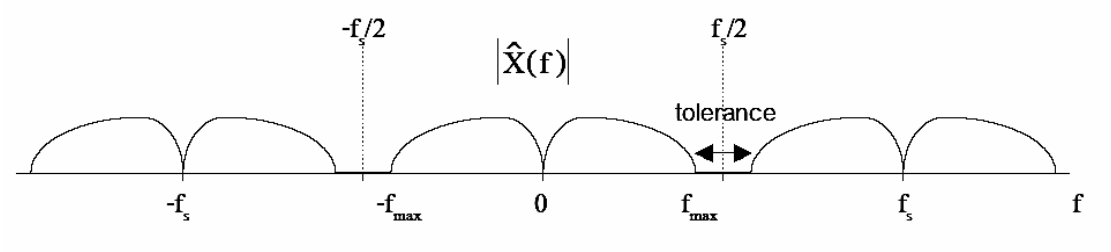
ภาพที่ 2-8 ผลตอบสนองเชิงความถี่ของตัวกรองป้องกัน Aliasing

ในความเป็นจริงตัวกรองแบบอุดมคติไม่สามารถทำได้ และถ้าต้องการสร้างตัวกรองที่ใกล้เคียงกับอุดมคติจะทำให้ต้นทุนสูง ตัวกรองที่สามารถทำได้ในทางปฏิบัติมีรูปร่างดังแสดงในภาพ 2-8 (ข) คือ มีความชันของการตัดความถี่ไม่คมเท่ากับตัวกรองอุดมคติจึงทำให้เกิด Aliasing บางส่วนขึ้นได้ถ้าใช้ความถี่ในการสุ่มพอดีเท่ากับ $2f_{max}$

2.1.4.2.2 การสุ่มโดยใช้ความถี่เกินกว่า $2f_{max}$

การสุ่มโดยใช้ความถี่เกินกว่า $2f_{max}$ มากๆ หรือเรียกว่าการทำ Over sampling การสุ่มด้วยความถี่สูงขึ้นเป็นวิธีที่ใช้ป้องกัน Aliasing เสริมจากการวิธีการใช้ตัวกรอง ถ้าพิจารณาในเชิงความถี่จะเห็นว่า เมื่อ f_s มีค่าสูงขึ้น จะมีช่วงความถี่ที่เผื่อให้เกิด Aliasing ได้กว้างขึ้น(ช่วงเผื่อนี้ คือ ย่านTolerance แสดงดังในภาพที่ 2-9 เป็นย่านที่ไม่มีความถี่ของสัญญาณที่สนใจอยู่) ช่วงเผื่อนี้เป็นย่านความถี่ที่ยอมให้มีความถี่ของสัญญาณที่ไม่ต้องการหลุดรอดเข้ามาในระบบได้บ้าง ดังนั้น

การสุ่มด้วยความถี่สูงกว่าอัตราไนควิสท์ ทำให้สามารถใช้ตัวกรองป้องกัน Aliasing ที่ไม่ต้องมีคุณสมบัติคมมากนักได้

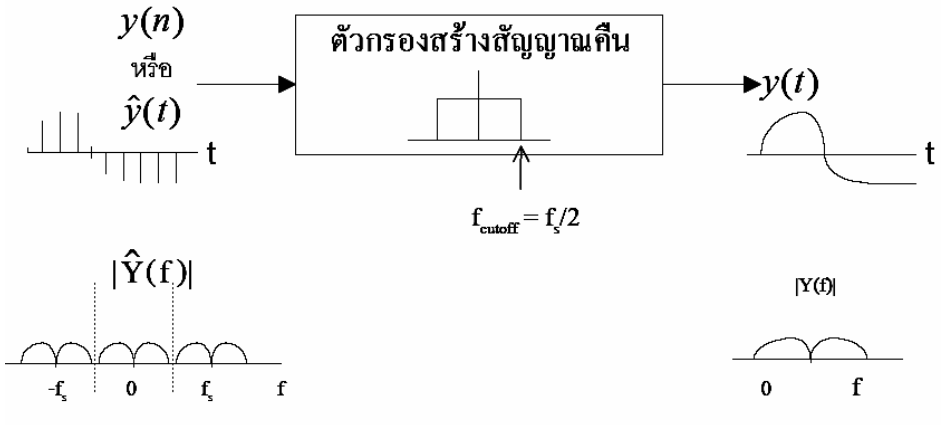


ภาพที่ 2-9 สเปกตรัมของสัญญาณที่เกิดขึ้น เมื่อสุ่มด้วยความถี่สูงกว่า $2f_{max}$ มากๆ

ในทางปฏิบัติใช้ $f_s \geq 2.5 f_{\max}$ เพื่อชดเชยผลของการที่ตัวกรอง Anti-Aliasing ไม่เป็นอุดมคติ สำหรับ f_s สูงสุดที่จะสุ่มได้โดยทั่วไป ขึ้นอยู่กับขีดจำกัดด้านความเร็วของตัวแปลงสัญญาณแอนะลอกเป็นดิจิตอล และตัวประมวลผลที่เลือกใช้ ถ้าความถี่ f_s สูงขึ้น หมายถึงว่า ต้องใช้วงจรแปลงสัญญาณที่เร็วขึ้น และใช้ปริมาณการประมวณผลที่มากขึ้น เพราะอัตราข้อมูลสูงขึ้น โดยช่วงเวลาที่ใช้ประมวลผล เพื่อให้ได้แต่ละค่าของสัญญาณขาออก ต้องมีค่าน้อยกว่าคาบของการสุ่ม

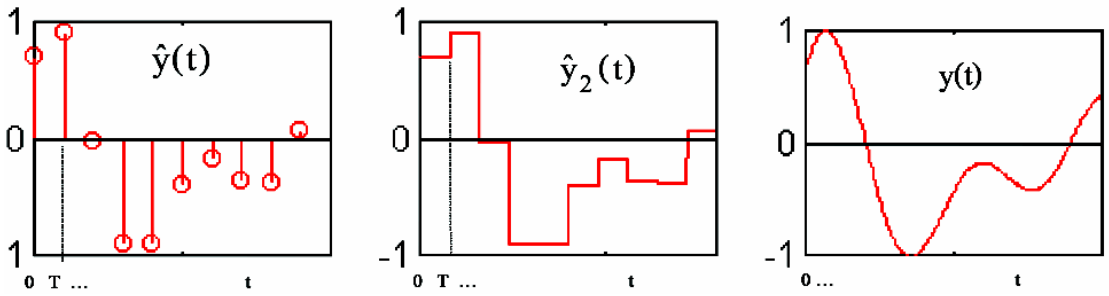
2.1.4.3 การสร้างสัญญาณคืน

การสร้างสัญญาณคืน ในทางทฤษฎีทำได้โดยผ่านสัญญาณแบบไม่ต่อเนื่องเข้าไปยังตัวกรอง (แอนะลอก) แบบผ่านความถี่ต่ำที่มีความถี่ตัดที่ $f_s/2$ ตัวกรองนี้บางครั้งเรียกว่า ตัวกรองสร้างสัญญาณคืน (Reconstruction Filter) ตัวกรองจะผ่านเฉพาะสัญญาณในช่วงความถี่ระหว่าง $-f_s/2$ ถึง $f_s/2$ หรือช่วงไนควิสท์ออกมา ผลที่ได้คือ จะได้สำเนาของสเปกตรัมที่อยู่รอบความถี่ 0 ออกมาเป็นสัญญาณขาออก ซึ่งก็คือ สัญญาณแอนะลอกที่มีรูปร่างเป็นขอบของสัญญาณไม่ต่อเนื่องก่อนสร้างกลับ ดังแสดงในภาพที่ 2-10



ภาพที่ 2-10 สัญญาณขาเข้าและออกของตัวกรองสร้างสัญญาณคืน

ในทางปฏิบัติ ไม่ใช้ $\hat{y}(t)$ เป็นสัญญาณขาเข้าของตัวกรองสร้างสัญญาณคืน เนื่องจากมองเห็นสัญญาณไม่ต่อเนื่องในลักษณะเป็นลำดับของข้อมูล สัญญาณที่มองแบบแอนะลอกคือ $\hat{y}(t)$ เป็นสัญญาณอุดมคติที่ใช้พิสูจน์ในทางคณิตศาสตร์เท่านั้น ดังนั้น การแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกในความเป็นจริง จะนำสัญญาณ $y(n)$ มาผ่านวงจรคงค่า (Hold) ที่ทำงานเข้าจังหวะกับอัตราสุ่มของสัญญาณ $y(n)$ เพื่อสร้างเป็นสัญญาณลักษณะขั้นบันไดดังในภาพที่ 2-11

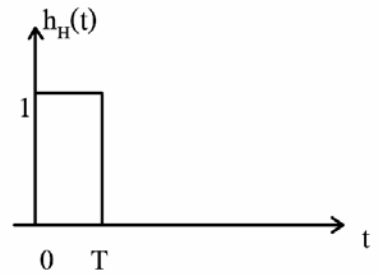


ภาพที่ 2-11 สัญญาณขาเข้า และขาออกของวงจรคงค่าสัญญาณ

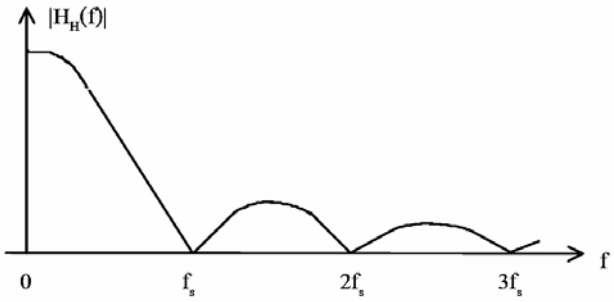
สัญญาณขั้นบันได $\hat{y}_2(t)$ ที่สร้างมา มีลักษณะของสเปกตรัมเหมือนหรือแตกต่างจากสเปกตรัมของสัญญาณ $\hat{y}(t)$ สามารถหาสเปกตรัมของสัญญาณ $\hat{y}_2(t)$ ได้โดยมองวงจรคงค่าสัญญาณเป็นระบบแอนะลอกแบบเชิงเส้นอันหนึ่ง มีผลตอบสนองสัญญาณอิมพัลส์คือ $h_H(t)$ ดังแสดงในภาพ 2-12(ก) คือ วงจรนี้เปลี่ยนอิมพัลส์ของสัญญาณขาเข้า เป็นพัลส์สี่เหลี่ยมที่มีความกว้างเท่ากับคาบของอัตราการสุ่ม เมื่อทำการแปลงฟูริเยร์ ผลตอบสนองอิมพัลส์นี้ จะได้ผลตอบสนองเชิงความถี่ของระบบมีลักษณะเป็นฟังก์ชันซิงค์ (Sinc) ดังแสดงในภาพที่ 2-12(ข)

สเปกตรัมของสัญญาณขาออกของวงจรคงค่า หาได้จากผลคูณของสเปกตรัมของสัญญาณขาเข้า และผลตอบสนองเชิงความถี่ของระบบ ได้ผลลัพธ์ดังแสดงในภาพที่ 2-12(ง) สังเกตได้ว่าสเปกตรัมของสัญญาณขั้นบันไดยังคงมีสำเนาของสัญญาณแอนะลอกที่ความถี่ 0, f_s, $2f_s$,...อยู่ครบเพียงแต่สำเนาแต่ละตัวถูกกดขนาดลงด้วยการคูณของฟังก์ชันซิงค์ ซึ่งสำเนาเหล่านี้จะต้องถูกกำจัดทิ้ง โดยตัวกรองสร้างสัญญาณคืนที่มีความถี่ตัดที่ $f_s/2$ เพื่อให้ได้สัญญาณแอนะลอกขาออกมีรูปร่างที่เรียบสมบูรณ์ สรุปว่า ถึงจะใช้วงจรคงค่าซึ่งให้ผลดูใกล้เคียงสัญญาณแอนะลอกที่ต้องการแล้ว ยังคงต้องใช้ตัวกรองสร้างสัญญาณคืนร่วมด้วย

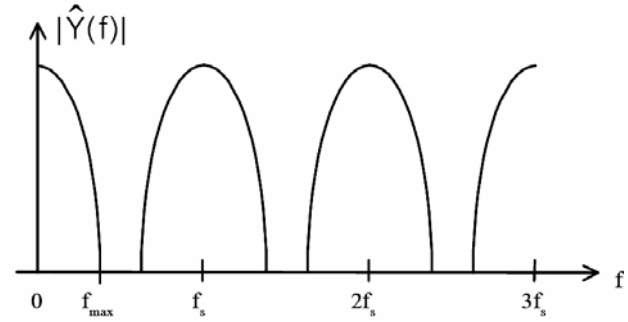
ก)



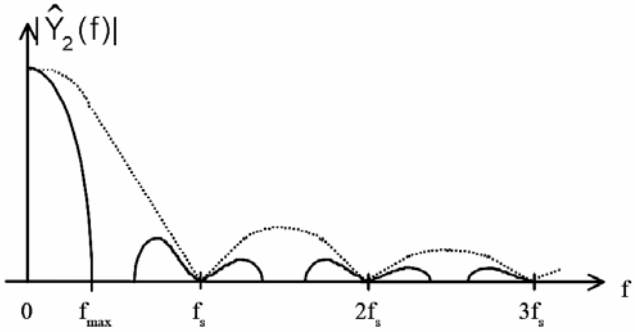
ข)



ค)



ง)



ภาพที่ 2-12 (ก) ผลตอบสนองสัญญาณอิมพัลส์ของวงจรคงค่า (ข) ผลตอบสนองเชิงความถี่ของวงจรคงค่า (ค) สเปกตรัมของสัญญาณ $\hat{y}(t)$ (ง) สเปกตรัมของสัญญาณ $\hat{y}_2(t)$

เช่นเดียวกับตัวกรองป้องกัน Aliasing คือ ไม่สามารถทำตัวกรองอุดมคติสำหรับสัญญาณคืนได้ ซึ่งผลข้างเคียงคือ จะทำให้สัญญาณที่อยู่ในช่วงความถี่สูงกว่า $f_s/2$ หลุดลอดออกมาที่สัญญาณขาออกด้วย กลายเป็นความเพี้ยนของสัญญาณขาออก อย่างไรก็ตาม ถ้าใช้ความถี่ในการสุ่มสูงๆ หรือ (Over Sampling) เพื่อให้มีระยะห่างระหว่างสำเนาความถี่แต่ละตัวมากขึ้น นอกจากจะแก้ปัญหาของการใช้ตัวกรองป้องกัน Aliasing ที่ไม่เป็นอุดมคติ ยังช่วยแก้ปัญหาของการใช้ตัวกรองสร้างสัญญาณคืนที่ไม่เป็นอุดมคติได้อีกด้วย

2.2 การประมวลผลแบบหลายอัตราสุ่ม [8, 9]

วิธีที่จะเปลี่ยนอัตราการสุ่มของสัญญาณอย่างง่าย คือ การแปลงสัญญาณนั้นกลับเป็นสัญญาณแอนะลอกก่อน แล้วสุ่มสัญญาณแอนะลอกใหม่ด้วยอัตราการสุ่มที่ต้องการ วิธีนี้ให้อิสระในการเปลี่ยนอัตราสุ่มพอสมควร โดยสามารถใช้อัตราสุ่มใหม่เป็นเท่าไรก็ได้ โดยไม่ขัดแย้งกับหลักการของการสุ่มที่ถูกต้อง คือ อัตราการสุ่มใหม่จะต้องมากกว่าสองเท่าของความถี่สูงสุดที่มีอยู่ในสัญญาณ

กรณีที่เป็นการเพิ่มอัตราการสุ่ม จะไม่มีปัญหาเกี่ยวกับ Aliasing เนื่องจาก อัตราการสุ่มใหม่มากกว่าอัตราเก่า หมายถึง ย่านไนควิทซ์กว้างขึ้น ดังนั้น ถ้าสัญญาณไม่ต่อเนื่องเดิมไม่มี Aliasing สัญญาณใหม่จะไม่เกิด Aliasing เช่นกัน แต่ในกรณีของการลดอัตราการสุ่มซึ่งย่านไนควิทซ์จะแคบลง จะต้องระวังไม่ให้ความถี่ของสัญญาณเดิมมากเกินกว่าครึ่งหนึ่งของอัตราการสุ่มใหม่ หากมากกว่า จำเป็นจะต้องกรองความถี่ในช่วงที่มากกว่านั้นทิ้งไป โดยอาจใช้ตัวกรองดิจิตอลกรองทิ้งไปก่อนแปลงเป็นสัญญาณแอนะลอก หรือใช้ตัวกรองแอนะลอกกรองก่อนที่จะทำการสุ่มใหม่ได้ วิธีเปลี่ยนอัตราการสุ่มโดยแปลงเป็นสัญญาณแอนะลอกก่อน เป็นวิธีที่ไม่นิยมทำเนื่องจากมีข้อเสีย คือ เพิ่มสัญญาณรบกวนจากการแบ่งขั้นสัญญาณ เข้ามาในระบบ ในจุดที่ทำการสุ่มใหม่ และเพิ่มความเพี้ยนจาก aliasing ถ้าใช้ตัวกรองแอนะลอกที่ D/Aและ A/D ไม่คมพอ อีกข้อเสียหนึ่งคือ ไม่สะดวกที่จะมีวงจรแอนะลอกเพิ่มเข้ามาในกึ่งกลางของการประมวลผล ดังนั้นจำเป็นที่จะต้องใช้ ตัวเปลี่ยนอัตราสุ่ม ที่ทำงานได้โดยไม่ต้องแปลงสัญญาณกลับเป็นแอนะลอกก่อน

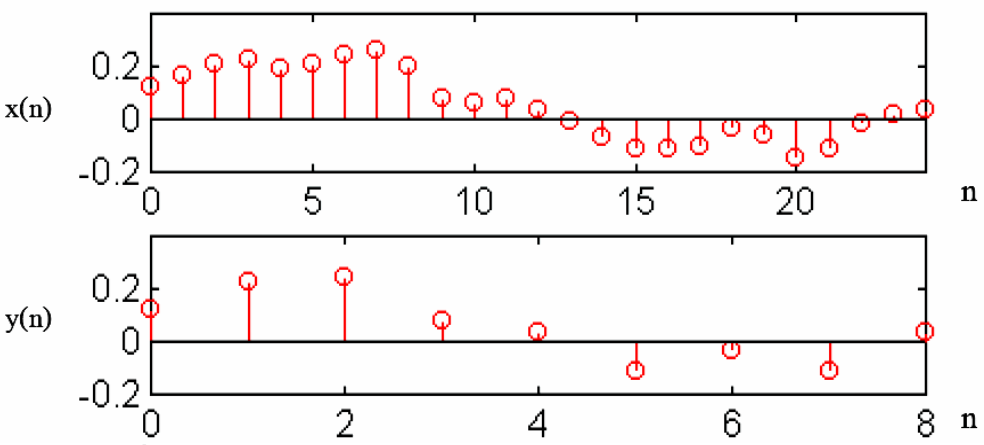
หลักการของการประมวลผลแบบหลายอัตราสุ่มจะแบ่งออกเป็น 2 ส่วนคือ เดซิเมชัน (Decimation) ใช้ในการลดอัตราการสุ่ม และอินเทอร์โพเลชัน (Interpolation) ใช้ในการเพิ่มอัตราการสุ่ม

2.2.1 เดซิเมชัน

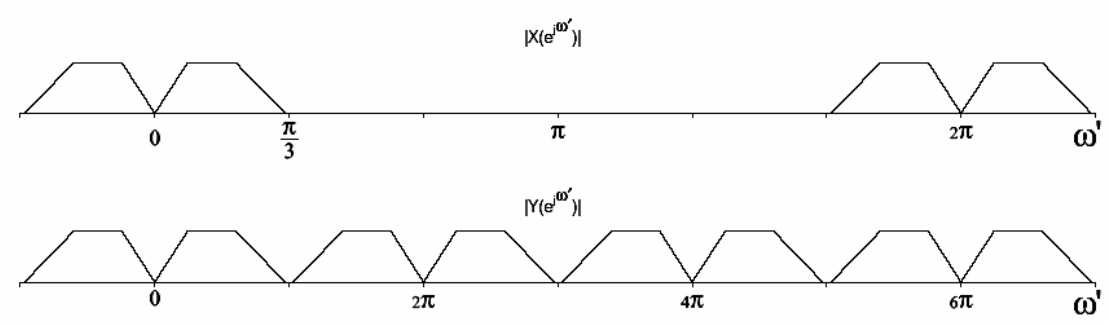
การลดอัตราการสุ่มของสัญญาณไม่ต่อเนื่องสามารถทำได้โดย ถ้าต้องการลดอัตราลงด้วยอัตราส่วน 1/D หรือ ลดอัตราลง D เท่า โดยที่ D เป็นจำนวนเต็ม ทำได้โดยสร้างสัญญาณใหม่ขึ้นมาโดยดึงสัญญาณเก่ามา 1 ค่า แล้วเว้น D-1 ค่า เช่น ถ้า D = 3 จะต้องสร้างสัญญาณใหม่ $y(n)$ โดยดึงเอาสัญญาณเก่า $x(n)$ 1 ค่า แล้วเว้น 2 ค่า จะได้อัตราข้อมูลของ $y(n)$ น้อยลง หรือช้าลงกว่า $x(n)$ 3 เท่า ถ้าคิดเป็นปริมาณของข้อมูล จะได้จำนวนข้อมูลของ $y(n)$ น้อยกว่า $x(n)$ 3 เท่า ดังแสดงในภาพที่ 2-13

ในโดเมนความถี่ การลดอัตราการสุ่มลง D เท่า จะทำให้ย่านไนควิทซ์แคบลงเป็น 1/D จากของเดิม สัญญาณใหม่จะมีความถี่ π ย้ายมาอยู่ที่ความถี่ $\pi/3$ ของเดิม ดังแสดงในภาพที่ 2-14 จะมีปัญหาเกิดขึ้นถ้าหากว่าสัญญาณเดิมมีองค์ประกอบความถี่ที่สูงกว่า $\pi/3$ อยู่ เมื่อลดอัตราการสุ่มจะ

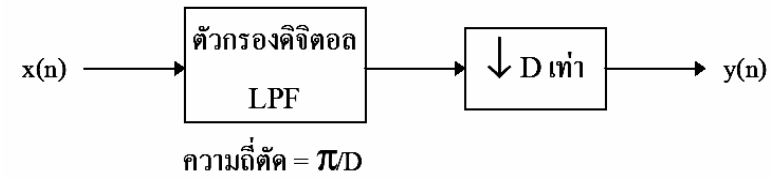
ทำให้เกิดการซ้อนทับของสเปกตรัม คือเกิด Aliasing การลดอัตราการสุ่มจะต้องกรองสัญญาณความถี่สูงทิ้งไป โดยใช้ตัวกรองดิจิตอลผ่านต่ำที่มีความถี่ตัดที่ $\pi/3$



ภาพที่ 2-13 สัญญาณทางเวลาก่อนและหลังการลดอัตราการสุ่มลง 3 เท่า



ภาพที่ 2-14 สเปกตรัมของสัญญาณก่อน และหลังการลดอัตราการสุ่มลง 3 เท่า



ภาพที่ 2-15 ส่วนประกอบของเดซิเมชัน

ในภาพที่ 2-15 แสดงส่วนประกอบของเดซิเมชัน การลดจำนวนข้อมูลสัญญาณ $x(n)$ ลด D เท่า ซึ่งประกอบด้วย ตัวกรองดิจิตอล LPF (Lowpass Filter) และ Sampling Rate Compressor โดยตัวลดขนาดสัญญาณจะทำการสร้างความถี่สุ่มขึ้นมาใหม่

2.2.2 อินเทอร์โพเลชัน

การเพิ่มอัตราสุ่มของสัญญาณไม่ต่อเนื่องสามารถทำได้โดย ถ้าต้องการอัตราการสุ่มของสัญญาณขึ้น I เท่า โดยที่ I เป็นจำนวนเต็ม กำหนดให้ I เท่ากับ 3 สัญญาณใหม่ $y(n)$ จะมีอัตราข้อมูลเพิ่มขึ้นจากสัญญาณเก่า $x(n)$ 3 เท่า ถ้าคิดในแง่ของการประมวลผลแบบไม่เป็นเวลาจริง จะได้ว่า $y(n)$ จะมีจำนวนข้อมูลมากขึ้น 3 เท่า

การเพิ่มอัตราการสุ่มกรณี I = 3 จะได้ย่านไนควิทซ์เพิ่มขึ้นจากเดิมสามเท่า จึงไม่มีปัญหาเรื่อง เกิด Aliasing เหมือนการลดอัตราการสุ่ม แต่จะมีปัญหาคือการสร้างสัญญาณ $y(n)$ ขึ้นมาใหม่ โดยจะต้องนำสัญญาณ $x(n)$ มาแทรกข้อมูลใหม่เพิ่มลงไป 2 จุดในระหว่างทุกๆจุดของ $x(n)$ เพื่อให้มีอัตราข้อมูลเพิ่มขึ้น 3 เท่า โดยจะต้องหาข้อมูลใหม่ที่มีค่าเฉลี่ยที่ถูกต้องแทรกเข้าไปโดยไม่ทำให้สเปกตรัมเดิมของสัญญาณผิดเพี้ยนไป

ถ้าลองแทรกศูนย์ลงไป 2 จุดในระหว่างค่าของสัญญาณเก่าทุกๆจุด เรียกสัญญาณใหม่นี้ว่า *w(n)* ดังแสดงในภาพที่ 2-16 จะได้ว่า อัตราข้อมูลของ *w(n)* มากขึ้นเป็นสามเท่าจริงแต่ ศูนย์ที่แทรกเข้าไปไม่ใช่ค่าเฉลี่ยที่ถูกต้อง เพราะทำให้รูปร่างสัญญาณเพี้ยนไป พิจารณาในภาคความถี่ว่าเกิดการเปลี่ยนแปลงกับสัญญาณ *w(n)* โดยพิจารณาการแปลง *z* ของ *w(n)* ดังนี้

$$W(z) = \sum_{n=0}^{\infty} w(n) z^{-n} \tag{2-8}$$

ถ้าสัญญาณ $w(n)$ เริ่มต้นที่ $w(0) = x(0)$ จะได้ค่า $w(n)$ ที่ไม่เท่ากับศูนย์ คือ *w(3), w(6), w(9),…* โดยที่ *w(3) = x(1), w(6) = x(2),…* เป็นเช่นนี้ไปเรื่อยๆหรือเขียนเป็นสูตรได้ว่า *w(n) = x(n/3)* จะได้

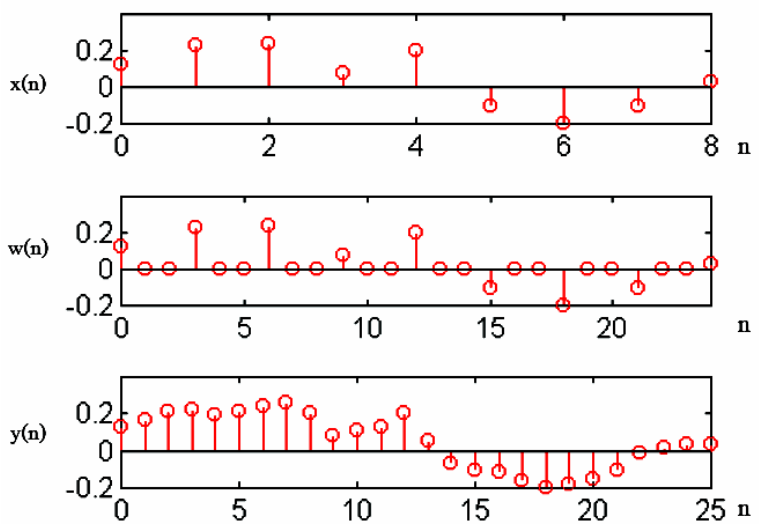
$$\begin{aligned} W(z) &= \sum_{n=0,3,6,9,\ldots}^{\infty} w(n) z^{-n} \\ &= \sum_{n=0,3,6,9,\ldots}^{\infty} x(n/3) z^{-n} \end{aligned}$$

แทน n ด้วย 3n จะได้ตัวชี้ที่เรียงลำดับเป็นปกติคือ n = 0, 1, 2, 3,…ดังนี้

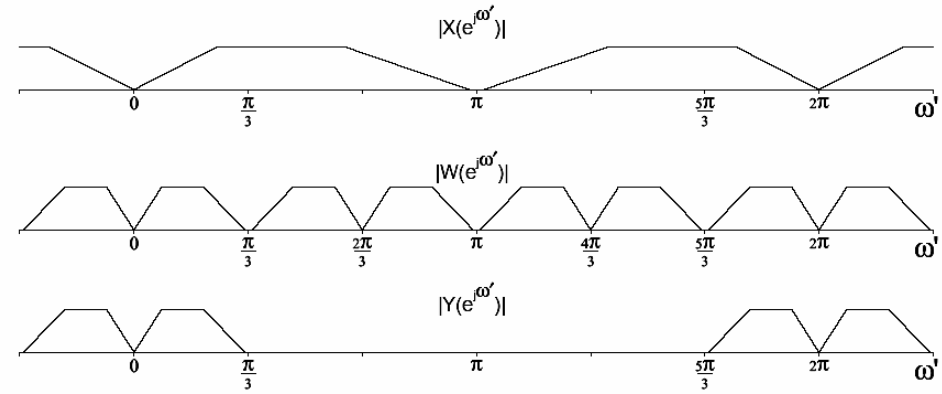
$$\begin{aligned} W(z) &= \sum_{n=0}^{\infty} x(n) z^{-3n} \\ &= X(z^3) \end{aligned}$$

แทน $z = e^{j\omega'}$ จะได้ความสัมพันธ์ระหว่างสเปกตรัมของ x(n) กับ w(n) ดังนี้

$$W(e^{j\omega'}) = X(e^{j3\omega'}) \tag{2-9}$$

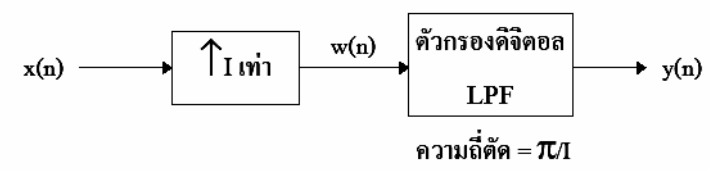


ภาพที่ 2-16 สัญญาณทางเวลาก่อนและหลังการเพิ่มอัตราการสุ่มขึ้น 3 เท่า



ภาพที่ 2-17 สเปกตรัมของสัญญาณก่อนและหลังการเพิ่มอัตราการสุ่มขึ้น 3 เท่า

จะได้ สเปกตรัมของ *w(n)* คือ สเปกตรัมของ *x(n)* ที่ถูกบีบให้แคบลง 3 เท่า ดังแสดงในภาพที่ 2-17 ย่านความถี่ในช่วง 0 ถึง π เดิมของ *x(n)* จะถูกบีบให้อยู่ในช่วงความถี่ 0 ถึง $\pi/3$ ของ *w(n)* ซึ่งช่วงความถี่นี้ คือสิ่งที่ต้องการ ส่วนความถี่ในช่วง $\pi/3$ ถึง π จะมีสำเนาของความถี่เดิมของ *x(n)* ปนเข้ามา ซึ่งสิ่งที่ไม่ต้องการจะถูกตัดทิ้งโดยใช้ตัวกรองดิจิตอล ส่วนประกอบของอินเทอร์โพเลชัน แสดงดังในภาพที่ 2-18



ภาพที่ 2-18 ส่วนประกอบของอินเทอร์โพเลชัน

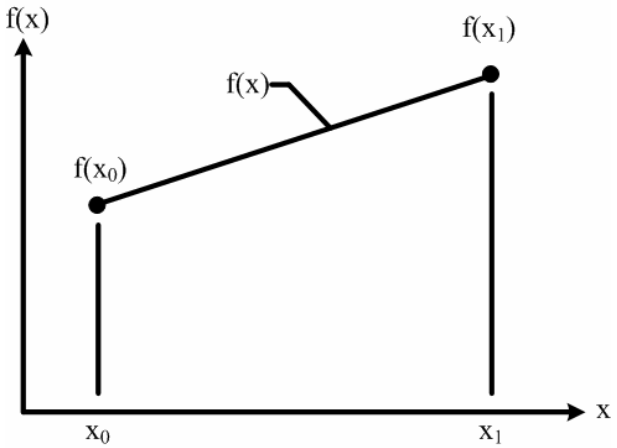
2.3 ฟังก์ชันพหุนามของลากรองจ์ (Lagrange Interpolation Polynomials) [10]

ปัญหาจากการเพิ่มความถี่สุ่ม คือ การหาสัญญาณมาชดเชยช่วงที่ขาดหายไป เพื่อให้ความถี่สุ่มเพิ่มขึ้น I เท่า โดยเงื่อนไขของข้อมูลใหม่ที่จะเติมลงไปคือ จะต้องเป็นค่าเฉลี่ยที่ถูกต้องระหว่างข้อมูลที่ถูกแทรกเข้าไปโดยไม่ทำให้สเปกตรัมเดิมของสัญญาณผิดเพี้ยนไป ซึ่งจะแก้ปัญหาโดยการประมาณค่าสัญญาณที่จะทำการชดเชยลงในสัญญาณเดิมโดยใช้ข้อมูล ของสัญญาณเดิมพยากรณ์หาสัญญาณใหม่โดยใช้ฟังก์ชันพหุนามของลากรองจ์

การประมาณค่าในช่วงโดยการใช้ฟังก์ชันพหุนามของลากรองจ์ (Lagrange Interpolation Polynomials) เป็นวิธีการที่นิยมใช้กันมากวิธีหนึ่ง วิธีการนี้เป็นรากฐานที่ถูกนำไปใช้ในระเบียบวิธีเชิงตัวเลขในระดับสูงขึ้นไป หลักการของวิธีการนี้คือการประดิษฐ์ฟังก์ชันพหุนามที่มีลักษณะของการกระจายซึ่งประกอบด้วยค่าของข้อมูลตามตำแหน่งต่างๆที่กำหนดมาให้ การศึกษาขั้นตอนการประดิษฐ์ฟังก์ชันพหุนาม เริ่มจากรูปแบบอย่างง่ายดังนี้

2.3.1 การประมาณค่าในช่วงเชิงเส้น

สมมุติว่ามีข้อมูลอยู่ 2 ข้อมูล เช่น ตำแหน่ง x_0 ค่าของข้อมูล คือ $f(x_0)$ และที่ตำแหน่ง x_1 ค่าของข้อมูลคือ $f(x_1)$ ถ้าต้องการประมาณค่าในช่วงระหว่าง x_0 และ x_1 ฟังก์ชันที่ง่ายที่สุดจะอยู่ในรูปแบบของสมการเส้นตรงดังแสดงในภาพที่ 2-19



ภาพที่ 2-19 ลักษณะฟังก์ชันเชิงเส้นสำหรับการประมาณค่าในช่วงระหว่าง x_0 และ x_1

อธิบายได้ด้วยสมการดังนี้

$$f(x) = ax + b \tag{2-10}$$

โดย a และ b เป็นค่าคงตัวที่ไม่รู้ค่า ซึ่งสามารถหาได้จากเงื่อนไขที่ตำแหน่ง x_0 และ x_1 ดังนี้

ที่ $x = x_0$: $f(x_0) = ax_0 + b$ (2-11)

$x = x_1$: $f(x_1) = ax_1 + b$ (2-12)

หากเอาสมการ (2-11)ลบออกจากสมการ(2-12) จะได้

$$f(x_1) - f(x_0) = a(x_1 - x_0)$$

$$a = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

จากนั้นแทนค่าคงตัวที่ a ที่ได้กลับลงในสมการ (2-11) จะได้

$$b = f(x_0) - \frac{f(x_1) - f(x_0)}{x_1 - x_0} x_0$$

ทำให้สมการเส้นตรง (2-10) กลายมาเป็น

$$f(x) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} x + f(x_0) - \frac{f(x_1) - f(x_0)}{x_1 - x_0} x_0$$

และหลังจากจัดพจน์ต่างๆจะได้

$$f(x) = \left(\frac{x_1 - x}{x_1 - x_0} \right) f(x_0) + \left(\frac{x_0 - x}{x_0 - x_1} \right) f(x_1) \qquad (2\text{-}13)$$

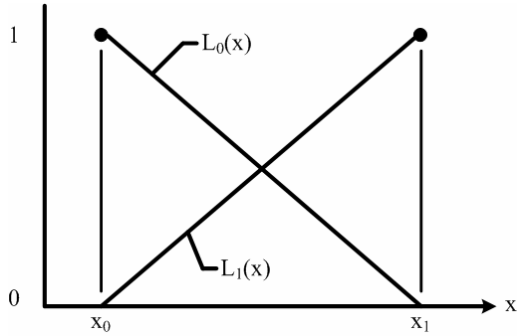
หรือเขียนได้ในรูปแบบดังนี้

$$f(x) = L_0(x) f(x_0) + L_1(x) f(x_1) \qquad (2\text{-}14)$$

โดย

$$L_0(x) = \frac{x_1 - x}{x_1 - x_0} \quad \text{และ} \quad L_1(x) = \frac{x_0 - x}{x_0 - x_1} \qquad (2\text{-}15)$$

ฟังก์ชัน $L_0(x)$ และ $L_1(x)$ นี้เรียกว่าฟังก์ชันประมาณค่าในช่วงของลากรองจ์(Lagrange interpolation functions) ซึ่งในกรีนี้อยู่ในรูปแบบของฟังก์ชันเชิงเส้นที่มีลักษณะการกระจายดังแสดงในภาพที่ 2-20



ภาพที่ 2-20 ฟังก์ชันประมาณค่าภายในของลากรองจ์แบบเชิงเส้น

โดย

$$L_0(x) = \begin{cases} 1 & \text{ณ } x = x_0 \\ 0 & \text{ณ } x = x_1 \end{cases} \qquad (2\text{-}16)$$

และ

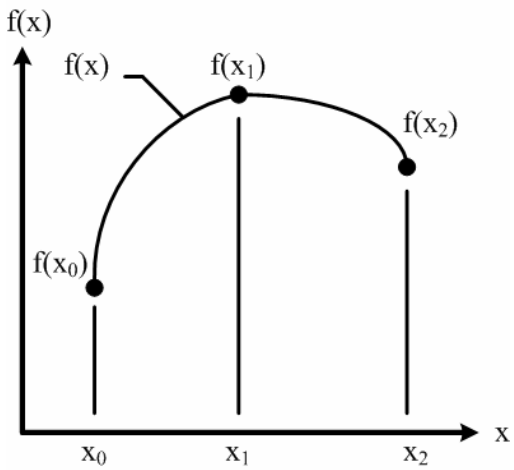
$$L_0(x) = \begin{cases} 0 & \text{ณ } x = x_0 \\ 1 & \text{ณ } x = x_1 \end{cases} \qquad (2\text{-}17)$$

สามารถสรุปได้ว่า

$$L_i(x) = \begin{cases} 1 & \text{ณ } x = x_i \\ 0 & \text{ณ } x \text{ ของตำแหน่งข้อมูลอื่นๆ} \end{cases} \qquad (2\text{-}18)$$

2.3.2 การประมาณค่าในช่วงกำลังสอง

หากมีข้อมูลทั้งหมด 3 ข้อมูล คือ $f(x_0), f(x_1), f(x_2)$ ที่ตำแหน่ง x_0, x_1, x_2 ตามลำดับ ถ้าต้องการประมาณค่าในช่วงระหว่าง x_0 และ x_2 ฟังก์ชันที่เหมาะสมที่สุดจะอยู่ในรูปแบบฟังก์ชันกำลังสอง (Quadratic Function) ดังแสดงในภาพที่ 2-21



ภาพที่ 2-21 ลักษณะฟังก์ชันกำลังสองสำหรับการประมาณค่าในช่วงระหว่าง x_0 และ x_2

ซึ่งอธิบายได้ด้วยสมการ ดังนี้

$$f(x) = ax^2 + bx + c \qquad (2\text{-}19)$$

โดย a,b,c เป็นค่าคงตัวที่ไม่รู้ค่า ซึ่งหาได้จากเงื่อนไขตำแหน่ง x_0, x_1, x_2 ได้ดังนี้

ที่ $x = x_0$: $f(x_0) = ax_0^2 + bx_0 + c$ (2-20)

ที่ $x = x_1$: $f(x_1) = ax_1^2 + bx_1 + c$ (2-21)

ที่ $x = x_2$: $f(x_2) = ax_2^2 + bx_2 + c$ (2-22)

หลังจากได้ค่าคงตัว แล้วแทนกลับลงในสมการ (2-19) แล้วทำการจัดพจน์สมการ (2-19) จะสามารถเขียนให้อยู่ในรูปแบบของฟังก์ชันประมาณค่าในช่วงของลากรองจ์ได้ดังนี้

$$f(x) = L_0(x) f(x_0) + L_1(x) f(x_1) + L_2(x) f(x_2) \quad (2\text{-}23)$$

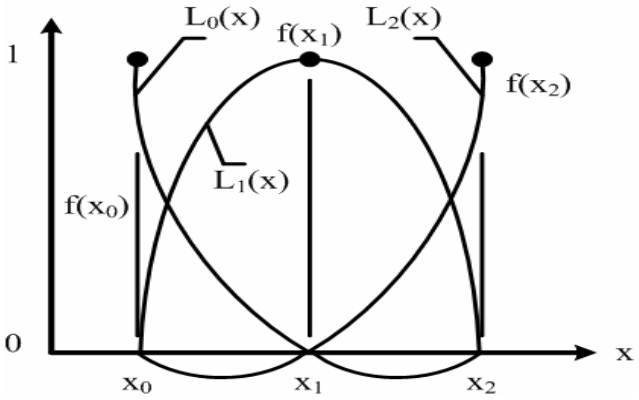
โดย

$$L_0(x) = \frac{(x_2 - x)(x_1 - x)}{(x_2 - x_0)(x_1 - x_0)} \quad (2\text{-}24)$$

$$L_1(x) = \frac{(x_2 - x)(x_0 - x)}{(x_2 - x_1)(x_0 - x_1)} \quad (2\text{-}25)$$

$$L_2(x) = \frac{(x_1 - x)(x_0 - x)}{(x_1 - x_2)(x_0 - x_2)} \quad (2\text{-}26)$$

ฟังก์ชันประมาณค่าในช่วงของลากรองจ์ดังแสดงในสมการ (2-24) ถึง (2-26) ต่างมีคุณสมบัติที่ว่า $L_i = 1$ ที่ x_i และ 0 ที่ตำแหน่ง x ของข้อมูลอื่นๆเช่นเดียวกันกับที่เกิดขึ้นกับฟังก์ชันเชิงเส้นในสมการ (2-18)



