การศึกษาบอร์ดประมวลผลสัญญาณดิจิตอลและการประยุกต์ใช้งาน

DIGITAL SIGNAL PROCESSING (DSP) BOARD STUDY AND ITS APPLICATION

โดย

นางสาวพรสุภา ศุภราศรี

นางสาวพันธุมวดี จิระชุติโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

การศึกษาบอร์ดประมวลผลสัญญาณดิจิตอลและการประยุกต์ใช้งาน

DIGITAL SIGNAL PROCESSING (DSP) BOARD STUDY AND ITS APPLICATION

โดย

นางสาวพรสุภา ศุภราศรี

นางสาวพันธุมวดี จิระชุติโรจน์

อาจารย์ที่ปรึกษา

อาจารย์สว่าง เลิศฤธิรสุนทร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การศึกษาบอร์ดประมวลผลสัญญาณดิจิตอลและการประยุกต์ใช้งาน
(Digital Signal Processing : DSP)

ผู้จัดทำ

นางสาวพรสุภา ศุภราศรี 39014357

นางสาวพันธุมวดี จิระชุติโรจน์ 39014363

..... อาจารย์ที่ปรึกษา

(อาจารย์สว่าง เลิศฤทธิรสุนทร)

การศึกษาบอร์ดประมวลผลสัญญาณดิจิตอลและการประยุกต์ใช้งาน

DIGITAL SIGNAL PROCESSING (DSP) BOARD STUDY AND ITS APPLICATION

โดย

นางสาวพรสุภา ศุภราศรี

นางสาวพันธุมวดี จิระชุติโรจน์

อาจารย์ที่ปรึกษา

อาจารย์สว่าง เลิศถิรสุนทร

ปีการศึกษา 2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ จะกล่าวถึงโครงสร้างสถาปัตยกรรมภายในและการประยุกต์ใช้งานบอร์ด DSP TMS320C3x โดยมีการต่อชุดบอร์ดอินพุท/เอาท์พุท อินเตอร์เฟส และเขียนโปรแกรมเพื่อแสดงผลการทำงานรวมทั้งเขียนโปรแกรมการทำงานพื้นฐาน ซึ่งสามารถพัฒนาโปรแกรมให้ใช้งานได้ในระดับที่สูงขึ้น

Abstract

This project refers to architectural overview and applications of TMS320C3x DSP Board.It is I/O Interface Board and program to display operation including basic programs that can develop to using in higher operation.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีก็เพราะได้รับความเมตตาจาก อาจารย์สว่าง เลิศฤทธิสุนทร อาจารย์ที่ปรึกษาโครงงานนี้ที่คอยให้ความช่วยเหลือ คำแนะนำ ชี้แนะ พร้อมกับให้ข้อมูลต่างๆ ซึ่งเป็นประโยชน์ต่อการทำโครงงานนี้เป็นอย่างยิ่ง

ขอขอบพระคุณ Mr.Teiichi Furukawa ที่ให้แนวทางในการเขียนโปรแกรมและตอบคำถามเกี่ยวกับโครงงานนี้

ขอขอบพระคุณคณาจารย์ทุกท่านที่ได้ให้ได้ให้คำชี้แนะ คำปรึกษาและประสิทธิ์ประสาทวิชาความรู้ให้กับผู้จัดทำ

ขอขอบพระคุณภาควิชาวิศวกรรมระบบควบคุม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้เอื้อเฟื้อเรื่องเครื่องมือและอุปกรณ์ต่างๆ ในการทำโครงงานนี้ให้ประสบผลสำเร็จลุล่วงไปด้วยดี

ขอขอบพระคุณคุณพ่อ คุณแม่ และคุณพี่ของผู้จัดทำ ที่ได้อุปการะผู้จัดทำและเป็นกำลังใจให้แก่ผู้จัดทำ

ขอขอบคุณเพื่อนๆ พี่ๆ ทุกคนที่คอยให้คำปรึกษาและคำแนะนำต่างๆ ตลอดจนอุปกรณ์ในการทำงานมาโดยตลอด

ผู้จัดทำ

นางสาวพรสุภา ศุภราศรี

นางสาวพันธุมวดี จิระชุติโรจน์

สารบัญ

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

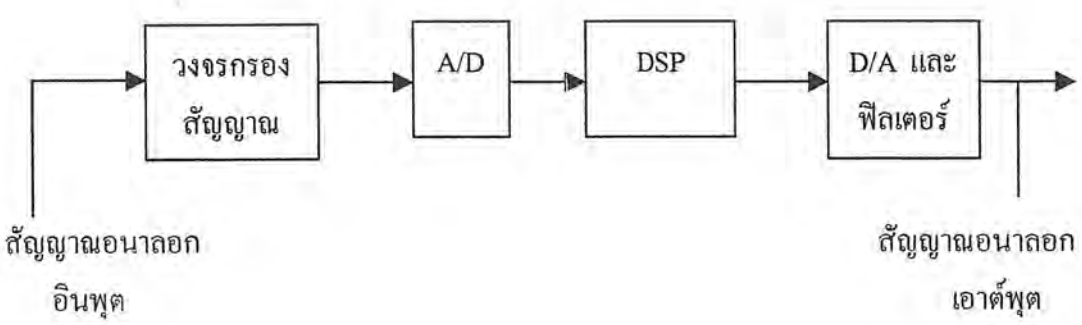
สารบัญตาราง

บทที่ 1
บทนำ

การประมวลผลสัญญาณดิจิตอล (Digital Signal Processing : DSP) เป็นการใช้ความรู้ทางคณิตศาสตร์แบบดิจิตอลมาจัดการสัญญาณต่างๆ ซึ่งมีบทบาทสำคัญในวงการอิเล็กทรอนิกส์ในปัจจุบันอย่างยิ่ง จนนำมาประยุกต์ใช้งานอย่างกว้างขวาง เนื่องจากขบวนการทางดิจิตอลเป็นการออกแบบระบบด้วยซอฟท์แวร์จึงสามารถสร้างและประมวลผลสัญญาณต่างๆ เพื่อให้ได้ผลลัพธ์ตามต้องการได้ง่าย ในขณะที่การจัดการสัญญาณอนาลอกซึ่งโดยปกติเป็นทั้งขบวนการแบบเชิงเส้นและไม่เชิงเส้น จะมีอุปกรณ์ช่วยในการประมวลผลมากมาย เช่น ตัวต้านทาน ตัวเก็บประจุ ทรานซิสเตอร์ ออปแอมป์ และอุปกรณ์อิเล็กทรอนิกส์ต่างๆ นอกจากนี้อุปกรณ์ที่เกี่ยวกับการประมวลผลสัญญาณดิจิตอลมีราคาถูกลงและทำงานได้รวดเร็วขึ้น การประยุกต์ใช้งานที่พบเห็นได้มาก ได้แก่เรื่องต่างๆ ต่อไปนี้

- การแปลงฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่องอย่างรวดเร็ว (Fast Fourier Transform : FFT)
- ฟิลเตอร์ อะแดปทีฟฟิลเตอร์ อีควอไลเซอร์ และตัวเลื่อนเฟส
- มิกเซอร์ มอดูเลเตอร์ และตัวเปรียบเทียบเฟส
- วงจรกำเนิดสัญญาณ ออสซิลเลเตอร์ปรับค่าได้ และแหล่งสัญญาณรบกวน
- อุปกรณ์ไม่เชิงเส้น ลิมิตเตอร์ คอมพาราเตอร์
- การควบคุม วงจรควบคุมแบบต่างๆ เซอร์โว ฯลฯ
- วงจรประมวลผลสัญญาณภาพ เสียงพูด ฯลฯ

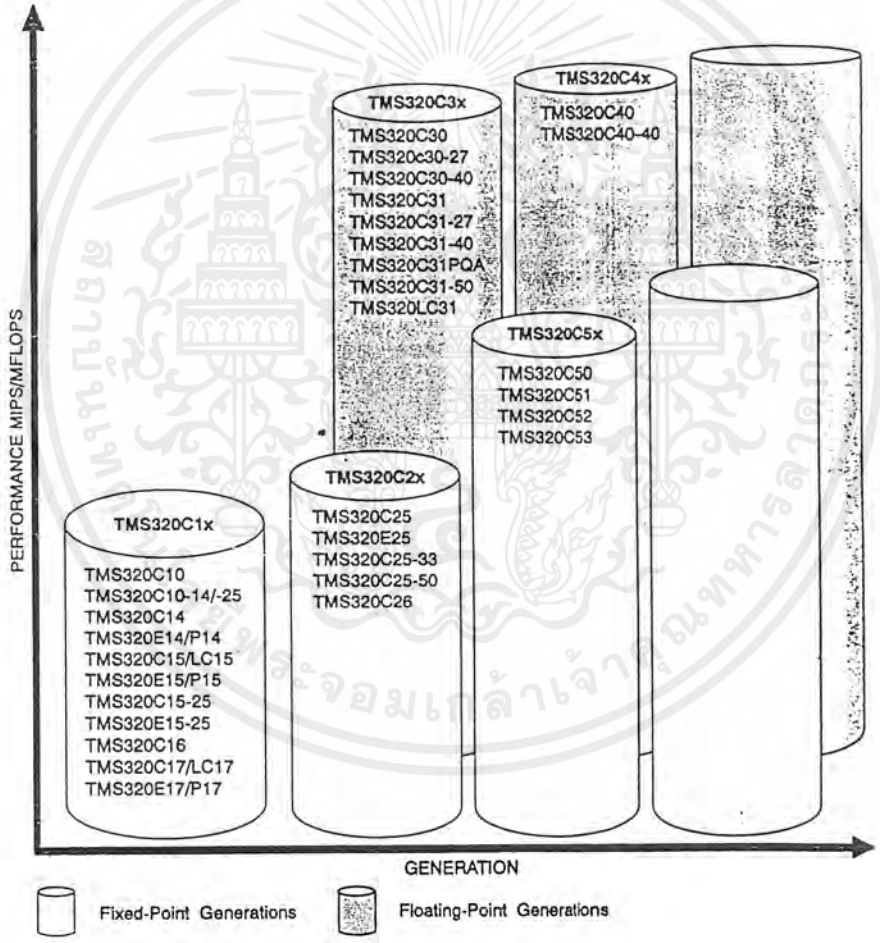
รูปแบบของการประมวลสัญญาณเขียนเป็นบล็อกไดอะแกรมได้ดังรูปที่ 1-1



รูปที่ 1-1 แผนผังการทำงานของระบบประมวลสัญญาณด้วย DSP

การประมวลผลสัญญาณในรูปแบบ DSP จะใช้ไมโครโปรเซสเซอร์ (microprocessor) ที่มีความเร็วสูง ดังนั้นจึงประยุกต์ใช้งานได้ง่าย และเป็นการออกแบบระบบด้วยซอฟท์แวร์ (software) ดังนั้นจึงมีการนำไปใช้งานด้านต่างๆ ที่เป็นผลิตภัณฑ์สำเร็จรูปหลายอย่าง เช่น โมเด็ม อุปกรณ์โทรศัพท์ วงจรสังเคราะห์เสียง วงจรควบคุมขบวนการผลิต เป็นต้น

สำหรับ DSP ตระกูล TMS320 ประกอบด้วย 5 รุ่น : TMS320C1x, TMS320C2x, TMS320C3x, TMS329C4x และ TMS320C5x (ดังรูปที่ 1-2) แสดงวิวัฒนาการทั้งใน DSP รุ่นแรกๆ และรุ่นใหม่ที่มีสมรรถภาพสูงขึ้น



รูปที่ 1-2 วิวัฒนาการของ TMS320

บัสภายในของ TMS320 และชุดคำสั่งพิเศษของ DSP มีความเร็วและความยืดหยุ่นในการทำงานถึง 50 ล้านคำสั่งอิงดรรชนีต่อวินาที (million floating – point instuctions per second : MFLOPS) สามารถทำให้มีความเร็วที่ดีที่สุดโดยเพิ่มฟังก์ชันในฮาร์ดแวร์ (hardware) ในขณะที่ตัวประมวลผล (processor) อื่นจะเพิ่มในซอฟท์แวร์ (software) หรือรหัสคำสั่งไมโคร (microcode) การที่จะเพิ่มฮาร์ดแวร์ได้ต้องมีสมรรถภาพ ซึ่งแต่ก่อนไม่สามารถทำได้ในชิปเดียว (single chip)

TMS320C30 และ TMS320C31 สามารถทำการคูณแบบขนาน (parallel) และหน่วยคำนวณและตรรกะ (Arithmetic Logic Unit :ALU) จัดการข้อมูลแบบจำนวนเต็ม (integer) หรือแบบอิงดรรชนี (floating-point) ภายในรอบเดียว (single cycle) ภายในตัวประมวลผลมีแฟ้มข้อมูลรีจิสเตอร์อเนกประสงค์ (general-purpose register file), โปรแกรมแคช (program cache), หน่วยคำนวณรีจิสเตอร์ช่วย (Auxiliary Register Arithmetic Unit : ARAU) สำหรับทำหน้าที่พิเศษโดยเฉพาะ, หน่วยความจำภายในที่สามารถเข้าถึงได้แบบคู่ (dual), ช่องการเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access : DMA) 1 ช่องที่สนับสนุนอินพุต/เอาท์พุตพร้อมกัน และรอบของเครื่อง (machine cycle) สั้น โครงสร้างเหล่านี้ถูกผลิตให้มีสมรรถภาพสูงและใช้งานได้ง่าย

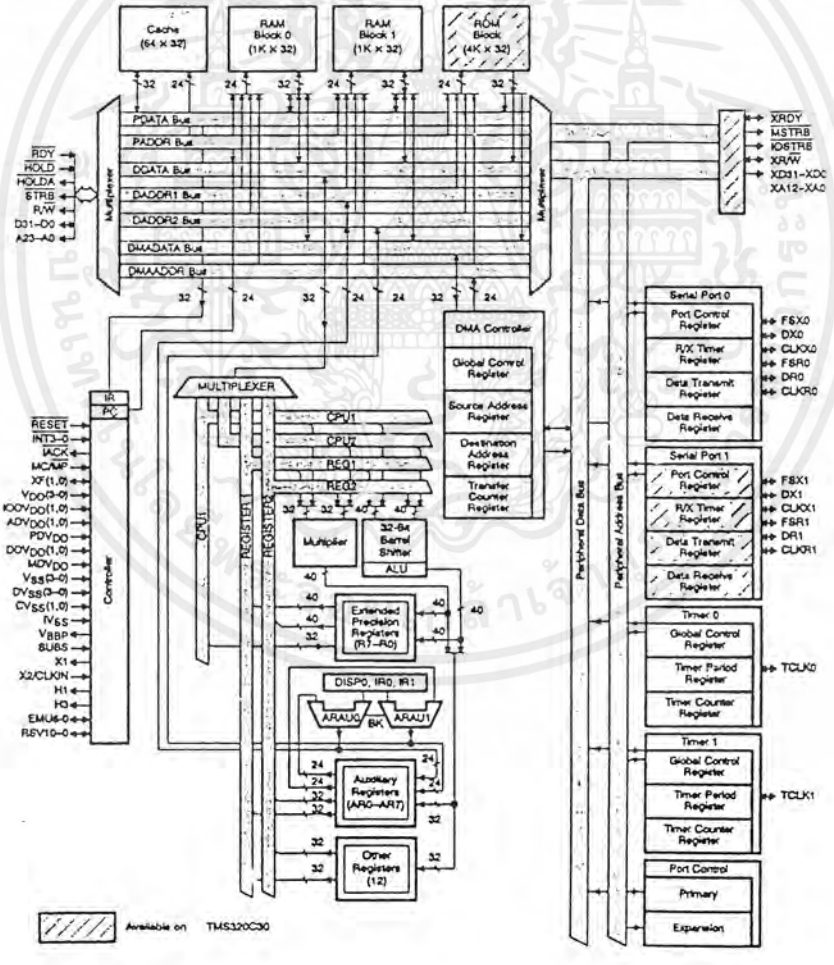
ในโครงงานนี้จะใช้ชิป DSP TMS320C31-50 ซึ่งเป็นรุ่น TMS320C3x ของบริษัท TEXAS INSTRUMENT เป็น DSP รุ่นอิงดรรชน์ขนาด 32 บิตแบบ CMOS สมรรถภาพสูงอยู่ในตระกูลของ TMS320 สามารถทำงานด้วยความเร็วสูงถึง 40 นาโนวินาทีต่อไซเคิล (ns /cycle) ประมวลผลได้ด้วยความเร็วถึง 60 ล้านคำสั่งอิงดรรชนีต่อวินาที และ 30 ล้านคำสั่งต่อวินาที (million instructions per second :MIPS)

บทที่ 2
ทฤษฎี

2.1 สถาปัตยกรรม TMS320C3x

สถาปัตยกรรม TMS320C3x สามารถตอบสนองความต้องการพื้นฐานของขั้นตอนวิธีคำนวณที่ซับซ้อน และเน้นการแก้ปัญหาทั้งทางด้านฮาร์ดแวร์และซอฟท์แวร์ หน่วยอิงครรชนีมีพิสัยแบบพลวัต (dynamic range) ที่กว้างและเที่ยงตรง, หน่วยความจำบนชิปขนาดใหญ่, ระบบแบบขนานดีกรีสูง, และตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง ทำให้ TMS320C3x มีสมรรถภาพสูง

รูปที่ 2-1 เป็นบล็อกไดอะแกรมของสถาปัตยกรรม TMS320C3x



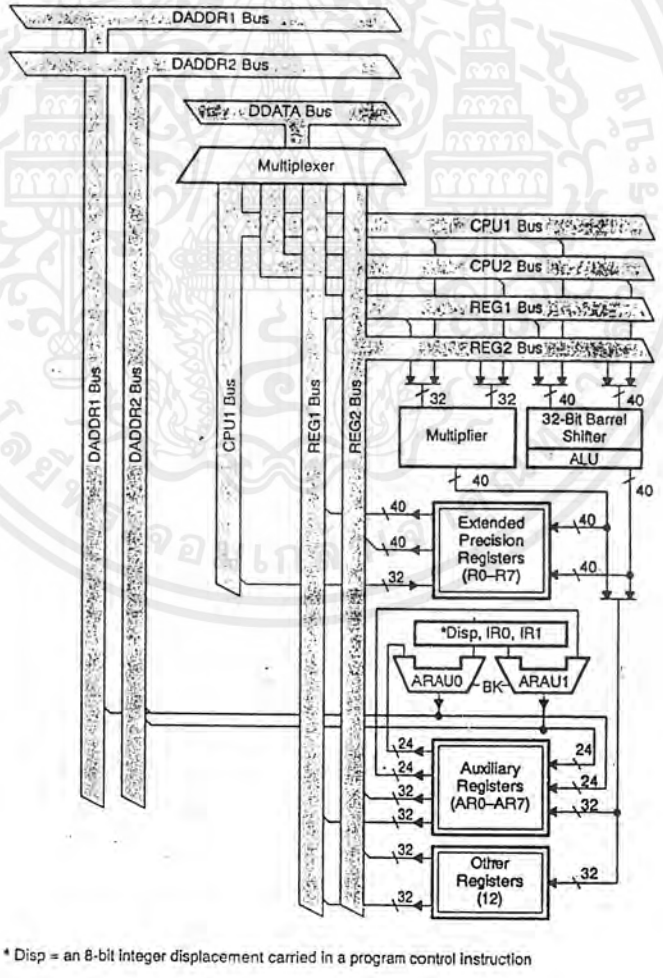
รูปที่ 2-1 บล็อกไดอะแกรมของ TMS320C3x

2.1.1 หน่วยประมวลผลกลาง (Central Processing Unit : CPU)

ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- ตัวคูณเลขอิงครรชนี/จำนวนเต็ม (Floating-point/integer multiplier)
- หน่วยคำนวณและตรรกะ (Arithmetic logic unit : ALU) สำหรับไว้ดำเนินการเลขอิงครรชนี, จำนวนเต็ม และตรรกะ
- ตัวเลื่อนหรือหมุนเวิร์ดข้อมูล (barrel shifter) 32 บิต
- บัสภายใน (CPU1/CPU2 และ REG1/REG2)
- หน่วยคำนวณรีจิสเตอร์ช่วย (Auxiliary register arithmetic units : ARAUs)
- แฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง (CPU register file)

รูปที่ 2-2 แสดงอุปกรณ์ต่างๆ ของหน่วยประมวลผลกลาง โดยจะตามหลังด้วยคำอธิบาย



รูปที่ 2-2 หน่วยประมวลผลกลาง

2.1.1.1 ตัวคูณ (Multiplier)

ตัวคูณจะทำการคูณแบบรอบเดียว โดยเป็นค่าจำนวนเต็ม 24 บิต และค่าอิงดรรชนี 32 บิต วิธีการคำนวณเลขอิงดรรชนีของ TMS320C3x จะให้ทำงานที่ความเร็วไม่อิงดรรชนี (fixed-point) ผ่านทางรอบคำสั่ง 50 นาโนวินาที และระบบแบบขนานดีกรีสูง

เมื่อตัวคูณทำการคูณเลขอิงดรรชนี, อินพุตจะเป็นเลขอิงดรรชนี 32 บิต และได้ผลลัพธ์เป็นเลขอิงดรรชนี 40 บิต

เมื่อตัวคูณทำการคูณเลขจำนวนเต็ม, อินพุตจะเป็นเลขจำนวนเต็ม 24 บิต และได้ผลลัพธ์เป็นเลขจำนวนเต็ม 32 บิต

2.1.1.2 หน่วยคำนวณและตรรกะ (ALU)

หน่วยคำนวณและตรรกะ แสดงการจัดการข้อมูลจำนวนเต็ม 32 บิต, ตรรกะ 32 บิต และอิงดรรชนี 40 บิต แบบรอบเดียว รวมทั้งการเปลี่ยนจำนวนเต็มและอิงดรรชนี แบบรอบเดียว ผลลัพธ์ของหน่วยคำนวณและตรรกะจะยังคงเป็นเลขจำนวนเต็ม 32 บิตหรืออิงดรรชนี 40 บิตเสมอ ส่วนตัวเลื่อนหรือหมุนเวิร์ดข้อมูล จะใช้เพื่อเลื่อนหรือหมุนบิตจนถึง 32 บิตซ้ายหรือขวาภายในรอบเดียว

บัสภายใน, CPU1/CPU2 และ REG1/REG2, บรรจุ 2 ตัวถูกดำเนินการ (operand) จากหน่วยความจำ และบรรจุ 2 ตัวถูกดำเนินการจากแฟ้มรีจิสเตอร์ ดังนั้นจึงสามารถคูณแบบขนาน และบวก/ลบเลขจำนวนเต็มหรือเลขอิงดรรชนี 4 ตัวได้ในรอบเดียว

2.1.1.3 หน่วยคำนวณรีจิสเตอร์ช่วย (ARAUs)

ARAU0 และ ARAU1 สามารถสร้าง 2 ตำแหน่งในรอบเดียว หน่วยคำนวณรีจิสเตอร์ช่วย ทำงานแบบขนานด้วยตัวคูณและALU ซึ่งทั้ง 2 ตัวนี้สนับสนุนการเข้าถึงตำแหน่งด้วยการแทนที่ (addressing with displacements), รีจิสเตอร์ดรรชนี (index register : IR0 และ IR1), การเข้าถึงตำแหน่งแบบวนรอบ (circular addressing) และการเข้าถึงตำแหน่งแบบกลับบิต (bit-reversed addressing)

2.1.1.4 แฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง (CPU register file)

TMS320C3x จะจัดให้รีจิสเตอร์ 28 ตัวอยู่ในแฟ้มรีจิสเตอร์ที่มีทางเข้าออกหลายทาง ซึ่งติดกับหน่วยประมวลผลกลาง ส่วนตัวนับโปรแกรม (Program Counter : PC) จะไม่รวมอยู่ในรีจิสเตอร์ 28 ตัวข้างต้น รีจิสเตอร์ทั้งหมดนี้สามารถทำงานโดยตัวคูณ, หน่วยคำนวณและ

ตรรกะ และสามารถใช้เป็นรีจิสเตอร์อเนกประสงค์ 32 บิต นอกจากนี้รีจิสเตอร์ยังมีฟังก์ชันพิเศษ เช่น รีจิสเตอร์เพิ่มเติม (extended-precision) 8 ตัวจะเหมาะสำหรับเก็บผลลัพธ์แบบอิงดรรชนี, รีจิสเตอร์ช่วย 8 ตัว จะสนับสนุนโหมดการเข้าถึงตำแหน่งโดยอ้อม (indirect addressing mode) ที่หลากหลาย และสามารถใช้เป็นรีจิสเตอร์จำนวนเต็ม 32 บิตทั่วไป และ รีจิสเตอร์แบบตรรกะ, รีจิสเตอร์ที่เหลือจะจัดให้แต่ละฟังก์ชันของระบบ เช่น การเข้าถึงตำแหน่ง, การจัดการสแตก (stack), สถานะของตัวประมวลผล, การอินเทอร์รัพต์ และบล็อก (block) ที่ทำซ้ำ

ชื่อของรีจิสเตอร์และฟังก์ชันที่กำหนด จะอยู่ในตาราง 2-1 ตามตารางจะบรรยายฟังก์ชันของแต่ละรีจิสเตอร์หรือกลุ่มของรีจิสเตอร์ไว้อย่างย่อๆ

ตาราง 2-1 แสดงรีจิสเตอร์ของหน่วยประมวลผลกลาง

Register name	Assigned Function
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