ภาพที่ 2-22 ฟังก์ชันประมาณค่าในช่วงของลากรองจ์แบบกำลังสอง

2.3.3 การประมาณค่าในช่วงฟังก์ชันพหุนามโดยทั่วไป

จากหัวข้อ 2.3.1 และ 2.3.2 จะเห็นได้ว่าสามารถสร้างฟังก์ชันพหุนามโดยทั่วไปของลากรองจ์เพื่อการประมาณค่าในช่วงได้ ฟังก์ชังพหุนามดังกล่าวมีลักษณะของการกระจายจะครอบคลุมค่าฟังก์ชัน $f(x_i)$ ณ ตำแหน่ง x_i ต่างๆที่กำหนดมาให้ ฟังก์ชันพหุนามของลากรองจ์นี้สามารถเขียนให้อยู่ในรูปแบบโดยทั่วไปได้ดังนี้

$$f(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \tag{2-27}$$

โดย

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \tag{2-28}$$

โดย $\prod$ เป็นสัญลักษณ์ของการคูณ ตัวอย่างของการใช้สัญลักษณ์นี้ เช่น

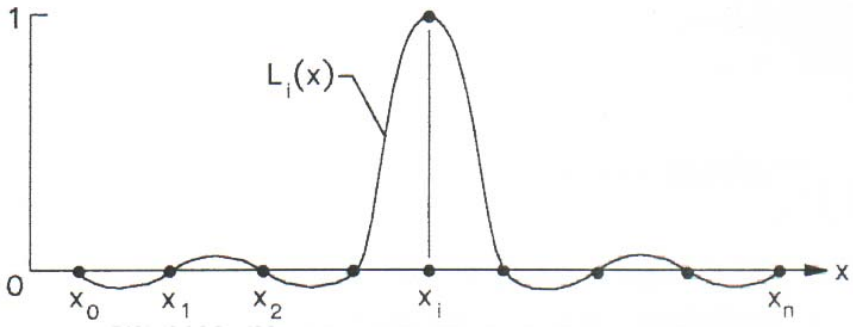
$$\prod_{k=1}^{3} (k+1) = (1+1)(2+1)(3+1) = (2)(3)(4) = 24$$

เป็นต้น ฟังก์ชันการประมาณค่าในช่วงของลากรองจ์ $L_i(x)$ ดังแสดงในสมการ (2-28) สามารถเขียนได้โดยละเอียดดังตัวอย่างต่อไปนี้ หากมี 3 ข้อมูล $(n = 2)$ และหากต้องการหา $L_1(x)$ นั้นคือ $i = 1$

$$L_1(x) = \prod_{\substack{j=0 \\ j \neq 1}}^{3} \frac{x - x_j}{x_1 - x_j} = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

ซึ่งให้ผลเท่ากับสมการ(2-25) เป็นต้น

ฟังก์ชันการประมาณค่าในช่วงของลากรองจ์แบบพหุนามทั่วไปดังแสดงในสมการ (2-28) นี้ยังมีคุณสมบัติเช่นเดียวกับฟังก์ชันการประมาณในช่วงที่สังเกตมาดังแสดงในสมการ (2-18) กล่าวคือ L_i มีค่าเท่ากับ1 ที่ตำแหน่ง x_i และมีค่าเท่ากับ 0 ที่ตำแหน่ง x ของข้อมูลอื่นๆ คุณสมบัติดังกล่าวก่อให้เกิดลักษณะการกระจายของฟังก์ชันดังแสดงในภาพที่ 2-23



ภาพที่ 2-23 ฟังก์ชันประมาณค่าในช่วงของลากรองจ์แบบพหุนามทั่วไป

2.4 การหาค่าความผิดพลาดของรูปสัญญาณ [1, 2, 3]

สัญญาณหลังจากผ่านกระบวนการอินเทอร์โพเลตเพื่อกลับสัญญาณเป็นรูปเดิมแล้ว จะเกิดความเพี้ยนของสัญญาณขึ้น โดยค่าความผิดพลาดหาได้จากค่า Percent root-mean square Difference (PRD) ซึ่งเป็นมาตราฐานการวัดความเพี้ยนของสัญญาณคลื่นไฟฟ้าหัวใจ ดังแสดงด้วยสมการที่ (2-29)

$$PRD = \sqrt{\frac{\sum_{n=0}^{N-1}(x(n)-\hat{x}(n))^2}{\sum_{n=0}^{N-1}x^2(n)}} \times 100 \qquad (2\text{-}29)$$

โดย $x(n)$ คือ สัญญาณขาเข้าที่ยังไม่ผ่านกระบวนการประมวลผล

$\hat{x}(n)$ คือ สัญญาณขาออกที่ผ่านการประมวลผลแล้ว

2.5 ขบวนการทางไฟฟ้าของหัวใจ [11]

การเกิดคลื่นไฟฟ้าหัวใจ เกิดจากการกระตุ้นเซลล์ต่างๆของหัวใจด้วยกระบวนการทางไฟฟ้า ซึ่งมีการนำไฟฟ้าและขั้นตอนการกระตุ้นของเซลล์ต่างๆ ดังต่อไปนี้

Sinoatrial Nodeเป็นจุดเริ่มต้นของการกำเนิดไฟฟ้าในหัวใจที่เกิดขึ้นเป็นจังหวะโดยอัตโนมัติ คลื่นไฟฟ้าจะเคลื่อนตัวไปตามทางเดิน คือ Internodal Tract ไปกระตุ้นเอเตรียมขวาก่อนเอเตรียมซ้าย แรงกระตุ้นจะแผ่ไปทุกทิศทาง แต่ทิศทางจะรวมพุ่งไปทางด้านซ้ายและลงล่างทำให้เกิดคลื่น P ขึ้น ส่วนต้นของคลื่น P เกิดจากการกระตุ้น เอเตรียมขวา และขณะที่เอเตรียมซ้ายถูกกระตุ้น AV Nodeก็ถูกกระตุ้นด้วยคลื่นไฟฟ้าจะเคลื่อนตัวช้าๆ โดยใช้เวลาประมาณ 34 มิลลิวินาที อยู่ในเนื้อเยื่อบริเวณ AV Node (Junctional Tissue) ก่อนจะเคลื่อนที่ไปที่เวนตริเคิลมีระยะเวลาการคลายตัวเพื่อรอรับเลือดจากเอเตรียม

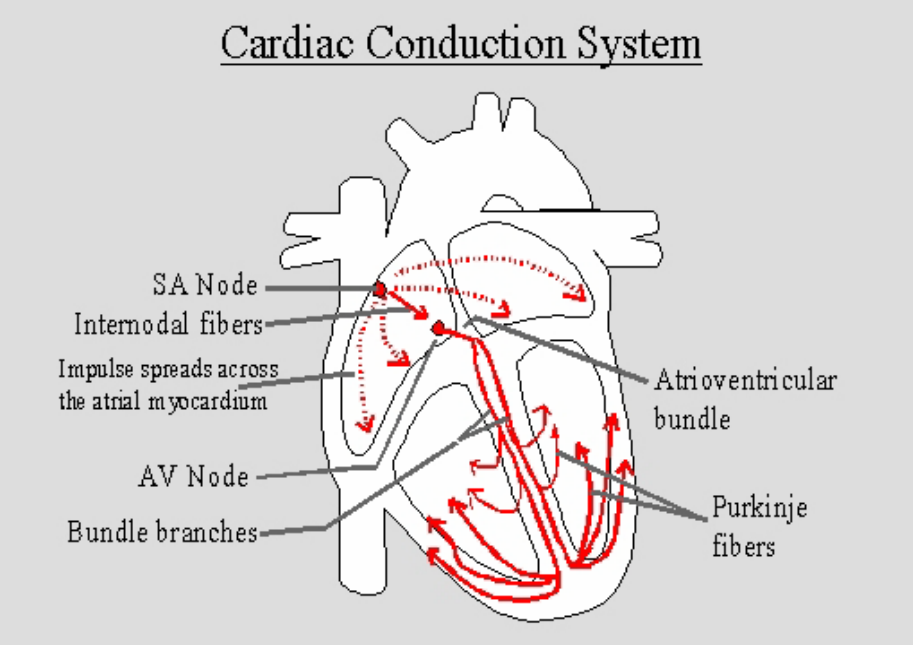
เมื่อ AV Node ถูกกระตุ้นแล้ว คลื่นไฟฟ้าจะเคลื่อนตัวตาม Bundle of HIS แผ่กระจายไปตามแขนงขวาและซ้าย (Right, Left Bundle Branch) เข้าสู่เส้นใยพูร์คินเย ซึ่งมีลักษณะเป็นเส้นใยคล้ายร่างแห กระจายไปทั่วกล้ามเนื้อทำให้เวนตริเคิลหดตัวบีบเลือดออกไปเลี้ยงส่วนต่างๆ ของร่างกายการกระตุ้นที่เวนตริเคิล เริ่มที่ผนังกั้นระหว่างเวนตริเคิลตอนกลางด้านซ้ายถูกกระตุ้นก่อนโดยมีทิศทางของแรงกระตุ้นพุ่งไปทางด้านหน้าและลงล่างใช้เวลา 8 มิลลิวินาทีจุดนี้ทำให้เกิดคลื่นQ ต่อมาเวนตริเคิลด้านขวาถูกกระตุ้นตามด้วยผนังด้านซ้าย คลื่นไฟฟ้าจะวิ่งเป็นทิศทางมุมฉากกับผนังเวนตริเคิลโดยออกจาก เอนโดคาเดียม (Endocardium) ไปพิคาเดียม (Epicardium) เนื่องจาก

ผนังเวนตริเคิลซ้ายหนากว่าเวนตริเคิลขวา ทำให้เกิดแรงรวมที่เกิดขึ้นพุ่งไปทางเวนตริเคิล ซ้าย คือ พุ่งลงล่างและไปทางซ้าย ทำให้เกิดคลื่น QRS Complex

2.6 การเกิดสัญญาณคลื่นไฟฟ้าหัวใจ [11]

การเต้นของหัวใจที่ปกติ จะเริ่มจากตำแหน่ง S-A (S-A: Sinoatrial Node) ภายในหัวใจห้องบนข้างขวา แสดงในภาพที่ 2-24 โดยเกิดสัญญาณไฟฟ้าเป็นจังหวะและจะกระจายไปยังหัวใจห้องบนทั้งสองห้อง ทำให้หัวใจห้องบนหดตัวและบีบเลือดเข้าไปในหัวใจห้องล่างทั้งสองห้อง หลังจากนั้น สัญญาณจะไปที่ตำแหน่ง A-V (A-V: Artioventricular Node) ซึ่งอยู่ที่ฐานหัวใจด้านล่างข้าง ความเร็วในการเคลื่อนที่ของสัญญาณผ่านตำแหน่ง A-V จะช้ามาก ทำให้เกิดการหน่วงเวลาก่อนที่สัญญาณจะไปถึงหัวใจห้องล่าง

สัญญาณที่ออกจากตำแหน่ง A-V จะเคลื่อนที่ไปตามเนื้อเยื่อนำไฟฟ้าในหัวใจ (Bundle of His) และกลุ่มเส้นใยพูร์คินเย (Purkinje Fibers) ซึ่งเป็นแขนงเส้นใยที่นำคลื่นไฟฟ้าแยกไปตามส่วนต่างๆ ของกล้ามเนื้อหัวใจห้องล่าง ทำให้เกิดการหดตัวสูบฉีดเลือดออกจากหัวใจไปเลี้ยงส่วนต่างๆ ของร่างกาย ในผู้ใหญ่ขณะอยู่นิ่ง ตำแหน่ง S-A จะให้กำเนิดสัญญาณไฟฟ้าออกมาด้วยอัตราประมาณ 70 ครั้งต่อนาที ช่วงเวลาที่ใช้ในการส่งผ่านสัญญาณไฟฟ้าจากตำแหน่ง S-A ถึงตำแหน่ง A-V โดยทั่วไปมีค่า 0.12-0.22 วินาที



ภาพที่ 2-24 ระบบเหนี่ยวนำไฟฟ้าของหัวใจ

บทที่ 3
ขั้นตอนการดำเนินงาน

3.1 ศึกษาค้นคว้าข้อมูล

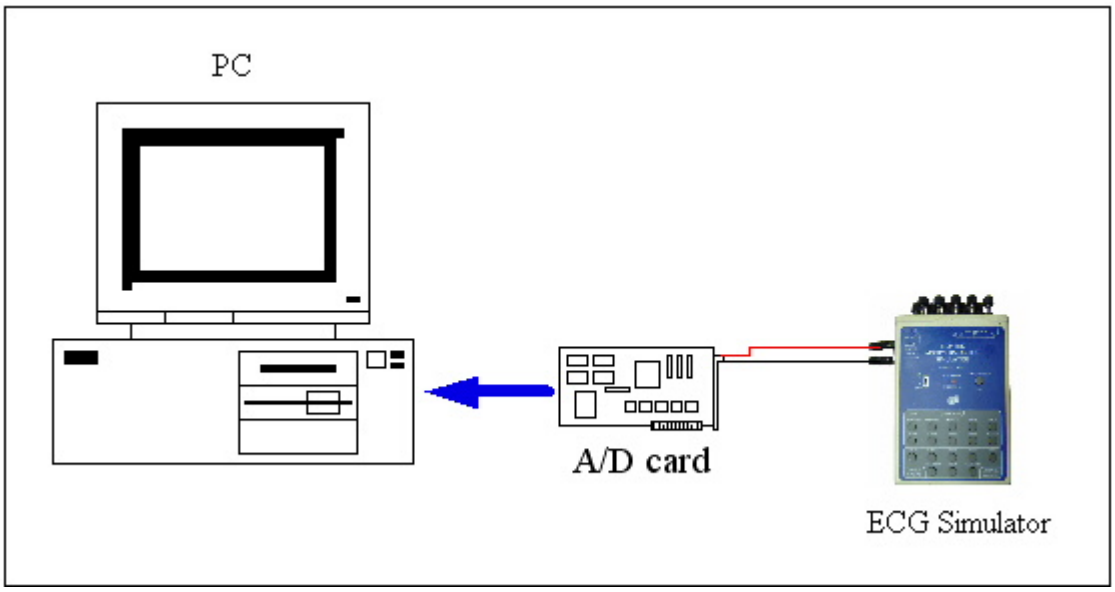
จากการศึกษาทฤษฎีในบทที่ 2 ผู้วิจัยเห็นว่าสามารถนำเอาหลักการของการประมวลผลแบบหลายอัตราสุ่ม มาใช้กับการลดขนาด (Compression) ของคลื่นไฟฟ้าหัวใจได้ ซึ่งลักษณะของสัญญาณคลื่นไฟฟ้าหัวใจที่ได้จากเครื่องวัด เป็นสัญญาณต่อเนื่องทางเวลา ดังนั้นขั้นตอน การออกแบบของงานวิจัยนี้ จะเริ่มจากการแปลงสัญญาณแบบต่อเนื่องทางเวลาให้เป็นสัญญาณไม่ต่อเนื่องทางเวลาก่อน แล้วนำมาผ่านกระบวนการ ประมวลผลเพื่อที่จะลดขนาดของสัญญาณคลื่นไฟฟ้าหัวใจ ในงานวิจัยนี้จะทำการลดขนาดของคลื่นไฟฟ้าหัวใจในปริมาณ 2 เท่า 4 เท่า และ 8 เท่า และจะเพิ่มขนาด (Decompression) สัญญาณคลื่นไฟฟ้าหัวใจกลับทันที โดยจะเพิ่มขนาด 2 เท่า 4 เท่า และ 8 เท่า เช่นกัน โดยขั้นตอนในการออกแบบวิธีการผู้วิจัยกระทำบนโปรแกรม Matlab โดยสัญญาณคลื่นไฟฟ้าหัวใจที่ใช้ในการออกแบบ จะเก็บสัญญาณจากเครื่องจำลองคลื่นไฟฟ้าหัวใจ โดยกำหนดอัตราการเต้นของหัวใจที่ 30, 60, และ120 ครั้งต่อนาที เป็นสัญญาณ อินพุต นำไปเขียนโปรแกรมจำลองการทำงาน บนโปรแกรม Matlab จากนั้นนำสัมประสิทธิ์ที่ใช้จำลองการทำงานไปทำการสร้างจริงโดยใช้บอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31 DSKทำงานที่ Sampling Frequency ที่ 600 Hz และ 1200 Hz หลังจากนั้นนำผลมาคำนวณหาค่า PRD มาเปรียบเทียบกับผลที่ได้จากการจำลองการทำงาน

3.2 การออกแบบการทำงาน

3.2.1 การเก็บสัญญาณ คลื่นไฟฟ้าหัวใจ จากเครื่องจำลองคลื่นไฟฟ้าหัวใจ

ในการออกแบบการทำงานของกระบวนการลดขนาดสัญญาณ คลื่นไฟฟ้าหัวใจ จำเป็นที่จะต้องหาสัญญาณ คลื่นไฟฟ้าหัวใจ มาทดสอบ โดยผู้วิจัยได้ทำการเก็บสัญญาณ คลื่นไฟฟ้าหัวใจ จากเครื่อง จำลองคลื่นไฟฟ้าหัวใจ โดย นำสัญญาณ คลื่นไฟฟ้าหัวใจ จากเครื่องจำลองคลื่นไฟฟ้าหัวใจผ่านการ์ดแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอล โดยสัญญาณ คลื่นไฟฟ้าหัวใจ จะถูกเก็บเข้ามาในเครื่องคอมพิวเตอร์ในรูปแบบของสัญญาณไม่ต่อเนื่อง ในลักษณะของลำดับค่า หรือลำดับข้อมูล เมื่อได้ข้อมูลของสัญญาณแล้วจึงนำมาทำการประมวลผล เครื่องจำลองคลื่นไฟฟ้าหัวใจจะแสดงดังในภาพที่ 3-1 และการเก็บสัญญาณ คลื่นไฟฟ้าหัวใจ จากเครื่องจำลองคลื่นไฟฟ้าหัวใจ จะแสดงดังในภาพที่ 3-2

ภาพที่ 3-1 เครื่องจำลองคลื่นไฟฟ้าหัวใจ (ECG Simulator)



ภาพที่ 3-2 การเก็บสัญญาณจากเครื่องจำลองคลื่นไฟฟ้าหัวใจ ลงในเครื่องคอมพิวเตอร์

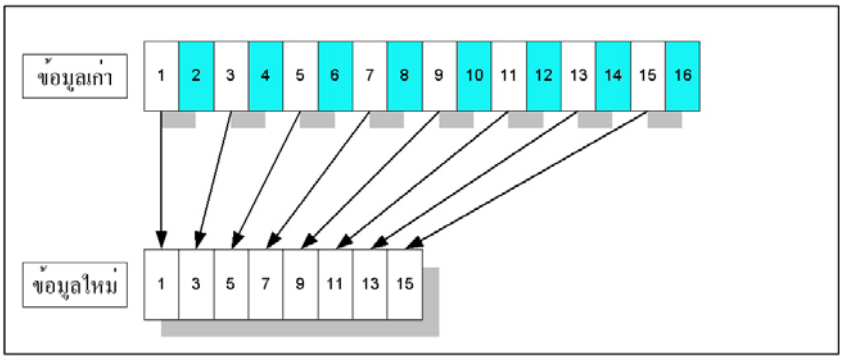
3.2.2 การลดจำนวนข้อมูลด้วยโปรแกรม Matlab

เมื่อได้สัญญาณ คลื่นไฟฟ้าหัวใจ ที่ต้องการนำมาประมวลผลแล้ว ขั้นตอนต่อไปคือ การนำหลักการของทฤษฎี ของการประมวลผลหลายอัตราสุ่ม มาใช้ในการออกแบบสร้างตัวประมวลผล เพื่อที่จะลดขนาดของสัญญาณ คลื่นไฟฟ้าหัวใจ ให้มีจำนวนข้อมูลที่น้อยลง โดยจะใช้ วิธีการเดซิเมชัน มาออกแบบในการลดจำนวนข้อมูลของสัญญาณ คลื่นไฟฟ้าหัวใจ การออกแบบ วิธีเดซิเมชันที่กระทำบนโปรแกรม Matlab ดังแสดงในภาพที่ 3-3

```
%------------------------(Compression)DECIMATION--------------------------%
w=size(we);
nnn=1;
for n=1:2:w(2);    %(Compression 2)
    x(nnn)=we(n);
    nnn=nnn+1;
end
```

ภาพที่ 3-3 การออกแบบบนโปรแกรม Matlab ในส่วนการลดข้อมูล 2 เท่า

ในภาพที่ 3-3 จะเป็นการลดข้อมูล 2 เท่า โดยโปรแกรมจะทำการเก็บข้อมูล ค่าที่ 1 ไปไว้ในค่าตัวแปรใหม่ แล้วเว้นข้อมูลเก่าในค่าที่ 2 ไปเก็บข้อมูลตัวที่ 3 ของข้อมูลเก่ามาไว้ในลำดับที่ 2 ของตัวแปรใหม่ แล้วทำอย่างนี้จนหมดข้อมูล ซึ่งจำนวนข้อมูลในตัวแปรใหม่จะมีจำนวนน้อยกว่าข้อมูลเก่า 2 เท่า



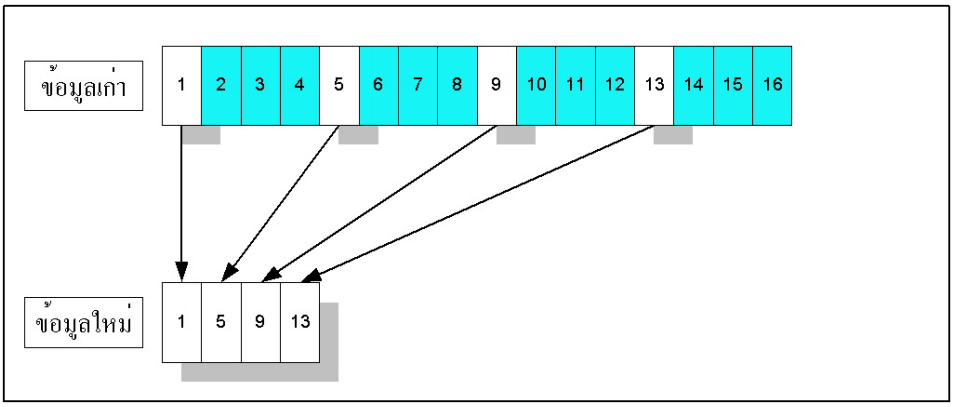
ภาพที่ 3-4 ลักษณะการเก็บข้อมูลของการลดข้อมูลจำนวน 2 เท่า

ในภาพที่ 3-4 แสดงให้เห็นถึงลักษณะการเก็บข้อมูลของการลดข้อมูล 2 เท่าโดยค่าของข้อมูลแต่ละค่า คือลำดับค่าของสัญญาณ คลื่นไฟฟ้าหัวใจ ที่เก็บเข้ามาได้ จะเห็นได้ว่าจำนวนข้อมูลใหม่มีจำนวนที่ลดลงครึ่งหนึ่ง

```
%------------------------(Compression)DECIMATION--------------------------%
w=size(we);
nnn=1;
for n=1:4:w(2);    %(Compression 4)
    x(nnn)=we(n);
    nnn=nnn+1;
end
```

ภาพที่ 3-5 การออกแบบบนโปรแกรม Matlab ในส่วนการลดข้อมูล 4 เท่า

ในภาพที่ 3-5 จะเป็นการลดข้อมูล 4 เท่า โดยโปรแกรมจะทำการเก็บข้อมูล ค่าที่ 1 ไปไว้ในค่าตัวแปรใหม่ แล้วเว้นข้อมูลเก่าในค่าที่ 2-4 ไปเก็บข้อมูลตัวที่ 5 ของข้อมูลเก่ามาไว้ในลำดับที่ 2 ของตัวแปรใหม่ แล้วทำอย่างนี้จนหมดข้อมูล ซึ่งจำนวนข้อมูลในตัวแปรใหม่จะมีจำนวนน้อยกว่าข้อมูลเก่า 4 เท่า



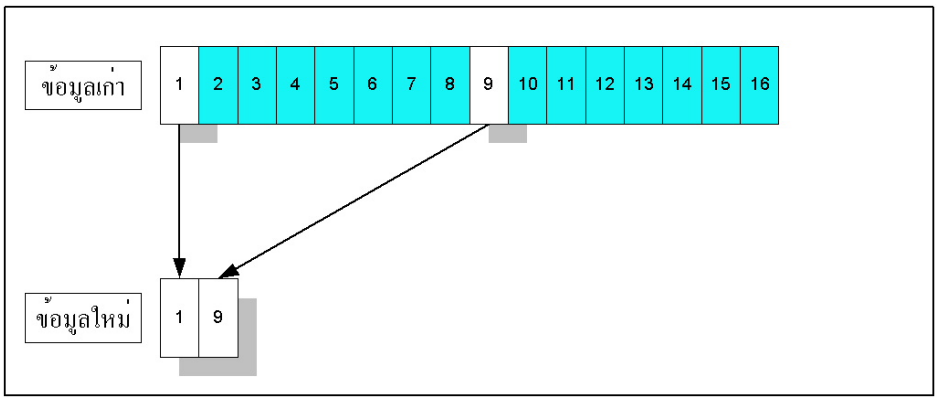
ภาพที่ 3-6 ลักษณะการเก็บข้อมูลของการลดข้อมูลจำนวน 4 เท่า

ในภาพที่ 3-6 แสดงให้เห็นถึงลักษณะการเก็บข้อมูลของการลดข้อมูล 4 เท่าโดยค่าของข้อมูลแต่ละค่า คือลำดับค่าของสัญญาณ คลื่นไฟฟ้าหัวใจ ที่เก็บเข้ามาได้ จะเห็นได้ว่าจำนวนข้อมูลใหม่มีจำนวนที่ลดลง 4 เท่า

```
%-------------------------(Compression)DECIMATION--------------------------%
w=size(we);
nnn=1;
for n=1:8:w(2);    %(Compression 8)
    x(nnn)=we(n);
    nnn=nnn+1;
end
```

ภาพที่ 3-7 การออกแบบบนโปรแกรม Matlab ในส่วนการลดข้อมูล 8 เท่า

ในภาพที่ 3-7 จะเป็นการลดข้อมูล 8 เท่า โดยโปรแกรมจะทำการเก็บข้อมูล ค่าที่ 1 ไปไว้ในค่าตัวแปรใหม่ แล้วเว้นข้อมูลเก่าในค่าที่ 2-8 ไปเก็บข้อมูลตัวที่ 9 ของข้อมูลเก่ามาไว้ในลำดับที่ 2



ภาพที่ 3-8 ลักษณะการเก็บข้อมูลของการลดข้อมูลจำนวน 8 เท่า

ในภาพที่ 3-8 แสดงให้เห็นถึงลักษณะการเก็บข้อมูลของการลดข้อมูล 8 เท่าโดยค่าของข้อมูลแต่ละค่า คือลำดับค่าของสัญญาณ คลื่นไฟฟ้าหัวใจ ที่เก็บเข้ามาได้ จะเห็นได้ว่าจำนวนข้อมูลใหม่มีจำนวนที่ลดลง 8 เท่า

จะเห็นได้ว่าปริมาณข้อมูลใหม่ที่ได้มีปริมาณที่น้อยกว่าข้อมูลเก่า ซึ่งในจุดนี้คือข้อมูลที่นำเอาไปใช้ประโยชน์ เช่น การบันทึกในหน่วยความจำ การส่งข้อมูลระยะไกล เป็นต้น

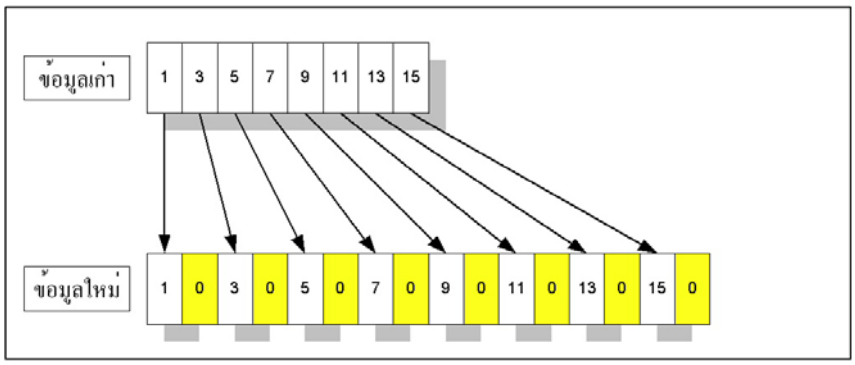
3.2.3 การเพิ่มข้อมูลด้วยโปรแกรม Matlab

หลังจากสามารถลดขนาดข้อมูลของสัญญาณ คลื่นไฟฟ้าหัวใจ ได้แล้ว การคืนค่าเดิมให้สัญญาณ คลื่นไฟฟ้าหัวใจ โดยที่ให้ปริมาณข้อมูลและคงค่าเดิมของสัญญาณไว้ เป็นสิ่งสำคัญมาก เพราะว่าการผิดเพี้ยนของสัญญาณ คลื่นไฟฟ้าหัวใจ ในปริมาณที่มาก ย่อมทำให้การอ่านค่าของสัญญาณ คลื่นไฟฟ้าหัวใจ ผิดเพี้ยนไปด้วย ในส่วนนี้จะแสดงการออกแบบการเพิ่มข้อมูลของสัญญาณ คลื่นไฟฟ้าหัวใจ ให้ปริมาณเท่ากับสัญญาณ คลื่นไฟฟ้าหัวใจ เดิม โดยจะนำค่า 0 ไปใส่แทรกระหว่างข้อมูลใหม่โดยจะแทน 0 เข้าไปในปริมาณที่เท่ากับจำนวนที่ลดขนาด ส่วนการออกแบบการคืนค่าเดิมให้กับสัญญาณ คลื่นไฟฟ้าหัวใจ นั้นจะกล่าวในหัวข้อถัดไป

```
%--------------------(Decompression) INTERPOLATION-------------------------%
y=we()*0;
uu = size(x);
nnn=1;
for n=1:1:uu(2);
    y(nnn)= x(n);
    nnn=nnn+2;
end
```

ภาพที่ 3-9 การออกแบบบนโปรแกรม Matlab ในส่วนการเพิ่มข้อมูล 2 เท่า

ในภาพที่ 3-9 แสดงการออกแบบการเพิ่มปริมาณข้อมูลบนโปรแกรม Matlab โดยโปรแกรมจะทำการแทรกค่า 0 ระหว่างข้อมูลของค่าเดิม ในโปรแกรมนี้เป็นการเพิ่มข้อมูล 2 เท่า



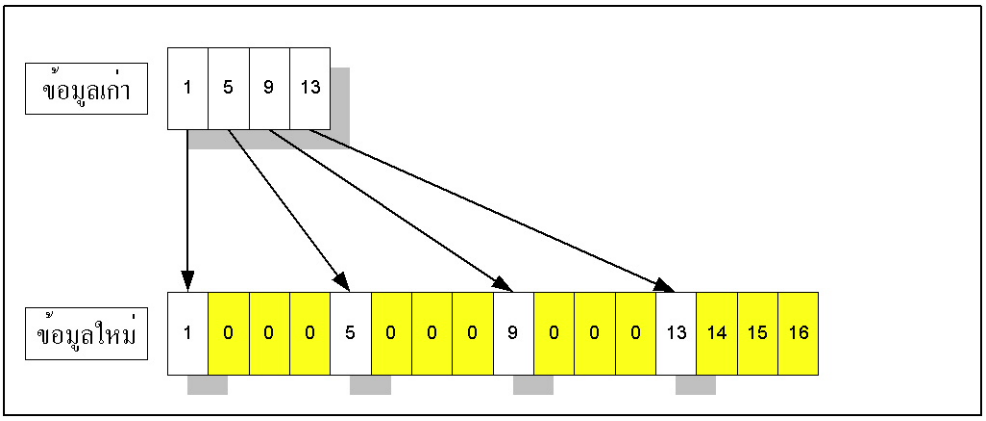
ภาพที่ 3-10 ลักษณะการแทรก 0 ของการเพิ่มข้อมูลจำนวน 2 เท่า

จากภาพที่ 3-10 จะเห็นได้ว่าผลของการแทรก 0 เข้าไประหว่างข้อมูลเก่าทำให้ปริมาณของข้อมูลเพิ่มขึ้น เป็น 2 เท่า

```
%--------------------(Decompression) INTERPOLATION--------------------------%
y=we()*0;
uu = size(we);
nnn=1;
for n=1:4:uu(2);
    y(nnn)= we(n);
    nnn=nnn+4;
end
```

ภาพที่ 3-11 การออกแบบบนโปรแกรม Matlab ในส่วนการเพิ่มข้อมูล 4 เท่า

ในภาพที่ 3-11 แสดงการออกแบบการเพิ่มปริมาณข้อมูลบนโปรแกรม Matlab โดยโปรแกรมจะทำการแทรกค่า 0 ระหว่างข้อมูลของค่าเดิม ในโปรแกรมนี้เป็นการเพิ่มข้อมูล 4 เท่า



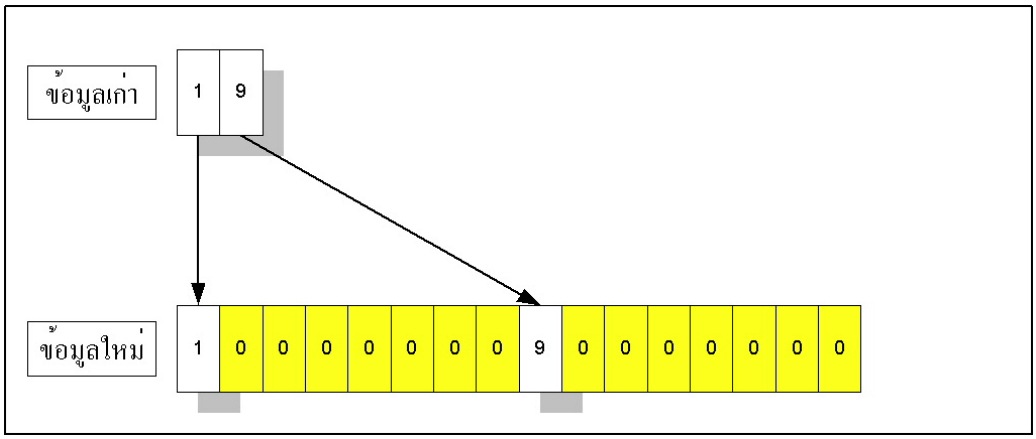
ภาพที่ 3-12 ลักษณะการแทรก 0 ของการเพิ่มข้อมูลจำนวน 4 เท่า

จากภาพที่ 3-12 ผลของการแทรก 0 เข้าไประหว่างข้อมูลเก่าทำให้ปริมาณของข้อมูลเพิ่มขึ้นเป็น 4 เท่า

```
%--------------------(Decompression) INTERPOLATION--------------------------%
y=we()*0;
uu = size(we);
nnn=1;
for n=1:8:uu(2);
    y(nnn)= we(n);
    nnn=nnn+8;
end
```

ภาพที่ 3-13 การออกแบบบนโปรแกรม Matlab ในส่วนการเพิ่มข้อมูล 8 เท่า

ในภาพที่ 3-13 แสดงการออกแบบการเพิ่มปริมาณข้อมูลบนโปรแกรม Matlab โดยโปรแกรมจะทำการแทรกค่า 0 ระหว่างข้อมูลของค่าเดิม ในโปรแกรมนี้เป็นการเพิ่มข้อมูล 8 เท่า



ภาพที่ 3-14 ลักษณะการแทรก 0 ของการเพิ่มข้อมูลจำนวน 8 เท่า

จากภาพที่ 3-14 ผลของการแทรก 0 เข้าไประหว่างข้อมูลเก่าทำให้ปริมาณของข้อมูลเพิ่มขึ้นเป็น 8 เท่า

เมื่อเพิ่มปริมาณข้อมูลของสัญญาณ คลื่นไฟฟ้าหัวใจ ให้มีขนาดกลับไปเท่าสัญญาณเดิม แต่สัญญาณที่ได้นั้นยังเป็นสัญญาณที่ไม่ถูกต้องเพราะว่าค่าที่เพิ่มเข้าไปนั้นเป็นค่า 0 ค่าสัญญาณที่ได้ในตอนนี้จึงยังเป็นค่าสัญญาณที่ผิดเพี้ยน

3.2.4 การสร้างสัญญาณกลับด้วยโปรแกรม Matlab

หลังจากที่ได้สัญญาณ คลื่นไฟฟ้าหัวใจ ที่มีขนาดของสัญญาณเท่าเดิมกับสัญญาณ คลื่นไฟฟ้าหัวใจ ก่อนที่จะถูกลดขนาดแล้ว แต่ยังเป็นค่าสัญญาณที่ไม่ถูกต้อง ในการพยากรณ์ค่าสัญญาณ คลื่นไฟฟ้าหัวใจ คืนให้มีค่าที่ถูกต้อง ในที่นี้จะใช้ ทฤษฎีของ ฟังก์ชันพหุนามของลากรองจ์ (Lagrange Interpolation Polynomials) โดยจะใช้วิธีการประมาณค่าแบบกำลังสอง โดยจะใช้สมการที่ 2-23 ถึง2-26 ในบทที่ 2 ในการพยากรณ์ค่าสัญญาณ คลื่นไฟฟ้าหัวใจ ในการใช้วิธีการประมาณค่ากำลังสอง เพื่อที่จะหาค่ากลางระหว่างข้อมูล ตำแหน่งที่ 1 และตำแหน่งที่ 2 จำเป็นที่จะต้องใช้ข้อมูลในตำแหน่งที่ 3 ด้วยในการคำนวณ ดังนั้นเมื่อถึงตำแหน่งข้อมูลสองหลักสุดท้ายจึงไม่สามารถคำนวณค่าได้ ทำให้จำเป็นที่จะต้องเสียข้อมูลในช่วงสุดท้ายไป

3.2.4.1 การคำนวณเพื่อหาค่ามาใส่ในสมการของข้อมูลที่เพิ่มขึ้น 2 เท่า

จากสมการที่ 2-23 จะต้องหาค่า $L_0(x), L_1(x), L_2(x)$ มาใส่ในสมการโดยสามารถหาค่า $L_0(x), L_1(x), L_2(x)$ ได้จากสมการที่ 2-24 ถึง 2-26

หาค่า $L_0(x)$ จากสมการที่ 2-24

โดยให้ค่า $x_0 = 1, x_1 = 3, x_2 = 5$

เมื่อแทนค่าตัวแปรในสมการที่ 2-24 จะได้

$$L_0(x) = \frac{(5-x)(3-x)}{(5-1)(3-1)} = \frac{(5-x)(3-x)}{8} \qquad (3\text{-}1)$$

หาค่า $L_1(x)$ จากสมการที่ 2-25

โดยให้ค่า $x_0 = 1, x_1 = 3, x_2 = 5$

เมื่อแทนค่าตัวแปรในสมการที่ 2-25 จะได้

$$L_1(x) = \frac{(5-x)(1-x)}{(5-3)(1-3)} = \frac{(5-x)(1-x)}{-4} \qquad (3\text{-}2)$$

หาค่า $L_2(x)$ จากสมการที่ 2-25

โดยให้ค่า $x_0 = 1, x_1 = 3, x_2 = 5$

เมื่อแทนค่าตัวแปรในสมการที่ 2-26 จะได้

$$L_2(x) = \frac{(3-x)(1-x)}{(3-5)(1-5)} = \frac{(3-x)(1-x)}{8} \qquad (3\text{-}3)$$

เมื่อแทนค่า $x = 2$ ในสมการ (3-1) ถึง (3-3) จะได้

$$L_0(2) = 0.375, L_1(2) = 0.75, L_2(2) = -0.125$$

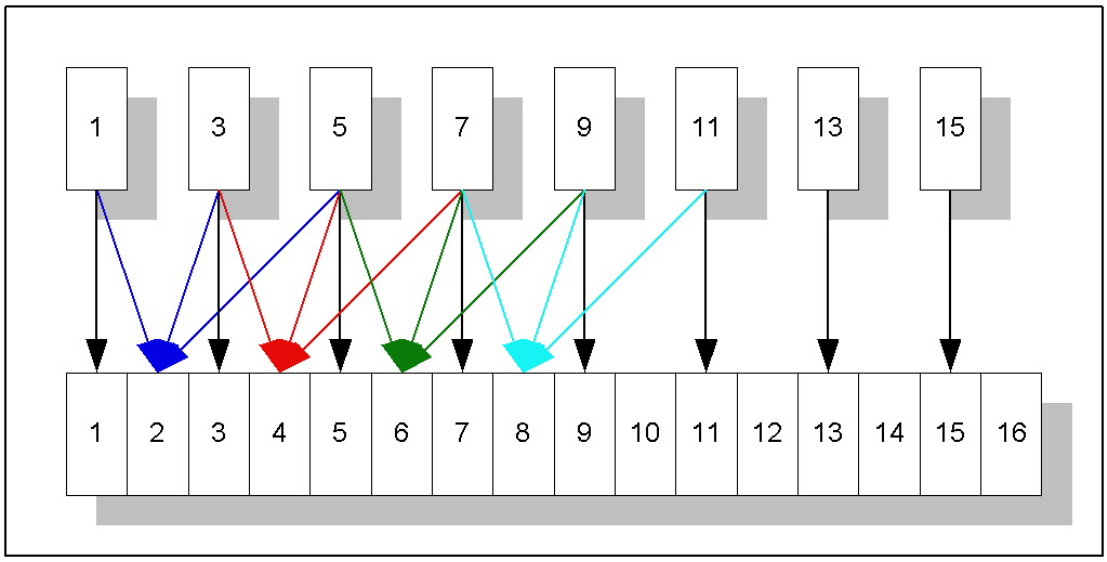
นำค่าที่ได้ไปแทนในสมการ 2-23 จะได้

$$f(2) = (0.375\, f(x_0)) + (0.75\, f(x_1)) + (-0.125\, f(x_2)) \qquad (3\text{-}4)$$

นำสมการที่ (3-4) ไปใช้ในโปรแกรม Matlab ดังแสดงในภาพที่ 3-15

```
%---------------------------LAGRANGE---------------------------------------%
f = [0 0 0 ];
ll = size(y);
nnn=1;yy=0;
for n = 1:2:(ll(2)-4);
    f(1)=y(nnn);
    nnn=nnn+1; nnn=nnn+1;
    f(2)=y(nnn);
    nnn=nnn+1; nnn=nnn+1;
    f(3)=y(nnn);
    yy =((0.375*f(1))+(0.75*f(2))+(-0.125*f(3)));
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    y(nnn)=yy; nnn=nnn+1;
end
```

ภาพที่ 3-15 โปรแกรม Matlab ในส่วน Lagrange ของการเพิ่มข้อมูล 2 เท่า



ภาพที่ 3-16 การใช้ข้อมูลของการคำนวณหาค่าที่แทรกข้อมูลเข้าไป 2 เท่า

ภาพที่ 3-16 แสดงให้เห็นถึงการนำข้อมูลที่ถูกลดขนาดแล้วมาหาค่าเฉลี่ยโดยใช้การประมาณค่ากำลังสองของฟังก์ชันพหุนามของลากรองจ์ จากภาพ 3-16 การคำนวณหาค่าในช่องที่ 2 จะต้องใช้ข้อมูล ที่ 1, 3 และ 5 เป็นต้น