R0 – R7	รีจิสเตอร์เพิ่มเติม (extended–precision register) 0–7, มีความสามารถในการจัดเก็บและสนับสนุนการจัดการเลขจำนวนเต็ม 32 บิต และเลขอิงดรรชนี 40 บิต
AR0 - AR7	รีจิสเตอร์ช่วย (auxillary register) 0-7 สามารถเข้าถึงได้โดยหน่วยประมวลผลกลาง และแก้ไขโดยหน่วยคำนวณรีจิสเตอร์ช่วย 2 ตัว โดยที่ฟังก์ชันพื้นฐานของรีจิสเตอร์ช่วยทำให้เกิดตำแหน่ง 24 บิต ซึ่งสามารถใช้เป็นตัวนับลูป (loop counter) หรือ เป็นรีจิสเตอร์อเนกประสงค์ 32 บิต ซึ่งสามารถแก้ไขโดยตัวคูณและหน่วยเลขคณิตแบบตรรกะ
DP	ตัวชี้เพจของข้อมูล (data page pointer) คือ รีจิสเตอร์ 32 บิต โดยที่ 8 บิตล่างถูกใช้เป็นโหมดการเข้าถึงตำแหน่งโดยตรง (direct addressing mode) เสมือนเป็นตัวชี้ตำแหน่งเพจของข้อมูล เพจของข้อมูลมีความยาว 64K เวิร์ด (word) จากทั้งหมด 256 เพจ
IR0, IR1	รีจิสเตอร์ดรรชนี 32 บิต จะเก็บค่าให้กับหน่วยคำนวณรีจิสเตอร์ช่วย เพื่อคำนวณตำแหน่งที่ชี้
BK	รีจิสเตอร์ขนาดบล็อก (block size register) 32 บิต ถูกใช้โดยหน่วยคำนวณรีจิสเตอร์ช่วยในการเข้าถึงตำแหน่งแบบวนรอบ (circular addressing) เพื่อระบุขนาดของบล็อกข้อมูล (data block)
SP	ตัวชี้สแตกของระบบ (system stack pointer) คือ รีจิสเตอร์ 32 บิต ซึ่งเก็บตำแหน่งของสแตกบนสุด (top stack) ซึ่ง SP จะชี้อยู่ที่ตัวสุดท้ายที่ถูกพุช (push) ลงมา (การพุชนั้นก่อนพุช ค่าของ SP จะถูกเพิ่มก่อนทุกครั้ง ส่วนการป็อป (pop) จะทำการป็อปก่อนจึงจะลดค่าใน SP) ค่า SP เปลี่ยนแปลงตามการอินเทอร์รัพต์, แทร็ป (trap), เรียก (call), รีเทิร์น (return), คำสั่งพุช และคำสั่งป็อป
ST	รีจิสเตอร์สถานะ (status register) เก็บข้อมูลทั่วไปที่เกี่ยวข้องกับสถานะของหน่วยประมวลผลกลาง
IE	รีจิสเตอร์อินเทอร์รัพต์หน่วยประมวลผลกลาง/การเข้าถึงหน่วยความจำโดยตรง (CPU/DMA interrupt enable register) เป็น รีจิสเตอร์ 32 บิต บิตอินเทอร์รัพต์หน่วยประมวลผลกลาง (CPU interrupt enable bit) คือ บิต 10-0 บิตอินเทอร์รัพต์การเข้าถึงหน่วยความจำโดยตรง (DMA interrupt enable bit) คือ บิต 26-16 โดย 1 = ได้ (enable), 0 = ไม่ได้ (disable)
IF	รีจิสเตอร์อินเทอร์รัพต์แฟลกของหน่วยประมวลผลกลาง (CPU interrupt flag register) เป็น รีจิสเตอร์ 32 บิต โดย 1 = อินเทอร์รัพต์, 0 = ไม่อินเทอร์รัพต์

IOF รีจิสเตอร์แฟลกของอินพุต/เอาท์พุต (I/O flag register) ควบคุมฟังก์ชันของขาภายนอกที่มีหน้าที่พิเศษ, (XF0 และ XF1) โดยที่ขาเหล่านี้อาจถูกกำหนดเป็นอินพุตหรือเอาท์พุตก็ได้

RC ตัวนับการทำซ้ำ (repeat counter) เป็น รีจิสเตอร์ 32 บิต ใช้ระบุจำนวนครั้งที่บล็อกของรหัส (code) ที่ทำซ้ำ เมื่อบล็อกที่ทำซ้ำกำลังทำงาน

RS รีจิสเตอร์ตำแหน่งเริ่มต้นการทำซ้ำ (repeat start address register) 32 บิต เก็บตำแหน่งเริ่มต้นของบล็อกของหน่วยความจำโปรแกรมที่ทำซ้ำ

RE รีจิสเตอร์ตำแหน่งจบการทำซ้ำ (repeat end address register) 32 บิต เก็บตำแหน่งสุดท้ายของบล็อกที่ทำซ้ำ

PC ตัวนับโปรแกรม (program counter) เป็น รีจิสเตอร์ 32 บิต เก็บตำแหน่งของคำสั่งต่อไปที่จะไปนำมา (fetch) ถึงแม้ PC จะไม่ใช่ส่วนของแฟ้มรีจิสเตอร์ของหน่วยประมวลผลกลาง แต่ก็สามารถแก้ไขโดยคำสั่งที่แก้ไขสายงานโปรแกรม

2.1.2 การจัดหน่วยความจำของ TMS320C31

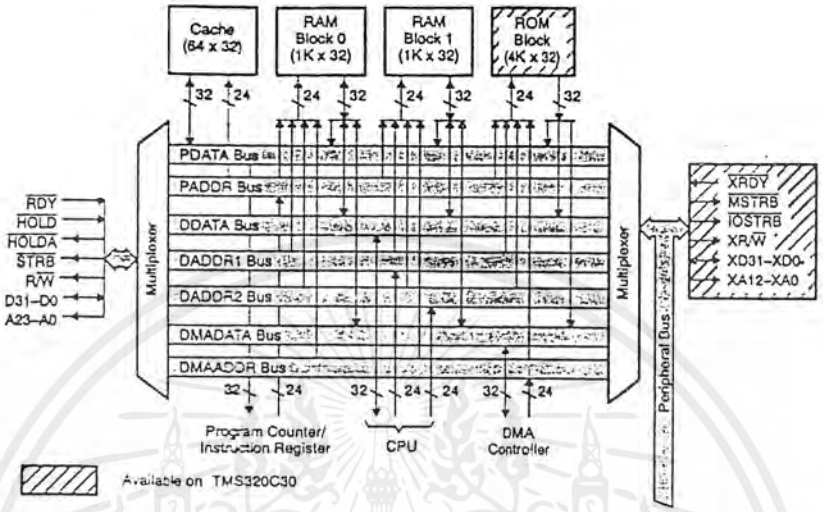
TMS320C3x มีพื้นที่ของหน่วยความจำทั้งหมดเป็น 16M (million) 32 บิต เวิร์ด โดยบรรจุโปรแกรม, ข้อมูล และเนื้อที่ของอินพุต/เอาท์พุตอยู่ภายใน ดังนั้นสามารถเก็บตาราง (table), สัมประสิทธิ์ (coefficient), รหัสของโปรแกรม (program code) หรือข้อมูล ไว้ได้ทั้งใน RAM หรือ ROM ก็ได้ ในกรณีนี้จะทำให้สามารถใช้หน่วยความจำได้มากที่สุด และ จัดพื้นที่หน่วยความจำได้ตามต้องการ

2.1.2.1 RAM, ROM และ Cache

รูปที่ 2-3 แสดงการจัดการหน่วยความจำของ TMS320c3x โดยใน TMS320C31 มี RAM 2 ตัว แต่ละตัวมีขนาด 1K x 32 บิต, ไม่มี ROM และมี Cache ขนาด 64 x 32 บิต เพื่อเก็บคำสั่งที่ใช้บ่อยๆ (ทำให้ทำงานได้ดีขึ้น)

TMS320C31 นี้มีการแยกหน่วยความจำข้อมูลกับหน่วยความจำโปรแกรมออกจากกัน และทำงานโดยวิธีไปป์ไลน์ (pipeline) จึงทำให้มีความเร็วสูงขึ้น การแยกบัสของโปรแกรม (program bus), บัสของข้อมูล (data bus) และบัสของการเข้าถึงหน่วยความจำโดยตรง (DMA bus) ทำให้สามารถไปนำโปรแกรมมา, อ่านข้อมูล และทำการเข้าถึงหน่วยความจำโดยตรงได้พร้อมๆกัน เช่น ในขณะที่หน่วยประมวลผลกลางกำลังจัดการกับข้อมูลอยู่ใน RAM ตัวหนึ่งอยู่

ก็ยังสามารถไปนำโปรแกรมมาโดยเข้าถึงหน่วยความจำโดยตรงผ่าน RAM อีกตัวหนึ่งได้ภายในรอบเดียว



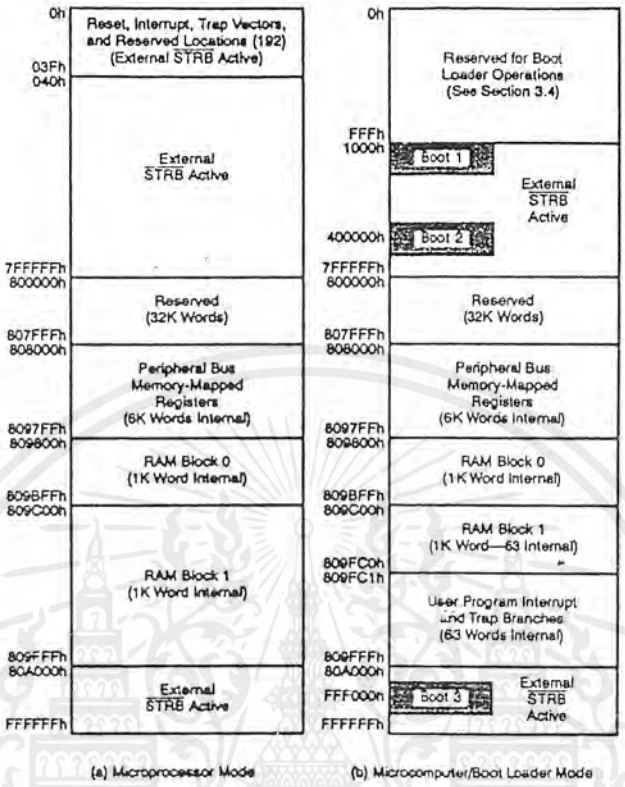
รูปที่ 2-3 การจัดการหน่วยความจำ

2.1.2.2 การแทนข้อมูล (Memory Map) ของ TMS320C31

การแทนข้อมูลขึ้นอยู่กับการตั้งค่าขา $\overline{\text{MCBL/MP}}$ ว่าอยู่ในโหมดใดของฟังก์ชันดังนี้

- โหมดไมโครโปรเซสเซอร์ ($\overline{\text{MC/MP}}$ หรือ $\overline{\text{MCBL/MP}} = 0$)
- โหมดไมโครคอมพิวเตอร์ ($\overline{\text{MC/MP}}$ หรือ $\overline{\text{MCBL/MP}} = 1$)

การแทนข้อมูลที่ได้จะคล้ายๆ กัน ดังรูปที่ 2-4



รูปที่ 2-4 การแทนข้อมูลของ TMS320C31

2 1.3 การจัดการบัสภายใน

ส่วนใหญ่ TMS3320C3x จะมีสมรรถภาพสูงเพราะบัสภายใน และการแยกแบบขนานของบัสของโปรแกรม (PADDR และ PDATA), บัสของข้อมูล (DADDR1, DADDR2 และ DDATA) และบัสของการเข้าถึงหน่วยความจำโดยตรง (DMAADDR และ DMADATA) จะสามารถไปนำโปรแกรมมาแบบขนาน, การเข้าถึงข้อมูล และ การเข้าถึงหน่วยความจำโดยตรง โดยบัสเหล่านี้จะต่อกับพื้นที่ทั้งหมด (หน่วยความจำบนชิป, หน่วยความจำนอกชิป และอุปกรณ์ภายนอกบนชิป) รูปที่ 2-3 แสดงบัสภายในเหล่านี้ และการติดต่อกับบล็อกของหน่วยความจำบนชิปและนอกชิป

ตัวนับโปรแกรมต่ออยู่กับบัสตำแหน่งโปรแกรม 24 บิต (PADDR) ส่วนรีจิสเตอร์คำสั่ง (Instruction register : IR) ต่ออยู่กับบัสข้อมูลของโปรแกรม 32 บิต (PDATA) โดยที่บัสเหล่านี้สามารถไปนำเวิร์ดคำสั่งเดี่ยวมาได้ทุกๆ รอบของเครื่อง

บัสตำแหน่งของข้อมูล 24 บิต (DADDR1 และ DADDR2) และ บัสของข้อมูล 32 บิต (DDATA) จะสนับสนุนการเข้าถึงหน่วยความจำของข้อมูลทั้ง 2 ในทุกรอบของเครื่อง บัส DDATA จะเก็บข้อมูลให้หน่วยประมวลผลกลางผ่านไป บัส CPU1 และ CPU2 จากนั้นบัส CPU1 และ CPU2 จะเก็บตัวถูกดำเนินการของหน่วยความจำข้อมูล 2 ตัวไปให้ตัวคูณ, หน่วยคำนวณและตรรกะ และแฟ้มรีจิสเตอร์ทุกรอบเครื่อง เช่นเดียวกันกับ บัสของรีจิสเตอร์ REG1 และ REG2 ก็จะเก็บข้อมูล 2 ค่าจากแฟ้มรีจิสเตอร์ไปให้ตัวคูณ และหน่วยคำนวณและตรรกะทุกรอบของเครื่อง รูปที่ 2-2 แสดงบัสภายในไปยังส่วนของหน่วยประมวลผลกลาง

ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA controller) ถูกสนับสนุนโดยบัสของตำแหน่ง 24 บิต (DMAADDR) และ บัสของข้อมูล 32 บิต (DMADATA) บัสเหล่านี้จะอนุญาตให้การเข้าถึงหน่วยความจำโดยตรงเป็นแบบขนาน โดยใช้บัสของข้อมูล และบัสของโปรแกรม