3.2.4.2 การคำนวณเพื่อหาค่ามาใส่ในสมการของข้อมูลที่เพิ่มขึ้น 4 เท่า

การคำนวณหาค่ามาใส่ในสมการของข้อมูลเพิ่มขึ้น 4 เท่านี้ จะกระทำเหมือนกับการคำนวณแบบ 2 เท่า แต่จะต้องเพิ่มสมการเข้าไปในส่วนโปรแกรม Matlab อีกเป็น 3 สมการ สามารถคำนวณหาได้ดังนี้

เมื่อ ค่า $x_0 = 1, x_1 = 5, x_2 = 9$ นำไปแทนค่าในสมการที่ 2-24 ถึง 2-26 จะได้ค่าดังนี้

$$L_0(x) = \frac{(9-x)(5-x)}{(9-1)(5-1)} = \frac{(9-x)(5-x)}{32} \tag{3-5}$$

$$L_1(x) = \frac{(9-x)(1-x)}{(9-5)(1-5)} = \frac{(9-x)(1-x)}{-16} \tag{3-6}$$

$$L_2(x) = \frac{(5-x)(1-x)}{(5-9)(1-9)} = \frac{(5-x)(1-x)}{32} \tag{3-7}$$

จากนั้นแทนค่า $x = 2, x = 3, x = 4$ แล้วนำไปเขียนให้อยู่ในรูปของสมการที่ 2-23 จะได้

$$f(2) = (0.65625 f(x_0)) + (0.4375 f(x_1)) + (-0.09375 f(x_2)) \tag{3-8}$$

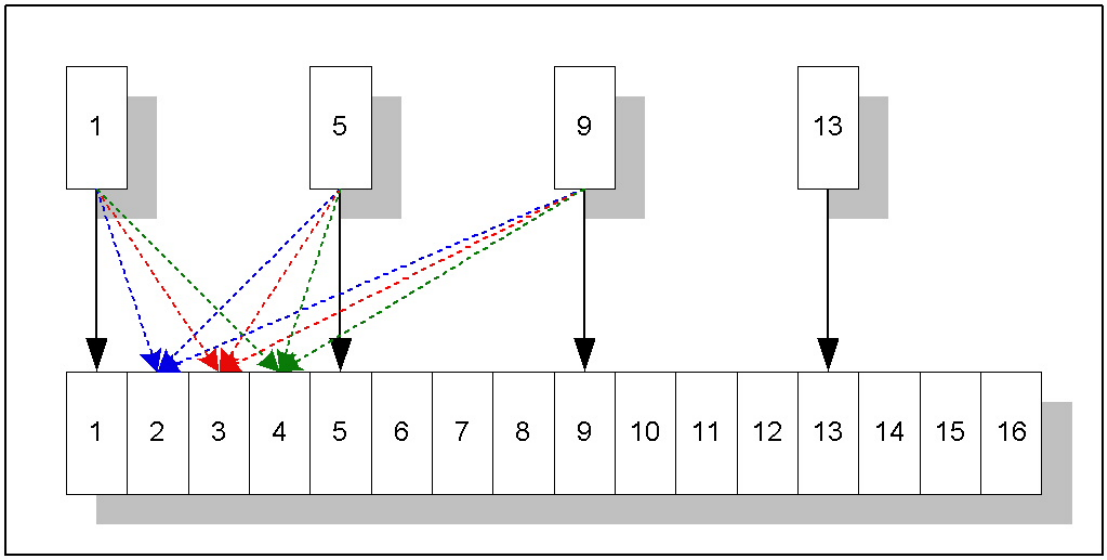
$$f(3) = (0.375 f(x_0)) + (0.75 f(x_1)) + (-0.125 f(x_2)) \tag{3-9}$$

$$f(4) = (0.15625 f(x_0)) + (0.9375 f(x_1)) + (-0.09375 f(x_2)) \tag{3-10}$$

นำสมการที่ (3-8) ถึง (3-10) ไปใช้ในโปรแกรม Matlab ดังแสดงในภาพที่ 3-17

```
%----------------------------LAGRANGE-------------------------------------%
f = [0 0 0 ];
ll = size(y);
nnn=1;yy =0;yy1=0;yy2=0;
for n = 1:4:(ll(2)-8);
    f(1)=y(nnn);
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    f(2)=y(nnn);
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    f(3)=y(nnn);
    yy  =((0.65625*f(1))+(0.4375*f(2))+(-0.09375*f(3)));
    yy1 =((0.375*f(1))+(0.75*f(2))+(-0.125*f(3)));
    yy2 =((0.15625*f(1))+(0.9375*f(2))+(-0.09375*f(3)));
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    y(nnn)=yy;  nnn=nnn+1;
    y(nnn)=yy1; nnn=nnn+1;
    y(nnn)=yy2; nnn=nnn+1;
    end
```

ภาพที่ 3-17 โปรแกรม Matlab ในส่วน Lagrange ของการเพิ่มข้อมูล 4 เท่า



ภาพที่ 3-18 การใช้ข้อมูลของการคำนวณหาค่าที่แทรกข้อมูลเข้าไป 4 เท่า

3.2.4.3 การคำนวณเพื่อหาค่ามาใส่ในสมการของข้อมูลที่เพิ่มขึ้น 8 เท่า

การคำนวณทำแบบเดียวกับ 2 หัวข้อที่ผ่านมา แต่เพิ่มสมการเข้าไปเป็น 7 สมการ สามารถคำนวณหาได้ดังนี้

เมื่อ ค่า $x_0 = 1, x_1 = 9, x_2 = 17$ นำไปแทนค่าในสมการที่ 2-24 ถึง 2-26 จะได้ค่าดังนี้

$$L_0(x) = \frac{(17-x)(9-x)}{(17-1)(9-1)} = \frac{(17-x)(9-x)}{128} \tag{3-11}$$

$$L_1(x) = \frac{(17-x)(1-x)}{(17-9)(1-9)} = \frac{(17-x)(1-x)}{-64} \tag{3-12}$$

$$L_2(x) = \frac{(9-x)(1-x)}{(9-17)(1-17)} = \frac{(9-x)(1-x)}{128} \tag{3-13}$$

จากนั้นแทนค่า $x = 2, x = 3, x = 4, x = 5, x = 6, x = 7, x = 8$ แล้วนำไปเขียนให้อยู่ในรูปของสมการที่ 2-23 จะได้

$$f(2) = (0.8203125\, f(x_0)) + (0.234375\, f(x_1)) + (-0.0546875\, f(x_2)) \tag{3-14}$$

$$f(3) = (0.65625\, f(x_0)) + (0.4375\, f(x_1)) + (-0.09375\, f(x_2)) \tag{3-15}$$

$$f(4) = (0.5078125\, f(x_0)) + (0.609375\, f(x_1)) + (-0.1171875\, f(x_2)) \tag{3-16}$$

$$f(5) = (0.375\, f(x_0)) + (0.75\, f(x_1)) + (-0.125\, f(x_2)) \tag{3-18}$$

$$f(6) = (0.2578125\, f(x_0)) + (0.859375\, f(x_1)) + (-0.1171875\, f(x_2)) \tag{3-19}$$

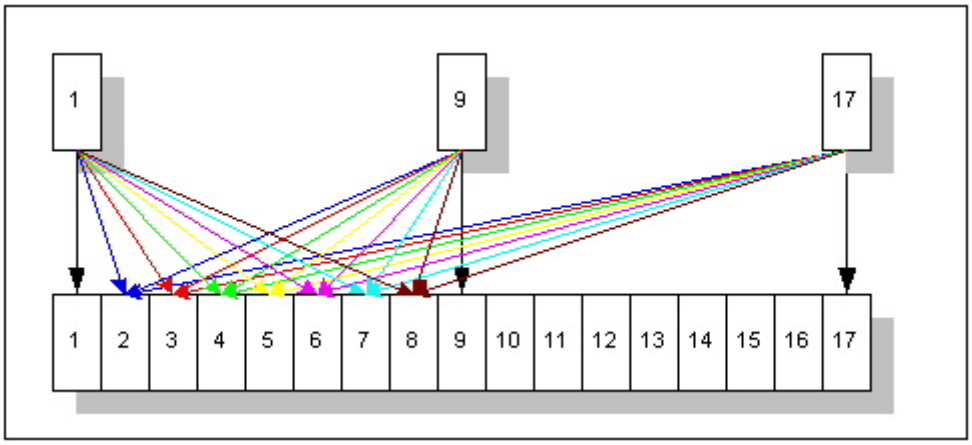
$$f(7) = (0.15625\, f(x_0)) + (0.9375\, f(x_1)) + (-0.09375\, f(x_2)) \tag{3-20}$$

$$f(8) = (0.0703125\, f(x_0)) + (0.984375\, f(x_1)) + (-0.0546875\, f(x_2)) \tag{3-21}$$

นำสมการที่ (3-14) ถึง (3-21) ไปใช้ในโปรแกรม Matlab ดังแสดงในภาพที่ 3-19

```
%----------------------------LAGRANGE---------------------------------------%
f = [0 0 0 ];
ll = size(y);
nnn=1;
yy =0;yy1=0;yy2=0;yy4=0;yy5=0;yy6=0;
for n = 1:8:(ll(2)-18);
    f(1)=y(nnn);
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    f(2)=y(nnn);
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    nnn=nnn+1; nnn=nnn+1; nnn=nnn+1; nnn=nnn+1;
    f(3)=y(nnn);
    yy  =((0.8203125*f(1))+(0.234375*f(2))+(-0.0546875*f(3)));
    yy1 =((0.65625*f(1))+(0.4375*f(2))+(-0.09375*f(3)));
    yy2 =((0.5078125*f(1))+(0.609375*f(2))+(-0.1171875*f(3)));
    yy3 =((0.375*f(1))+(0.75*f(2))+(-0.125*f(3)));
    yy4 =((0.2578125*f(1))+(0.859375*f(2))+(-0.1171875*f(3)));
    yy5 =((0.15625*f(1))+(0.9375*f(2))+(-0.09375*f(3)));
    yy6 =((0.0703125*f(1))+(0.984375*f(2))+(-0.0546875*f(3)));
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    nnn=nnn-1; nnn=nnn-1; nnn=nnn-1;
    y(nnn)=yy;  nnn=nnn+1;
    y(nnn)=yy1; nnn=nnn+1;
    y(nnn)=yy2; nnn=nnn+1;
    y(nnn)=yy3; nnn=nnn+1;
    y(nnn)=yy4; nnn=nnn+1;
    y(nnn)=yy5; nnn=nnn+1;
    y(nnn)=yy6; nnn=nnn+1;
end
%----------------------------------------------------------------------------%
```

ภาพที่ 3-19 โปรแกรม Matlab ในส่วน Lagrange ของการเพิ่มข้อมูล 8 เท่า



ภาพที่ 3-20 การใช้ข้อมูลของการคำนวณหาค่าที่แทรกข้อมูลเข้าไป 8 เท่า

3.2.5 การหาค่า (Percent Root mean square Different: PRD) บนโปรแกรม Matlab

เมื่อได้สัญญาณ คลื่นไฟฟ้าหัวใจ ที่ผ่านการกระบวนการคืนค่ามาแล้ว จะต้องนำสัญญาณที่ได้ มาคำนวณเพื่อค่าผิดพลาดจากค่า คลื่นไฟฟ้าหัวใจ ต้นฉบับ โดยอาศัยสมการที่ (2-29) มาช่วยในการหาค่า โดยนำมาเขียนในโปรแกรม Matlab ได้ดังแสดงในภาพที่ 3-21

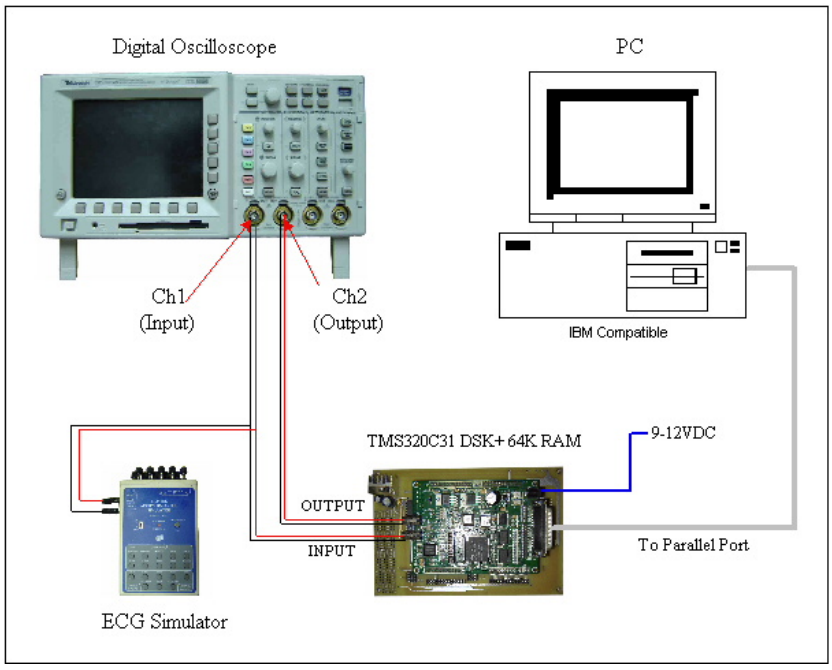
```
%-----------------------------------PRD-----------------------------------%
Y1=0;
YY=0;
for n=1:1:2995
    YY = (we(n)-y(n))^2;
    Y1 = Y1+YY;
end

YY=0;
Y2=0;
for n=1:1:2998
    YY = we(n)^2;
    Y2 = Y2+YY;
end

Y3 = sqrt(Y1/Y2);
PRD = (Y3*100)
%-------------------------------------------------------------------------%
```

ภาพที่ 3-21 การหาค่า PRD ของสัญญาณ คลื่นไฟฟ้าหัวใจ บนโปรแกรม Matlab

3.3 การสร้างจริงบนบอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31



ภาพที่ 3-22 ลักษณะการต่อการทดลองจริงบนบอร์ด TMS320C31 DSK

จากภาพที่ 3-22 สัญญาณคลื่นไฟฟ้าหัวใจจากเครื่อง จำลองคลื่นไฟฟ้าหัวใจ จะถูกป้อนเข้าทางช่องสัญญาณ อินพุต ของบอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31 DSK และจะแสดงผลของสัญญาณโดยต่อเข้ากับ Digital Oscilloscope ทางช่องรับสัญญาณที่ 1 เมื่อบอร์ด TMS320C31 DSK รับสัญญาณคลื่นไฟฟ้าเข้ามาแล้ว จะนำสัญญาณเข้ามาประมวลผล โดยคำสั่งในการประมวลผลจะรับมาจากเครื่องคอมพิวเตอร์ ผ่านพอร์ตขนาน (Parallel Port) สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการประมวลผลในการลดขนาดสัญญาณแล้วจะถูกนำเข้าไปเก็บที่หน่วยความจำภายนอก ในชุดที่ 1 หลัง จากนั้นจะนำสัญญาณที่ถูกเก็บในหน่วยความจำชุดที่ 1 มาทำการประมวลผลอีกครั้งในการคืนค่าสัญญาณให้กับคลื่นไฟฟ้าหัวใจ จากนั้นจะนำผลของการประมวลผลที่ได้ไปเก็บในหน่วยความจำชุดที่ 2 และจะส่งผลของสัญญาณที่ได้ออกทางช่องสัญญาณ เอาท์พุต ของบอร์ด TMS320C31 และนำผลที่ได้ไปแสดงโดยจะต่อกับ Digital Oscilloscope ทางช่องรับสัญญาณที่ 2

3.4 การประเมินระบบ

ในการประเมินระบบของการทดลองในวิทยานิพนธ์นี้ จะดูที่ผลของค่าความผิดพลาดของคลื่นไฟฟ้าหัวใจ ที่ผ่านขั้นตอนการบีบอัดและขยายสัญญาณกลับคืนมา นำเอามาเปรียบเทียบกับสัญญาณก่อนที่ผ่านระบบ โดยจะดูว่าค่าความผิดพลาดของระบบ ค่า (PRD) เกิน 5% หรือไม่ ถ้าค่าความผิดพลาดของรูปคลื่นไฟฟ้าหัวใจผิดพลาดไม่เกิน 5% แสดงว่าระบบที่ออกแบบมาสามารถใช้งานได้ โดยที่ค่า (PRD) เป็นค่าที่ใช้ในการหาความผิดพลาดของการประมวลผลคลื่นไฟฟ้าหัวใจ ซึ่งค่าความผิดพลาดที่แพทย์ยอมรับได้นั้นจะอยู่ที่ค่า (PRD) ไม่เกิน 5%

3.5 การเปรียบเทียบผลการทดลอง

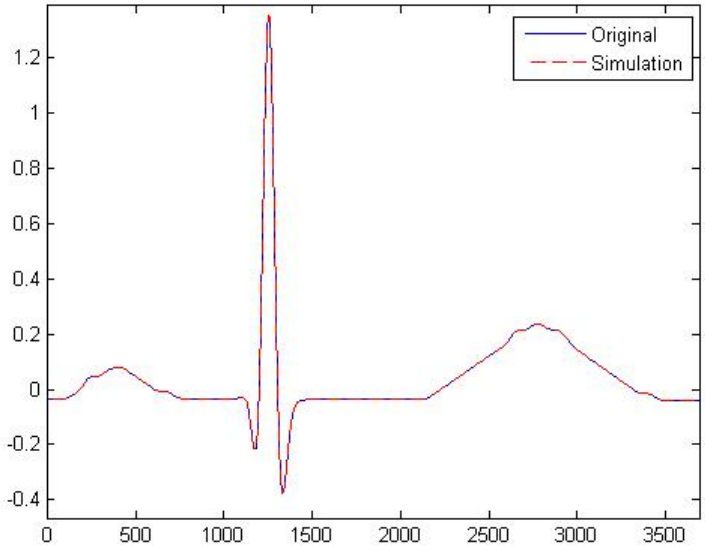
การเปรียบเทียบผลการทดลองจะเป็นการเปรียบเทียบกัน ของค่าความผิดพลาดระหว่างการจำลองการทำงานบนโปรแกรม Matlab กับ ผลความผิดพลาดของการทดลองสร้างจริงบนบอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31 ที่ใช้ความถี่ในการสุ่มสัญญาณที่ 600 Hz และ 1200 Hz ดูว่าผลของการจำลองการทำงานกับผลของการสร้างจริงมีความแตกต่างกันอย่างไร

บทที่ 4
การทดลองและผลการทดลอง

การทดลองจะแบ่งออกเป็น 2 ส่วน คือในส่วนแรกคือส่วนของการจำลองการทำงานของระบบบนโปรแกรม Matlab ในส่วนนี้จะนำสัญญาณคลื่นไฟฟ้าที่เก็บจากเครื่องจำลองคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30, 60, และ 120 ครั้งต่อนาที มาทำการประมวลผลโดยจะนำสัญญาณคลื่นไฟฟ้าหัวใจ 1 ลูกคลื่น มาทำการประมวลผลระบบบีบอัดและขยายคลื่นไฟฟ้าหัวใจที่ได้ทำการออกแบบไว้โดย จะสั่งให้โปรแกรมแสดงผลออกมาเป็นกราฟคลื่นไฟฟ้าหัวใจและกราฟแสดงค่าผิดพลาดเพื่อแสดงความผิดพลาดของการสัญญาณกลับ และให้โปรแกรมทำการคำนวณค่าเปอร์เซนความผิดพลาด (PRD) ออกมาเป็นค่าตัวเลข ในส่วนที่สองคือส่วนของการนำระบบไปทำการสร้างจริงบนบอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31 DSK โดยการทดลองในส่วนนี้จะทำการตั้งอัตราการสุ่มของบอร์ดประมวลผลสัญญาณดิจิตอลที่ 600 Hzและ 1200 Hz และใช้อัตราการเต้นของหัวใจของเครื่องจำลองคลื่นไฟฟ้าหัวใจ เท่ากับที่ใช้ในการจำลองการทำงานบนโปรแกรม Matlab คือ 30, 60 และ 120 ครั้งต่อนาที เช่นกัน แล้วจะนำสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการประมวลผลแล้ว และสัญญาณคลื่นไฟฟ้าหัวใจต้นฉบับมาทำการคำนวณเพื่อหาค่าเปอร์เซนความผิดพลาด เพื่อนำมาเปรียบเทียบผลกับการจำลองการทำงานของระบบบนโปรแกรม Matlab

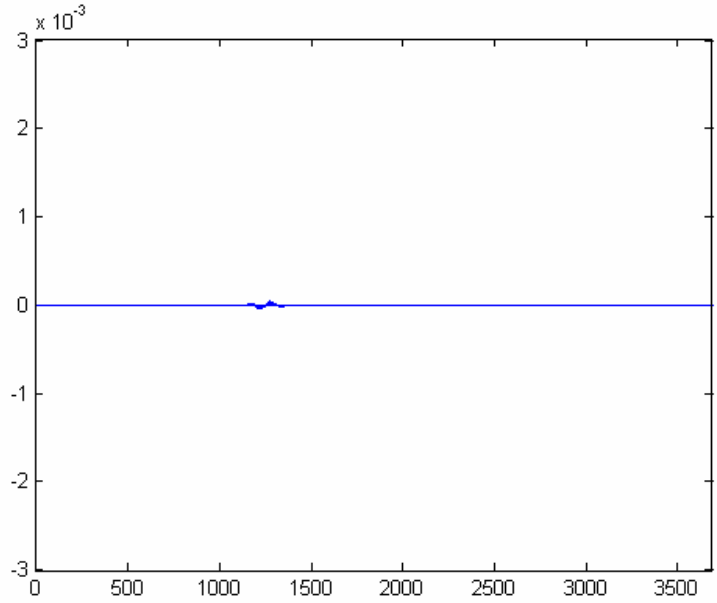
4.1 ผลการจำลองการประมวลผลด้วยโปรแกรม Matlab

4.1.1 อัตราการเต้นของหัวใจที่ 30 ครั้งต่อนาที



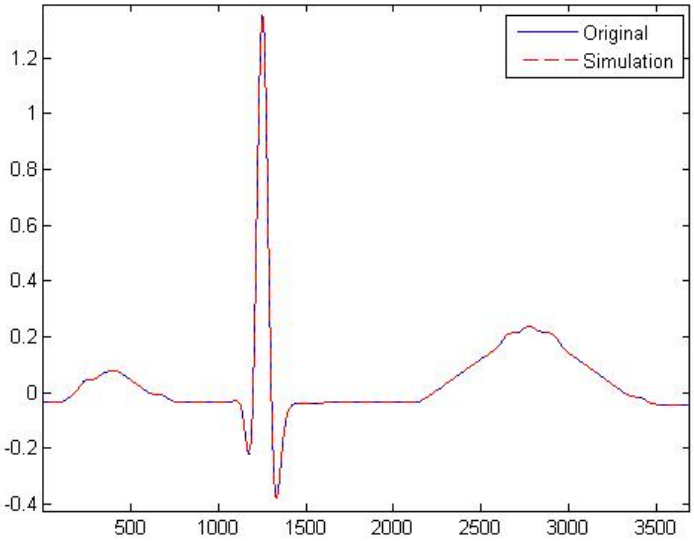
ภาพที่ 4-1 คลื่นไฟฟ้าหัวใจอัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า

ภาพที่ 4-1 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 30 ครั้งต่อนาที โดยลดขนาดข้อมูลที่ 2 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



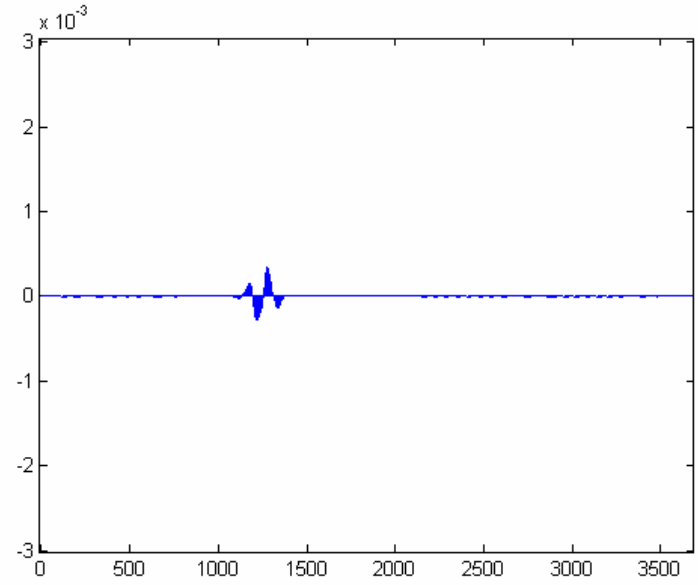
ภาพที่ 4-2 กราฟแสดงความผิดพลาดที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 2เท่า

ภาพที่ 4-2 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 2 เท่า อัตราการเต้น 30 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0021 %



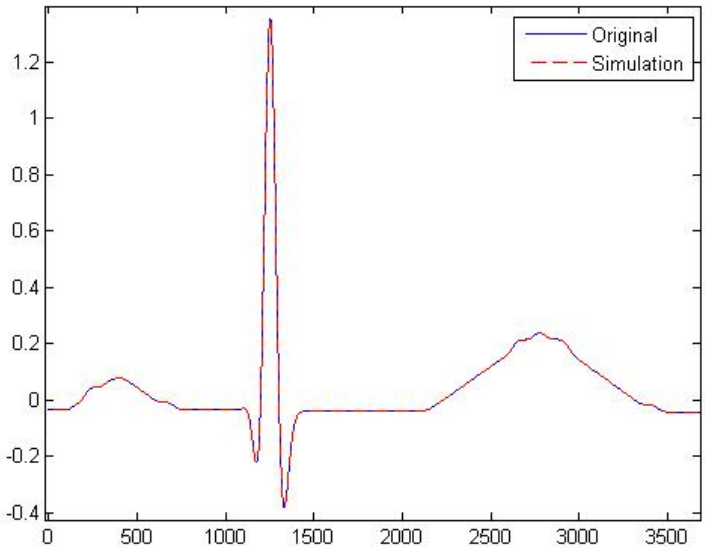
ภาพที่ 4-3 คลื่นไฟฟ้าหัวใจอัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า

ภาพที่ 4-3 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 30 ครั้งต่อนาที โดยลดขนาดข้อมูลที่ 4 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



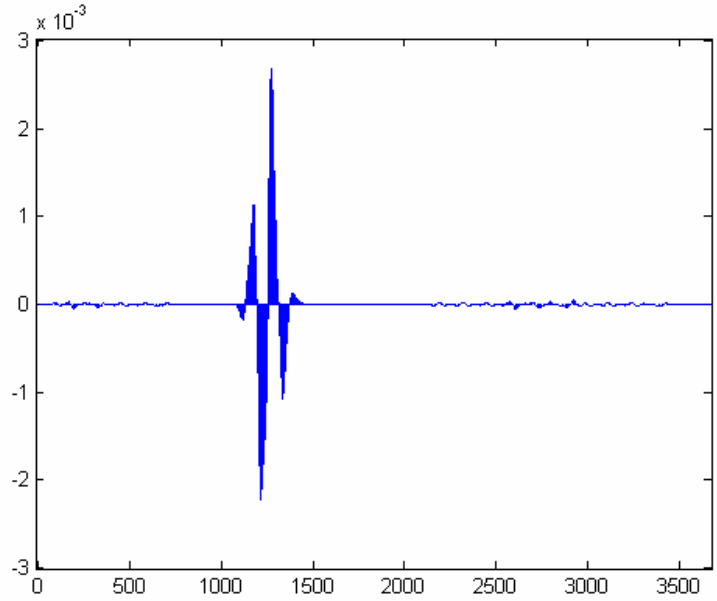
ภาพที่ 4-4 กราฟแสดงความผิดพลาดที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า

ภาพที่ 4-4 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 4 เท่า อัตราการเต้น 30 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0170 %



ภาพที่ 4-5 คลื่นไฟฟ้าหัวใจอัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า

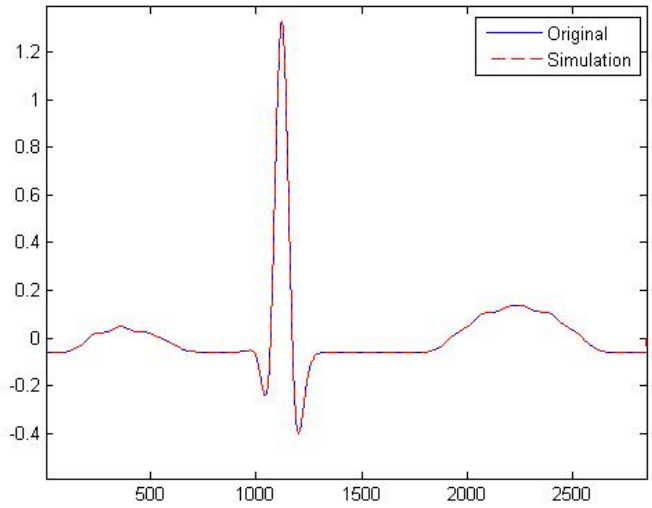
ภาพที่ 4-5 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 30 ครั้งต่อนาที โดยลดขนาดข้อมูลที่ 8 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



ภาพที่ 4-6 กราฟแสดงความผิดพลาดที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า

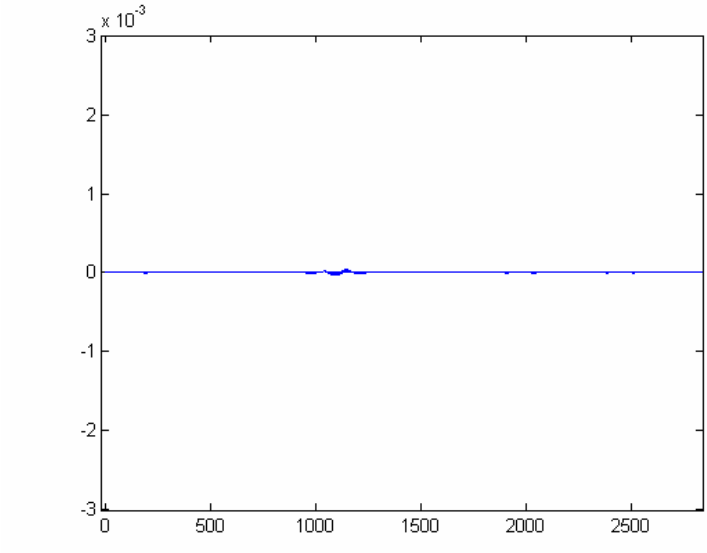
ภาพที่ 4-6 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 8 เท่า อัตราการเต้น 30 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.1352 %

4.1.2 อัตราการเต้นของหัวใจที่ 60 ครั้งต่อนาที



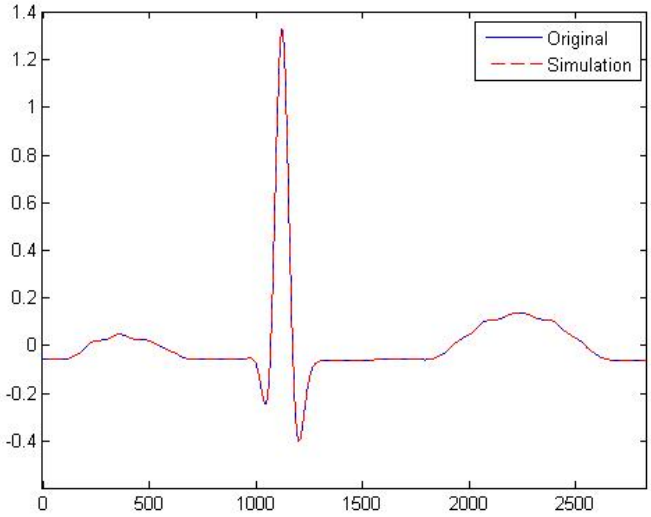
ภาพที่ 4-7 คลื่นไฟฟ้าหัวใจอัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า

ภาพที่ 4-7 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 60 ครั้งต่อนาที โดยลดขนาดข้อมูล 2 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



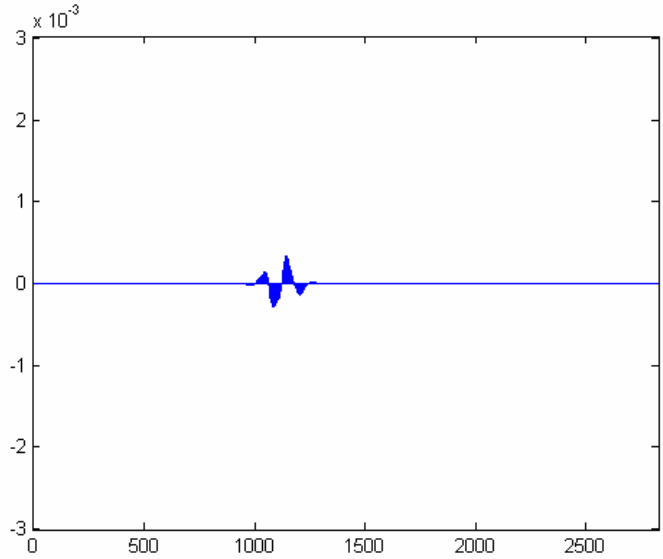
ภาพที่ 4-8 กราฟแสดงความผิดพลาดที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า

ภาพที่ 4-8 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 2 เท่า อัตราการเต้น 60 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0022 %



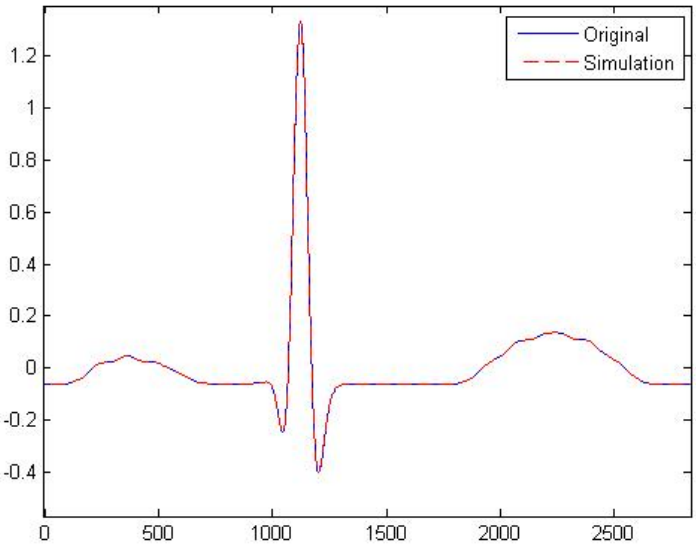
ภาพที่ 4-9 คลื่นไฟฟ้าหัวใจอัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า

ภาพที่ 4-9 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 60 ครั้งต่อนาที โดยลดขนาดข้อมูล 4 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



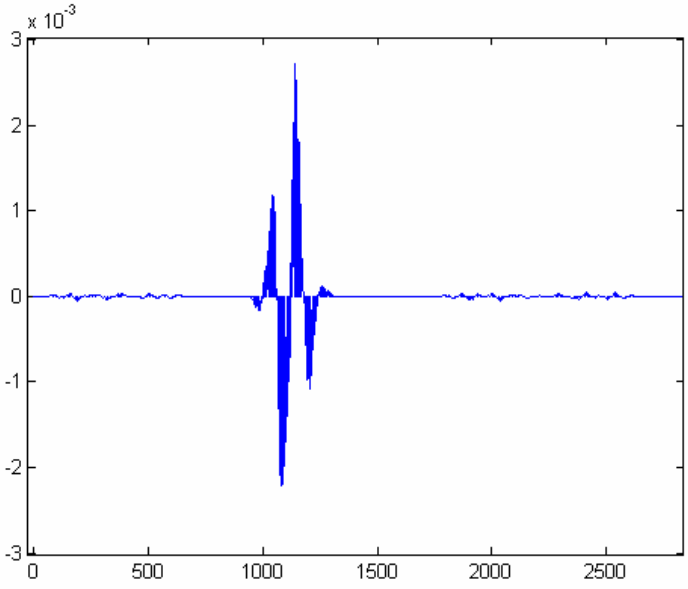
ภาพที่ 4-10 กราฟแสดงความผิดพลาดที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาด 4 เท่า

ภาพที่ 4-10 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 4 เท่า อัตราการเต้น 60 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0185 %



ภาพที่ 4-11 คลื่นไฟฟ้าหัวใจอัตราการเต้น 60 ครั้งต่อนาที ลดขนาด 8 เท่า

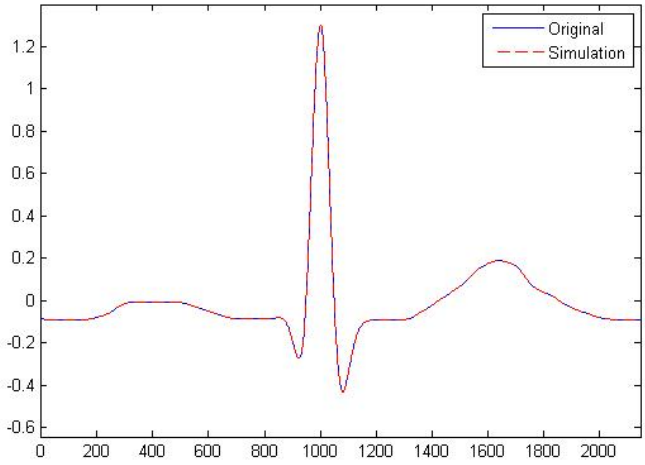
ภาพที่ 4-11 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 60 ครั้งต่อนาที โดยลดขนาดข้อมูล 8 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



ภาพที่ 4-12 กราฟแสดงความผิดพลาดที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาด 8 เท่า

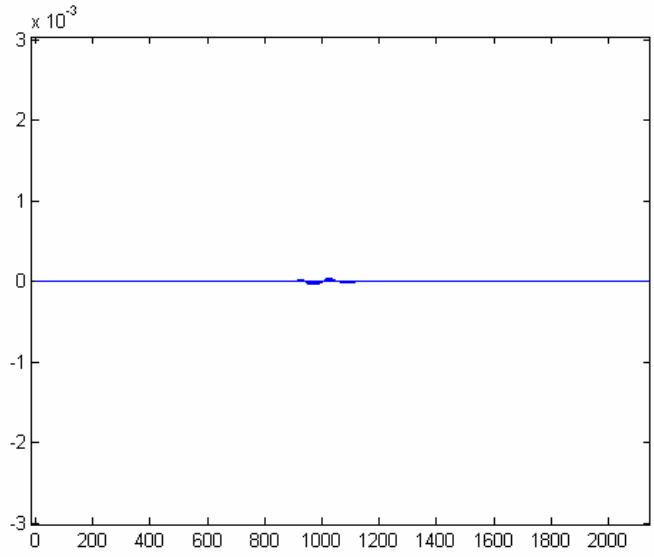
ภาพที่ 4-12 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 8 เท่า อัตราการเต้น 60 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.1467 %

4.1.3 อัตราการเต้นของหัวใจที่ 120 ครั้งต่อนาที



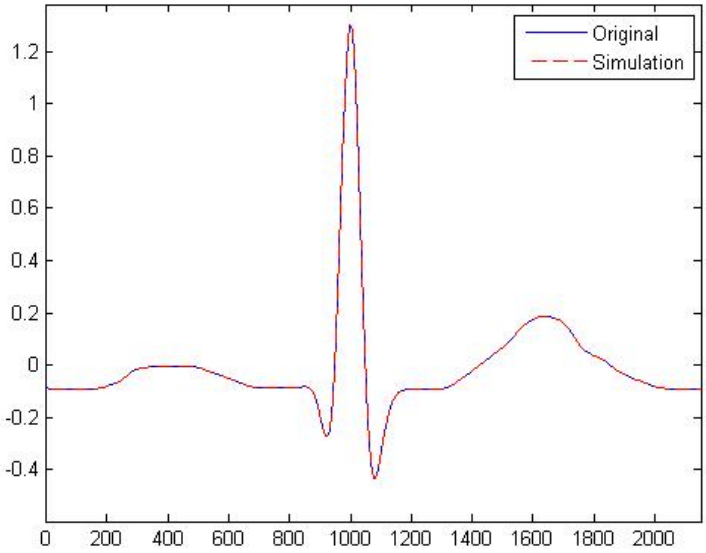
ภาพที่ 4-13 คลื่นไฟฟ้าหัวใจอัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า

ภาพที่ 4-13 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 120 ครั้งต่อนาที โดยลดและเพิ่มขนาดที่ 2 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



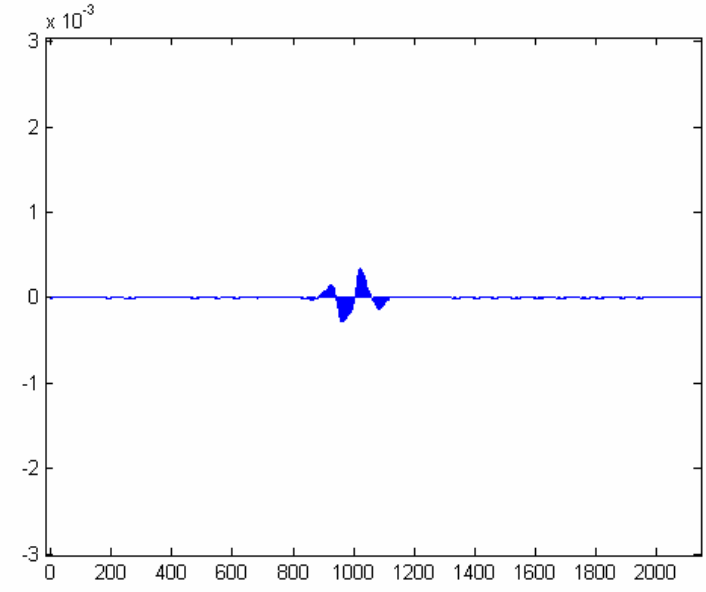
ภาพที่ 4-14 กราฟแสดงความผิดพลาดที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า

ภาพที่ 4-14 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 2 เท่า อัตราการเต้น 120 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0022 %



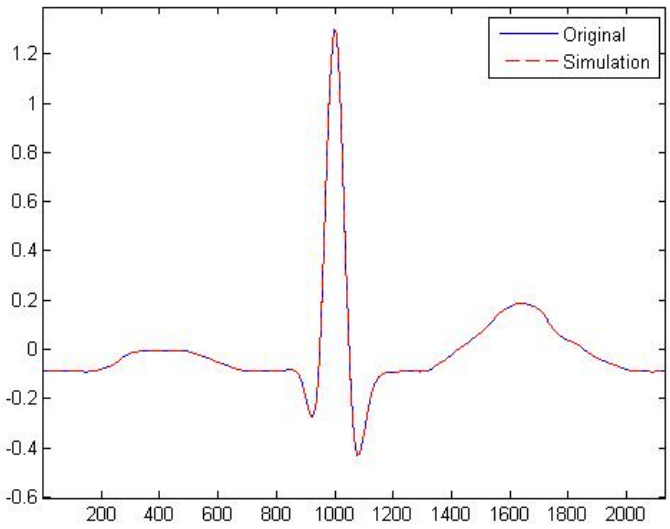
ภาพที่ 4-15 คลื่นไฟฟ้าหัวใจอัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า