2.1.4 อุปกรณ์รอบนอก (Peripherals)

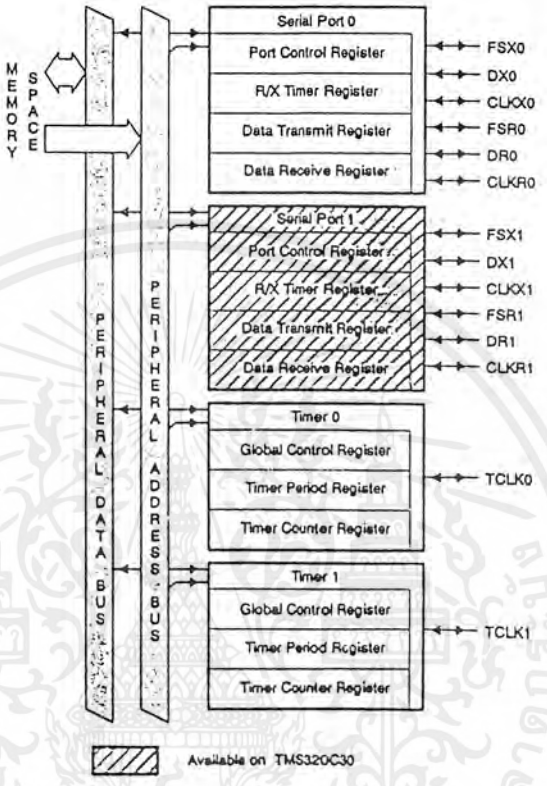
รีจิสเตอร์การแทนข้อมูลจะควบคุมอุปกรณ์ภายนอกของ TMS320C3x ทั้งหมดโดยใช้บัสของอุปกรณ์รอบนอกซึ่งเป็นบัสของข้อมูล 32 บิต และบัสของตำแหน่ง 24 บิต ที่จะยอมให้ติดต่อกับอุปกรณ์รอบนอกโดยตรง อุปกรณ์รอบนอกของ TMS320C31 ประกอบด้วยตัวจับเวลา (timer) 2 ตัว และพอร์ตอนุกรม (serial port) 1 ตัว รูปที่ 2-5 แสดงอุปกรณ์รอบนอกกับบัสและสัญญาณที่เกี่ยวข้อง

2.1.4.1 ตัวจับเวลา (Timer)

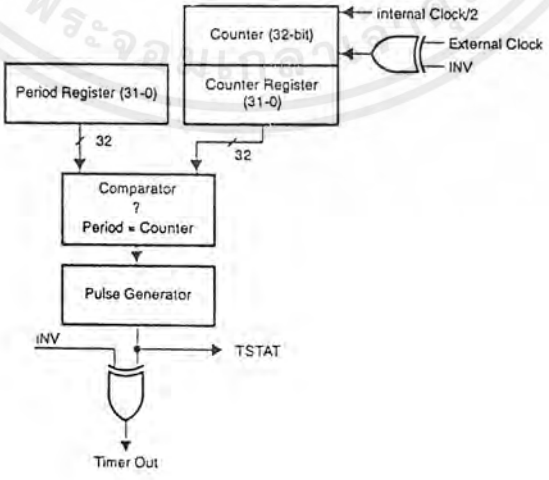
ตัวจับเวลา 2 ตัวของ TMS320C3x เป็นตัวจับเวลาอเนกประสงค์ (general-purpose timer) / ตัวนับเหตุการณ์ (event-counter) 32 บิตที่มีโหมดด้านสัญญาณ 2 ตัว และสัญญาณนาฬิกา (clocking) ภายในหรือภายนอก (ดูรูปที่ 2-6) สามารถใช้ตัวจับเวลา เพื่อส่งสัญญาณให้ TMS320C3x หรือโลกภายนอกเป็นช่วงๆ ตามที่กำหนด หรือนับเหตุการณ์ภายนอก

การใช้สัญญาณนาฬิกาภายในจะทำให้สามารถใช้ตัวจับเวลาเพื่อส่งสัญญาณให้ตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (A/D converter) ภายนอกไปเริ่มต้นการเปลี่ยนแปลง, หรือสามารถอินเทอร์รัพต์ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงของTMS320C3x ให้เริ่มต้นส่งย้ายข้อมูล การอินเทอร์รัพต์ตัวจับเวลาเป็นการอินเทอร์รัพต์ภายในแบบหนึ่ง ส่วนการใช้สัญญาณนาฬิกาภายนอกจะทำให้ตัวจับเวลาสามารถนับเหตุการณ์ภายนอกและอินเทอร์รัพต์หน่วยประมวลผลกลางหลังจากระบุจำนวนเหตุการณ์ ตัวจับเวลาแต่ละตัวจะมีขาอินพุต/เอาท์พุต (I/O pin) ซึ่งสามารถใช้เสมือนเป็นสัญญาณนาฬิกาอินพุต (input clock) ให้ตัวจับ

เวลา, สัญญาณนาฬิกาเอาท์พุต (output clock) หรือขาอินพุต/เอาท์พุตอเนกประสงค์ (general-purpose I/O pin)



รูปที่ 2-5 แสดงอุปกรณ์ภายนอกกับบัสและสัญญาณที่เกี่ยวข้อง



รูปที่ 2-6 บล็อกไดอะแกรมของตัวจับเวลา

การแทนที่ข้อมูลสำหรับตัวจับเวลา แสดงในรูปที่ 2-7

Register	Peripheral address	
	Timer 0	Timer 1
Timer Global Control(ดู table-1)	808020h	808030h
Reserved	808021h	808031h
Reserved	808022h	808032h
Reserved	808023h	808033h
Timer Counter	808024h	808034h
Reserved	808025h	808035h
Reserved	808026h	808036h
Reserved	808027h	808037h
Timer Period	808028h	808038h
Reserved	808029h	808039h
Reserved	80802Ah	80803Ah
Reserved	80802Bh	80803Bh
Reserved	80802Ch	80803Ch
Reserved	80802Dh	80803Dh
Reserved	80802Eh	80803Eh
Reserved	80802Fh	80803Fh

รูปที่ 2-7 ตำแหน่งการแทนข้อมูลของตัวจับเวลา

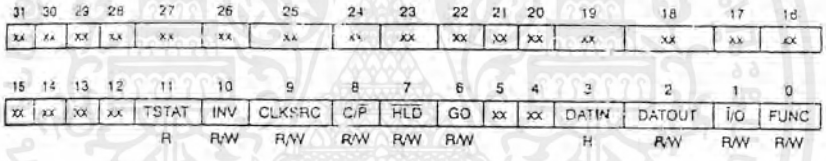
ในแต่ละตัวจับเวลา จะมีรีจิสเตอร์การแทนข้อมูล 3 ชนิด ดังนี้