ภาพที่ 4-15 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 120 ครั้งต่อนาที โดยลดขนาดข้อมูล 4 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ



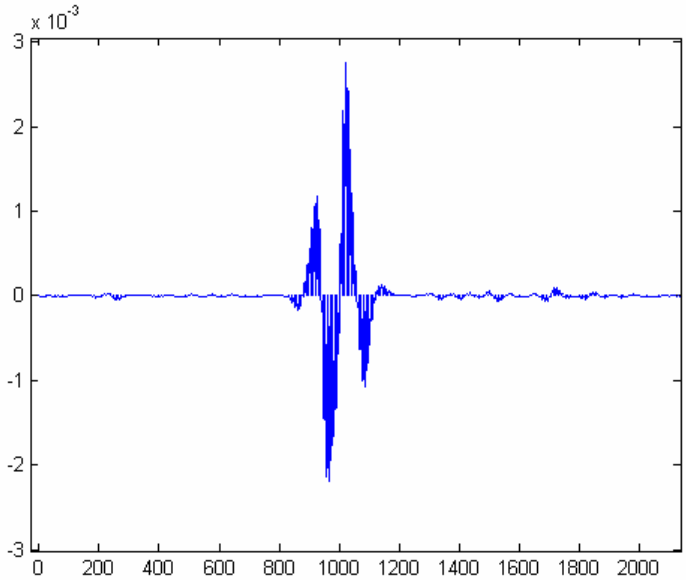
ภาพที่ 4-16 กราฟแสดงความผิดพลาดที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า

ภาพที่ 4-16 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 4 เท่า อัตราการเต้น 120 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.0185 %



ภาพที่ 4-17 คลื่นไฟฟ้าหัวใจอัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า

ภาพที่ 4-17 แสดงรูปสัญญาณคลื่นไฟฟ้าหัวใจที่อัตรา 120 ครั้งต่อนาที โดยลดขนาดข้อมูล 8 เท่า สัญญาณต้นแบบแสดงเป็นเส้นตรง ส่วนสัญญาณที่ผ่านการลดขนาดข้อมูลแสดงเป็นเส้นปะ

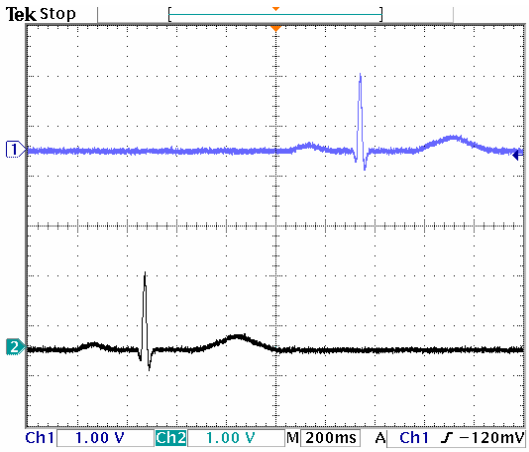


ภาพที่ 4-18 กราฟแสดงความผิดพลาดที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า

ภาพที่ 4-18 แสดงเส้นกราฟความผิดพลาดของคลื่นไฟฟ้าหัวใจก่อนการลดขนาดข้อมูลเทียบกับคลื่นไฟฟ้าหัวใจที่ผ่านการลดขนาดข้อมูล 8 เท่า อัตราการเต้น 120 ครั้งต่อนาที ค่าความผิดพลาดเท่ากับ 0.1466 %

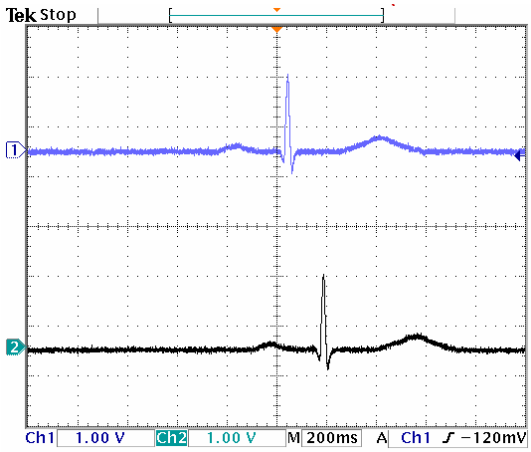
4.2 ผลการทดลองจริงบนบอร์ดประมวลผล TMS320C31 DSK

4.2.1 อัตราการเต้นของหัวใจที่ 30 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 600 Hz



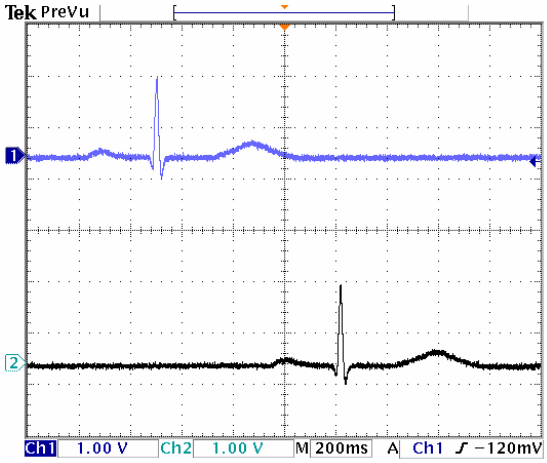
ภาพที่ 4-19 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 600Hz

ภาพที่ 4-19 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-20 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 600Hz

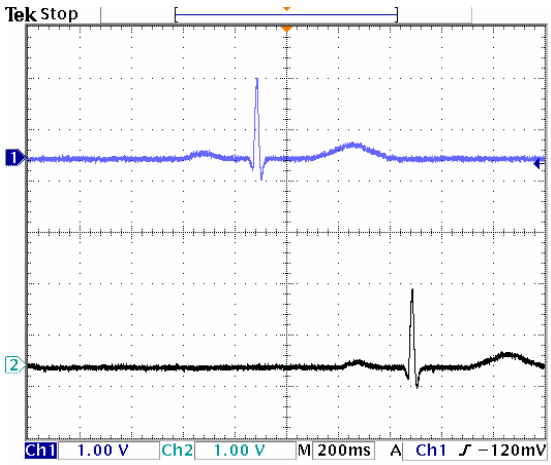
ภาพที่ 4-20 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-21 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 600Hz

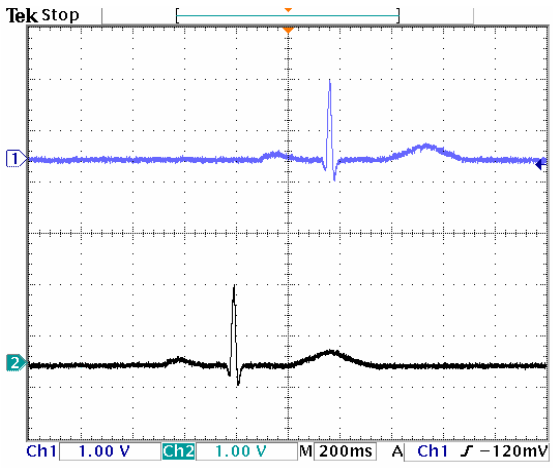
ภาพที่ 4-21 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาด 8 เท่า

4.2.2 อัตราการเต้นของหัวใจที่ 30 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 1200 Hz



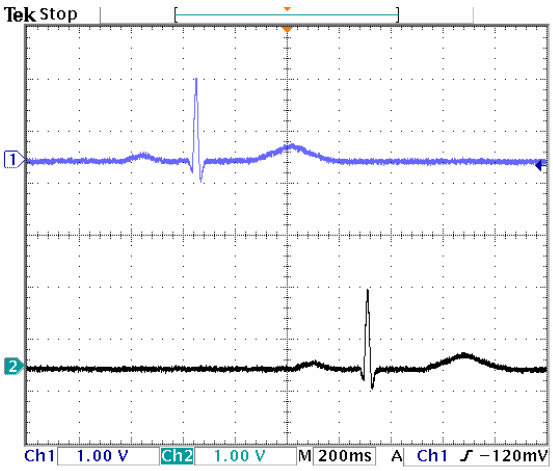
ภาพที่ 4-22 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 1200Hz

ภาพที่ 4-22 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-23 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 1200Hz

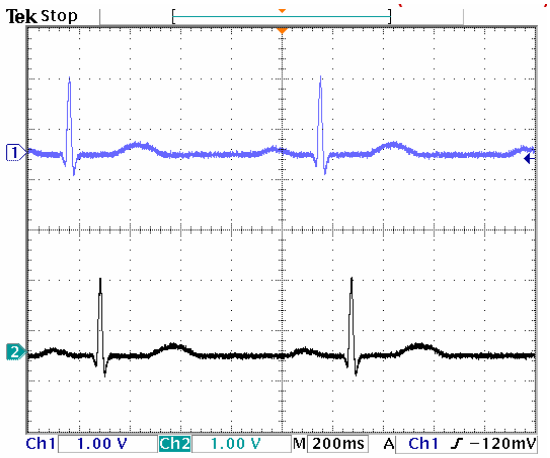
ภาพที่ 4-23 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-24 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 1200Hz

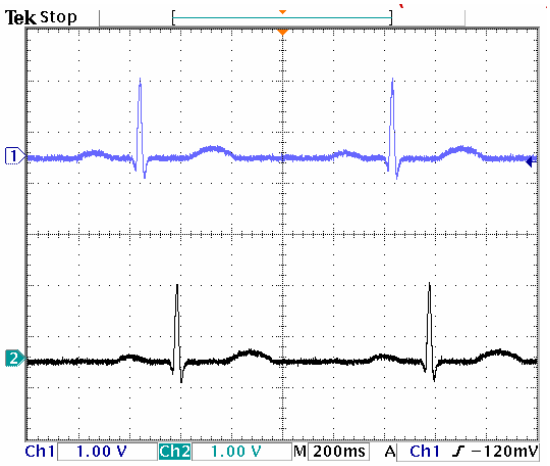
ภาพที่ 4-24 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 30 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 8 เท่า

4.2.3 อัตราการเต้นของหัวใจที่ 60 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 600 Hz



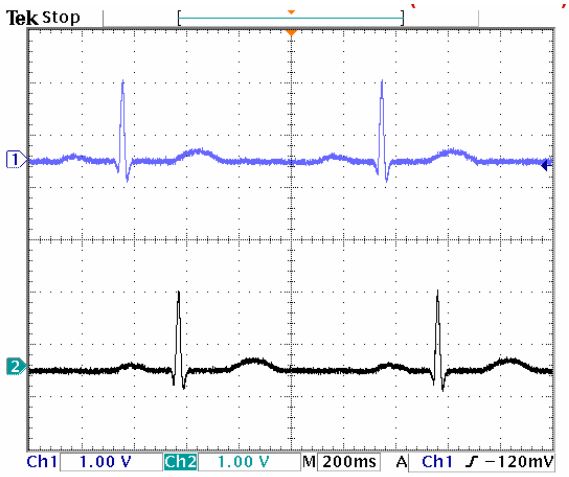
ภาพที่ 4-25 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 600Hz

ภาพที่ 4-25 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-26 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 600Hz

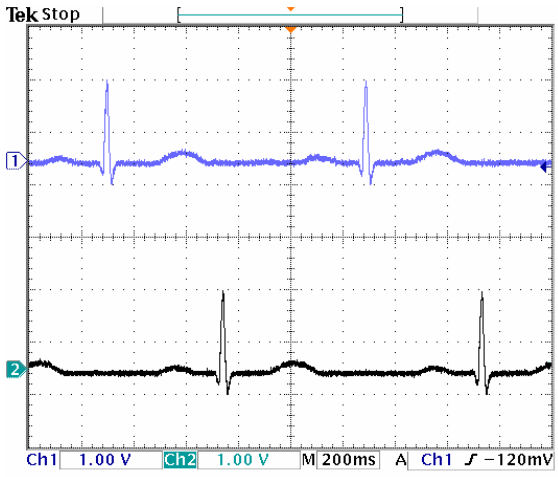
ภาพที่ 4-26 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-27 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 600Hz

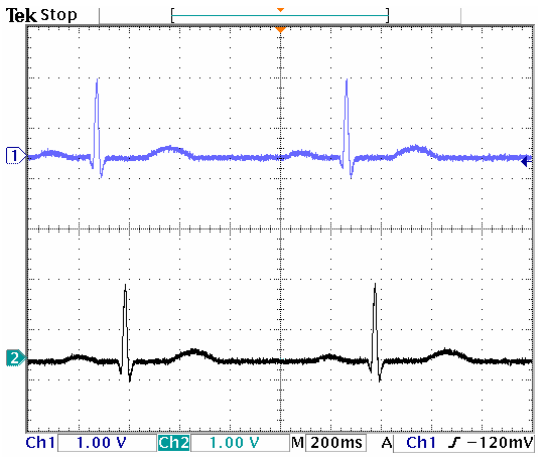
ภาพที่ 4-27 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 8 เท่า

4.2.4 อัตราการเต้นของหัวใจที่ 60 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 1200 Hz



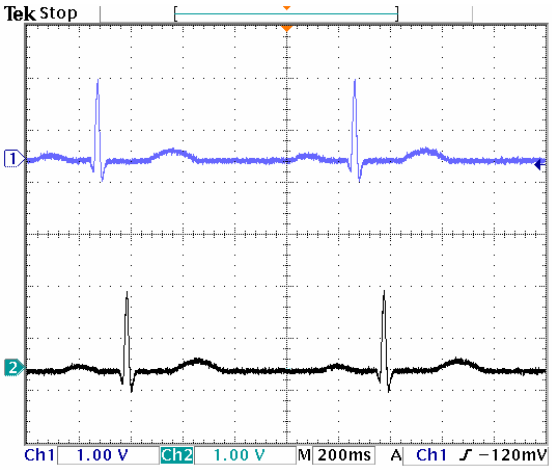
ภาพที่ 4-28 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 1200Hz

ภาพที่ 4-28 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-29 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 1200Hz

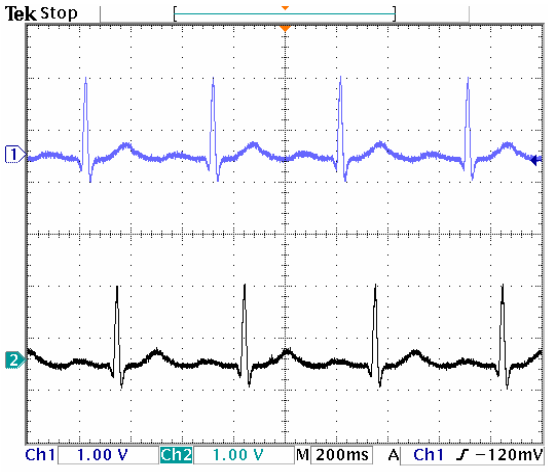
ภาพที่ 4-29 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-30 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 1200Hz

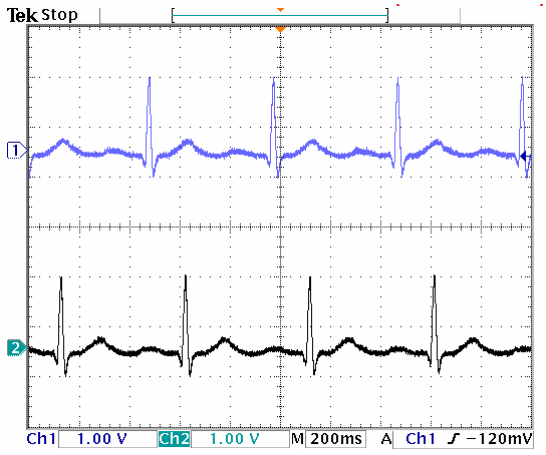
ภาพที่ 4-30 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 60 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาด 8 เท่า

4.2.5 อัตราการเต้นของหัวใจที่ 120 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 600 Hz



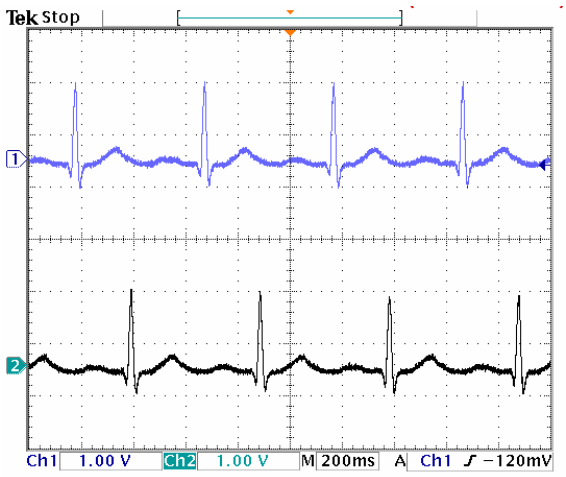
ภาพที่ 4-31 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 600Hz

ภาพที่ 4-31 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-32 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 600Hz

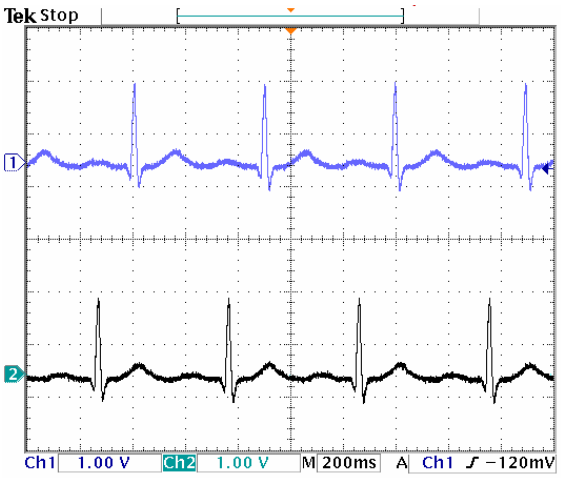
ภาพที่ 4-32 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-33 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 600Hz

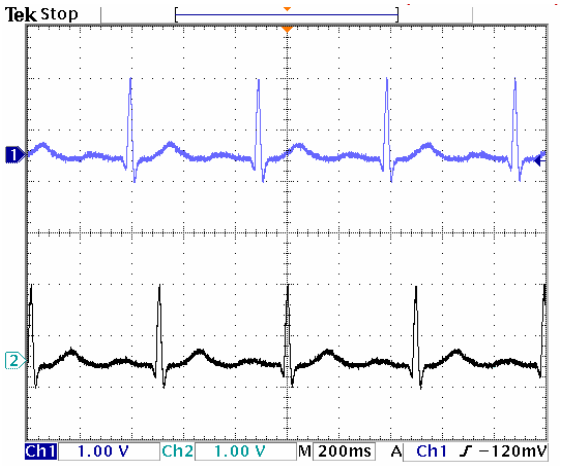
ภาพที่ 4-33 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 600 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 8 เท่า

4.2.6 อัตราการเต้นของหัวใจที่ 120 ครั้งต่อนาที โดยอัตราการสุ่มสัญญาณที่ 1200 Hz



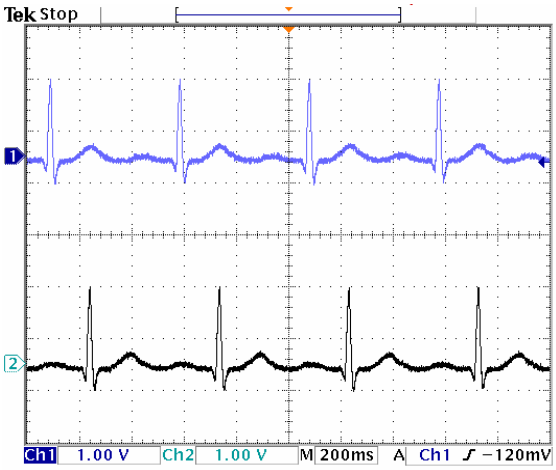
ภาพที่ 4-34 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 2 เท่า fs = 1200Hz

ภาพที่ 4-34 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 2 เท่า



ภาพที่ 4-35 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 4 เท่า fs = 1200Hz

ภาพที่ 4-35 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 4 เท่า



ภาพที่ 4-36 สัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที ลดขนาดข้อมูล 8 เท่า fs = 1200Hz

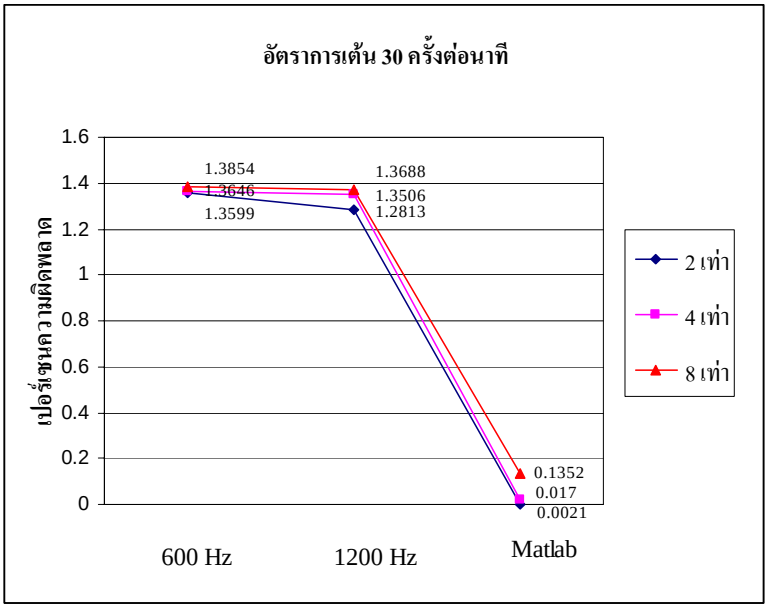
ภาพที่ 4-36 แสดงหน้าจอภาพของออสซิลโลสโคปที่วัดคลื่นไฟฟ้าหัวใจที่อัตราการเต้น 120 ครั้งต่อนาที อัตราการสุ่ม 1200 Hz โดยที่ สัญญาณช่อง 1 เป็นสัญญาณก่อนการลดขนาดข้อมูล สัญญาณช่อง 2 เป็นสัญญาณที่ผ่านการลดและเพิ่มขนาดข้อมูล 8 เท่า

4.3 ผลการหาค่าเปอร์เซนความผิดพลาด

ตารางที่ 4-1 ค่าเปอร์เซนความผิดพลาดของการทดลอง

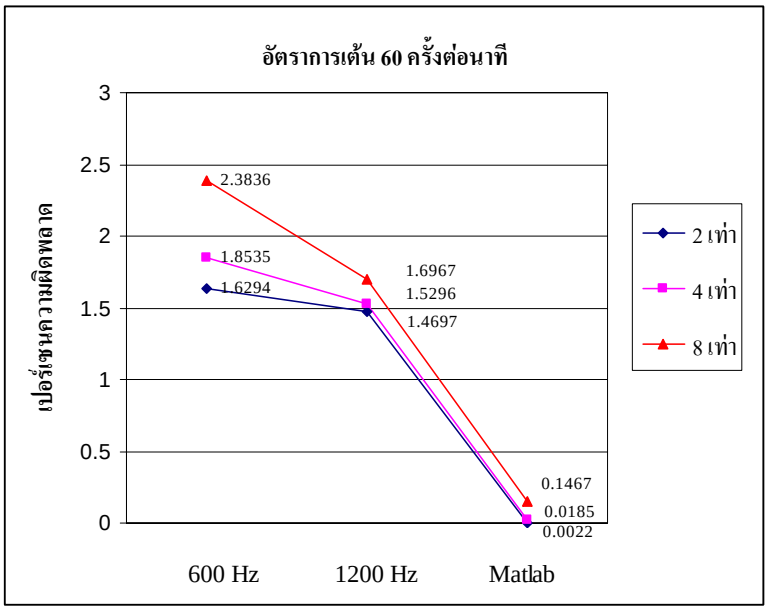
อัตราการเต้นของหัวใจ **(ครั้ง/นาที)**	อัตราการลดขนาด **(เท่า)**	ค่าเปอร์เซนความผิดผลาด (PRD) **(%)**		
		การสร้างจริง อัตราการสุ่ม (600 Hz)	การสร้างจริง อัตราการสุ่ม (1200 Hz)	การจำลองการทำงานบน (MATLAB)
30	2	1.3599	1.2813	0.0021
	4	1.3646	1.3506	0.0170
	8	1.3854	1.3688	0.1352
60	2	1.6294	1.4697	0.0022
	4	1.8535	1.5296	0.0185
	8	2.3836	1.6967	0.1467
120	2	1.7086	1.3353	0.0022
	4	3.3572	3.0550	0.0185
	8	4.2484	3.2714	0.1466

ตารางที่ 4-1 แสดงค่าเปอร์เซนความผิดพลาดที่ได้จากการทดลองโดยจะมีผลที่ได้จากการจำลองการทำงานบนโปรแกรม Matlab และผลจากการสร้างจริง โดยกำหนดค่าอัตราการสุ่มที่ใช้ในการทดลองที่ 600Hz และ 1200Hz



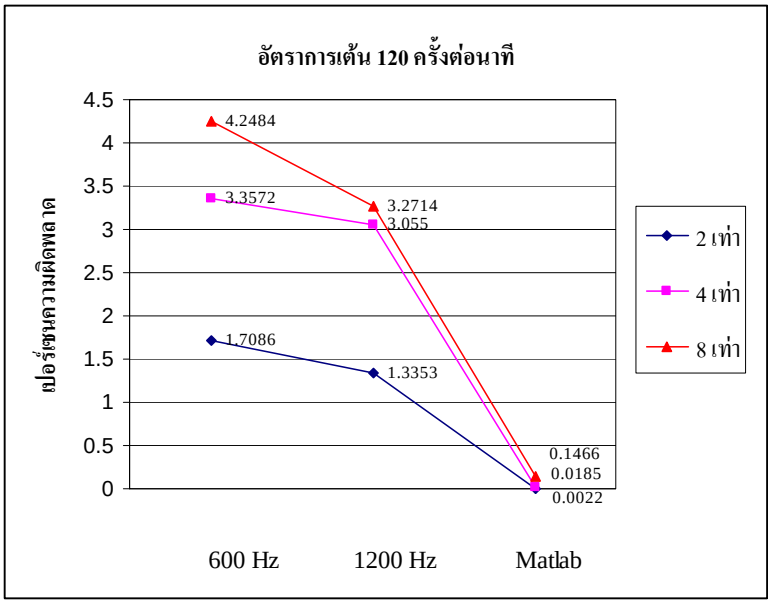
ภาพที่ 4-37 กราฟแสดงค่าความผิดพลาด ที่อัตราการเต้นของหัวใจ 30 ครั้งต่อนาที

ภาพที่ 4-37 แสดงเส้นกราฟค่าความผิดพลาดของการลดขนาดข้อมูลที่อัตราการเต้นของหัวใจที่ 30 ครั้งต่อนาที



ภาพที่ 4-38 กราฟแสดงค่าความผิดพลาด ที่อัตราการเต้นของหัวใจ 60 ครั้งต่อนาที

ภาพที่ 4-38 แสดงเส้นกราฟค่าความผิดพลาดของการลดขนาดข้อมูลที่อัตราการเต้นของหัวใจที่ 60 ครั้งต่อนาที



ภาพที่ 4-39 กราฟแสดงค่าความผิดพลาด ที่อัตราการเต้นของหัวใจ 120 ครั้งต่อนาที

ภาพที่ 4-39 แสดงเส้นกราฟค่าความผิดพลาดของการลดขนาดข้อมูลที่อัตราการเต้นของหัวใจที่ 120 ครั้งต่อนาที

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

จากผลการทดลองในบทที่ 4 จะเห็นได้ว่าค่าเปอร์เซนความผิดพลาดที่เกิดจากการจำลองการทำงานการลดขนาดสัญญาณและขยายสัญญาณคลื่นไฟฟ้าหัวใจที่อัตราการลดขนาดต่างๆ มีค่าความผิดพลาดที่น้อยมาก เมื่อเทียบกับการนำระบบที่ออกแบบได้ไปทำการทดลองสร้างจริงบนบอร์ด ประมวลผลสัญญาณดิจิตอล TMS320C31 DSK ค่าเปอร์เซนความผิดพลาดของการสร้างจริงจะมีความผิดพลาดที่มากกว่า แต่ยังเป็นค่าที่สามารถยอมรับได้ สาเหตุที่ค่าความผิดพลาดของการทดลองสร้างจริงมีค่ามากกว่าการจำลองการทำงานบนโปรแกรม Matlab เนื่องมาจากการที่จะต้องต่อสายสัญญาณหลายตำแหน่งในการทดลองทำให้เกิดสัญญาณรบกวนเกิดขึ้นกับสัญญาณคลื่นไฟฟ้าหัวใจทั้งในส่วนสัญญาณอินพุตและสัญญาณเอาท์พุต ทำให้ค่าความผิดพลาดเกิดขึ้นมาก และจากผลการทดลองจะเห็นได้ว่าอัตราการสุ่มของสัญญาณที่ใช้ในการทดลองมีผลกับค่าความผิดพลาดของสัญญาณที่อัตราการสุ่มที่ 600 Hz ค่าความผิดพลาดจะมากกว่าการใช้อัตราการสุ่มที่ 1200 Hz เพราะว่าจำนวนข้อมูลของอัตราการสุ่มที่ 600 Hz จะมีข้อมูลที่น้อยกว่า การใช้อัตราการสุ่มที่ 1200 Hz แต่การใช้อัตราการสุ่มที่ 600 Hz จะใช้หน่วยความจำน้อยกว่าการใช้อัตราการสุ่มที่ 1200 Hz ทำให้ใช้หน่วยความจำน้อยกว่าในการบันทึก และอัตราการลดขนาดก็มีผลกับค่าความผิดพลาดโดยที่อัตราการลดขนาดที่ 2 เท่า จะให้ค่าความผิดพลาดที่น้อยกว่าอัตราการลดขนาดที่ 4 เท่า และ 8 เท่า สาเหตุที่ค่าความผิดพลาดของการลดขนาด 2 เท่า น้อยกว่า 4 เท่า และ 8 เท่า นั้นเพราะว่าข้อมูลที่ใช้ในการคืนค่าสัญญาณที่ผ่านการลดขนาดสัญญาณมีมากกว่าการลดขนาด ที่ 4 เท่า และ 8 เท่า แต่ขนาดของสัญญาณที่ผ่านการลดขนาดที่น้อยที่สุดคือการใช้อัตราการลดขนาดที่ 8 เท่า ดังนั้นที่หน่วยความจำเท่ากัน การใช้อัตราการลดขนาดที่ 8 เท่า จะสามารถเก็บข้อมูลได้มากกว่า จากกราฟแสดงความผิดพลาด จะเห็นว่าช่วงที่มีความผิดพลาดจะเกิดบริเวณช่วงของยอดคลื่นไฟฟ้าหัวใจ เพราะว่าบริเวณช่วงยอดคลื่นนั้นมีการเปลี่ยนแปลงของข้อมูลที่ต่างกันมากทำให้เป็นบริเวณที่เกิดความผิดพลาดขึ้นมาก

จากผลการทดลองสามารถสรุปได้ว่า ค่าความผิดพลาดทั้งการจำลองการทำงานและการสร้างจริงบนบอร์ดประมวลผลสัญญาณดิจิตอล TMS320C31 DSK ของการสร้างระบบบีบอัดและขยายคลื่นไฟฟ้าหัวใจโดยการใช้การประมวลผลเชิงเลขหลายอัตรา มีค่าต่ำ ไม่เกิน 5% ซึ่งเป็นค่าที่แพทย์

ยอมรับได้ที่ไม่ทำให้เกิดความผิดเพี้ยนในการอ่านค่าคลื่นไฟฟ้าหัวใจ ทำให้สามารถนำระบบที่ออกแบบไปใช้ในเครื่องบันทึกคลื่นไฟฟ้าหัวใจเพื่อช่วยลดพื้นที่หน่วยความจำได้ และจากการทดลองจะทำในลักษณะ Real time ทำให้สามารถนำไปประยุกต์ใช้ในการ รับ/ส่ง สัญญาณคลื่นไฟฟ้าหัวใจระยะไกลได้

5.2 ข้อเสนอแนะ

จากการทดลอง จะเห็นได้ว่าผลของการลดขนาดคลื่นไฟฟ้าหัวใจเป็นที่น่าพอใจและยอมรับได้ในความผิดพลาด แต่ในการนำระบบไปใช้งานจริงควรจะทำการออกแบบของบอร์ดประมวลผลสัญญาณดิจิตอลและบอร์ดหน่วยความจำภายนอกให้มีขนาดเล็กลง หรือออกแบบให้สามารถใช้หน่วยความจำภายนอกชนิดที่มีขายทั่วไป เพื่อความสะดวกในการนำมาใช้งาน ในการนำไปใช้งานจริงควรจะเพิ่มอัตราการสุ่มที่สูงขึ้นเพื่อลดค่าความผิดพลาดให้ลดลงอีก เพื่อเพิ่มความแม่นยำของผู้ที่อ่านค่าสัญญาณคลื่นไฟฟ้าหัวใจ เพราะว่าในการใช้งานจริงจะมีปัญหาของเรื่องสัญญาณรบกวนในสายสัญญาณเพิ่มขึ้นมา และควรมีการพัฒนาต่อให้ระบบที่สร้างสามารถตรวจจับเฉพาะบริเวณค่ายอดของคลื่นไฟฟ้าหัวใจ แล้วให้ความสำคัญคือทำการลดขนาดในปริมาณที่น้อยในการบีบอัดเฉพาะบริเวณที่เป็นค่าของยอดคลื่น ส่วนบริเวณอื่นที่มีค่าใกล้เคียงกันให้ลดขนาดในปริมาณหลายเท่าเพื่อที่จะสามารถลดปริมาณข้อมูลลงได้อีก

เอกสารอ้างอิง

1. Akay, Metin. **Biomedical Signal Processing.** Academic Press, 1994.
2. Willis J. Tompking, **Biomedical Digital Signal Processing**, Prentice-Hall, 1993.
3. Alfred Mertins. **Signal Analysis Wavelets, Filter Bank, Time-Frequency Transforms and Applications**. John Wiley&Sons, Inc. 1996.
4. R.E. Crochiere and L. R. Rabiner. **Multirate Digital Signal Processing**. Prentice-Hall, 1993.
5. M. Vetterli. "**A Theory of Multirate Filter Banks**." IEEE Trans. Signal Processing, vol. ASSP-35, No 3,pp. 356-372, Mar. 1987.
6. Xiang-Gen Xia and Bruce W. Suter, " **Multirate Filter Banks with Block ampling**, " IEEE Trans. Signal Processing, vol. 44 No. 3, pp. 484-496, Mar. 1996.
7. VX Afonso, WJ Tompkins, TQ Nguyen, s Luo, " **Multirate Processing of the ECG using Filter Banks**, " IEEE Computers in Cardiology, pp. 245-248,1996.
8. พรชัย ภววงษ์ศักดิ์. **การประมวลผลสัญญาณดิจิตอลเบื้องต้น**. [ออนไลน์] 2542. [สืบค้นวันที่ 14 กรกฎาคม 2544] จาก http://www.ee.mut.ac.th/home/pornchai
9. Sanjit. K.Mitra, **Digital Signal Processing** : Computer-Based Approach, McGRAW-HILL. 1996.
10. ปราโมทย์ เดชะอำไพ. **ระเบียบวิธีเชิงตัวเลขในงานวิศวกรรม.** กรุงเทพฯ : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2544.
11. วรมนต์ ตรีพรหม. **สัญญาณชีพ**. โครงการตำรา ภาควิชาการพยาบาลพื้นฐาน. คณะพยาบาลศาสตร์ มหาวิทยาลัยเชียงใหม่ โอเดี้ยนสโตร์, 2537.
12. **TMS320C3x User's Guide**. : Texas Instruments Inc, 1997.
13. **TMS320C3x DSP Starter kit**. : Texas Instruments Inc, 1996.
14. Rulph Chassing. **Digital Signal Processing Laboratory Experiments Using C and the TMS320C31 DSK**. : John Wiley & Sons, Inc., 1999.

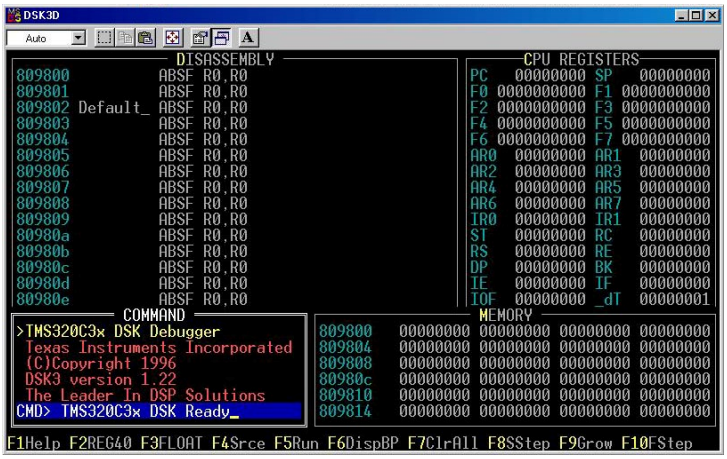
ภาคผนวก ก

การใช้งานบอร์ด TMS320C31 DSP STARTER KIT

ก.1 การใช้งานบอร์ด TMS320C31 DSP STARTER KIT

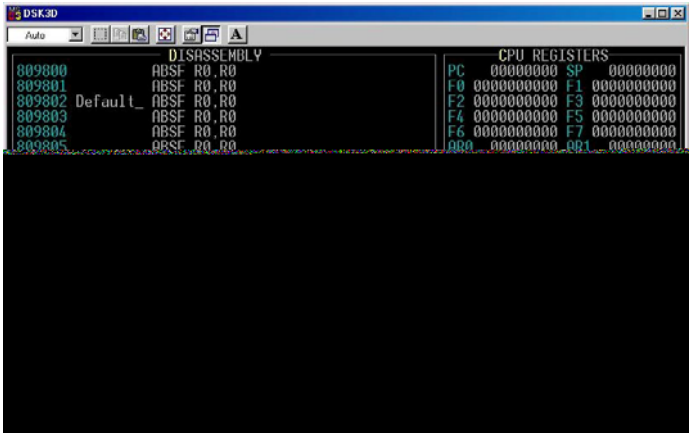
การใช้งานบอร์ด TMS320C31 DSP STARTER KIT มีขั้นตอนการใช้งานดังนี้

1. ต่อบอร์ด TMS320C31 DSK เข้ากับ Port Printer ของเครื่องไมโครคอมพิวเตอร์
2. จ่ายไฟ 10 – 12 Vdc เข้ากับบอร์ด TMS320C31 DSK
3. สร้าง Sub-Directory ด้วยคำสั่ง md c:\c3xtools
4. Copy โปรแกรมของ TMS320C31 DSK ด้วยคำสั่ง copy d:*.* c:\c3xtools
5. Run คำสั่ง c:\c3xtools\dsk3d ถ้าไม่มีข้อผิดพลาดบนจอภาพของเครื่องไม่โครคอมพิวเตอร์จะแสดงดังในภาพที่ ก-1



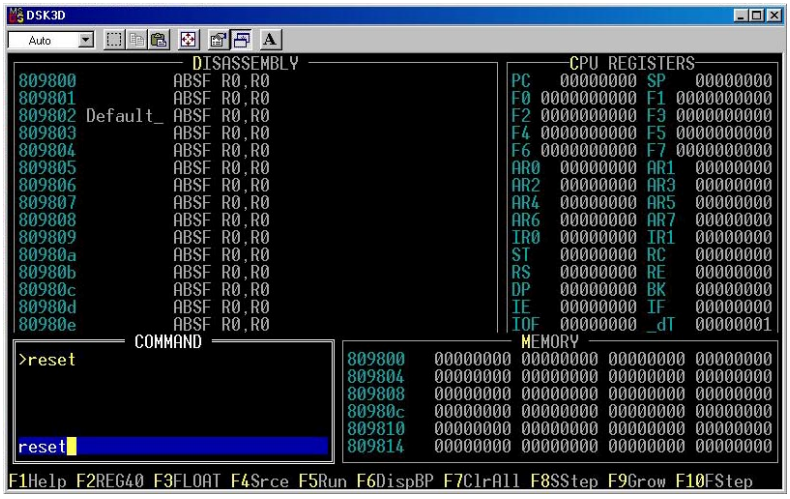
ภาพที่ ก-1 การรันโปรแกรม DSK3D

6. เมื่อต้องการจะใช้งานหน้าต่าง Debugger หลังจากใช้คำสั่ง dsk3d จะปรากฏจอภาพดังในภาพที่ ก-2



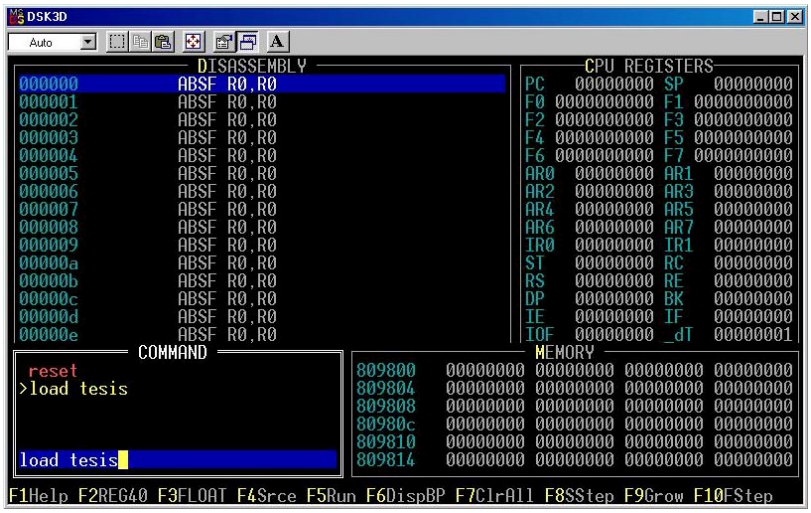
ภาพที่ ก-2 หน้าจอโปรแกรม dsk3d พร้อมใช้งาน

7. พิมพ์คำสั่ง reset ที่หน้าต่าง Command เพื่อที่จะพร้อมในการโหลดโปรแกรม



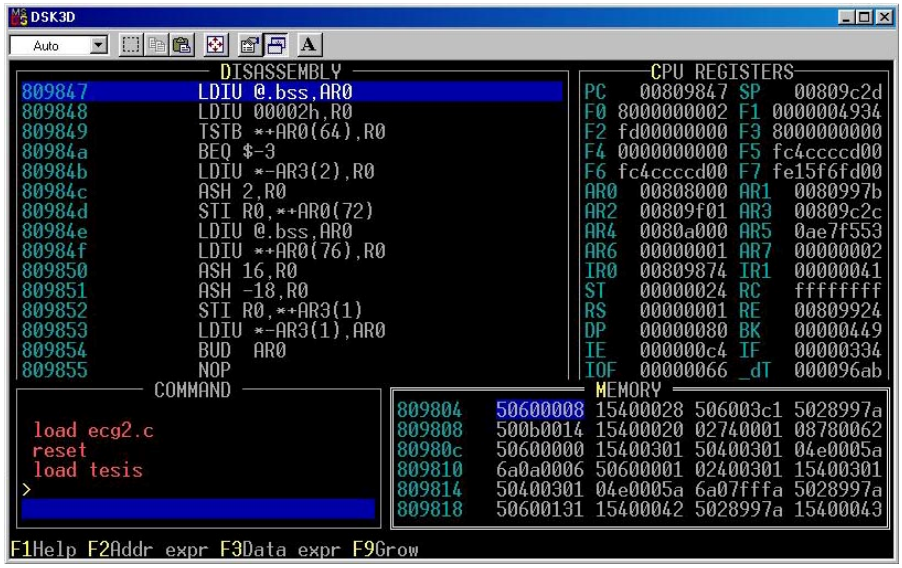
ภาพที่ ก-3 พิมพ์คำสั่ง reset ที่หน้าต่าง Command

8. เมื่อต้องการที่จะทดสอบโปรแกรม ให้ใช้คำสั่ง load ตามด้วยชื่อ ไฟล์ที่ต้องการทดสอบ

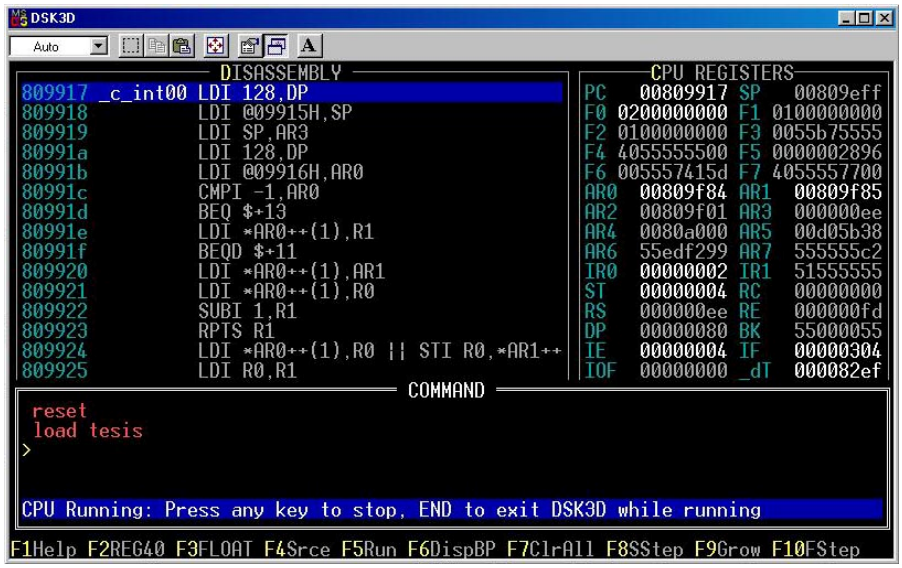


ภาพที่ ก-4 พิมพ์คำสั่ง load ตามด้วยชื่อไฟล์ที่ต้องการทดสอบ

9. หลังการใช้คำสั่ง load ตามด้วยชื่อไฟล์ที่ต้องการทดสอบแล้วกดปุ่ม Enter โปรแกรมจะถูกนำไปเก็บไว้ใน Memory ภายในบอร์ด TMS320C31 แสดงในภาพที่ ก-5
10. เมื่อต้องการที่จะรัน(RUN)โปรแกรมที่ต้องการทดสอบหลังจากโหลดโปรแกรมเข้ามาไว้ในบอร์ดแล้ว ให้กดปุ่ม F5 เพื่อรัน(RUN)โปรแกรม แสดงในภาพที่ ก-6



ภาพที่ ก-5 หน้าจอหลังจากโหลดโปรแกรมที่ต้องการทดสอบ



ภาพที่ ก-6 หน้าจอหลังจากรันโปรแกรมโดยกดปุ่ม F5

11. เมื่อต้องการหยุดการรันโปรแกรมทำได้โดยกดปุ่ม Esc
12. ถ้าต้องการออกจากโปรแกรม dsk3d ให้พิมพ์คำสั่ง Exit หรือ Quit

<u>หมายเหตุ</u> สามารถอ่านการใช้งานอย่างละเอียดได้จาก TMS320C3x DSP Starter Kit User's Guide

ก.2 การเรียกใช้หน้าต่าง Debugger

คำสั่งที่เรียกใช้หน้าต่าง Debugger

Dsk3d [option]

dsk3d เป็นคำสั่งที่เรียกใช้ หน้าต่าง Debugger

option เป็นการเพิ่มรายละเอียดของ Debugger

ตารางที่ ก-1 แสดงรายการ option ของ Debugger โดยในตารางจะอธิบาย option บาง option ที่ถูกเรียกใช้บ่อย

ตารางที่ ก-1 รายการ option ของ Debugger บางคำสั่ง ที่ถูกเรียกใช้บ่อย

Option	**อธิบายสรุป**
? หรือ Help	การแสดงรายการของ Option
AUTO	ค้นหาอัตโนมัติถ้า พอร์ตขนานสนับสนุนโหมด 8 บิต หรือ 4 บิต
BW = 4, Nibble	การสื่อสารใช้พอร์ตขนานแบบมาตรฐาน 4 บิต mode unidirectional
BW = 5, Byte	การสื่อสารที่ใช้พอร์ตขนานแบบมาตรฐาน 8 บิต mode bidirectional
LPTx, LPT = x	เลือกพอร์ตพรินเตอร์ขนาน (LPT1 เป็น default)
PORT = 0x378	เลือก address ของ พอร์ตต่างๆ
RESET	รีเซต DSK (cold boot)
TEST	ค้นหาอัตโนมัติทั้ง LPT1, LPT2, และ LPT3 เพื่อติดต่อกับ DSK
T = xx	เพิ่มบัส xx I/O ไซเคิลพิเศษในการส่งแต่ละครั้งสำหรับสายที่ยาวหรือมี noise
WIN = 1	การจัดการหน้าต่าง Time Slice ให้ทำงาน
WIN = 0	การจัดการหน้าต่าง Time Slice ให้หยุดการทำงาน และ เคลียร์อินเตอร์รัฟท์

ก.2.1 การแสดงรายการของ Option (? หรือ Help Option)

สามารถที่จะดู Option ที่แสดงในตารางที่ ก-1 โดยใช้ ? หรือ Help

ก.2.2 การเลือกพอร์ตพรินเตอร์ขนาน (LPT = 3 หรือ LPT # Option)

LPT Option จะเลือกพอร์ตพริ้นเตอร์ขนานจากการติดต่อของ DSK กับ Host

ตารางที่ ก-2 พอร์ตพรินเตอร์ขนานและหน้าที่

พอร์ตพรินเตอร์ขนาน	หน้าที่
LPT1 หรือ LPT = 1	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x378
LPT 2 หรือ LPT = 2	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x278
LPT 3 หรือ LPT = 3	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x3BC

หมายเหตุ สำหรับเครื่อง ESSA และ IBM PS/2s ใช้ LPTx ที่แตกต่างกัน

AT Convertion	EISA และ PS/2	I/O Address
LPT1	LPT2	0x378
LPT2	LPT3	0x278
LPT3	LPT1	0x3BC

ก.2.3 การเลือกพอร์ตพรินเตอร์ที่จะใช้ (พอร์ต option)

พอร์ต option เลือกพอร์ตพรินเตอร์ขนานใน Address ที่มีให้ดังตัวอย่าง

port =0x378

หมายความว่า เลือกพอร์ตพรินเตอร์ขนานของ host ใน Address 0x378

ก.2.4 การค้นหาพอร์ตพรินเตอร์อัตโนมัติ (TEST option)

ใช้ test option ไปค้นหาในระบบเพื่อหาพอร์ตขนานที่จะติดต่อกับ DSK

หมายเหตุ ถ้ามีพอร์ตพรินเตอร์หรือการติดต่ออื่นๆ ไปยัง PC จะต้องหยุดการทำงานก่อนการใช้ test option

ก.3 การใช้หน้าต่าง Debugger

ก.3.1 หน้าต่าง DISASSEMBLY

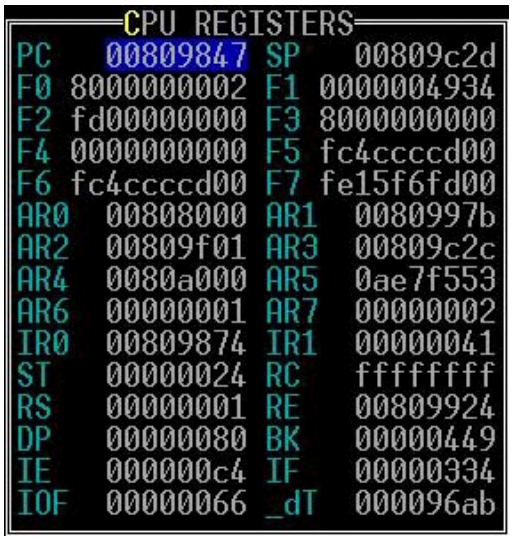
หน้าต่าง DISASSEMBLY แสดงการจองพื้นที่หน่วยความจำของ assembly ดังแสดงในภาพที่ ก-7 ในหน้าต่างนี้จะแสดงบรรทัดของ code ซึ่งแต่ละบรรทัดจะแสดงให้เห็นถึงโครงสร้างของตำแหน่ง โครงสร้าง opcode, ชื่อ และโครงสร้างของ mnemonic บรรทัดที่แสดงแถบสว่างจะเป็นโครงสร้างต่อไปที่ถูกดำเนินการ

ภาพที่ ก-7 หน้าต่าง DISASSEMBLY

ในการเลือกหน้าต่าง DISASSEMBLY โดยกด ALT + D ขณะที่อยู่ในหน้าต่างนี้สามารถใช้เคอร์เซอร์เลือกบรรทัดและเลือกใช้ function key เพื่อตั้งเคลียร์ Break Point จากตารางที่ มีรายละเอียดของ Function key

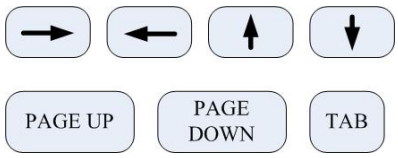
ก.3.2 หน้าต่าง CPU รีจิสเตอร์

หน้าต่าง CPU รีจิสเตอร์แสดงให้เห็นรีจิสเตอร์ที่มีอยู่ใน CPU ดังแสดงในภาพที่ ก-8 ตามปกติค่าที่บรรจุในรีจิสเตอร์จะเป็นเลขฐาน 16 สามารถกด F3 ค่าในรีจิสเตอร์จะเป็นแบบ Floating-Point-Decimal ถ้ากด F2 ค่าภายในรีจิสเตอร์จะเป็นแบบ 40 บิต (Hexadeadimal)



ภาพที่ ก-8 หน้าต่าง CPU รีจิสเตอร์

การเปลี่ยนแปลงค่ารีจิสเตอร์ทำได้โดยกด ALT+C สามารถเขียนค่าที่แถบข้อมูลที่เป็นแถบสีและกด ENTER เพื่อยอมรับการเปลี่ยนแปลง เมื่อแน่ใจและใช้ key ดังนี้เพื่อเลือก data ที่ต้องการแก้ไข



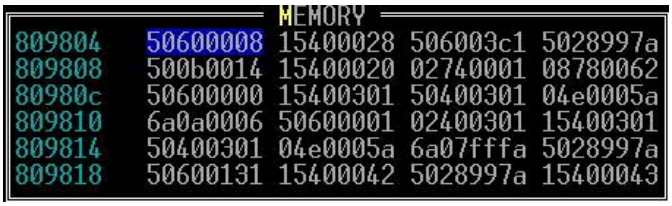
ก.3.3 หน้าต่าง MEMORY

หน้าต่าง MEMORY แสดงให้เห็นช่วงของ MEMORY ดังแสดงในภาพที่ ก-9 ในหน้าต่างจะมี 2 ส่วน คือ

- Address ในคอลัมน์แรกจะเป็นเลขของตำแหน่งแรกในหน้าต่างของคอลัมน์ข้อมูล ตำแหน่งที่คอลัมน์ข้อมูลในหน้าจอจะมีตำแหน่งเดียวกับตำแหน่งของคอลัมน์ แต่ละตำแหน่งในคอลัมน์จะแสดงลักษณะของข้อมูลที่ถูกต้อง

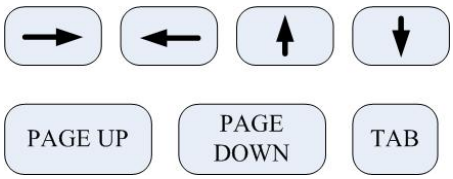
- Data ค่าที่มีอยู่ในคอลัมน์จะแสดงค่าให้เห็นในตำแหน่งนั้น

ตัวอย่างหน้าต่าง MEMORY มีข้อมูล 4 คอลัมน์ ดังนั้นในแต่ละตำแหน่งเริ่มต้นค่าจะเพิ่มขึ้นทีละ 4 ถึงหน้าต่างจะแสดงข้อมูลเพียง 4 คอลัมน์แต่ยังคงเป็นเพียงตำแหน่งของคอลัมน์เดียว เช่นที่ตำแหน่ง 0x00809804 มีค่าเท่ากับ 0x50600008, ที่ตำแหน่ง 0x00809805 มีค่าเท่ากับ 0x15400028 ที่ตำแหน่ง 0x00809808(ค่าแรกในแถวที่ 2) มีค่าเท่ากับ 0x500b0014, ที่ตำแหน่ง 0x00809809 มีค่าเท่ากับ 0x15400020 เป็นต้น



ภาพที่ ก-9 หน้าต่าง MEMORY

การเปลี่ยนแปลงค่าของหน้าต่าง MEMORY สามารถทำได้โดยกด ALT+M จึงสามารถเขียนข้อมูลทับได้ การเลือก cell สามารถใช้ key ดังนี้

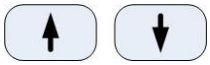


ก.3.4 หน้าต่าง COMMAND

หน้าต่าง COMMAND จะมีพื้นที่สำหรับส่งผ่านคำสั่ง ตอบรับคำสั่ง แสดงคำสั่งที่ผิดพลาด และข้อความผิดพลาด หน้าต่าง COMMAND มี 2 ส่วนคือ

- Command line ส่วนนี้มีไว้ป้อนคำสั่ง เมื่อต้องการส่งผ่านคำสั่ง
- ส่วนแสดงผล เป็นส่วนตอบรับคำสั่งที่ป้อนเข้ามาหรือแสดงเอาท์พุต อื่นๆ จากคำสั่งและแสดงข้อความผิดพลาด

ภาพที่ ก-10 หน้าต่าง COMMAND

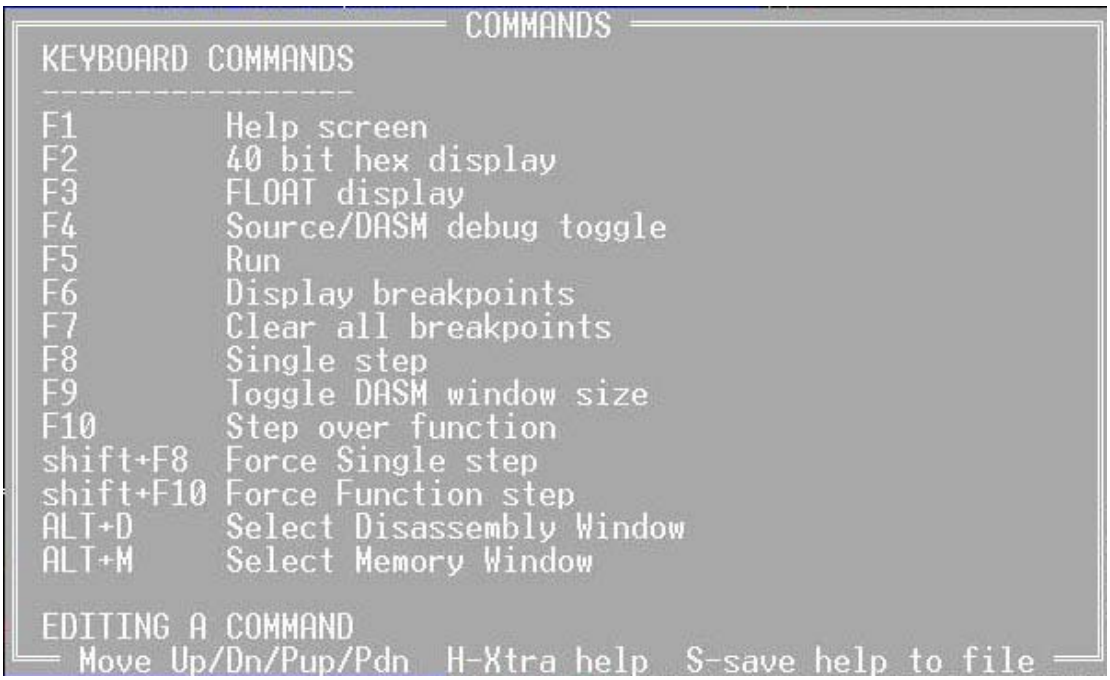
สามารถใช้  เลือกคำสั่งที่ป้อนเข้ามาก่อนจาก Buffer (เครื่องหมาย > คือตัวที่ใช้แสดง Buffer) การแก้ไข้คำสั่งแสดงในตารางที่ ก-3

ตารางที่ ก-3 การแก้ไขคำสั่ง

สิ่งที่ได้	**ใช้ Command ต่อไปนี้**
เลื่อนไปตลอดคำสั่ง	⬅ ➡
แทรกและเขียนทับ	INS
ลบตัวอักษรที่ตำแหน่งเคอร์เซอร์	DEL
เลื่อนไปยังบรรทัดเริ่มต้น	HOME
เลื่อนไปยังบรรทัดสุดท้าย	END
เคลียร์คำสั่ง	ESC
เลือกคำสั่งจาก Buffer	⬆ ⬇

ก.3.5 การใช้เมนูช่วยเหลือ

สามารถกด F1 หรือปุ่ม H เพื่อให้หน้าต่างช่วยเหลือเปิดขึ้นมาแสดงในภาพที่ ก-11 เลือกจากรายการเมนูข้างล่างในการหารายละเอียดเพิ่มเติม



ภาพที่ ก-11 รายละเอียดของหน้าต่าง Help Menu

การเลื่อนไปยังหน้าต่าง Help สามารถใช้

- PGUP เลื่อนไปยังรายการด้านบน
- PGDN เลื่อนกลับไปยังรายการเดิม
- HOME กลับไปยังรายการแรกของ Help Menu
- END ไปยังรายการสุดท้ายของ Help Menu
- S เซฟ Help Text เป็นไฟล์
- ESC ออกจาก Help Menu และกลับไปยัง Debugger
- H เพื่อผ่านไปยัง Help อันดับที่ 2 ส่วนใหญ่จะเกี่ยวกับฮาร์ดแวร์และการใช้คำสั่งเฉพาะของ Debugger

ก.4 คำสั่ง Debugger

ส่วนนี้จะแสดงตารางสรุปปุ่ม Function และคำสั่ง Debugger

ตารางที่ ก-4 คำสั่งแก้ไขในบรรทัด

สิ่งที่ได้	ใช้คำสั่ง
เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของ Command line	HOME
เลื่อนเคอร์เซอร์ไปยังจุดสุดท้ายของ Command line	END
ลบตัวอักษรด้านซ้ายของเคอร์เซอร์	DEL
ลบตัวอักษรด้านขวาของเคอร์เซอร์	SHIFT+END
เลื่อนเคอร์เซอร์ไปทางด้านซ้าย	⬅
เลื่อนเคอร์เซอร์ไปทางด้านขวา	➡

ตารางที่ ก-5 Command-Line Buffer Manipulation

สิ่งที่ได้	ใช้คำสั่ง
เรียกคำสั่งสุดท้าย	PAGE UP หรือ ⬆
เรียกคำสั่งแรกใน Command line Buffer อีกครั้ง	PAGE DOWN หรือ ⬇
ปฏิบัติในคำสั่งสุดท้ายอีกครั้ง	TAB

ตารางที่ ก-6 การสั่งงานโปรแกรม

สิ่งที่ได้	ใช้คำสั่ง
ปฏิบัติทีละโครงสร้างใน Single-step	SS
ปฏิบัติทีละ n โครงสร้าง	XN n
ปฏิบัติจนถึงโครงสร้างที่ถูกกำหนดใน addr ใน Sigle-step	XG addr
ปฏิบัติโปรแกรมจนถึง breakpoint	RUN
ปฏิบัติโปรแกรมและละเลย breakpoint run-free	RUNF

ตารางที่ ก-7 การแสดงผลและการเปลี่ยนแปลงข้อมูล

สิ่งที่ได้	ใช้คำสั่ง
การแสดงค่าที่อยู่ในหน่วยความจำที่ addr ในหน้าต่าง memory	MEM addr
การเปลี่ยนแปลงค่าในหน่วยความจำที่ addr	MM addr
ใส่ค่า lang ไปในหน่วยความจำโดยเริ่มต้นที่ addr ด้วย val เป็นค่า floating-point เฉพาะจะถูกเปลี่ยนแปลงไปเป็น ค่า floating-point ของ TMS320	MM addr leng val
แสดงภาษา Assembly ที่ addr ในหน้าต่าง DISASSEMBLY	DASM addr
แสดงรีจิสเตอร์ extended-precision แบบ 40 บิต ในหน้าต่าง Register	REG40
แสดงรีจิสเตอร์ extended-precision แบบ floating-point ในหน้าต่าง Register	FLOAT
การเปลี่ยนแปลง reg ในหน้าต่าง CPU REGISTER กับค่าจาก expression ดังตัวอย่าง PC = 0x809800 R0 = 1.34	reg = expression

ตารางที่ ก-8 การจัดการ Breakpoint

สิ่งที่ได้	ใช้คำสั่ง
เซต Breakpoint ที่ addr	SB addr
เคลียร์ Breakpoint ที่ addr	CB addr
เคลียร์ทุก Breakpoint	CB
แสดงรายการ Breakpoint ทั้งหมดที่ถูกเซท	DB

ตารางที่ ก-9 การโหลดโปรแกรม

สิ่งที่ได้	ใช้คำสั่ง
โหลดไฟล์	LOAD filename
โหลดสัญลักษณ์	SLOAD filename
โหลดไบนารี	BLOAD filename
เคลียร์สัญลักษณ์	SCLEAR

ตารางที่ ก-10 Performing System Tasks

สิ่งที่ได้	ใช้คำสั่ง
รีเซท DSK	RESET
ออกจาก Debugger	QUIT, EXIT
ออกไปยัง DOS และปฏิบัติตามคำสั่ง, และพิมพ์ EXIT เพื่อกลับมาที่ Debugger	DOS (Expression to Run)
ออกไปยัง DOS และทำการแก้ไขไฟล์(ถ้าไม่มีไฟล์ให้แก้ไขจะโหลดไฟล์ปัจจุบันมาใช้)	EDIT filename
ออกไปยัง DOS และแปลง DSK แอสเซมเบอไปเป็นไฟล์แอสเซมเบอ	dsk3a filename.asm

ก.5 Quick Reference Guide

ตารางที่ ก-11 ปุ่ม Shortcuts ฟังก์ชันสำหรับหน้าต่าง DISASSEMBLY

ปุ่มฟังก์ชัน	คำอธิบาย
F1	หน้าจอ Help
F2	เซท Breakpoint ที่เคอร์เซอร์
F3	เคลียร์ Breakpoint ที่เคอร์เซอร์
F4	Run ที่เคอร์เซอร์
F5	Run
F6	หน้าจอ Breakpoint
F7	เคลียร์ Breakpoint ทั้งหมด
F8	Run โปรแกรมทีละขั้น
F9	Grow windows
F10	Step over
SHIFT+F9	เลือกหน้าต่าง DISASSEMBLY
ESC หรือ ENTER	Ecsape

ตารางที่ ก-12 ปุ่ม Shortcuts ฟังก์ชันสำหรับหน้าต่าง CPU

ปุ่มฟังก์ชัน	คำอธิบาย
F1	หน้าจอ Help
ESC	ออกจากหน้าต่าง CPU
HOME	เลื่อนขึ้นด้านบน
END	เลื่อนลงด้านล่าง
⬆,⬇	เลื่อน cell ในแนวตั้ง
TAB	เลื่อน cell ในแนวนอน

ตารางที่ ก-13 ปุ่ม Shortcuts ฟังก์ชันสำหรับหน้าต่าง MEMORY

ปุ่มฟังก์ชัน	คำอธิบาย
F1	หน้าจอ Help
F9	Toggle windows size
ESC	ออกจากหน้าต่าง MEMORY
HOME	เลื่อนขึ้นด้านบน
END	เลื่อนลงด้านล่าง
PAGE UP, PAGE DOWN	เลื่อนหน้าขึ้นหรือลง
⬆,⬇	เลื่อน cell ในแนวตั้ง
TAB	เลื่อน cell ในแนวนอน

ตารางที่ ก-14 ปุ่ม Shortcuts ฟังก์ชันสำหรับหน้าต่าง COMMAND

ปุ่มฟังก์ชัน	คำอธิบาย
F1	แสดงรายการคำสั่ง
F2	รีจิสเตอร์ extended-precistion ฐาน 16 แบบ 40 บิต
F3	รีจิสเตอร์ extended-precistion ฐาน 10 แบบ floating-point
F4	Toggle ระหว่างการแสดงไฟล์ source และ หน่วยความจำ DISASSEMBLY
F5	Run โปรแกรมจนถึง Breakpoint ต่อไป
F6	แสดง Breakpoint ทั้งหมด

ตารางที่ ก-14 (ต่อ)

ปุ่มฟังก์ชัน	คำอธิบาย
F7	เคลียร์ Breakpoint ทั้งหมด
F8	Run โปรแกรมทีละขั้น
F9	Toggle the DISASSEMBLY eindows size
F10	Run โปรแกรมทีละขั้นและกลับไปขั้นตอนที่ผ่านมาด้วย
ALT+D	เลือกหน้าต่าง DISASSEMBLY
ALT+M	เลือกหน้าต่าง MEMORY
ALT+C	เลือกหน้าต่าง COMMAND
ESC	ออกจากหน้าต่างที่ทำงาน

ภาคผนวก ข

โปรแกรมที่ใช้บนบอร์ด TMS320C31 DSP STARTER KIT

ข.1 โปรแกรมลดขนาดสัญญาณคลื่นไฟฟ้าหัวใจ 2 เท่า

```
#include "c:\C3xTOOL\aiccomc.c"
#include "c:\C3xTOOL\math.h"

int AICSEC[4] = {0x162c,0x1,0x72e6,0x63};
volatile int *mem1 = (volatile int*) 0x100000;
volatile int *mem2 = (volatile int*) 0x700000;
main()
{
int result = 1,x0,i;
float fx[8];
float y2 = 0.1, y1 = 0.1,y0;
float x2,x1;
float a0 = 1.0000, a1 = 0 , a2 = 0.1716;
float b0 = 1.0000, b1= 0 , b2 = 0.2929;
int u;
int j;
int x[8];
float yy;
  //******************save 2  *****************
   AICSET();
   i = 32766;
       mem1 = (volatile int*)0x100000;
       u=1;
        while(i ! = 1)
        {
    x0 = UPDATE_SAMPLE(result);
    j = u%2; // time 2
          if (j == 0)
          {
            *mem1=x0;
             mem1++;
             i--;
          }
     u = u +1;
    }
```

```
//*****************************************
      mem1 = (volatile int*)0x100000;
      mem2 = (volatile int*)0x700000;
        i = 32764;
         while(i > 0)
         {
                *mem2=*mem1;
                 mem1++;
                 mem2++;
                 mem2++;
   i--;
                i--;
        }

  mem2 = (volatile int*)0x700000;
    i =32764;
        while(i > 0)
        {
                 fx[1]=*mem2;
                 mem2++;
                 mem2++;
      fx[2]=*mem2;
                 mem2++;
                 mem2++;
                 fx[3]=*mem2;
                 //---------- lagpol---------------
                yy = ((0.375*fx[1])+(0.75*fx[2])+(-0.125*fx[3]));
     //---------------------------------
                Mem2--;
                Mem2--;
                Mem2--;
                *mem2= (int)(yy);
                Mem2++;
```

```
                i--;
                i--;
          }

        //************ loader output ***************
    while(1)
        {
        mem2 = (volatile int*)0x700000;
        i =32764;
    while(i != 0)
        {
                result = *mem2;
                x0 = UPDATE_SAMPLE(result);
                mem2++;
                i--;
        }
        }
        while(1);
}
```

ข.2 โปรแกรมลดขนาดสัญญาณคลื่นไฟฟ้าหัวใจ 4 เท่า

```
#include "c:\C3xTOOL\aiccomc.c"
#include "c:\C3xTOOL\math.h"

int AICSEC[4] = {0x162c,0x1,0x72e6,0x63};
volatile int *mem1 = (volatile int*) 0x100000;
volatile int *mem2 = (volatile int*) 0x700000;
main()
{
int result = 1,x0,i;
float fx[8];
float y2 = 0.1, y1 = 0.1,y0;
float x2,x1;
float a0 = 1.0000, a1 = 0 , a2 = 0.1716;
float b0 = 1.0000, b1=0 , b2 = 0.2929;
int u;
int j;
int x[8];
float yy[5];
  //******************save  4  *****************
   AICSET();
   i = 32766;
       mem1 = (volatile int*)0x100000;
```

```
    u=1;
     while(i != 0)
     {
  x0 = UPDATE_SAMPLE(result);
  j=u%4; // time  4
        if (j == 0)
        {
           *mem1=x0;
            mem1++;
            i--;
        }
   u=u+1;
  }
//***********************************************
      mem1 = (volatile int*)0x100000;
      mem2 = (volatile int*)0x700000;
      i=32764;
         while(i > 0)
         {
                *mem2=*mem1;
                 mem1++;
                 mem2++;
                 mem2++;
                 mem2++;
                 mem2++;
                 i--;
     i--;
                 i--;
                 i--;
        }

  mem2 = (volatile int*)0x700000;
  i=32764;
         while(i > 0)
         {
                 fx[1]=*mem2;
                 mem2++;
                 mem2++;
                 mem2++;
                 mem2++;
       fx[2]=*mem2;
                 mem2++;
                 mem2++;
                 mem2++;
                 mem2++;
                 fx[3]=*mem2;
                 //----------- lagpol----------
```

```
                yy[0] = ((0.6565*fx[1])+(0.4375*fx[2])+(-0.0938*fx[3]));
        yy[1] = ((0.375*fx[1])+(0.750*fx[2])+(-0.1250*fx[3]));
        yy[2] = ((0.15625*fx[1])+(0.9375*fx[2])+(-0.0938*fx[3]));
        //---------------------------------
                mem2--;
                mem2--;
                mem2--;
                mem2--;
                mem2--;
                mem2--;
                mem2--;
                *mem2= (int)(yy[0]);
                mem2++;
        *mem2= (int)(yy[1]);
                mem2++;
        *mem2= (int)(yy[2]);
                mem2++;

                i--;
                i--;
                i--;
                i--;
        }

        //************** loader output ****************
        while(1)
        {
        mem2 = (volatile int*)0x700000;
        i=32764;
    while(i != 0)
        {
                result = *mem2;
                x0 = UPDATE_SAMPLE(result);
                mem2++;
                i--;
        }
        }
        while(1);
}
```

ข.3 โปรแกรมลดขนาดสัญญาณคลื่นไฟฟ้าหัวใจ 8 เท่า

```
#include "c:\C3xTOOL\aiccomc.c"
#include "c:\C3xTOOL\math.h"

int AICSEC[4] = {0x162c,0x1,0x72e6,0x63};
volatile int *mem1 = (volatile int*) 0x100000;
volatile int *mem2 = (volatile int*) 0x700000;
main()
{
int result = 1,x0,i;
float fx[8];
float y2 = 0.1, y1 = 0.1,y0;
float x2,x1;
float a0 = 1.0000, a1 = 0 , a2 = 0.1716;
float b0 = 1.0000, b1=0 , b2 = 0.2929;
int u;
int j;
int x[8];
float yy[10];
  //******************save  8 *****************
   AICSET();
   i = 32766;
       mem1 = (volatile int*)0x100000;
       u=1;
        while(i != 0)
        {
    x0 = UPDATE_SAMPLE(result);
    j=u%8; // time  8
          if (j == 0)
          {
             *mem1=x0;
              mem1++;
              i--;
          }
     u=u+1;
    }
  //*********************************************
        mem1 = (volatile int*)0x100000;
        mem2 = (volatile int*)0x700000;
        i=32764;
       while(i > 0)
          {
                  *mem2=0x00000000;
                  mem2++;
                  i--;
          }
        mem2 = (volatile int*)0x700000;
```

```
    i=32764;
      while(i > 0)
       {
              *mem2=*mem1;
               mem1++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               i--;
   i--;
               i--;
               i--;
               i--;
   i--;
               i--;
               i--;
      }
mem2 = (volatile int*)0x700000;
i=32764;
       while(i > 0)
       {
               fx[1]=*mem2;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
     fx[2]=*mem2;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               mem2++;
               fx[3]=*mem2;
               //----------- lagpol---------------
              yy[0] = ((0.8203*fx[1])+(0.2343*fx[2])+(-0.0546*fx[3]));
```

```
yy[1] = ((0.6562*fx[1])+(0.4375*fx[2])+(-0.0937*fx[3]));
yy[2] = ((0.5078*fx[1])+(0.6093*fx[2])+(-0.1171*fx[3]));
        yy[3] = ((0.3750*fx[1])+(0.7500*fx[2])+(-0.1250*fx[3]));
yy[4] = ((0.2578*fx[1])+(0.8593*fx[2])+(-0.1171*fx[3]));
yy[5] = ((0.1562*fx[1])+(0.9375*fx[2])+(-0.0937*fx[3]));
        yy[6] = ((0.0703*fx[1])+(0.9843*fx[2])+(-0.0546*fx[3]));
//---------------------------------
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        mem2--;
        *mem2= (int)(yy[0]);
        mem2++;
*mem2= (int)(yy[1]);
        mem2++;
*mem2= (int)(yy[2]);
        mem2++;
*mem2= (int)(yy[3]);
        mem2++;
*mem2= (int)(yy[4]);
        mem2++;
*mem2= (int)(yy[5]);
        mem2++;
*mem2= (int)(yy[6]);
        mem2++;
        i--;
        i--;
        i--;
        i--;
        i--;
        i--;
        i--;
        i--;
}
```

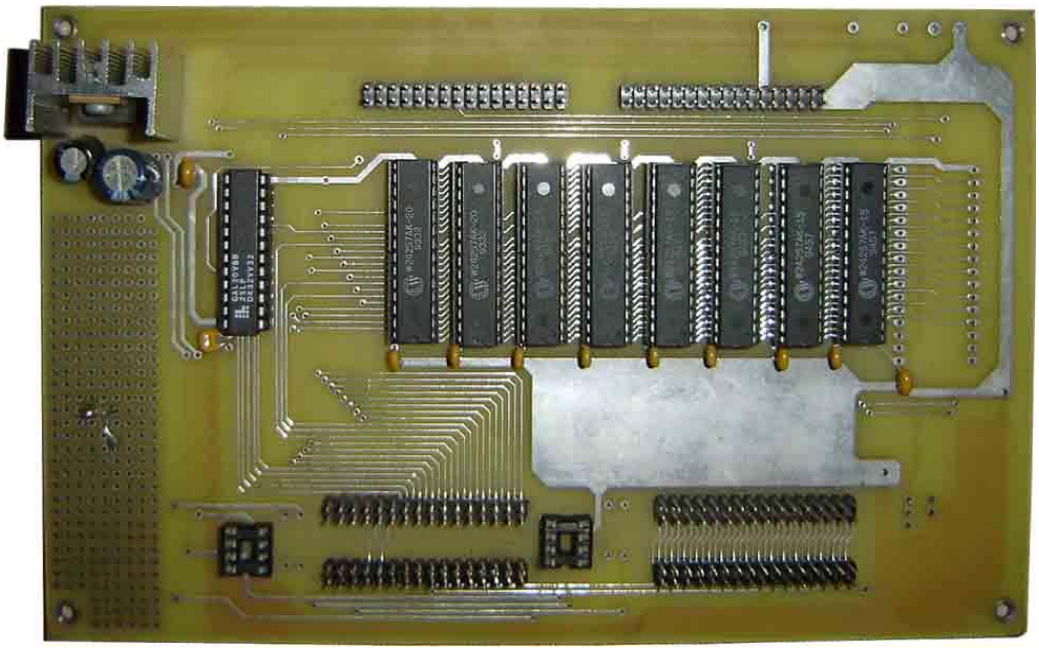
```
        //************** loader output ****************
        while(1)
        {
        mem2 = (volatile int*)0x700000;
        i=32764;
    while(i != 0)
        {
                result = *mem2;
                x0 = UPDATE_SAMPLE(result);
                mem2++;
                i--;
        }
        }
        while(1);
}
```

ภาคผนวก ค

บอร์ดหน่วยความจำภายนอก

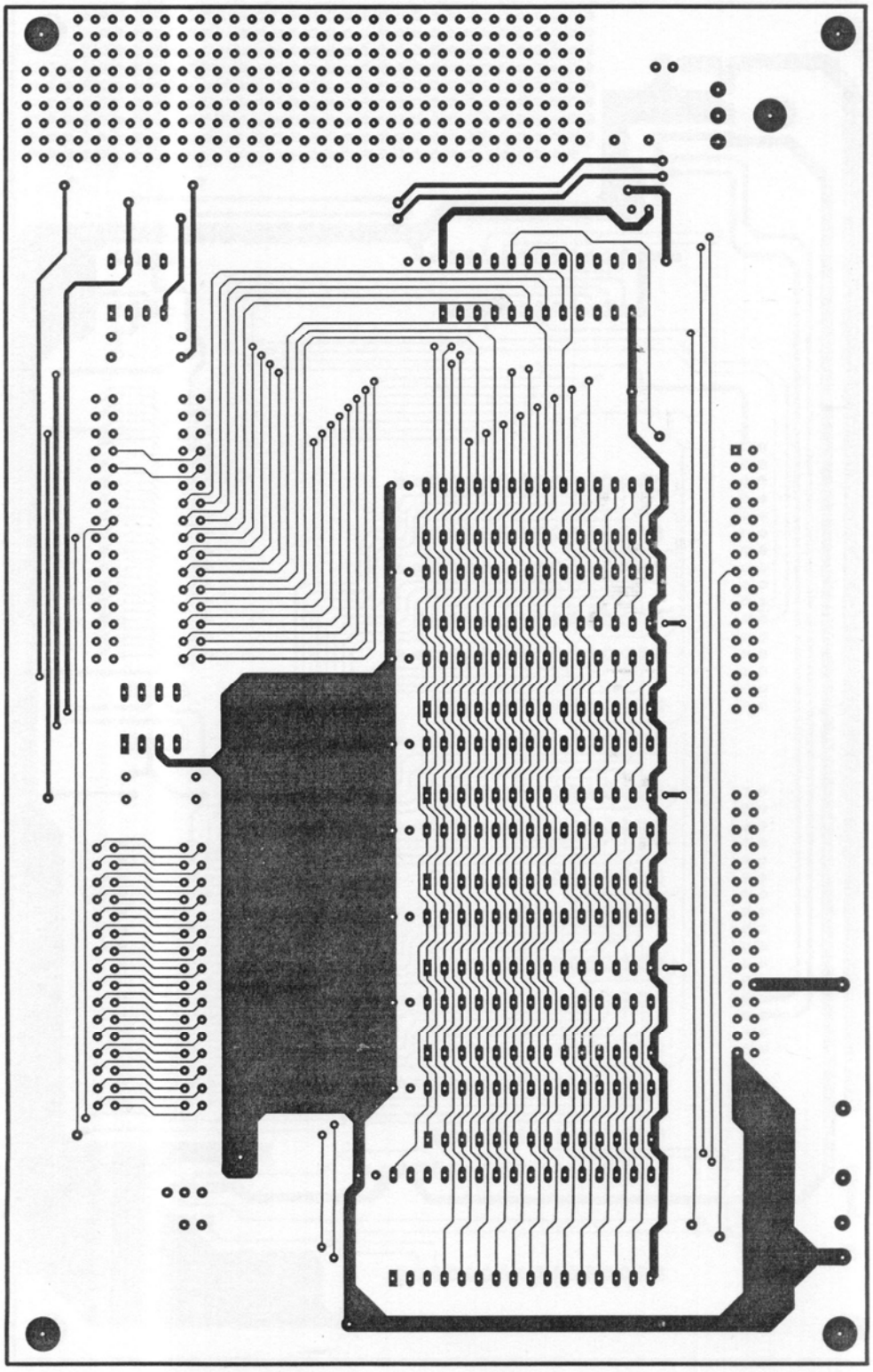
ค.1 การออกแบบบอร์ดหน่วยความจำภายนอก

หน่วยความจำข้อมูลหรือ RAM ที่ใช้ในการออกแบบคือ IC เบอร์ 61256 เป็นหน่วยความจำขนาด 32 K ที่สามารถอ่านและเขียนข้อมูลเข้าไปเก็บไว้ในตัวใน ข้อมูลที่เก็บไว้ภายใน RAM จะสูญหายไป เมื่อไม่มีการจ่ายไฟเลี้ยง จึงใช้เป็นตัวเก็บโปรแกรมขอมูลชั่วคราว RAM ที่นำมาใช้มีขนาด 8 บิต จึงต้องใช้ RAM 4 ตัวมาต่อ DATA BUSES ขนาด 32 บิต จึงจะได้ RAM ขนาด 32 K เมื่อต้องการ RAM ขนาด 64 K จึงต้องใช้ RAM ทั้งหมด 8 ตัว ส่วนการถอดรหัสสัญญาณ Address จะใช้ GAL เบอร์ PAL22V10 ทำหน้าที่เป็นตัวถอดรหัสสัญญาณ Address ที่ใช้ในการเลือกตำแหน่งของ RAM ซึ่งสามารถถอดรหัสออกมาได้ 2 ช่วงตามสถานะของสัญญาณ A8 ถึง A23 คือช่วง 100000h-107FFFh ซึ่งจะเป็นของ RAM ชุดที่ 1 และช่วง 108000h-10FFFEh จะเป็น RAM ของชุดที่ 2 โดยในงานวิจัยนี้จะออกแบบโดยใช้ RAM ชุดที่ 1 เป็น RAM ในการเก็บสัญญาณ คลื่นไฟฟ้าหัวใจ ต้นฉบับจากเครื่อง จำลองคลื่นไฟฟ้าหัวใจ ที่ผ่านการลดขนาดสัญญาณลงแล้ว และจะใช้ หน่วยความจำภายในบอร์ด TMS320C31 DSKเป็นส่วนที่เก็บซอฟท์แวร์ที่ใช้ในการประมวลผล และนำสัญญาณคลื่นไฟฟ้าหัวใจ ที่ผ่านกระบวนการในคืนค่าสัญญาณ มาเก็บไว้ในส่วน RAM ชุดที่ 2 ของ บอร์ดหน่วยความจำภายนอก



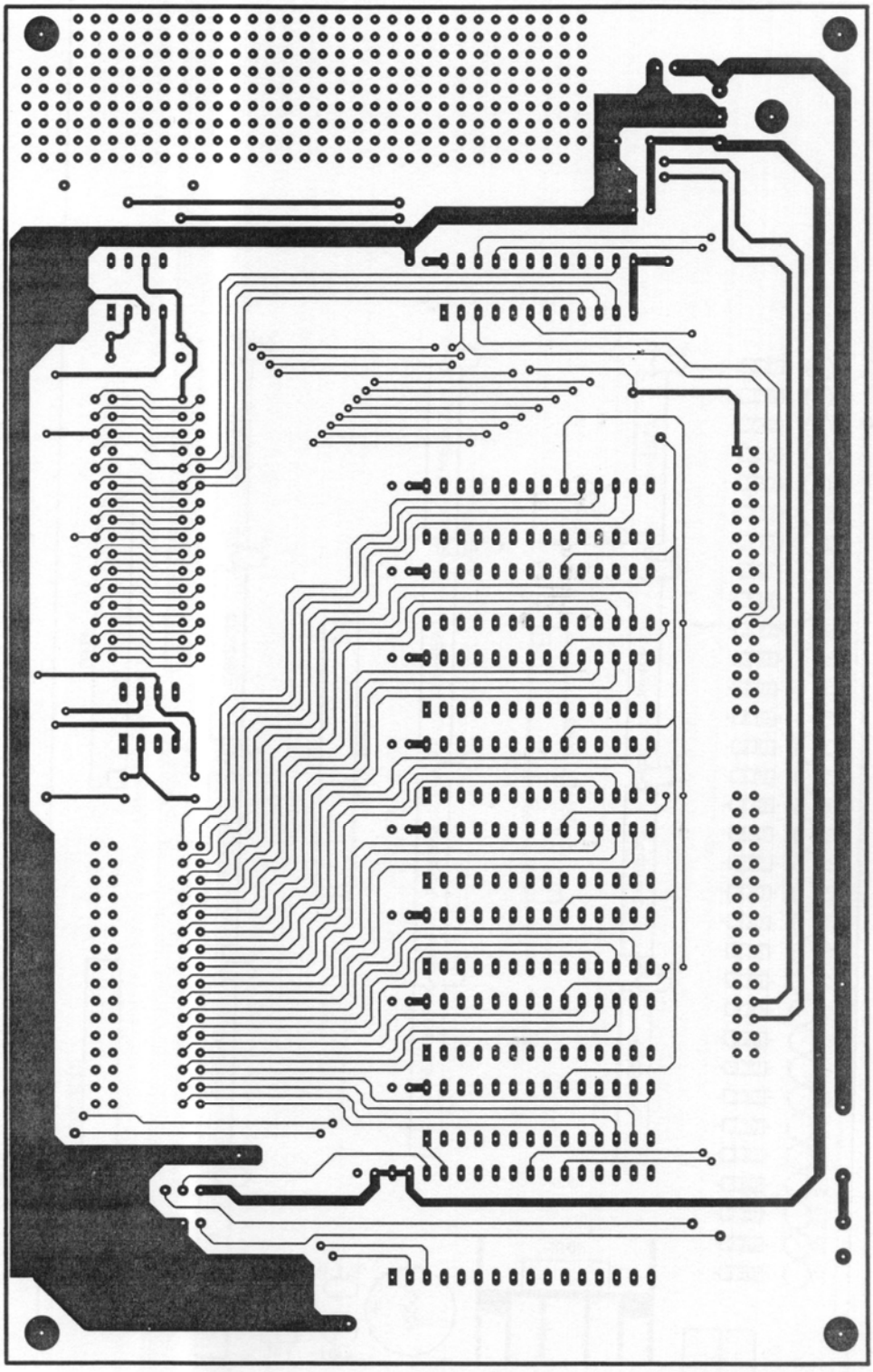
ภาพ ค-1 บอร์ดหน่วยความจำภายนอก

ค.2 ลายวงจรบอร์ดหน่วยความจำภายนอก(ด้านหน้า)



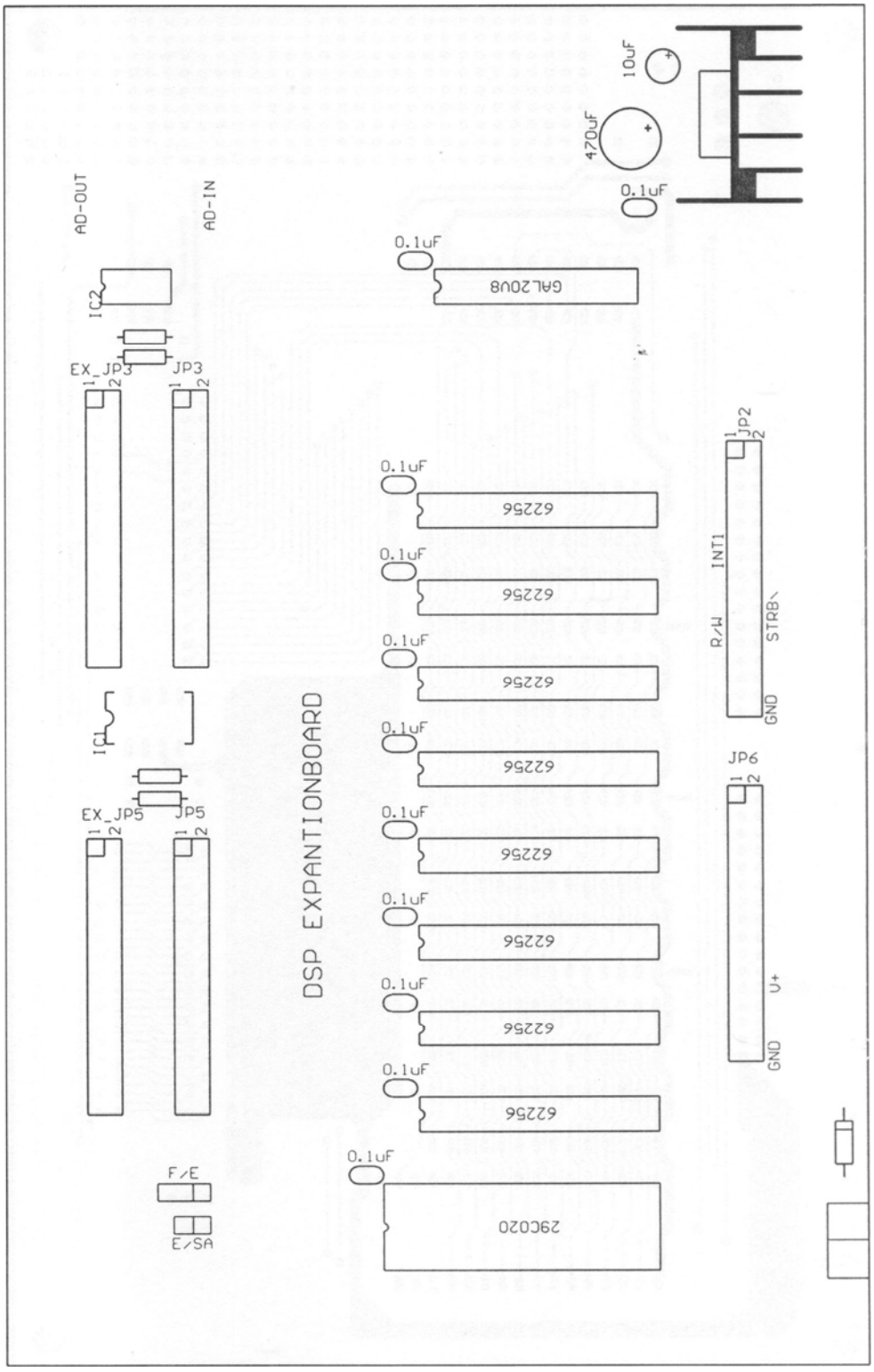
ภาพ ค-2 ลายวงจรบอร์ดหน่วยความจำภายนอก(ด้านหน้า)

ค.3 ลายวงจรบอร์ดหน่วยความจำภายนอก(ด้านหลัง)



ภาพ ค-3 ลายวงจรบอร์ดหน่วยความจำภายนอก(ด้านหลัง)

ค.4 ตำแหน่งการวางอุปกรณ์บนบอร์ดหน่วยความจำภายนอก



ภาพ ค-4 การวางอุปกรณ์บนบอร์ดหน่วยความจำภายนอก

ค.5 โปรแกรม PAL20V8

```
CHIP C3x GAL20V8 ; 7000000-70ffff
NC RW STRB A15 A16 A17 A18 A19 A20 A21 A22 GND
OE A23 T1 T2 T11 T22 CES2 CES1 OEF CEF NC VCC
EQUATIONS
/CES1 = /STRB*/A23*/A22*/A21*A20*/A19*/A18*/A17*/A16*/A15 ; 0x100000-107fff
/CES2 = /STRB*/A23*A22*A21*A20*/A19*/A18*/A17*/A16*/A15   ; 0x700000-707fff
/CEF = /STRB*/A23*A22*/A21*/A20*/A19*/A18*/A17
OEF = /RW
/T1 = /A23*/A22*A21*A20*/A19*/A18*/A17*/A16*/A15       ; 300000
/T2 = /A23*/A22*/A21*A20*/A19*/A18*/A17*/A16*/A15      ; 900000
/T11 = /STRB*/A23*/A22*A21*A20*/A19*/A18*/A17*/A16*/A15     ; 300000
/T22 = /STRB*A23*/A22*/A21*A20*/A19*/A18*/A17*/A16*/A15     ; 900000
```

ภาคผนวก ง

สถาปัตยกรรมของบอร์ด TMS320C31 DSP STARTER KIT

ง.1 สถาปัตยกรรมของ TMS320C31

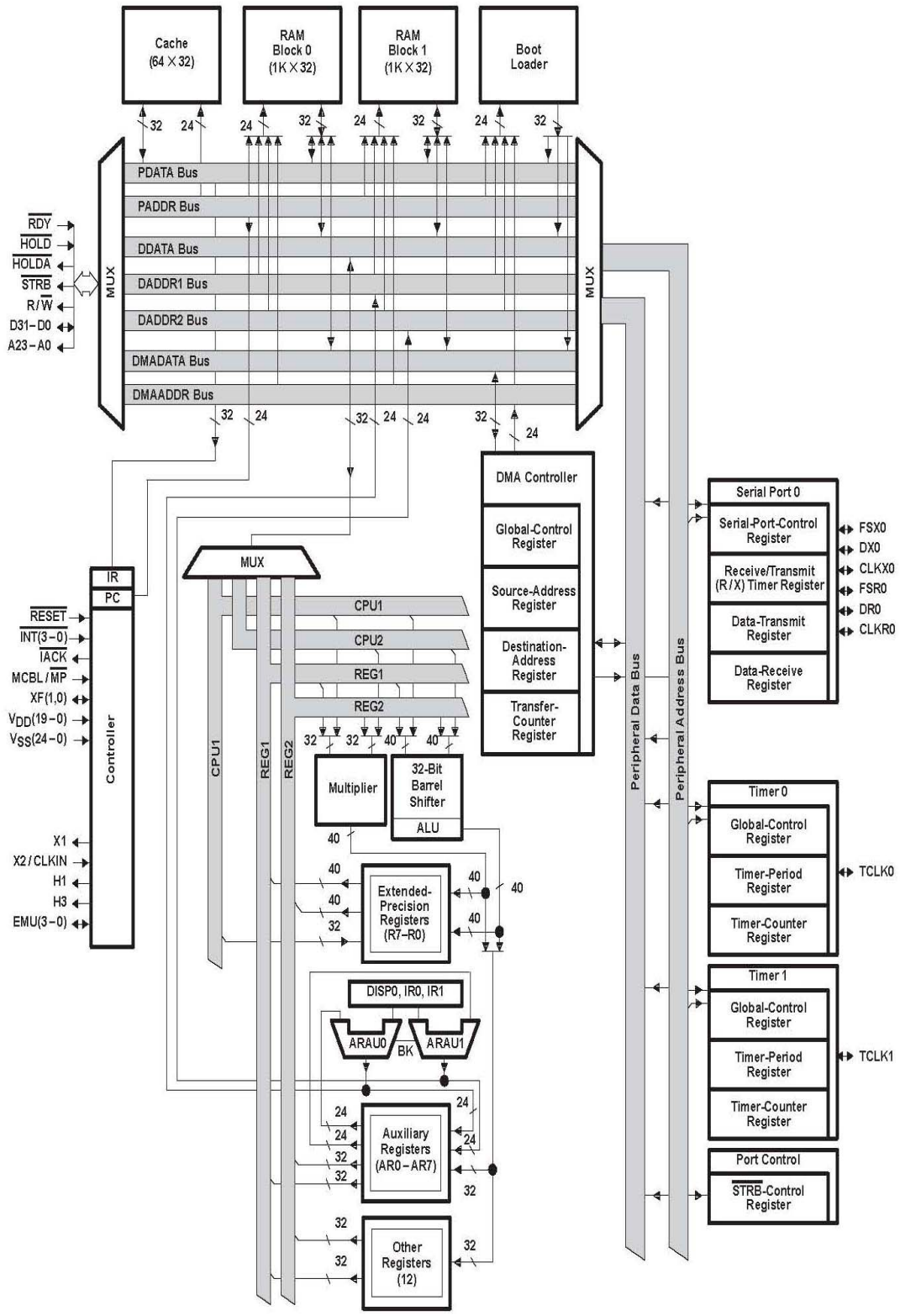
สถาปัตยกรรมของ TMS320C31 ประกอบด้วยส่วนหลักๆดังนี้

- หน่วยประมวลผลกลาง (CPU)
- หน่วยความจำ (Memory Organization)
- ลักษณะบัสภายใน (Internal Bus Operation)
- ลักษณะบัสภายนอก (External Bus Operation)
- อุปกรณ์สนับสนุน (Periphrasis)
- ชุดควบคุมการเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access Controller)

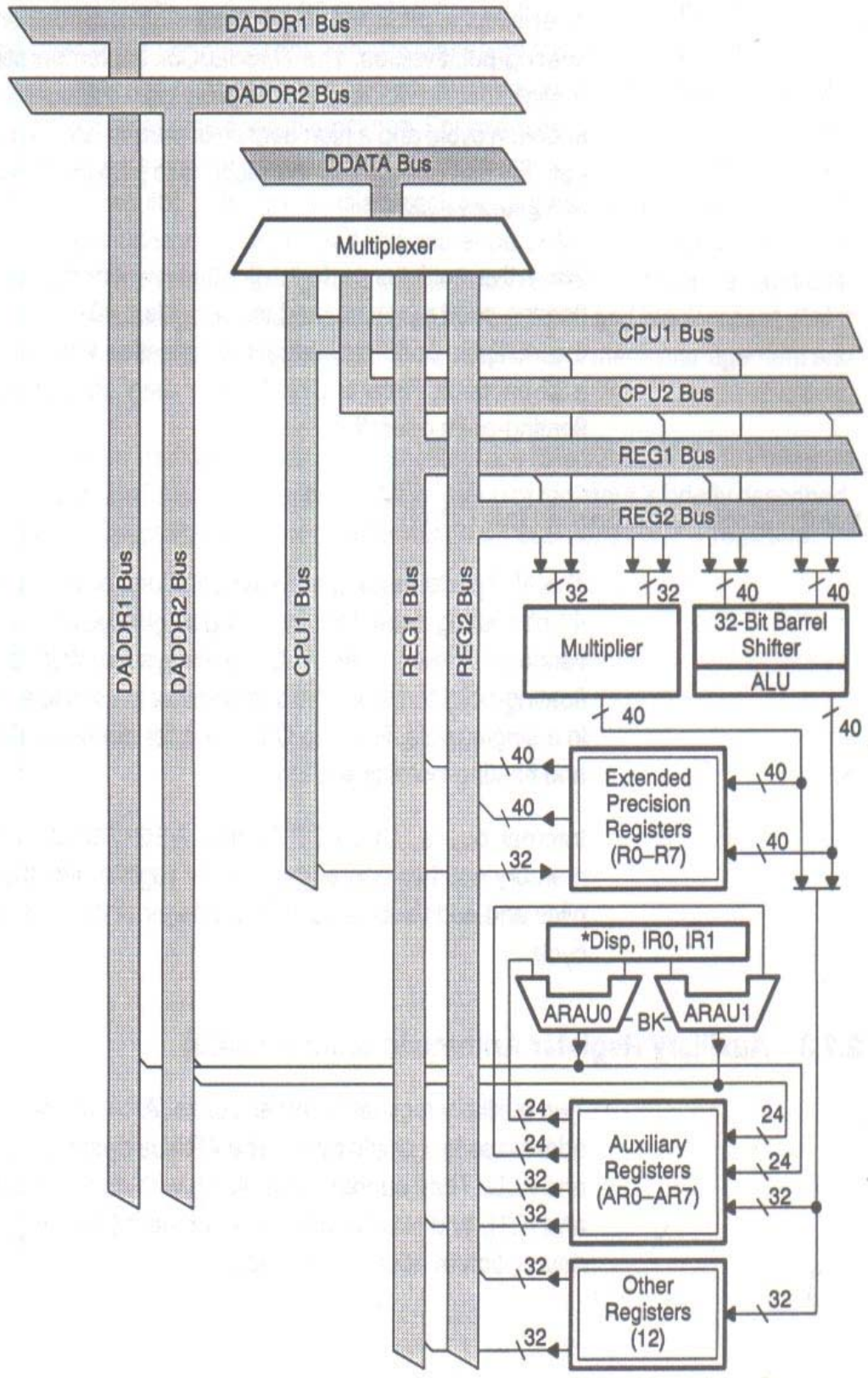
ง.1.1 หน่วยประมวลผลกลาง (CPU)

สถาปัตยกรรมของ TMS320C31 ถูกสร้างแบบ Harvest Architecture ทำให้หน่วยประมวลผลกลางมีประสิทธิภาพเพิ่มขึ้น สามารถทำงานไปพร้อมๆกันได้ สำหรับหน่วยประมวลผลกลางจะประกอบด้วยส่วนย่อยต่างๆ ดังนี้

- ตัวคูณ (Multiplier)
- หน่วยคำนวณ (ALU)
- 32-บิต barrel Shifter
- บัสภายใน (Internal bus)
- รีจิสเตอร์ช่วย (Auxiliary register)
- แฟ้มรีจิสเตอร์ของซีพียู (CPU register file)



ภาพที่ ง-1 บล็อกไดอะแกรมของ TMS320C31



ภาพที่ ง-2 หน่วยประมวลผลกลางของ (CPU) ของ TMS320C31

ง.1.1.1 ตัวคูณ (Multiplier)

ตัวคูณจะทำงานภายใน 1 ไซเคิล ซึ่งจัดการเกี่ยวกับตัวเลขจำนวนเต็ม (Integer) ขนาด 24, 32 บิต และตัวเลขแบบจุดทศนิยม (Floating Point) สามารถทำงานแบบขนานไปพร้อมกับการทำงานของ ALU ซึ่งจะมีคำสั่งการทำงานแบบขนาน

เมื่อตัวคูณทำการคูณข้อมูลขนาด 32 บิต แบบจุดทศนิยม จะถูกนำมาทำการคำนวณจะได้ผลลัพธ์เป็นขนาด 40 บิต (Extend precision) หรือถ้าเป็น 24 บิต จะได้ผลลัพธ์เป็นขนาด 32 บิต

ง.1.1.2 หน่วยคำนวณALU

ALU จะทำงานภายใน 1 ไซเคิล บนเลขจำนวนเต็มขนาด 32 บิต และตัวเลขแบบจุดทศนิยมขนาด 40 บิต สามารถแปลงตัวเลขระหว่างเลขจำนวนเต็มกับตัวเลขแบบจุดทศนิยมผลลัพธ์ของ ALU จะเป็น 32 และ 40 บิต เสมอภายใน ALU มีวงจรเลื่อน (Barrel Shifter) ขนาด 32 บิต ระบบบัสภายใน CPU1/CPU2 และ REG1/REG2 ซึ่งสามารถนำโอเปอร์แรน สองชุดจากหน่วยความจำและโอเปอร์แรนสองชุดจากแฟ้มรีจิสเตอร์ ซึ่งการทำเช่นนี้จะเห็นว่าสามารถทำงานแบบขนานระหว่าง ALU ไปพร้อมกับการทำงานของตัวคูณ (Multiplier) ได้ภายใน 1 ไซเคิล

ง.1.1.3 Auxiliary Register Arithmetic Unit (ARAUs)

หน่วยคำนวณสำหรับรีจิสเตอร์ช่วย (ALU) สองตัวสามารถกำเนิดสัญญาณแอดเดรสสองสัญญาณได้พร้อมกัน ARAU สามารถทำงานได้พร้อมกับตัวคูณและ ALU การอ้างแอดเดรสเป็นแบบ displacement, index register (IR0, IR1), circular, บิต-reversed

ง.1.1.4 แฟ้มรีจิสเตอร์ (CPU Register File)

TMS320C31 มี 28 รีจิสเตอร์ที่สนับสนุนการทำงานของตัวคูณ และ ALU สามารถใช้รีจิสเตอร์เหล่านี้ในลักษณะงานทั่วๆไปตามต้องการ ซึ่งอย่างไรก็ตามรีจิสเตอร์จะถูกแบ่งเพื่อให้ทำงานเฉพาะอย่าง เช่น รีจิสเตอร์ R0 – R7 (Extended Precision Register) จะถูกออกแบบมาเพื่อให้เก็บผลลัพธ์ที่เป็นเลขแบบจุดทศนิยม รีจิสเตอร์ช่วย (Auxiliary AR0 – AR7) ถูกออกแบบมาเพื่อใช้งานทั่วๆไป การอ้างแอดเดรสเป็นแบบตรง (Direct Addressing Mode) ใช้เก็บที่เป็นตัวเลขจำนวนเต็มขนาด 32 บิต และการทำงานทางตรรกะ (Logical) รีจิสเตอร์ที่เหลือจะทำหน้าที่เกี่ยวกับระบบ เช่น การอ้างแอดเดรส การจัดการสแตก(Stack) การจัดการสถานะ (Status) การขัดจังหวะ (Interrupt) การทำงานซ้ำๆเป็นกลุ่ม (Block Repeat)

ตารางที่ ง-1 รีจิสเตอร์ต่างๆและความหมาย

ชื่อรีจิสเตอร์	หน้าที่การทำงาน
R0 – R7	Extended – precision register 0 – 7
AR0 – AR7	Auxiliary register 0 – 7
DP	Data – page pointer
IR0	Index register 0
IR1	Index register 1
Bk	Block size
SP	System stack pointer
ST	Status register
IE	CPU/DMA Interrupt enable
IF	CPU Interrupt flags
IOF	I/O flags
RS	Repeat start address
RE	Repeat end address
RC	Repeat counter
PC	Program Counter

รีจิสเตอร์ขนาดพิเศษ (Extended – Precision Register R0 – R7) เป็นรีจิสเตอร์ขนาด 40 บิต สนับสนุนการทำงานของตัวเลขจำนวนเต็มขนาด 32 บิต และ 40 บิต สำหรับเลขจุดทศนิยม คำสั่งใดๆจะถือว่าเป็นเลขจุดทศนิยมบิตที่ 0 – 39 ถ้าคำสั่งเป็นการทำงานของเลขจำนวนเต็ม บิตที่ 0 – 31 จะถูกใช้ ส่วนบิตที่ 32 – 39 จะยังคงเหมือนรูปแบบการเก็บข้อมูลเดิม

รีจิสเตอร์ช่วย (Auxiliary Register AR0 – AR7) สามารถที่จะเข้าถึงโดย CPU และสามารถถูกเปลี่ยนแปลงแก้ไขโดย ARAUs ซึ่งเป็นหน่วยความจำสำหรับรีจิสเตอร์ช่วย (มีสองตัว) ซึ่งมีหน้าที่พื้นฐานของรีจิสเตอร์ช่วย คือ การกำเนิดสัญญาณแอดเดรสขนาด 24 บิต สามารถใช้งานในลักษณะที่เป็นตัวนับจำนวนรอบ (Loop Counter) และยังคงสามารถเก็บข้อมูลทั่วๆไปขนาด 32 บิต ซึ่งสามารถถูกแก้ไขเปลี่ยนแปลงโดยตัวคูณ (Multiplier) และ ALU

Data Page Pointer (DP) เป็นรีจิสเตอร์ขนาด 32 บิต บิต0 – 7 (8 บิต LSB) ของรีจิสเตอร์ DP ถูกใช้โดยโหมด การอ้างแอดเดรสแบบตรง ทำหน้าที่เสมือนตัวชี้เพจ (Page) ของข้อมูล 1 เพจ มีขนาด 64 กิโลเวิร์ด (64 Kwords) ซึ่งมีทั้งหมด 256 เพจ

Index Register (IR0, IR1) เป็นรีจิสเตอร์ขนาด 32 บิต เป็นตัวเก็บข้อมูลโดย ARAU (Auxiliary Register Arithmetic Unit) เพื่อทำการประมวลแอดเดรสแบบ อินเด็กส์ (Index Addressing)

System Stack Pointer (SP) เป็นรีจิสเตอร์ขนาด 32 บิต เป็นตัวเก็บค่าของแอดเดรสของสแตค ส่วนบนสุดของคำสั่งพุช (Push) จะมีผลทำให้ SP มีค่าเพิ่มขึ้น 1 และ POP มีผลทำให้

SP มีค่าลดลง 1 นอกจากคำสั่ง Push และ POP แล้วคำสั่ง TRAP, CALLS, RETURN ก็จะมีผลต่อรีจิสเตอร์ SP ด้วย

Status Register (ST) เป็นตัวเก็บข้อมูลซึ่งแสดงถึงสถานะต่างๆของ ซีพียู เช่น ผลจากการกระทำคำสั่งต่างๆสถานะของ Status Register จะเปลี่ยนแปลงไป เช่น ผลลัพธ์เป็น 0, เป็นค่าลบเป็นต้น

CPU/DMA Interrupt Enable Register (IE) เป็นรีจิสเตอร์ขนาด 32 บิต บิตที่ 0 – 10 จะเป็นการอนุญาตให้ทำการ Interrupt ได้หรือไม่ การอนุญาตให้ทำการเข้าถึงหน่วยความจำโดยตรง (DMA) จะอยู่ที่ตำแหน่งบิต 16 – 26

CPU Interrupt Flag Register (IF) เป็นรีจิสเตอร์ขนาด 32 บิต บิตที่ 0 – 10 จะบอกสถานะของการ interrupt และบิตที่ 16 – 26 จะบอกสถานะของการทำดีเอ็มเอ

I/O Flags Register (IOF) ใช้ควบคุมการทำงานของขา XF0 และขา XF1 ซึ่งขาเหล่านี้จะถูกติดตั้งให้เป็น Input และ Output

Program Counter (PC) เป็นรีจิสเตอร์ขนาด 32 บิต ใช้เก็บตำแหน่งแอดเดรสของคำสั่งก่อนหน้าคำสั่งปัจจุบัน

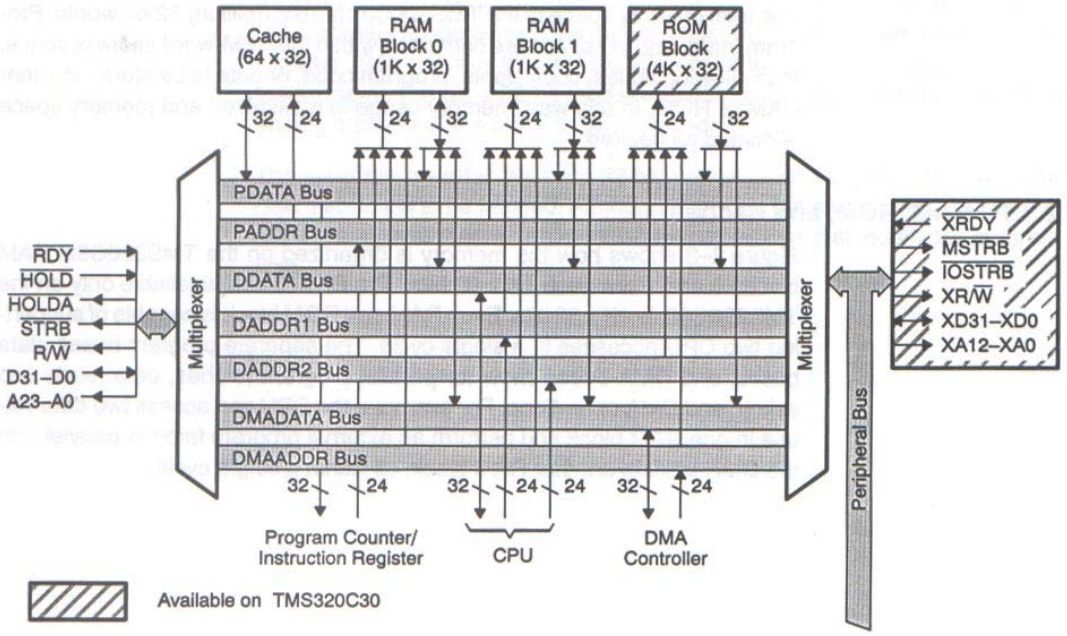
Repeat Counter (RC) เป็นรีจิสเตอร์ขนาด 32 บิต สำหรับเก็บค่าจำนวนครั้งในการกระทำคำสั่งแบบซ้ำๆกันเป็นกลุ่ม (Block Repeat) ถ้าซีพียูทำงานในโหมดกระทำซ้ำ (Repeat Mode) ซีพียูจะกระทำซ้ำตั้งแต่ตำแหน่งที่ถูกเก็บอยู่ในรีจิสเตอร์ Repeat Start Address (RS) จนถึงตำแหน่งที่ถูกเก็บในรีจิสเตอร์ Repeat End Address (RE)

ง.1.2 การจัดหน่วยความจำ (Memory Organization)

ขนาดของหน่วยความจำที่ TMS320C31 RAM บล็อก 0 และ 1 แต่ละบล็อกจะมีขนาด 1K 32 บิต word ในส่วนของโปรแกรม, ข้อมูล, พอร์ท, อินพุตก็อยู่ใน 16 Mbyte นี้

ง.1.2.1 RAM,ROM และ Cache

หน่วยความจำที่อยู่ภายใน TMS320C31 RAM บล็อก 0 และ 1 แต่ละบล็อกจะมีขนาด 1K 32 บิต ROM มีขนาด 4K 32 บิต ระบบบัสถูกแบ่งออกเป็นโปรแกรมบัส (Program buses) บัสข้อมูล (Data Buses) บัสดีเอ็มเอ (DMA Buses) เพื่อให้สามารถทำงานแบบขนานได้ เช่น CPU สามารถเข้าถึงข้อมูลที่อยู่ใน RAM Block และสามารถ Fetch โปรแกรมจากหน่วยความจำภายนอกพร้อมกับการนำข้อมูลจาก RAM บล๊อกอื่นเข้ามา (DMA) ซึ่งทำงานภายใน 1 Cycle หน่วยความจำแคชขนาด 64 × 32 บิต ใช้สำหรับเก็บคำสั่งที่มีการใช้งานบ่อยๆ

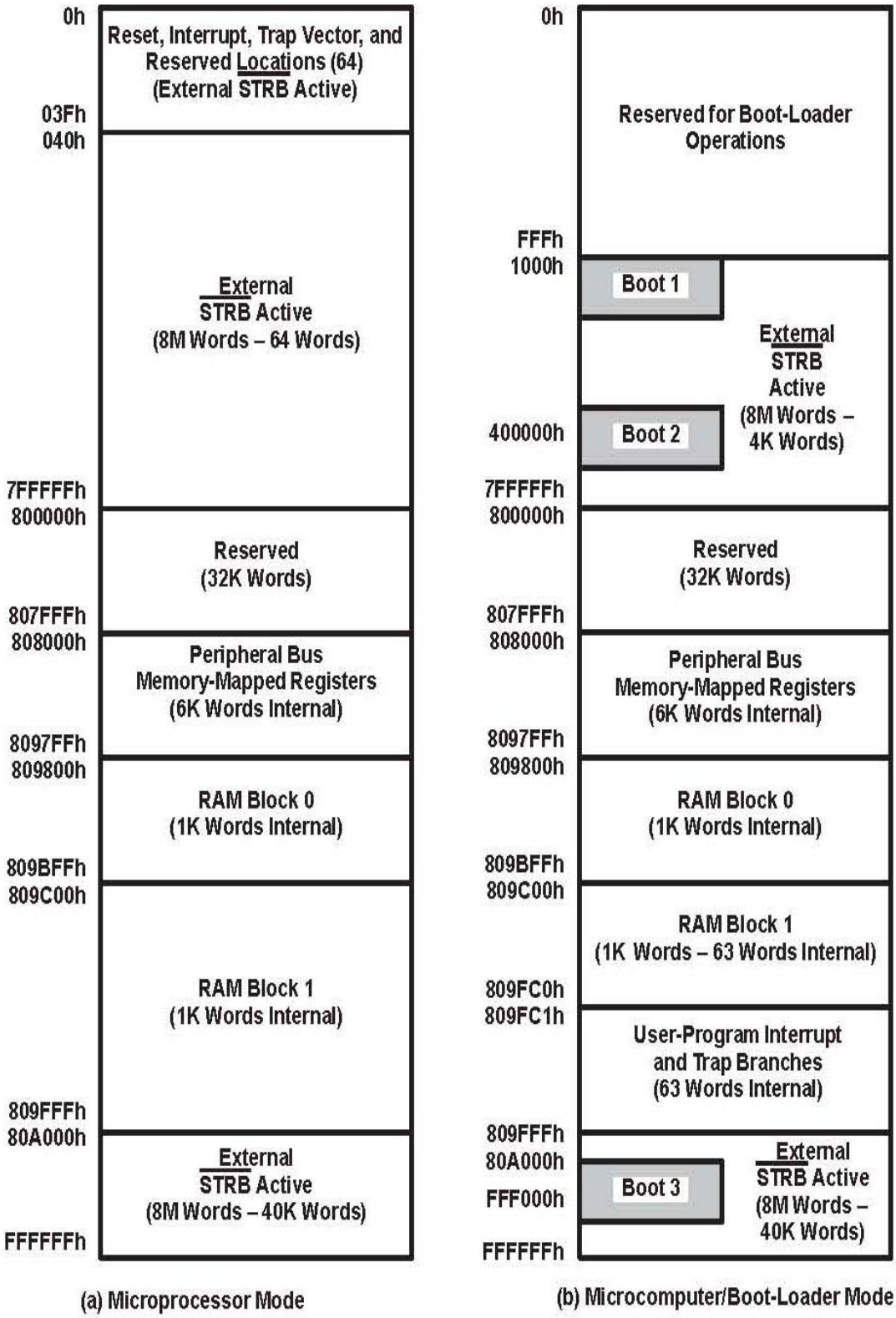


ภาพที่ ง-3 การจัดหน่วยความจำของ TMS320C31

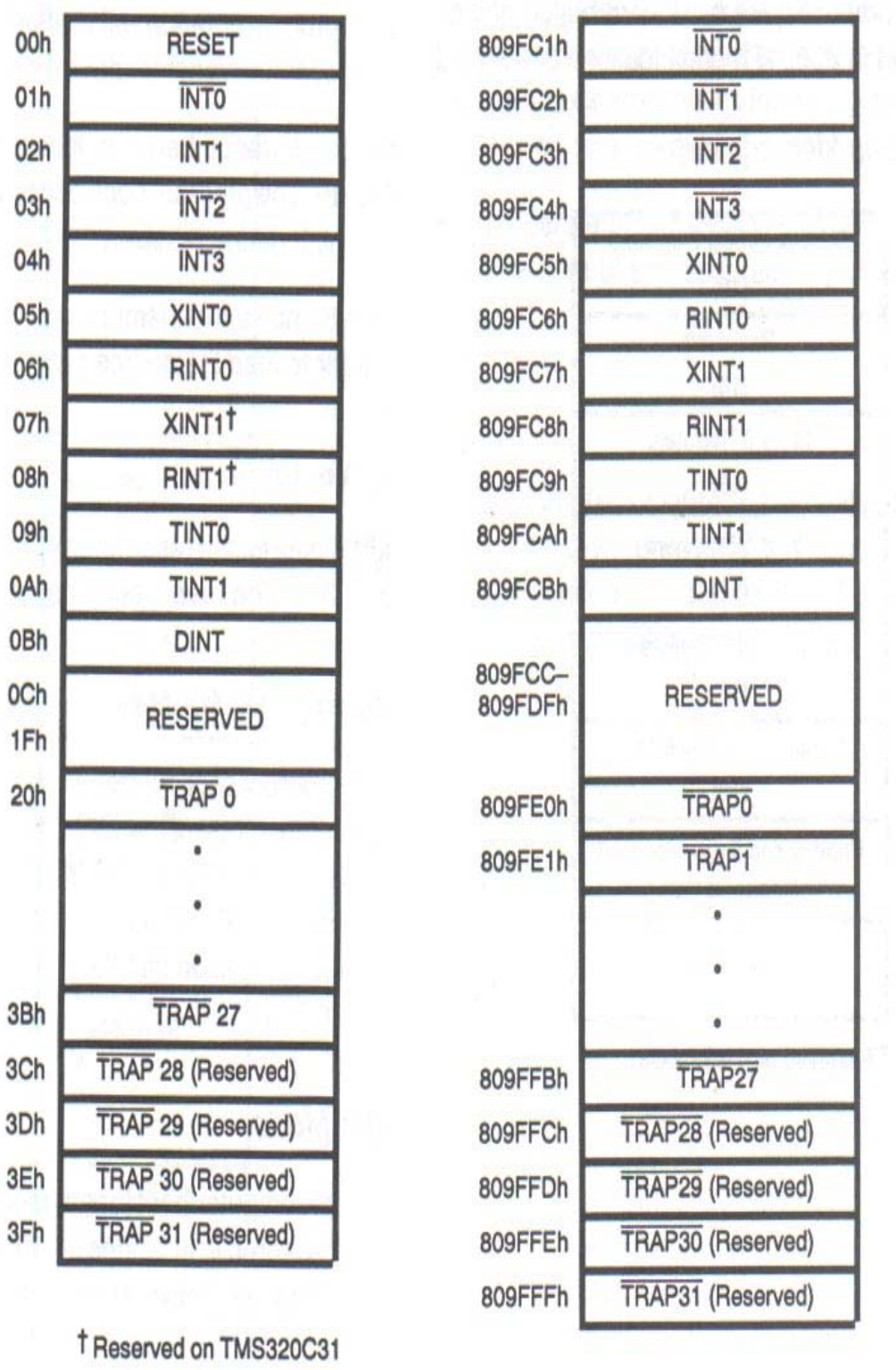
ง.1.2.2 การจัดแบ่งหน่วยความจำ (Memory Maps)

การจัดแบ่งหน่วยความจำจะขึ้นอยู่กับว่าโปรเซสเซอร์ทำงานอยู่ในโหมดใด ไมโครโปรเซสเซอร์โหมด (MC / $\overline{MP} = 0$) หรือไมโครคอมพิวเตอร์โหมด (MC / $\overline{MP} = 1$) ทั้งสองโหมดจะมีส่วนที่เหมือนกันคือที่ตำแหน่งแอดเดรส 80000h ถึง 801FFh ต้องถูกต่อใช้งานโดย Expansion bus เมื่อมีการอ้างอิงหน่วยความจำในส่วนนี้ ขาสัญญาณ $\overline{MSTRB}$ จะเป็น “0”(MSTRB active) ที่ตำแหน่งแอดเดรส 80200h ถึง 803FFFh ถูกสงวนไว้(ห้ามใช้) ตำแหน่งแอดเดรส 80400h ถึง 803FFFh ถูกสงวนไว้ตำแหน่งแอดเดรส 804000h ถึง 805FFFh จะต้องถูกต่อโดย Expansion bus เมื่อมีการอ้างอิงหน่วยความจำส่วนนี้ของสัญญาณ $\overline{IOSTRB}$ จะเป็น “0” ตำแหน่งแอดเดรส 80600h ถึง 807FFFh ถูกสงวนไว้

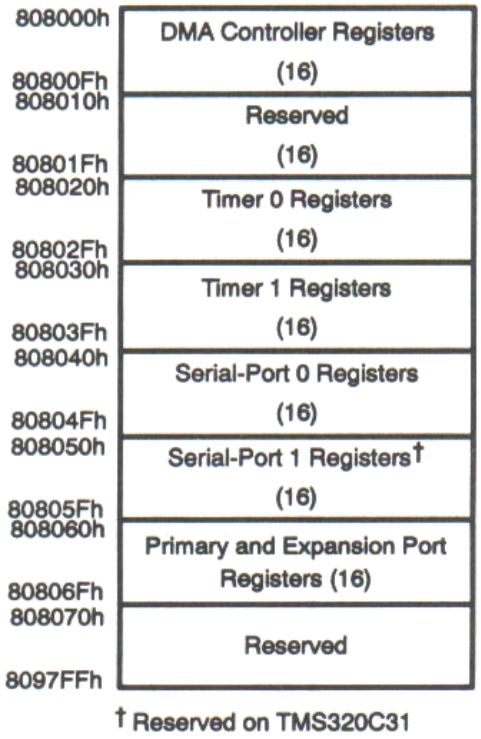
ในโหมดไมโครคอมพิวเตอร์จะมีหน่วยความจำแบบ ROM ได้ถูกกำหนดไว้ที่ 000000h ถึง 000FFFh ซึ่งตำแหน่ง 00 – 08h (192 ตำแหน่ง) จะใช้เก็บอินเตอร์รัพท์ เวคเตอร์(Interrupt vector) แทร็ปเวคเตอร์ (Trap vector) และพื้นที่สงวนไว้สำหรับระบบ ในการเข้าถึงหน่วยความจำตำแหน่ง 00100h – 7FFFFh จะต้องมีหน่วยความจำภายนอกต่อร่วมอยู่ด้วย ในโหมดไมโครโปรเซสเซฮร์จะไม่มีหน่วยความจำแบบ ROM ดังนั้นจึงต้องมีหน่วยความจำภายนอก (External RAM) ต่อร่วมด้วย



ภาพที่ ง-4 การจัดแบ่งหน่วยความจำของ TMS320C31



ภาพที่ ง-5 การจัดหน่วยความจำในโหมดไมโครโปรเซสเซอร์ในช่วง 00h – 3Fh และการจัดหน่วยความจำในโหมดไมโครคอมพิวเตอร์ในช่วง 809FC1h – 809FFFh



ภาพที่ ง-6 การจัดหน่วยความจำของอุปกรณ์สนับสนุน

ง.1.3 ลักษณะบัสภายใน (Internal Bus Operation)

สิ่งสำคัญมากที่ทำให้ TMS320C31 มีประสิทธิภาพสูงเป็นเพราะบัสภายในมีลักษณะความสัมพันธ์ที่ขนานกัน บัสเหล่านี้ประกอบด้วยบัสโปรแกรม (PADDR และ PDATA) บัสข้อมูล (DADDR1, DADDR2 และ DDATA) และบัสดีเอ็มเอ (DMAADDR และ DMADATA) สำหรับการเข้าถึงข้อมูลและดีเอ็มเอสามารถทำได้ลักษณะโปรแกรมพาราเลล (parallel program)

สำหรับการติดต่อกับ PC นั้นควบคุมโดยโปรแกรมบัสแอดเดรส 24 บิต (PADDR) รีจิสเตอร์คำสั่ง (IR) เป็นตัวควบคุมโปรแกรมบัสข้อมูล 32 บิต (PDATA) บัสเหล่านี้สามารถทำให้เกิดคำสั่งเวิร์ดเดี่ยวในทุกๆรอบการทำงาน

บัสแอดเดรสข้อมูล 24 บิต (DADDR1 และ DADDR2) และบัสข้อมูล 32 บิต สามารถเข้าถึงหน่วยความจำข้อมูล 2 ค่า ในทุกๆรอบการทำงานบัส DDATA นำข้อมูลไปที่ CPU บัส CPU1 และ CPU2 สามารถนำหน่วยความจำของข้อมูล 2 ค่าไปคำนวณที่มัลติไพลเออร์ (ALU) และไปที่รีจิสเตอร์ในทุกๆรอบการทำงาน

ชุดควบคุม ดีเอ็มเอ จะมีบัสแอดเดรส 24 บิต (DMADATA) และ บัสข้อมูล 32 บิต (DMADATA) บัสเหล่านี้ยินยอมให้ ดีเอ็นเอ เข้าถึงหน่วยความจำในลักษณะพาราเลลพร้อมกับการเข้าถึงหน่วยความจำที่เกิดขึ้นจากบัสข้อมูลและบัสโปรแกรม

ง.1.4 ลักษณะบัสภายนอก (External Bus Operation)

TMS320C31 มีการติดต่อภายนอกโดยใช้บัสไพรมาลี่ (Primary Bus) โดยบัสไพรมาลี่จะประกอบด้วย บัสแอดเดรส 24 บิต บัสข้อมูล 32 บิต และสัญญาณควบคุม 1 ชุด ($R/\overline{W}, \overline{STRB}, \overline{RDY}$)

ง.1.4.1 อินเทอร์รัพพ์ภายนอก

TMS320C31 มีการสนับสนุนอินเทอร์รัพพ์จากภายนอก 4 ชุด ($\overline{INT3} - \overline{INT0}$) และสามารถใช้สัญญาณ $\overline{RESET}$ จากภายนอกได้ สิ่งเหล่านี้สามารถใช้อินเทอร์รัพพ์อย่างใดอย่างหนึ่งได้ระหว่าง ดีเอ็มเอหรือซีพียู เมื่อซีพียูตอบสนองการอินเทอร์รัพพ์ ขา $\overline{IACK}$ จะส่งสัญญาณการตอบสนองการอินเทอร์รัพพ์ของซีพียูออกสู่ภายนอก

ง.1.4.2 สัญญาณ Interlocked – Instruction

I/O ภายนอก 2 ตัว XF0 และ XF1 สามารถเป็นอินพุทหรือเอาท์พุทภายใต้การควบคุมด้วยซอฟต์แวร์ ขาเหล่านี้ใช้โดยการทำงานแบบ interlocked ของ TMS320C31 กลุ่มคำสั่งการทำงานแบบ interlocked นั้นสนับสนุนการติดต่อแบบมัลติโปรเซสเซอร์

ง.1.5 อุปกรณ์สนับสนุน

อุปกรณ์สนับสนุนของ TMS320C31 ใช้รีจิสเตอร์พิเศษทำหน้าที่ควบคุมการทำงานของบัสอุปกรณ์สนับสนุน บัสอุปกรณ์สนับสนุนนั้นแบ่งออกเป็น บัสข้อมูล 32 บิต และบัสแอดเดรส 24 บิต บัสของอุปกรณ์สนับสนุนนั้นสามารถติดต่อกับอุปกรณ์สนับสนุนได้โดยตรง

อุปกรณ์สนับสนุนของ TMS320C31 ประกอบด้วย 2 ไทเมอร์ และ 1 ซีเรียลพอร์ท

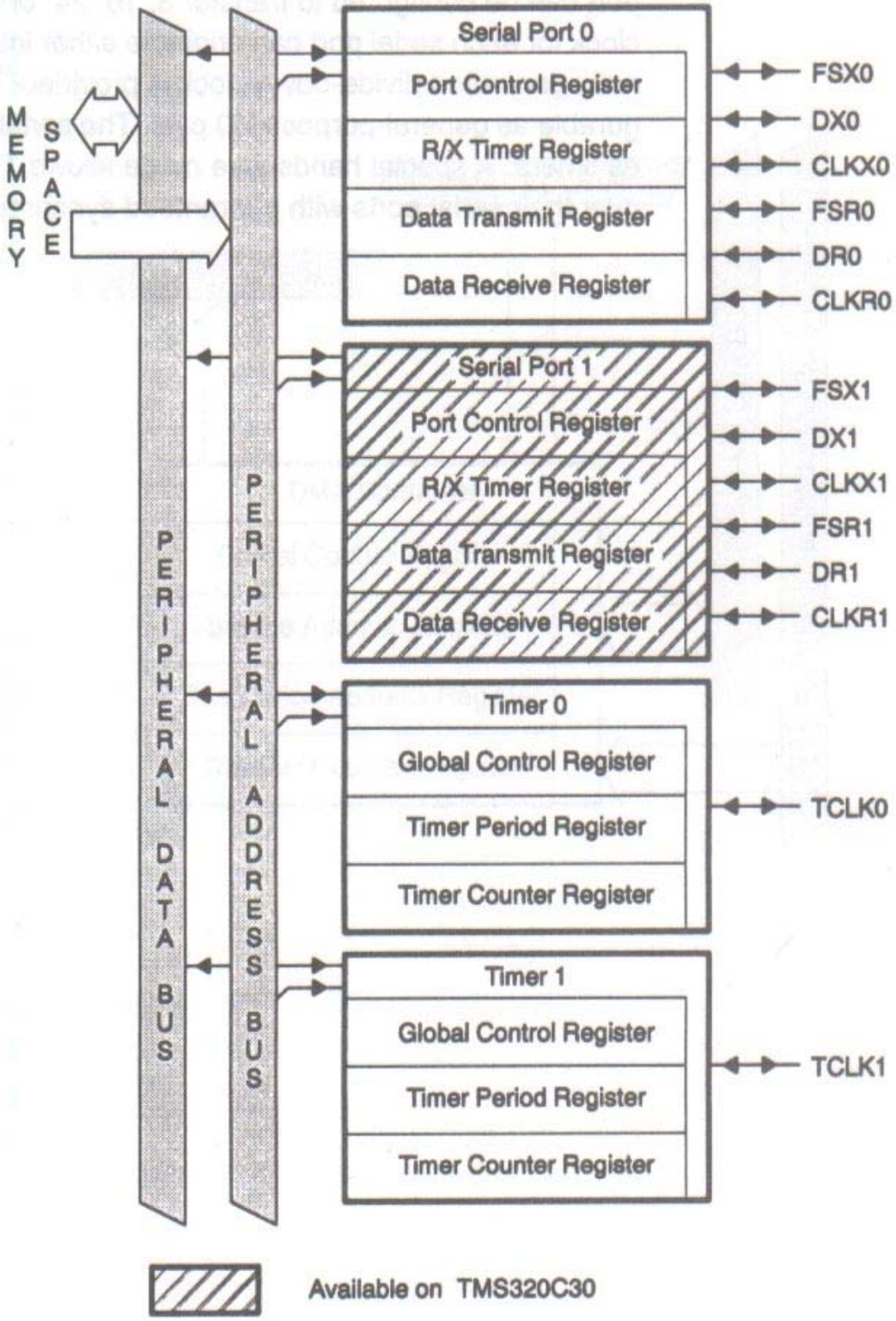
- ไทเมอร์

ลักษณะโดยทั่วไปของไทเมอร์ทั้ง 2 ตัวนั้นเป็นแบบ 32 บิต timer/event นับด้วยสัญญาณโหมด 2 สัญญาณคือ คล็อกกิ้งภายในหรือภายนอกในแต่ละขาของ I/O สามารถใช้เป็นอินพุทคล็อกที่ไทเมอร์หรือเป็นสัญญาณเอาท์พุทของไทเมอร์ ขานี้จะมีรูปร่างภายนอกเป็นขาของ I/O ได้

- ซีเรียลพอร์ท

ซีเรียลพอร์ทมี 1 ตัวทำงานเป็นอิสระไม่ขึ้นการควบคุมกับรีจิสเตอร์ใดๆนอกจากซีพียู ซีเรียลพอร์ทสามารถเปลี่ยนแปลงเป็น 8, 16, 24 หรือ 32 บิต คล็อกของซีเรียลพอร์ทนั้นสามารถใช้ได้ทั้งภายในหรือภายนอกก็ได้ สำหรับภายในนั้นมีการแบ่งส่วนคล็อก

ซีเรียลพอร์ทสามารถเป็นโครงสร้างของไทเมอร์โหมดแฮนด์เช็คพิเศษที่ยินยอมให้ TMS320C31 ติดต่อกับซีเรียลพอร์ทอื่นๆที่อยู่ในช่วงเวลาเดียวกัน

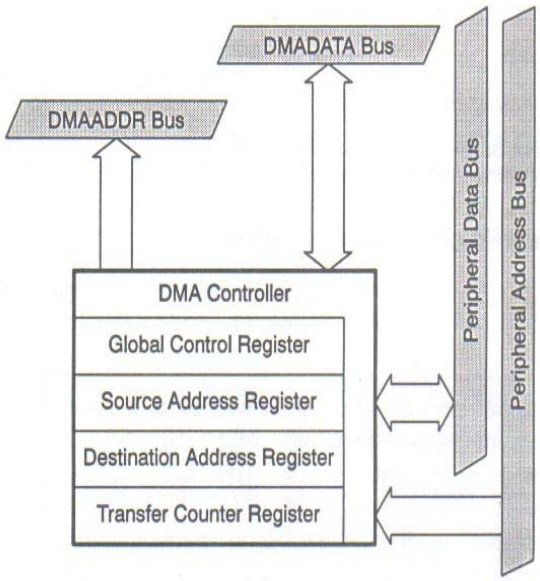


ภาพที่ ง-7 Peripheral Modules ของ TMS320C31

ง.1.6 ชุดควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA)

ชุดควบคุม ดีเอ็มเอ สามารถที่จะอ่านและเขียนได้ทุกๆที่ในหน่วยความจำรวมทั้งการติดต่อกับภายนอกด้วยการทำงาน(หรือการควบคุม) ของซีพียู เพราะฉะนั้น TMS320C31 สามารถติดต่อกับหน่วยความจำภายนอกและอุปกรณ์เสริมภายนอกได้โดยตรง

ชุดควบคุม ดีเอ็มเอ ประกอบด้วยตัวกำเนิดแอดเดรสของตัวมันเอง แหล่งกำเนิด และรีจิสเตอร์เป้าหมายและตัวนับโอน การใช้บัสแอดเดรสและบัสข้อมูลของ ดีเอ็มเอ เองจะทำให้การทำงานระหว่าง ซีพียู และชุดควบคุมดีเอ็มเอมีผลกระทบน้อยที่สุด การทำงานของดีเอ็มเอ จะใช้บล็อกหรือการโอนเวิร์ดเดียวส่งไปยังหน่วยความจำ

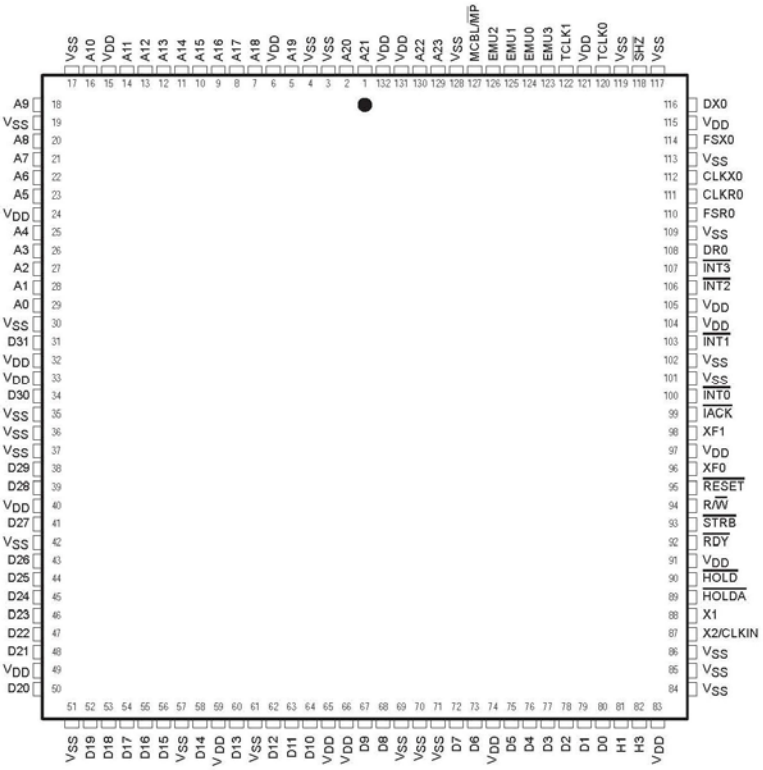


ภาพที่ ง-8 การควบคุม DMA ของ TMS320C31

ง.2 ลักษณะของ TMS320C31

- ประมวลผลภายใน 1 cycle (60ns)
- ประมวลผลเลขจุดทศนิยม 33.3 ล้านส่งคำต่อ 1 วินาที (33.3 MFLOPS)
- ประมวลผลคำสั่งทั่วไป 16.7 ล้านคำต่อ 1 วินาที (16.7 MIPS)
- หน่วยความจำแบบ RAM ภายในขนาด 1K × 32 บิต 2 ชุด
- หน่วยความจำแคช (Cache) ภายในขนาด 64 × 32 บิต
- คำสั่งและข้อมูลขนาด 32 บิต, แอดเดรสขนาด 24 บิต
- เลขจุดทศนิยมขนาด 40 บิต, เลขจำนวนเต็มขนาด 32 บิต
- แบเรอร์ชิฟเตอร์ (Barrel Shifter)

- รีจิสเตอร์ขนาด 40 บิต 8 ตัว (Extended – Precision Register) สำหรับประมวลผลเลขจุดทศนิยม และเลขจำนวนเต็ม
- มีตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access) สำหรับการทำงานแบบขนานระหว่างอุปกรณ์ภายนอก และการทำงานของซีพียู
- ระบบตัวเลขแบบจำนวนเต็ม (Integer), จุดทศนิยม (Floating Point) และการทำงานแบบตรรกะ (Logic)
- คำสั่งแบบมีตัวที่ถูกกระทำ 2 และ 3 ตัว (Two and Three Operand)
- คำสั่งที่ทำงานพร้อมกันระหว่าง ALU และ Multiplier ภายใน 1 clock cycle
- ความสามารถในการทำงานแบบซ้ำเป็นกลุ่ม (Block Repeat)
- เงื่อนไขสำหรับการเรียก (CALL) และการกลับ (RETURN)
- คำสั่งที่ประสานงานกันเพื่อสนับสนุนการทำงานแบบหลายงาน (Multiprocessing)
- Flexible boot program loader
- พอร์ทอนุกรมหนึ่งพอร์ท สนับสนุนการส่งแบบ 8, 16, 24, 32 บิต
- ตัวตั้งเวลา (Timer) สองชุด ขนาด 32 บิต
- การขัดจังหวะสี่ช่องจากภายนอก (4 Interrupt)
- 132 pin PQFP package, 8 μm CMOS



ภาพที่ ง-9 ลักษณะของชิป TMS320C31

TERMINAL NAME	NO.	TERMINAL NAME	NO.	TERMINAL NAME	NO.	TERMINAL NAME	NO.	TERMINAL NAME	NO.
A0	29	D4	76	EMU0	124	V_{DD}	40	V_{SS}	84
A1	28	D5	75	EMU1	125	V_{DD}	49	V_{SS}	85
A2	27	D6	73	EMU2	126	V_{DD}	59	V_{SS}	86
A3	26	D7	72	EMU3	123	V_{DD}	65	V_{SS}	101
A4	25	D8	68	FSR0	110	V_{DD}	66	V_{SS}	102
A5	23	D9	67	FSX0	114	V_{DD}	74	V_{SS}	109
A6	22	D10	64	H1	81	V_{DD}	83	V_{SS}	113
A7	21	D11	63	H3	82	V_{DD}	91	V_{SS}	117
A8	20	D12	62	$\overline{\text{HOLD}}$	90	V_{DD}	97	V_{SS}	119
A9	18	D13	60	$\overline{\text{HOLDA}}$	89	V_{DD}	104	V_{SS}	128
A10	16	D14	58	$\overline{\text{IACK}}$	99	V_{DD}	105	X1	88
A11	14	D15	56	$\overline{\text{INT0}}$	100	V_{DD}	115	X2/CLKIN	87
A12	13	D16	55	$\overline{\text{INT1}}$	103	V_{DD}	121	XF0	96
A13	12	D17	54	$\overline{\text{INT2}}$	106	V_{DD}	131	XF1	98
A14	11	D18	53	$\overline{\text{INT3}}$	107	V_{DD}	132		
A15	10	D19	52	MCBL/$\overline{\text{MP}}$	127	V_{SS}	3		
A16	9	D20	50	$\overline{\text{RDY}}$	92	V_{SS}	4		
A17	8	D21	48	$\overline{\text{RESET}}$	95	V_{SS}	17		
A18	7	D22	47	R/$\overline{\text{W}}$	94	V_{SS}	19		
A19	5	D23	46	$\overline{\text{SHZ}}$	118	V_{SS}	30		
A20	2	D24	45	$\overline{\text{STRB}}$	93	V_{SS}	35		
A21	1	D25	44	TCLK0	120	V_{SS}	36		
A22	130	D26	43	TCLK1	122	V_{SS}	37		
A23	129	D27	41			V_{SS}	42		
CLKR0	111	D28	39			V_{SS}	51		
CLKX0	112	D29	38	V_{DD}	6	V_{SS}	57		
D0	80	D30	34	V_{DD}	15	V_{SS}	61		
D1	79	D31	31	V_{DD}	24	V_{SS}	69		
D2	78	DR0	108	V_{DD}	32	V_{SS}	70		
D3	77	DX0	116	V_{DD}	33	V_{SS}	71		

† V_{DD} and V_{SS} pins are on a common plane internal to the device.

ภาพที่ ง-10 หน้าที่ของขาสัญญาณของ TMS320C31 เรียงตามลำดับตัวอักษร

TERMINAL NO.	TERMINAL NAME	TERMINAL NO.	TERMINAL NAME	TERMINAL NO.	TERMINAL NAME	TERMINAL NO.	TERMINAL NAME	TERMINAL NO.	TERMINAL NAME
1	A21	31	D31	61	V_{SS}	91	V_{DD}	121	V_{DD}
2	A20	32	V_{DD}	62	D12	92	$\overline{RDY}$	122	TCLK1
3	V_{SS}	33	V_{DD}	63	D11	93	$\overline{STRB}$	123	EMU3
4	V_{SS}	34	D30	64	D10	94	R/$\overline{W}$	124	EMU0
5	A19	35	V_{SS}	65	V_{DD}	95	$\overline{RESET}$	125	EMU1
6	V_{DD}	36	V_{SS}	66	V_{DD}	96	XF0	126	EMU2
7	A18	37	V_{SS}	67	D9	97	V_{DD}	127	MCBL/$\overline{MP}$
8	A17	38	D29	68	D8	98	XF1	128	V_{SS}
9	A16	39	D28	69	V_{SS}	99	$\overline{IACK}$	129	A23
10	A15	40	V_{DD}	70	V_{SS}	100	$\overline{INT0}$	130	A22
11	A14	41	D27	71	V_{SS}	101	V_{SS}	131	V_{DD}
12	A13	42	V_{SS}	72	D7	102	V_{SS}	132	V_{DD}
13	A12	43	D26	73	D6	103	$\overline{INT1}$		
14	A11	44	D25	74	V_{DD}	104	V_{DD}		
15	V_{DD}	45	D24	75	D5	105	V_{DD}		
16	A10	46	D23	76	D4	106	$\overline{INT2}$		
17	V_{SS}	47	D22	77	D3	107	$\overline{INT3}$		
18	A9	48	D21	78	D2	108	DR0		
19	V_{SS}	49	V_{DD}	79	D1	109	V_{SS}		
20	A8	50	D20	80	D0	110	FSR0		
21	A7	51	V_{SS}	81	H1	111	CLKR0		
22	A6	52	D19	82	H3	112	CLKX0		
23	A5	53	D18	83	V_{DD}	113	V_{SS}		
24	V_{DD}	54	D17	84	V_{SS}	114	FSX0		
25	A4	55	D16	85	V_{SS}	115	V_{DD}		
26	A3	56	D15	86	V_{SS}	116	DX0		
27	A2	57	V_{SS}	87	X2/CLKIN	117	V_{SS}		
28	A1	58	D14	88	X1	118	$\overline{SHZ}$		
29	A0	59	V_{DD}	89	$\overline{HOLDA}$	119	V_{SS}		
30	V_{SS}	60	D13	90	$\overline{HOLD}$	120	TCLK0		

† V_{DD} and V_{SS} pins are on a common plane internal to the device.

ภาพที่ ง-11 หน้าที่ของขาสัญญาณของ TMS320C31 เรียงตามลำดับขาสัญญาณ

ง.3 ลักษณะสัญญาณของ TMS320C31

ในตารางที่ ง-2 จะอธิบายสัญญาณของ TMS320C31 ที่ใช้ในโหมดไมโครโปรเซสเซอร์

ตารางที่ ง-2 ตำแหน่งขาและหน้าที่การทำงานของแต่ละขา

Primary Bus Interface (61 Pins)		
ชื่อ	จำนวนขา	คำอธิบาย
$D_0 - D_{31}$	32	พอร์ตข้อมูล 32 บิต
$A_0 - A_{23}$	24	พอร์ตแอดเดรส 24 บิต
$\overline{HOLD}$	1	เมื่อ $\overline{HOLD}$ เป็นลอจิกต่ำจะทำให้สัญญาณ $A_0 - A_{23}$, $D_0 - D_{31}$, $\overline{STRB}$ และ $R/\overline{W}$ จะมีสภาพเป็น Hight – Impedance และการติดต่อทั้งหมดที่อยู่บน บัสอินเตอร์เฟสจะถูก Delay ไว้จนกระทั้งมีสัญญาณ $\overline{HOLD}$ อีกครั้งเป็นลอจิกสูง หรือจนกระทั่งบิต $\overline{NOHOLD}$ ของคอนโทรลบิตรีจิสเตอร์ Primary Bus ถูกเซต
$\overline{HOLDA}$	1	สัญญาณแสดงการเริ่มต้นหรือสิ้นสุดของสัญญาณ $\overline{HOLD}$ คือสัญญาณ $\overline{HOLDA}$ การตอบสนองการทำงานของลอจิกต่ำของสัญญาณ $\overline{HOLD}$ เมื่อ $\overline{HOLDA}$ เป็นลอจิกต่ำจะทำให้สัญญาณ $A_0 - A_{23}$, $D_0 - D_{31}$,STRB และ $R/\overline{W}$,มีสภาวะเป็น High – Impedance และการติดต่อทั้งหมดที่อยู่บนบัสจะถูก Delay ไว้จนกระทั่งสัญญาณ $\overline{HOLDA}$ ตอบสนองสภาวะสูงจากลอจิกของ $\overline{HOLD}$ หรือจนกระทั่งบิต $\overline{NOHOLD}$ ของคอนโทรลรีจิสเตอร์ Primary Bus ถูกเซต
$R/\overline{W}$	1	สัญญาณ Read/Write จะเป็นสภาวะสูงเมื่อมีการอ่านและจะเป็นสภาวะต่ำเมื่อมีการเขียนบนอินเตอร์เฟสแบบขนาน
$\overline{RDY}$	1	สัญญาณ Ready ขานี้จะทำงานก็ต่อเมื่ออุปกรณ์ภายนอกทำการติดต่อชิป
$\overline{STRB}$	1	สัญญาณ Strobe จากภายนอก

ตารางที่ ง-2 (ต่อ)

Control – Signal (10 pins)		
$\overline{INT3} - \overline{INT0}$	4	อินเตอร์เฟสภายนอก
$\overline{IACK}$	1	สัญญาณแสดงการเริ่มต้นหรือ สิ้นสุดของสัญญาณอินเตอร์รัพพ์ โดยปกติ $\overline{IACK}$ จะถูกเซตให้เป็น 1 เพราะฉะนั้น $\overline{IACK}$ จึงแสดงการเริ่มต้น หรือจุดสิ้นสุดของการใช้อินเตอร์รัพพ์รูทีน
$MCBL / \overline{MP}$	1	สัญญาณกำหนดการทำงานระหว่าง Microcomputer Boot Loader กับ Microprocessor
$\overline{RESET}$	1	สัญญาณ Reset เมื่อสัญญาณ Reset เป็นลอจิกต่ำจะยกเลิกการทำงานภายใต้เงื่อนไขที่กำหนดให้
$\overline{SH2}$	1	สัญญาณ Shut Down High Z เมื่อมีสภาวะต่ำจะยกเลิกการทำงานของ C31 และขาทั้งหมดจะมีสภาวะเป็น High – Impedance สัญญาณนี้ใช้ทดสอบ Broad Level นั้นคือจะไม่ปรากฏสภาวะอื่น <u>ข้อควรระวัง</u> เมื่อ $\overline{SH2}$ มีสภาวะต่ำจะทำให้ข้อมูลที่อยู่ภายในหน่วยความจำและรีจิสเตอร์หายไป การรีเซตด้วย $\overline{SH2}$ = 1 เป็นการคืนค่าในสภาวะของการทำงาน
XF0, XF1	2	สัญญาณ Flag ภายนอกโดยทั่วไปใช้เป็นขา I/O หรือสนับสนุนการทำงานของ Interlocked Processor Instructions
Serial Port 0 Signals(6 pins)		
CLK R0, CLK X0	2	สัญญาณนาฬิกา ที่ใช้สำหรับควบคุมการรับ – ส่งข้อมูลของ Serial Port 0 Receiver และ Serial Port 0 Transmitter
DR0	1	ใช้สำหรับรับข้อมูลจาก Serial Port 0 Receives
DX0	1	ใช้สำหรับส่งข้อมูลจาก Serial Port 0 Transmits
FSR0	1	FRS 0 จะเริ่มต้นนับเมื่อมีการรับข้อมูลบน DR 0
FSX0	1	FRS 0 จะเริ่มต้นนับเมื่อมีการส่งข้อมูลบน DX 0

ตารางที่ ง-2 (ต่อ)

Timer Signals (2 Pins)		
TCLK0	1	Timer 0 จะใช้ขา TCLK 0 ในการนับ Pulse จากภายนอกหรือกำเนิดสัญญาณ Pulse ออกสู่ภายนอก
TCLK1	1	Timer 1 จะใช้ขา TCLK 0 ในการนับ Pulse จากภายนอกหรือกำเนิดสัญญาณ Pulse ออกสู่ภายนอก
Supply and Oscillator Signals (49 pins)		
H1, H3	2	สัญญาณนาฬิกาภายนอก สัญญาณนาฬิกาจะมีคาบเท่ากับ 2 เท่า ของ CLKIN
V_{DD}	20	จ่ายไฟให้ได้ไม่เกิน +5 Vdc
V_{SS}	25	ใช้สำหรับต่อกราวด์
X_1	1	เป็นขา Output จากตัวคริสตัลภายในที่จะส่งออกภายนอก ถ้าไม่ใช้คริสตัลขานี้ไม่ต้องต่อ
X_2/CLKIN	1	เป็นขา Input สำหรับสัญญาณนาฬิกาหรือคริสตัลจากภายนอกที่จะส่งเข้าไปภายในชิป
Reserved (4 pins)		
EMU0 – EMU2	3	ขอสงวนไว้ : ใช้ตัวต้านทาน 20 KΩ ต่อ Pull – Up กับไฟ +5 Vdc
EMU3	1	ขอสงวนไว้

ง.4 บอร์ด TMS320C31 DSP STARTER KIT

บอร์ด TMS320C31 DSP Starter Kit เป็นบอร์ดพัฒนาโปรแกรมแบบ Stand – Alone สามารถประมวลผลแบบเวลาจริง (Real time) ได้ โดยสามารถที่จะสั่งรันโปรแกรมหรือเคลียร์โปรแกรมได้โดยผ่านพอร์ตของ PC และสามารถต่อบอร์ดอินเตอร์เฟสแบบต่างๆเข้ากับ บอร์ด DSK ได้

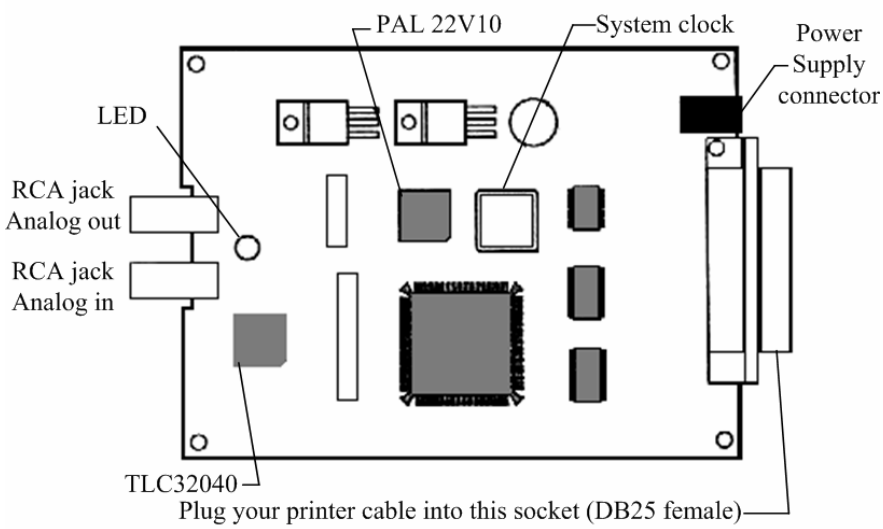
ง.4.1 ลักษณะของบอร์ด TMS320C31 DSP STARTER KIT

ฮาร์ดแวร์ของบอร์ด TMS320C31 DSK มีส่วนประกอบพื้นฐานที่ประกอบด้วย

- ชิป DSP TMS320C31
- ชิป A/D – D/A TLC32040
- พอร์ต I/O

- พอร์ตขนานของ พริ้นเตอร์

- LED แบบ 3 สี



ภาพที่ ง-12 ฮาร์ดแวร์ของบอร์ด TMS320C31 DSP Starter Kit

จากภาพที่ ง-12 แสดงอุปกรณ์บนบอร์ด DSK ประกอบด้วย

- Header ขนาด 32 pin จำนวน 4 แถว สัญญาณทุกเส้นของ C31 ที่จะส่งออกไปภายนอกบอร์ดจะต้องผ่าน Header ชุดนี้ซึ่งประกอบด้วย JP2 JP3 JP5 JP6

- Jumper Block header ขนาด 11 Pin (JP4) จะทำหน้าที่ควบคุมการส่งข้อมูลของพอร์ตอนุกรมของ TLC32040

- Host Interface Logic จะใช้ PLA 22V10Z และ 74ACT245 ควบคุมการสื่อสารระหว่างบอร์ด DSK กับ Host ของ PC

- Oscillator บนบอร์ด DSK จะใช้สัญญาณนาฬิกาขนาด 50 MHz เพื่อป้อนให้กับชิป TMS3320C31

- RCA Jack จะทำหน้าที่รับสัญญาณ Analog Input และส่งสัญญาณ Analog Output ของบอร์ด DSK โดยจะต่ออยู่กับขา I/O ของ AIC

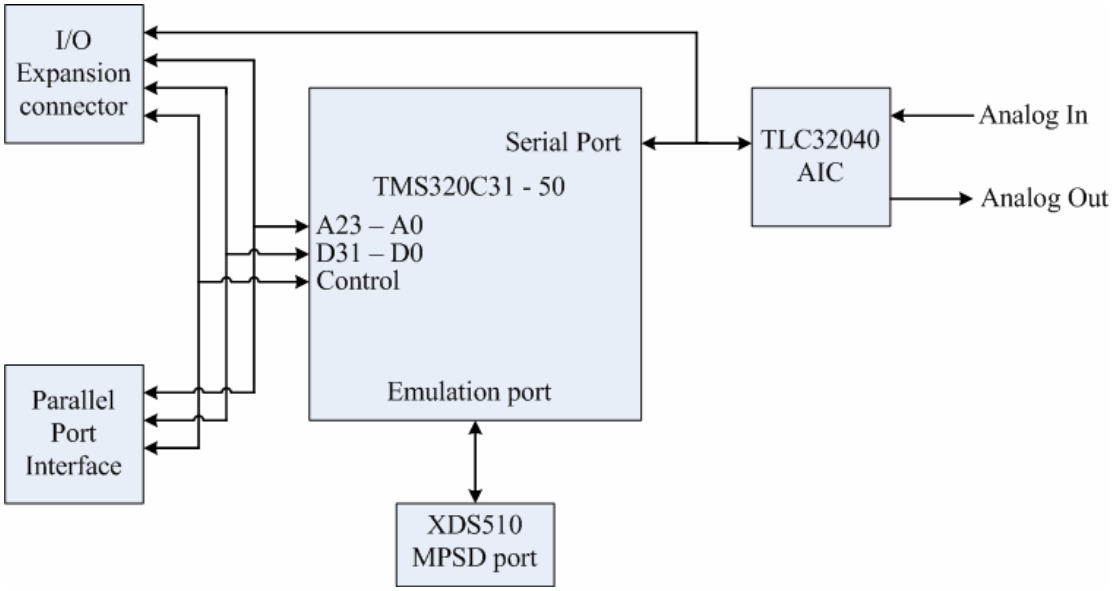
- TLC32040 AIC จะทำหน้าที่เป็น A/D และ D/A ของบอร์ด DSK

- TMS320C31 เป็นตัวประมวลผลขนาด 32 บิต แบบ Floating – Point

- Voltage regulators บอร์ด DSK สามารถใช้ไฟ 7 – 12 Vdc หรือ 6 – 9 Vac โดยไฟ DC และ AC จะต้องผ่าน IC Regulator เบอร์ LM7805 และ LM7905 ซึ่งจะได้ไฟ DC +5V และ -5V ตามลำดับ ไฟ DC ที่ได้จะใช้เลี้ยงอุปกรณ์ต่างๆภายในบอร์ดตลอดเวลา ที่บอร์ดยังทำงานอยู่

- XDS Emulator Port เป็น header ขนาด 12 Pin (JP1) ใช้สำหรับการ Upgrade โปรแกรม XDS Debugger ในอนาคต

จากที่กล่าวมาข้างต้นสามารถแสดงเป็นรูปบล็อกไดอะแกรมได้ดังภาพที่ ง-13



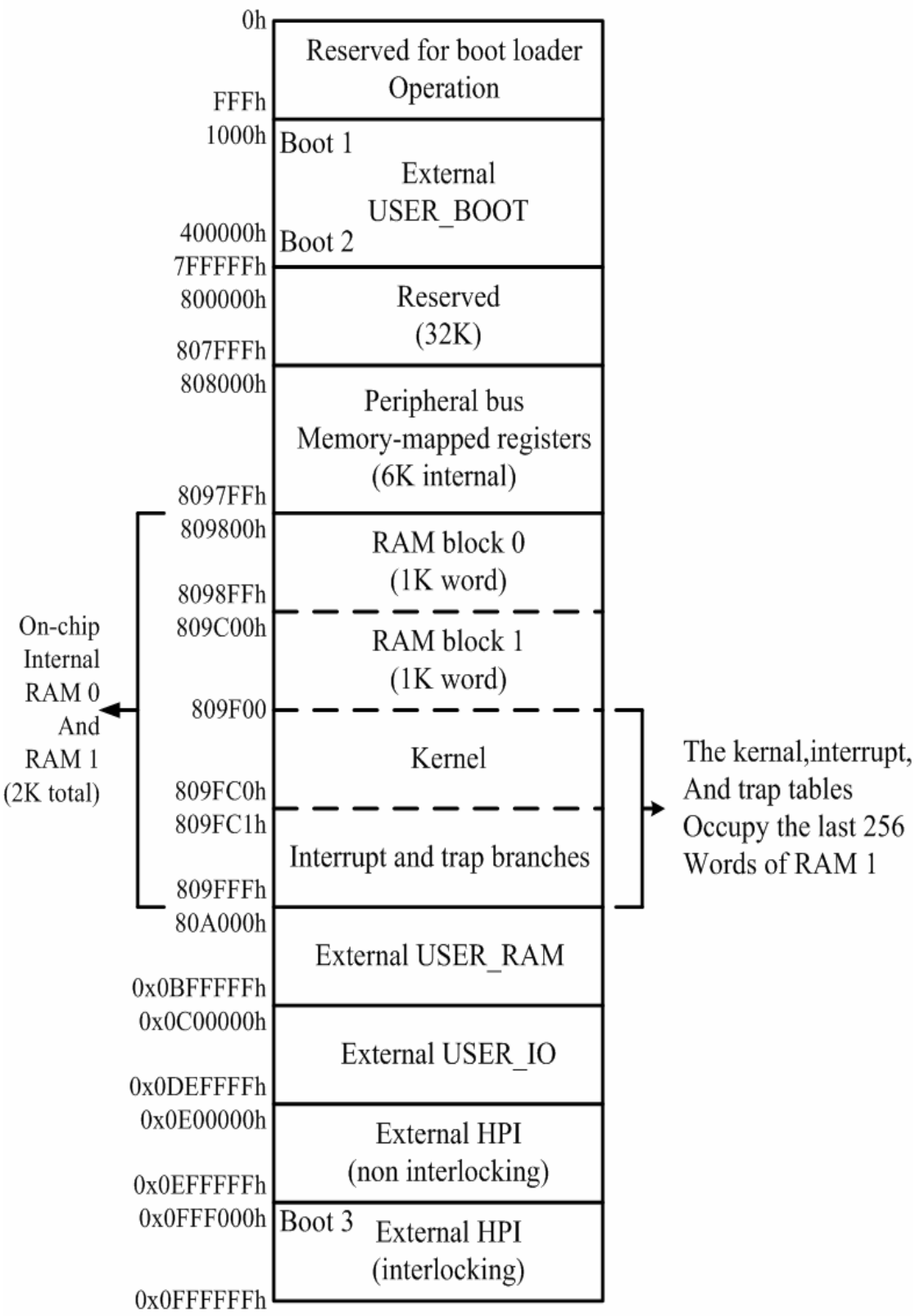
ภาพที่ ง-13 บล็อกไดอะแกรมของบอร์ด TMS320C31 DSP Starter Kit

ง.4.2 คุณลักษณะของบอร์ด TMS320C31 DSP Starter Kit

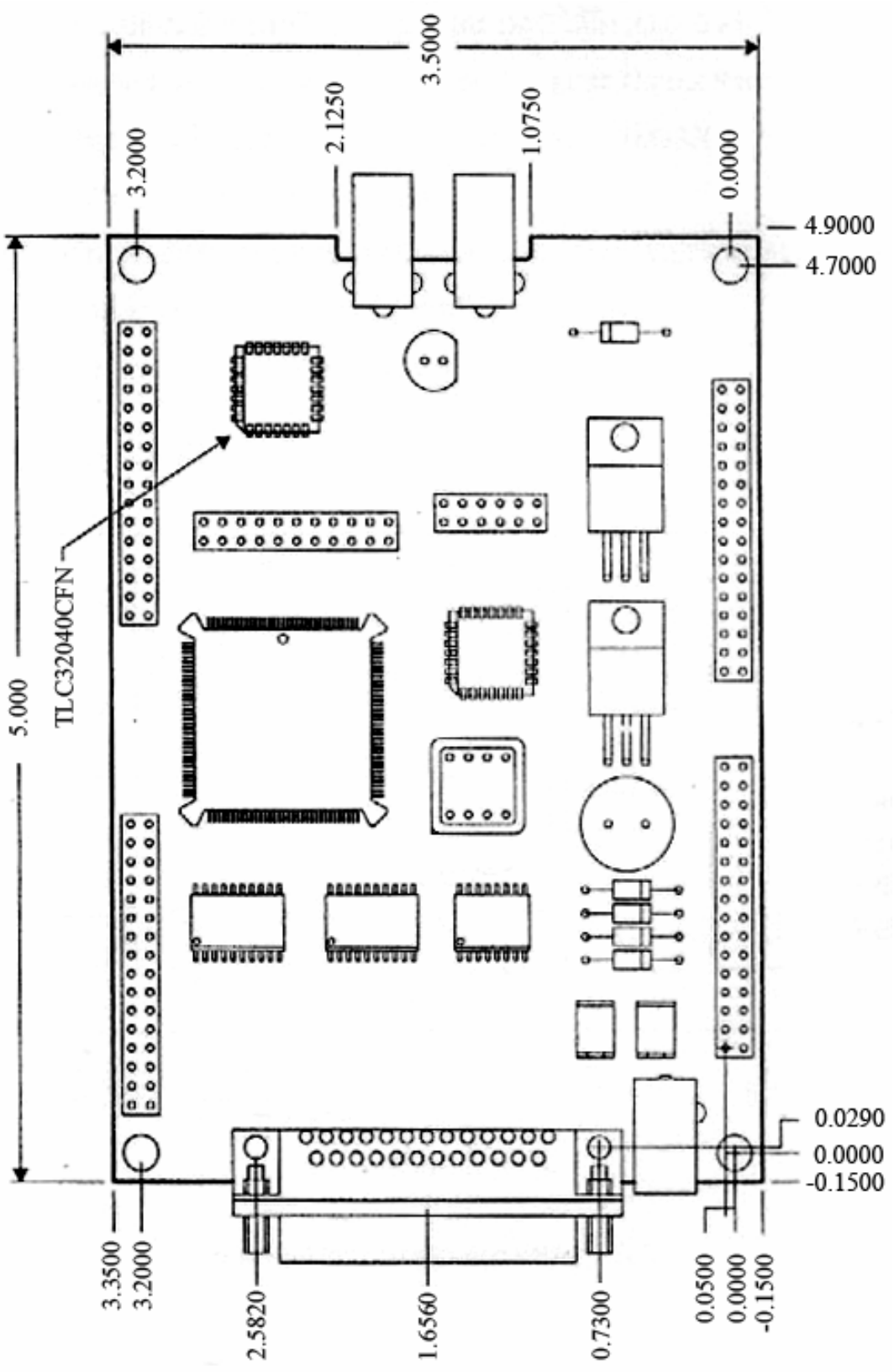
- ใช้ชิป TMS320C31 เป็นตัวประมวลผลขนาด 32 บิต แบบ Floating - point
- ใช้เวลาในการประมวลผลคำสั่ง 40 ns ต่อ 1 คำสั่งหรือ 50 MFLOP และ 25 MIPS
  MFLOP: Million Floating – point Intention Per Second
  MIPS : Million Intention Per Second
- บอร์ด DSK สามารถสื่อสารกับ PC ได้โดยใช้พอร์ตขนานของพริ้นเตอร์หรือใช้ host ของ PC
- ใช้ชิป TLC32040 AIC ขนาด 14 บิต อัตราการสุ่มสัญญาณ 20,000 ครั้งต่อวินาที
- ใช้ RCA Jack เป็น Jack มาตราฐานสำหรับการต่อสายสัญญาณ Analog Input และ Output ทำให้สามารถนำไปต่อกับไมโครโฟนและลำโพงได้

ง.4.3 การจัดหน่วยความจำบนบอร์ด TMS320C31 DSP Starter Kit

บอร์ด TMS320C31 DSK ได้จัดแบ่งหน่วยความจำให้อยู่ในโหมด Microcomputer /Boot Loader ดังแสดงในภาพที่ ง-14



ภาพที่ ง-14 การจัดแบ่งหน่วยความจำในโหมด Microcomputer/Boot Loader ของ บอร์ด TMS320C31 DSP Starter Kit



ภาพที่ ง-15 ลักษณะโดยรวมของบอร์ด TMS320C31 DSP Starter Kit

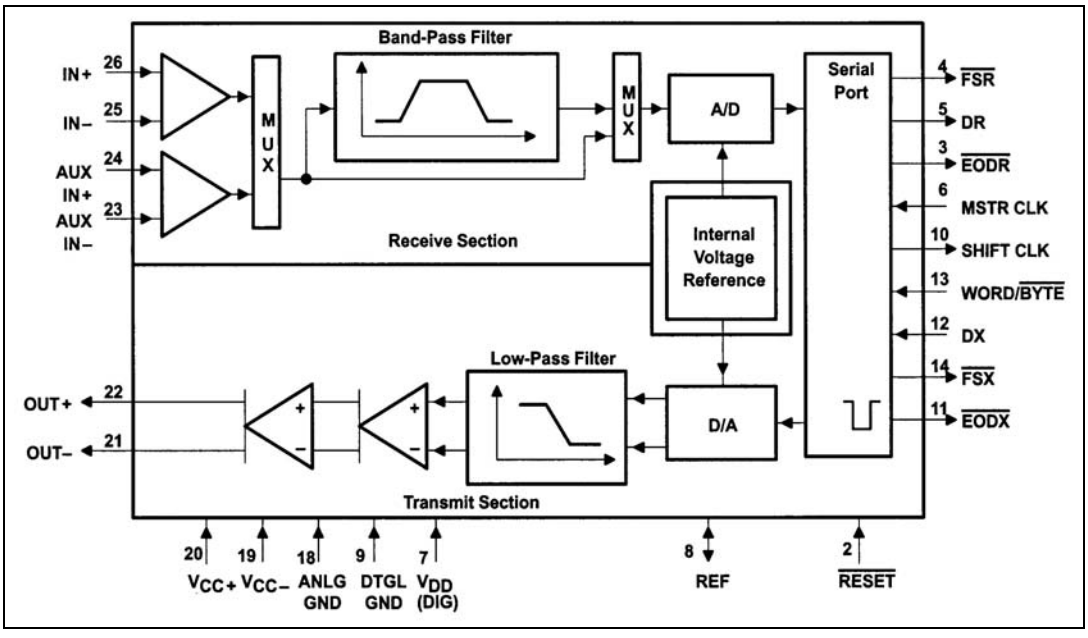
ง.5 วงจรอินเตอร์เฟสสัญญาณแอนะลอก TLC32040

TLC32040 เป็นชิป Analog Interface Circuit (AIC) ที่ใช้เชื่อมต่อกับ TMS320C31 ในการทำงานประมวลผลด้าน DSP

ง.5.1 ลักษณะของ TLC32040

- ใช้เทคโนโลยีการผลิต Advanced LinCMOS
- ความละเอียดของ ADC และ DAC เป็น 14 บิต
- สามารถเปลี่ยนอัตราแซมปิ้งของ ADC และ DAC ได้ถึง 20,000 ครั้ง /วินาที
- มี Switched Capacitor Anti Aliasing Input Filter และ Output Reconstruction Filter
- มีพอร์ทอนุกรมสำหรับติดต่อโดยตรงกับ TMS320C11, TMS320C17, TMS320C20, TMS320C25 Digital Signal Process
- สามารถปรับอัตราการแปลงของ ADC และ DAC ได้โดยใช้โปรแกรมควบคุม
- สัญญาณอินเตอร์เฟสของพอร์ทอนุกรมเข้าสู่ SN74C299
- 600 – Mil wide N Package (C_L to C_L)

ฟังก์ชันไดอะแกรมแสดงดังภาพที่ ง-16



ภาพที่ ง-16 ฟังก์ชันไดอะแกรมของ TLC32040

ง.5.2 ตำแหน่งขาและหน้าที่ การทำงานของแต่ละขาของ TLC32040

ตารางที่ ง-3 ตำแหน่งและหน้าที่การทำงานของแต่ละขา

ชื่อ	หมายเลข	คำอธิบาย
ANLG GND	17, 18	กราวด์อนาลอก (แยกกับกราวด์ดิจิตอล)
AUX IN+	24	Non-inverting auxiliary input
AUX IN-	23	Inverting auxiliary input
DGTL GND	9	กราวด์ดิจิตอล
DR	5	ใช้สำหรับส่ง output ADC จาก AIC (analog interface circuit) ไปยัง C31 ผ่านทางพอร์ทอนุกรมจะต้องซิงก์กับ shift CLK
DX	12	ใช้สำหรับ input DAC หรือคำสั่งการควบคุมจาก C31 ซึ่งการส่งผ่านทางพอร์ทอนุกรมจะต้องซิงก์กับ shift CLK
$\overline{EODR}$	3	สัญญาณหยุดรับข้อมูล (end of data receive) ในการติดต่อผ่านพอร์ทอนุกรมในโหมดเวิร์ด สัญญาณ $\overline{EODR}$ อยู่ในสถานะต่ำทันทีเมื่อ 16 บิตของ output A/D ได้ถูกส่งจาก AIC ไปยัง C31 ซึ่งสามารถใช้สัญญาณนี้ในการอินเตอร์รัพท์ไมโครโปรเซสเซอร์ให้ทราบว่าสิ้นสุดการติดต่อแล้ว หรือใช้สโครป และรีจิสเตอร์เลื่อนข้อมูลออก (enable external serial to parallel shift register) ก็ได้ แต่ถ้าเป็นโหมดไบต์ (byte mode) สัญญาณจะ $\overline{EODR}$ อยู่ในสถานะต่ำ หลังจากไบต์แรกได้ส่งไปยัง C31 แล้ว และยังคงรักษาสถานะต่ำจนกระทั่งไบต์ที่สองได้ส่งไป ทั้งนี้ก็เพื่อให้รู้ว่าไบต์แรกหรือไบต์ที่สองได้ส่งออกไป
$\overline{EODX}$	11	สัญญาณหยุดรับส่งข้อมูล (end of data transmit) คล้ายกับ $\overline{EODX}$ ซึ่งจะบอกให้ทราบว่าการติดต่อจาก C31 ไปยัง A/C นั้นเสร็จแล้ว ทั้งในโหมดเวิร์ดและโหมดไบต์ก็คล้ายกับ $\overline{EODR}$

ตารางที่ ง-3 (ต่อ)

ชื่อ	หมายเลข	คำอธิบาย
$\overline{FSR}$	4	สัญญาณซิงก์การรับ (Frame sync receive) ในการติดต่อแบบพอร์ทอนุกรม $\overline{FSR}$ จะมีสถานะต่ำตลอดการส่งจาก A/C ไปยัง C31 (โดยผ่านทางขา DR) ซึ่งบิตแรกที่จะส่งต้องพร้อมอยู่ที่ขา DR ก่อน $\overline{FSR}$ ก่อนจะ low
$\overline{FSX}$	14	สัญญาณซิงก์การส่ง (Frame sync transmit) เมื่อสัญญาณนี้อยู่ในสถานะต่ำ พอร์ทอนุกรม C31 จะส่งบิตไปยัง A/C โดยส่งมาที่ขา DX ในการติดต่ออนุกรมทุกโหมด $\overline{FSX}$ จะมีสถานะต่ำตลอดการส่ง
IN+	26	Non-inverting input
IN-	25	Inverting input
MSTR CLK	6	Master clock จะใช้การควบคุมทุกส่วนภายใน A/C ไม่ว่าจะเป็นสัญญาณนาฬิกาเลื่อน (shift clock) , สัญญาณนาฬิกาควบคุมฟิลเตอร์ (switched capacitor filter clock) , A/D และ D/A timing
OLT+	22	Non-inverting input
OLT-	21	Inverting input
REF	8	สำหรับ TLC 32040 และ TLC 32042 แรงดันอ้างอิงภายในจะถูกต่อเข้าที่ขานี้ แต่ถ้าเป็น TLC32040 , TLC32041 และ TLC32042 แรงดันอ้างอิงจากภายนอกจะถูกต่อเข้าที่ขานี้
$\overline{RESET}$	2	รีเซตจะทำการตั้งค่า TA, TA′, TB, RA′, RB และรีจิสเตอร์ควบคุมให้เป็นค่าเริ่มต้น รวมทั้งการติดต่อทางพอร์ทอนุกรมระหว่าง A/C และ DSP
SHIFT CLK	10	สัญญาณนาฬิกาเลื่อน จากการหารความถี่สัญญาณนาฬิกามาสเตอร์ด้วย 4 ซึ่งสัญญาณนี้จะใช้ในการติดต่อทางพอร์ทอนุกรม

ตารางที่ ง-3 (ต่อ)

ชื่อ	หมายเลข	คำอธิบาย
VDD	7	ไฟเลี้ยงวงจรดิจิตอล (digital supply voltage 5V ±5%)
Vcc+	20	ไฟเลี้ยงวงจรอนาลอกด้านบวก 12V±5%
Vcc-	19	ไฟเลี้ยงวงจรอนาลอกด้านลบ -12V±5%
WORD/ $\overline{BYTE}$	13	ขานี้จะทำงานร่วมกับรีจิสเตอร์ควบคุมเพื่อใช้ในการเลือกโหมดการติดต่ออนุกรม ซึ่งมี 4 แบบดังนี้ <u>การติดต่อแบบอะซิงโครนัสในโหมดไบต์</u> (WORD/BYTE = low) พอร์ทอนุกรมจะติดต่อโดยตรงกับ C31 และจะส่งค่าทีละ 8 บิต 2 ครั้ง ซึ่งมีขั้นตอนการทำงานดังนี้ 1. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะต่ำ 2. 8 บิตแรกจะถูกส่งออกไปหรือรับเข้ามา 3. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะต่ำ 4. $\overline{FSX}$ หรือ $\overline{FSR}$ high ประมาณ 4 สัญญาณนาฬิกาเลื่อนแล้วอยู่ในสถานะต่ำ 5. บิตต่อมา (ไบต์ที่ 2) ถูกส่งหรือรับเข้ามา 6. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะสูง 7. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะสูง <u>ในโหมดเวิร์ด</u> พอร์ทอนุกรมจะต่อตรงกับพอร์ทอนุกรมของ C31 และส่งค่าครั้งเดียว 16 บิต ซึ่งมีขั้นตอนการทำงานดังนี้ 1. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะต่ำ 2. 16 บิต ถูกส่งหรือรับเข้ามา 3. $\overline{FSX}$ หรือ $\overline{FSR}$ อยู่ในสถานะสูง 4. $\overline{EODX}$ หรือ $\overline{EODR}$ อยู่ในสถานะต่ำ การติดต่อแบบซิงโครนัส

ตารางที่ ง-3 (ต่อ)

ชื่อ	หมายเลข	คำอธิบาย
		ในกรณีนี้ bandpass filter และอัตราการเปลี่ยนแปลง A/D จะถูกกำหนดจาก TX counter A, TX counter B และ TA, TA′ และ TB แทนส่วนในการติดพต่อมีขั้นตอนเหมือนกับการติดต่อแบบอะซิงโครนัส

ง.5.3 การทำงานของ TLC32040

ง.5.3.1 Analog Input

แอนะลอกอินพุทมี 2 กลุ่มคือ IN+, IN- และ AUX IN+, AUX IN- ซึ่งสามารถเลือกใช้กลุ่มใดกลุ่มหนึ่งโดยจะใช้ในแบบดิฟเฟอเรนเซียลหรือซิงเกลเอนด์ (single ended) และค่าเกณฑ์สำหรับ IN+, IN- และ AUX IN+, AUX IN – สามารถใช้โปรแกรมตั้งค่าได้ (มี 3 ค่า คือ 1, 2, 4) การเลือกใช้กลุ่มอินพุทใดจะเลือกใช้ซอร์ฟแวร์ควบคุม

ง.5.3.2 A/D Bandpass Filter

A/D Bandpass Filter Clocking และ A/D Conversion Timing สามารถที่จะเลือกใช้หรือไม่ก็ได้ โดยใช้ซอฟแวร์ควบคุมความถี่ของสัญญาณนาฬิกาควบคุมฟิลเตอร์ (Filter Clock) จะเป็นตัวกำหนดทรานเฟอร์ฟังก์ชันของฟิลเตอร์ โดยจะคิดอัตราส่วนจากความถี่สัญญาณนาฬิกาควบคุมฟิลเตอร์ 288 kHz ที่ความถี่ต่ำที่เริ่มมีลักษณะเป็นความถี่สูงผ่านจะมีความถี่เป็น 300 kHz อัตราการเปลี่ยนแปลง D/A ก็จะหาจากความถี่ที่หาร 228 kHz ด้วย Rx Counter B

ง.5.3.3 Analog Output

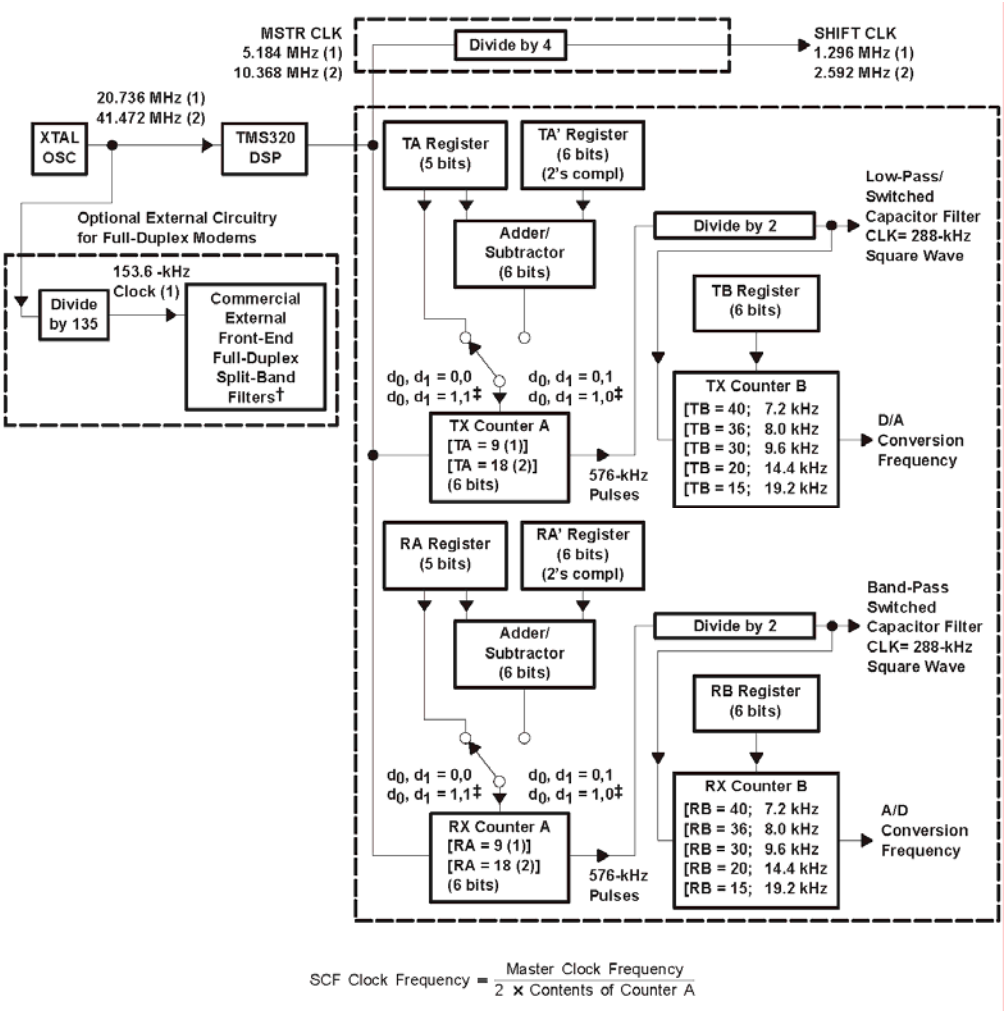
Analog Output จะมี Power Amplifier มี Output ทั้งแบบ Non – Inverting และแบบ Inverting เนื่องจากมี Amplifier ทำให้เอาท์พุทสามารถขับ Transformer Hybrid หรือโหลดอิมพีแดนซ์ต่ำได้ โดยใช้ทั้งแบบดิฟเฟอเรนเชียล หรือซิงเกิลเอนด์ (Single Ended)

ง.5.3.4 วงจรกรองความถี่ของ D/A

วงจรความถี่แบบ Bandpass ของ D/A, สัญญาณนาฬิกาควบคุม วงจรกรองความถี่ต่ำ และอัตราการแปลงสัญญาณดิจิตอลเป็นแอนะลอก (D/A Lowpass Filter, D/A Lowpass Filter Clocking, D/A Conversion Timing) เช่นเดียวกับ A/D Filter โดย Transfer Function ของ Filter ถูกกำหนดจากอัตราส่วนกับความถี่ 288 kHz หารด้วย TX Counter B โดยจะปลดออกไม่ได้

ง.5.3.5 การป้อนกลับ (Loop Back)

จะให้ผู้ใช้ตรวจสอบวงจรโดย OUT+ และ OUT- จะต่อมาในภายหลังกลับ IN+ และ IN- ดังนั้นบิต DAC (D15-D2) จะถูกส่งไปยัง DX แลกเปลี่ยนกับบิต ADC ที่ได้รับมาจากขา DR ซึ่งปกติจะต้องมีค่าเท่ากัน (ในทางปฏิบัติอาจไม่เท่ากันก็ได้) ในการตรวจสอบถ้าใช้ขา IN+ และ IN- สัญญาณภายนอกที่ต่อกับ IN+ และ IN- จะไม่มีผล แต่ถ้าใช้ AUX IN+, AUX IN- สัญญาณภายนอกจะถูกรวมกับ OUT+ และ OUT- สำหรับการควบคุมการต่อกลับ จะทำโดยตั้งค่ารีจิสเตอร์ควบคุม



ภาพที่ ง-17 Timing ภายใน TLC32040

ง.6 การควบคุม AIC

การส่งผ่านข้อมูลใน AIC จะกระทำกันในรีจิสเตอร์สำหรับการรับข้อมูล (Data Receive: DR) และรีจิสเตอร์สำหรับการส่งข้อมูล (Data Transmit: DX) รีจิสเตอร์ทั้ง 2 จะทำการส่งข้อมูลในแบบอนุกรม ในการควบคุมรีจิสเตอร์การส่งผ่านข้อมูลของ AIC จะใช้บิตล่าง 2 บิต (LSBs)

เป็นตัวกำหนดการติดต่อเมื่อบิตทั้ง 2 มีค่าเป็น 0 จะเป็นการส่งผ่านปกติ เมื่อบิตทั้ง 2 มีค่าเป็น 1 จะเป็นการติดต่อระดับ 2 การควบคุมการติดต่อระดับ 2 นี้ AIC จะยอมให้ส่งผ่านข้อมูลครั้งแรกก่อน Switching Back ในภาพที่ ง-18 แสดงข้อกำหนดของการติดต่อระดับ 2

secondary DX serial communication protocol

x x | ← to TA register → | x x | ← to RA register → |	0 0	d13 and d6 are MSBs (unsigned binary)
x | ← to TA' register → | x | ← to RA' register → |	0 1	d14 and d7 are 2's complement sign bits
x | ← to TB register → | x | ← to RB register → |	1 0	d14 and d7 are MSBs (unsigned binary)
x x x x x x x x d7 d6 d5 d4 d3 d2 (Control register)	1 1	d2 = 0/1 deletes/inserts the bandpass filter d3 = 0/1 disables/enables the loopback function d4 = 0/1 disables/enables the AUX IN+ and AUX IN– terminals d5 = 0/1 asynchronous/synchronous transmit receive sections d6 = 0/1 gain control bits (see gain control section) d7 = 0/1 gain control bits (see gain control section)

ภาพที่ ง-18 ข้อกำหนดของการติดต่อระดับ 2

การควบคุมฟังก์ชันการทำงานของรีจิสเตอร์ใน AIC สามารถกระทำได้โดยตรงที่รีจิสเตอร์เหล่านั้น เช่นเดียวกับ พอร์ต Input และ Input Filter เป็นการใช้รีจิสเตอร์เฉพาะของ AIC ดังตัวอย่างในภาพที่ ง-18 เป็นการเซทบิต C2 และ C4 ในรีจิสเตอร์ควบคุมเป็น 1 จะเป็นการเพิ่มค่า Input Bandpass และใช้ Auxiliary Input (AUX IN)

รีจิสเตอร์ A และ B บน AIC จะเป็นตัวควบคุมการทำงานของ AIC โดยรีจิสเตอร์ A ประกอบด้วย TA, RA และเครื่องหมายที่ใช้ควบคุม Filter (Represent Filter Control) รีจิสเตอร์ B จะประกอบด้วย TB, RB และเครื่องหมายที่ใช้ควบคุม A/D และ D/A รีจิสเตอร์เหล่านี้เป็นส่วนหนึ่งของการเซตค่า Timing ภายใน

ตำแหน่งบิตที่ใช้สำหรับควบคุมการส่งและรับของรีจิสเตอร์ TA และ RA เป็น

บิต 0 -1 ⟶ 0, 0

บิต 2 – 6 ⟶ RA

บิต 7 – 8 ⟶ don’t care (x)

บิต 9 – 13 ⟶ TA

บิต 14 – 15 ⟶ don’t care (x)

ตำแหน่งบิตที่ใช้สำหรับควบคุมการส่งและรับของรีจิสเตอร์ TB และ RB เป็น

บิต 0 – 1 ⟶ 0, 1

บิต 2 – 7 ⟶ RB

บิต 8 ⟶ don’t care (x)

บิต 9 – 14 ⟶ TB

บิต 15 ⟶ don’t care (x)

AIC สามารถกำหนดค่า Sampling Frequency และ Filter Bandwidth โดยการใช้การติดต่อระดับ 2 โดยให้เซตค่าเป็น 1 ใน 2 บิตแรก (LSBs) ทำให้การติดต่อระดับ 2 มีค่าเหมือนการติดต่อระดับ 1 ลำดับข้อมูลจะถูกโหลดจากรีจิสเตอร์ส่งข้อมูลพอร์ตอนุกรม (Serial Port Data Transmit Register) และเซต LSBs ทั้ง 2 เป็น 1 สำหรับการติดต่อระดับ 2 ในแต่ละครั้งมีดังนี้

- 0×3(3h) ใช้เรียกการติดต่อระดับ 2
- ค่ารีจิสเตอร์ A
- 0×3 เรียกการติดต่อระดับ 2 ครั้งที่ 2
- ค่ารีจิสเตอร์ B
- 0×3 เรียกการติดต่อระดับ 2 ครั้งที่ 3
- ค่ารีจิสเตอร์ควบคุม

สามารถหาค่า A และ B เพื่อออกแบบค่า Sampling Frequency และ Input Filter Bandwidth (BW)

ง.7 การคำนวณหาค่า A และ B เพื่อออกแบบค่า Fs และ Filter BW

บอร์ด TMS320C31DSK มี Input Clock (CLKIN) เป็น 50 MHz สามารถกำเนิดสัญญาณความถี่ไทเมอร์สูงสุด เป็น MCLK = (CLKIN/4) = 12.5 MHz ซึ่งสูงกว่า Master Clock Frequency ของ AIC ที่มีค่ากับ 10MHz AIC master clock (MCLK) ที่เข้ามาสามารถวัดสัญญาณได้จากขา 8 ใน JP1 สัญญาณสูงสุดที่ได้จาก A/C เราสามารถหาได้จาก Input Clock หารด้วย 8 หรือ

$$MCLK = CLKIN/8 = (50MHz / 8) = 6.25\ MHz$$

Switched – capacitor filter frequency (SCF) เป็นความสัมพันธ์จากรีจิสเตอร์การส่งของ A หรือ

$$SCF = MCLK / (2\times TA) \qquad (2\text{-}30)$$

และความถี่สุ่ม (Sampling frequency) เป็นความสัมพันธ์จาก การส่งในรีจิสเตอร์ของ A และ B

$$Fs = MCLK / (2\times TA\times TB) \qquad (2\text{-}31)$$

Input Filter Bandwidth หรือ Cutoff Frequency เซตที่ 3600 MHz สำหรับ SCF ที่ 288 kHz ค่า SCF ใหม่จะได้ค่า BW นำไปคำนวณในตัวอย่าง และหาค่า A และ B เพื่อไปเซต A/C

ออกแบบ Fs = 8 kHz

การออกแบบ Cutoff Frequency ของ Input Anti Aliasing Filter เป็น 3600 Hz ที่ SCF 288 kHz จาก (2-30)

$$\begin{aligned} TA &= MCLK / (2 \times SCF) = 6.25\ MHz / (2 \times 288\ kHz) \\ &= 10.85 \approx 11 = (01011)_b \end{aligned} \quad (2\text{-}32)$$

จาก (2-31)

$$\begin{aligned} TB &= MCLK / (2 \times TA \times Fs) \\ &= 6.25\ MHz / (2 \times 11 \times 8000) \\ &= 35.51 \approx 36 = (100100)_b \end{aligned}$$

จาก (2-32) จะได้ SCF (ปฏิบัติ) เป็น

$$\begin{aligned} SCF &= 6.25\ MHz / (2 \times TA) \\ &= 284.09\ KHz \end{aligned}$$

Cutoff frequency หรือ Input filter bandwidth

$$\begin{aligned} BW &= 3600\ (New\ SCF / Set\ SCF) \\ &= 3600\ (284.09\ KHz / 288\ KHz) \\ &= 3551.14\ Hz \end{aligned}$$

จะได้ความถี่สุ่ม (Sampling frequency) เป็น

$$\begin{aligned} Fs &= 6.25\ MHz / (2 \times TA \times TB) \\ &= 6.25\ MHz / (2 \times 11 \times 36) \\ &= 7891.41\ Hz \end{aligned}$$

จาก (2-32) จะได้ตำแหน่งบิตที่ใช้ ในการรีจิสเตอร์ควบคุม และเซต TA = RA ซึ่ง TA มี 5 บิต, TB มี 6 บิต และ ×× จะไม่สนใจ (don’t care) เช่น

00	01011	00	01011	00	⇨ 162Ch
xx	*TA*	*xx*	*RA*		

แยกบิตที่จะนำไปใช้เป็น 4 กลุ่มโดยค่า A = 162Ch TB = RB ก็เช่นเดียวกัน

0	100100	0	100100	10	⇨ 4892h
x	*TB*	*x*	*RB*		

ออกแบบ Fs = 10 kHz

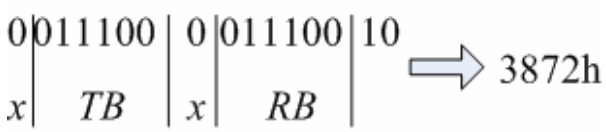
ใช้ Cutoff Frequency หรือ BW สำหรับ Input Anti Aliasing Filter เหมือนกับ Fs = 8 KHz, TA = 11 จะได้

TB = 6.25 MHz / (2×11×10000)
= 28.41 ≈ 28 = $(011100)_b$

ความถี่สุ่ม (Sampling frequency) เป็น

Fs = 6.25 MHz / (2×TA×TB)
= 6.25 MHz / (2×11×28)
= 10146 MHz

ค่า B ได้



หรือ B = 3872 h

ออกแบบ Fs = 20 kHz

ออกแบบ BW = 6.25 MHz / (2×640k)
BW = 3600 (New SCF / Set SCF)

ค่า Switched – capacitor filter frequency ค่าใหม่ เป็น

SCF = 8000 (288K) / 3600 KHz

ค่า TA และ TB เป็น

TA = 6.25 MHz / (2×640 k)
= 4.88 ≈ 5 = $(00101)_b$
TB = 6.25 MHz / (2×5×20000)
= 31.25 ≈31 = $(011111)_b$

ได้ SCF (ปฏิบัติ) เป็น

SCF = 6.25 MHz / (2×5) = 625 kHz

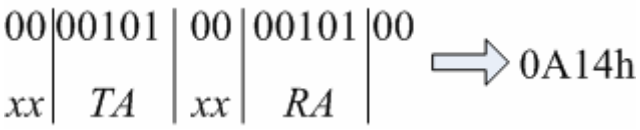
ได้ Bandwidth (ปฏิบัติ) เป็น

BW = 3600(625 k/288 k)
= 7812.5 Hz

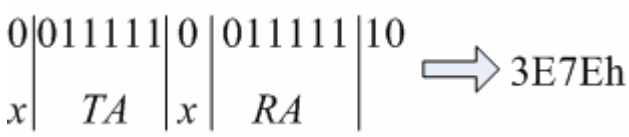
ความถี่สุ่ม (ปฏิบัติ) เป็น

Fs = 6.25 MHz/(2×5×31) = 20161.29 Hz

ค่า A จะได้



หรือ A = 0A14h



หรือ B = 3E7Eh

ตารางที่ ง-4 ค่ารีจิสเตอร์ที่มี Sampling rate แตกต่างกัน 4 ค่า

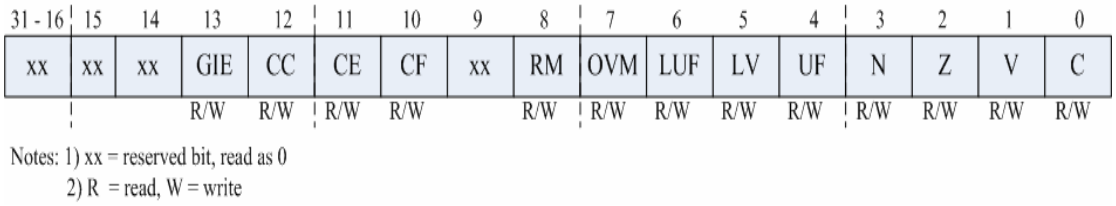
Fs(ออกแบบ),Hz	Fs(ปฏิบัติ)	A	B
8000	7891.41	0x162C	0x4892
10000	10146	0x162C	0x3872
16000	15943	0x0E1C	0x3872
20000	20161.29	0x0A14	0x3E4E

สำหรับ Fs = 16 kHz มี BW (ปฏิบัติ) 5580 Hz

การเพิ่มค่าที่เซตในรีจิสเตอร์ TA’ และ TB’ จะทำให้ค่า Sampling Rate มีความละเอียดมากขึ้น

ง.8 การขัดจังหวะ (Interrupt และ Peripherals)

TMS320C31 สนับสนุนการอินเทอร์รัพท์ทั้งภายในและภายนอกนั้นคือสามารถอินเทอร์รัพท์ CPU หรือ DMA เช่นเดียวกับการใช้ Nonmaskable External Interrupt ในภาพที่ ง-19 ซึ่งแสดงให้เห็นรีจิสเตอร์ Global Interrupt Enable (GIE) บิต ซึ่งอยู่ภายในรีจิสเตอร์สถานะ (Status Register: ST) และเป็นตัวการควบคุมการอินเทอร์รัพท์ทั้งหมดโดยการเซตค่าบิต GIE ให้มีค่าเป็น 1 เพื่อให้มีการอินเทอร์รัพท์ การสั่งให้อินเทอร์รัพท์ไม่ทำงานโดยเซตค่าให้รีจิสเตอร์ Interrupt Enable (IE) เป็น 0 ในภาพที่ ง-20 และเซตค่าให้ GIE เป็น 0 ด้วย ภาพที่ ง-21 แสดง Memory – mapped เป็นตำแหน่งที่ใช้สำหรับอินเทอร์รัพท์

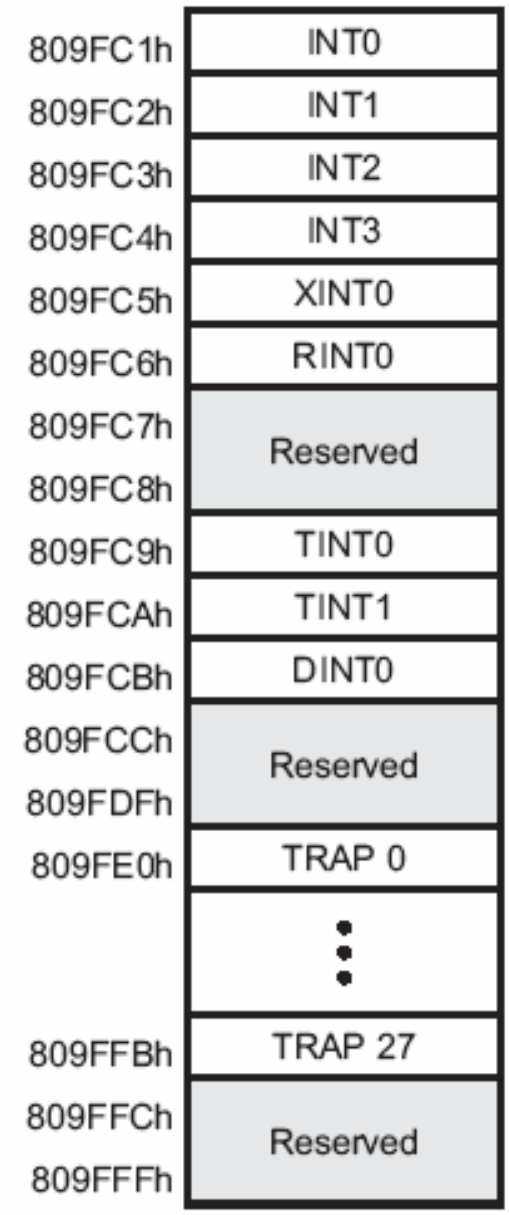


ภาพที่ ง-19 Status (ST) register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	EDINT (DMA)	ETINT1 (DMA)	ETINT0 (DMA)	ERINT1 (DMA)	EXINT1 (DMA)	ERINT0 (DMA)	EXINT0 (DMA)	EINT3 (DMA)	EINT2 (DMA)	EINT1 (DMA)	EINT0 (DMA)
					R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xx	xx	xx	xx	xx	EDINT (CPU)	ETINT1 (CPU)	ETINT0 (CPU)	ERINT1 (CPU)	EXINT1 (CPU)	ERINT0 (CPU)	EXINT0 (CPU)	EINT3 (CPU)	EINT2 (CPU)	EINT1 (CPU)	EINT0 (CPU)
					R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ภาพที่ ง-20 Interrupt enable (IE) register



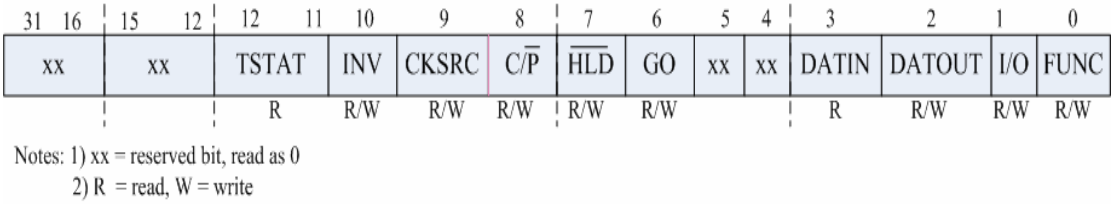
ภาพที่ ง-21 Memory – mapped interrupt location

ง.9 ตัวตั้งเวลา (Timer)

TMS320C31 สนับสนุน Timer 2 ตัว นั่นคือสามารถใช้สัญญาณนับภายนอกได้ สิ่งเหล่านี้ต้องเตรียมสัญญาณเวลาที่จำเป็นเพื่อเป็นค่าเริ่มต้นการแปลงค่าของ ADC ภาพที่ ง-22 แสดงรีจิสเตอร์ Peripheral Bus Memory – mapped รีจิสเตอร์ Timer Global Control (ภาพที่ ง-23) อยู่ที่ตำแหน่ง 808020 รีจิสเตอร์ Timer's status และรีจิสเตอร์ Timer's period อยู่ที่ตำแหน่ง 808028 เป็นตำแหน่งเฉพาะของ Timer's frequency รีจิสเตอร์ Timer's counter อยู่ที่ตำแหน่ง 808024 จะบรรจุค่าที่เพิ่มของ Counter เมื่อค่าของรีจิสเตอร์ Period เท่ากับค่าของรีจิสเตอร์ Timer Counter รีจิสเตอร์ Counter จะรีเซตเป็น 0 โดยที่ตำแหน่งรีเซต Timer Counter และรีจิสเตอร์ Period ถูกเซตเป็น 0 สามารถใช้รีจิสเตอร์เหล่านี้ออกแบบอัตราการอินเทอร์รัพท์ ซึ่งมีผลต่อการออกแบบ f_s

808000h	DMA Global Control
808004h	DMA Source Address
808006h	DMA Destination Address
808008h	DMA Transfer Count
808020h	Timer 0 Global Control
808024h	Timer 0 Counter
808028h	Timer 0 Period Register
808030h	Timer 1 Global Control
808034h	Timer 1 Counter
808038h	Timer 1 Period Register
808040h	Serial Port Global Control
808042h	FSX/DX/CLKX Serial Port Control
808043h	FSR/DR/CLKR Serial Port Control
808044h	Serial R/X Timer Control
808045h	Serial R/X Timer Counter
808046h	Serial R/X Timer Period Register
808048h	Data Transmit
80804Ch	Data Receive
808064h	Primary-Bus Control

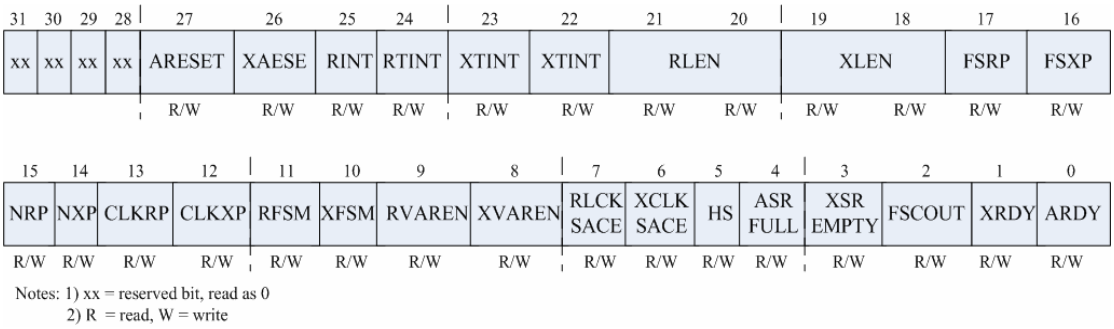
ภาพที่ ง-22 Peripheral bus memory – mapped registers



ภาพที่ ง-23 Timer global register

ง.10 พอร์ทอนุกรม (Serial Port)

TMS320C31 สนับสนุน Serial port 1 พอร์ท (C30 มี 2 พอร์ต) และการเซตค่ารีจิสเตอร์ควบคุมการแสดงในภาพที่ ง-22 ส่วนในภาพที่ ง-24 แสดงรูปแบบของรีจิสเตอร์ Serial Port Global Control AIC บนบอร์ด DSK ต่อกับ C31 โดยพอร์ทอนุกรม ขาที่ 22 ต่อกับตัว Jumper (JP1) ซึ่ง C31 ส่งสัญญาณไปยัง AIC โดย Jumper เหล่านี้สามารถเอาออกได้เพื่อหยุดการติดต่อจาก C31 ไปยัง AIC และ JP1 จะเข้าถึงสัญญาณของ C31 และสามารถต่อไปยังภายนอกบอร์ดได้



ภาพที่ ง-24 Serial port control register

ตารางที่ ง-5 ตำแหน่งและการเซตค่ารีจิสเตอร์ TMS320C31

รีจิสเตอร์	ตำแหน่ง	คำสั่ง
Timer 0 period	0x808020	Load 0x1
Timer 0 global control	0x808020	Load 0c3C1
I/O flag	IOF	Load 0x2
SP0 transmit port control	0x808042	Load 0x131
SP0 receive port control	0x808043	Load 0x131
SP0 global control	0x808040	Load 0x0E970300
SP0 data transmit	0x808048	Load 0x0
I/O	IOF	Load 0x6
Interrupt flag	IF	Load 0x0
Interrupt enable	IE	OR 0x10
Status register	ST	OR 0x2000

ประวัติผู้วิจัย

ชื่อ : นายอภิชิต กระจ่างเย่า

ชื่อวิทยานิพนธ์ : การสร้างระบบบีบอัดและขยายสัญญาณคลื่นไฟฟ้าหัวใจแบบเวลาจริง โดยใช้การประมวลผลสัญญาณเชิงเลขหลายอัตรา

สาขาวิชา : อุปกรณ์การแพทย์

ประวัติ

ประวัติการศึกษา

สำเร็จการศึกษาระดับปริญญาตรี สาขาเทคโนโลยีอุตสาหกรรม คณะวิทยาศาสตร์และเทคโนโลยี สถาบันราชภัฏพระนครศรีอยุธยา สำเร็จการศึกษา วันที่ 23 มิถุนายน 2543

สำเร็จการศึกษาระดับประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.) สาขาวิชาช่างอิเล็กทรอนิกส์ วิทยาลัยเทคนิคชัยนาท สำเร็จการศึกษาปี 2541

สำเร็จการศึกษาระดับประกาศนียบัตรวิชาชีพ (ปวช.) สาขาวิชาช่างอิเล็กทรอนิกส์ วิทยาลัยเทคนิคลพบุรี สำเร็จการศึกษาปี 2539

ประวัติการทำงาน

ปฏิบัติงานในตำแหน่ง เจ้าหน้าที่ห้องปฏิบัติการประจำโปรแกรมวิชาเทคโนโลยีอุตสาหกรรม คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏพระนครศรีอยุธยา (ธันวาคม 2543 – กุมภาพันธ์ 2549)

สถานที่ติดต่อ

บ้านเลขที่ 2/1 หมู่ 1 ตำบล บ้านลี่ อำเภอ บางปะหัน จังหวัดพระนครศรีอยุธยา 13220