- รีจิสเตอร์ควบคุมส่วนกลาง (Global-control register) กำหนดโหมดการทำงาน (operating mode) ของตัวจับเวลา, ตรวจสอบสถานะของตัวจับเวลา และ ควบคุมฟังก์ชันต่างๆ ของขาอินพุต/เอาท์พุตของตัวจับเวลา
- รีจิสเตอร์คาบเวลา (Period register) ระบุความถี่ทางสัญญาณของตัวจับเวลา
- รีจิสเตอร์ตัวนับ (Counter register) เก็บค่าในขณะปัจจุบันของตัวนับที่เพิ่มค่า (incrementing counter) ซึ่งสามารถเพิ่มค่าตัวจับเวลา โดยการนับขอบขาขึ้น

หรือขอบขาลงของตัวนับสัญญาณนาฬิกาอินพุต (input clock counter) ถ้าตัวนับเป็น 0 จะสามารถทำให้เกิดการอินเทอร์รัพต์ภายในได้เมื่อค่าของมันเท่ากับรีจิสเตอร์คาบเวลา โดยตัวสร้างสัญญาณพัลซ์ (pulse generator) จะสร้างสัญญาณนาฬิกาภายนอก 2 ตัวคือ สัญญาณพัลซ์ หรือ สัญญาณนาฬิกา

2.1.4.1.1 รีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา (Timer Global-Control Register)

เป็นรีจิสเตอร์ 32 บิต บรรจุบิตควบคุมพอร์ตและทั้งหมด สำหรับตัวจับเวลา ตาราง 2-2 อธิบายบิตของรีจิสเตอร์, ชื่อ และฟังก์ชัน โดย บิต 3-0 เป็น บิตควบคุมพอร์ต; บิต 11-6 เป็นบิตควบคุมส่วนกลางของตัวจับเวลา รูปที่ 2-8 แสดงรีจิสเตอร์ 32 บิต จะสังเกตได้ว่าเมื่อตั้งใหม่ (reset) บิตทุกบิตจะถูกตั้งค่าเป็น 0 ยกเว้น DATIN ที่ถูกตั้งค่าด้วยค่าที่ได้จาก TCLK



R = Read , W = Write , xx = reserved bit , read as 0

รูปที่ 2-8 รีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา

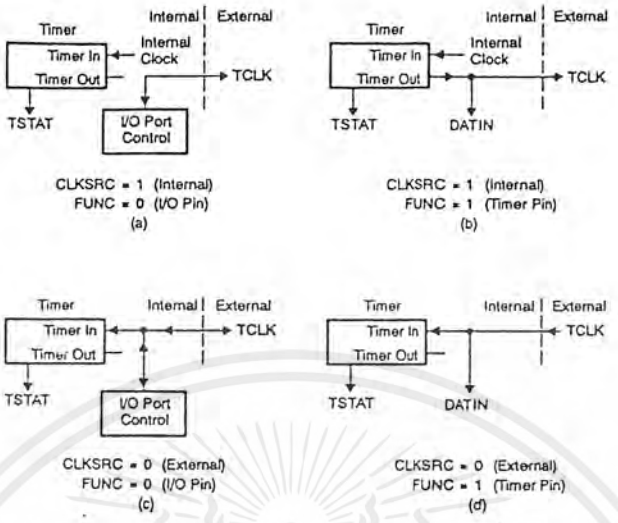
ตาราง 2-2 สรุปบิตของรีจิสเตอร์ควบคุมส่วนกลางของตัวจับเวลา

Bit	Name	Reset Value	Function
0	FUC	0	FUNC ควบคุมฟังก์ชันของ TCLK ถ้า FUNC = 0 แล้ว TCLK จะถูกกำหนดคล้ายพอร์ตอินพุต/เอาท์พุตแบบดิจิตอลอเนกประสงค์ ถ้า FUNC = 1 แล้ว TCLK จะถูกกำหนดคล้ายขาตัวจับเวลา (ดูรูปที่ 2-9 สำหรับอธิบายความสัมพันธ์ระหว่าง FUNC และ CLKSRC)
1	I/O	0	ถ้า FUNC = 0 และ CLKSRC = 0 แล้ว TCLK ถูกกำหนดคล้ายกับขาอินพุต/เอาท์พุตอเนกประสงค์ ในกรณีที่ I/O = 0 แล้ว TCLK ถูกกำหนดคล้ายขาอินพุต

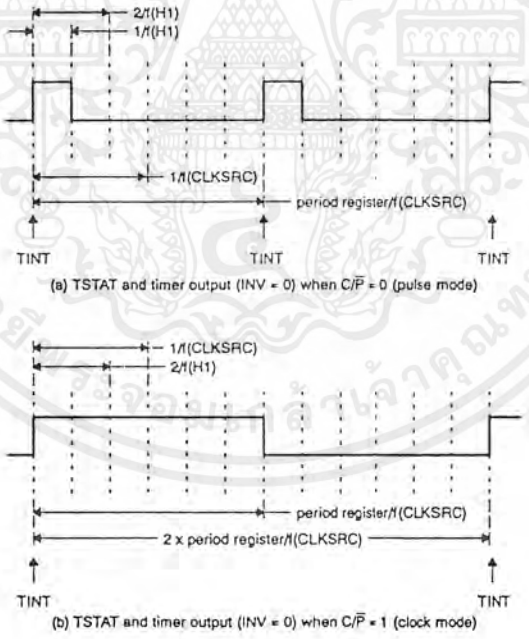
			อเนกประสงค์ ถ้า I/O = 1 แล้ว TCLK ถูกกำหนดเสมือนขาเอาท์พุต
2	DATOUT	0	DATOUT เป็นตัวกำหนด TCLK เมื่อ TMS320C3x อยู่ในโหมดพอร์ตอินพุต/เอาท์พุต สามารถใช้ DATOUT เช่นเดียวกับอินพุตไปสู่ตัวจับเวลา
3	DATIN	X^t	อินพุตข้อมูลบน TCLK หรือ DATOUT การเขียนไม่มีผลใดๆ
5-4	Reserved	0-0	สำรอง
6	GO	0	บิต GO ตั้งและเริ่มการทำงานของตัวจับเวลา-ตัวนับใหม่ เมื่อ GO = 1 และตัวจับเวลาไม่ถูกคงค่า (hold) ไว้ ตัวนับจะถูกทำให้เป็น 0 และเริ่มเพิ่มค่าในขอบขาขึ้นถัดไปของสัญญาณนาฬิกาอินพุตของตัวจับเวลา โดยที่บิต GO นี้จะถูกทำให้ว่าง (clear) บนขอบขาขึ้นเดิม ถ้า GO = 0 จะไม่มีผลต่อตัวจับเวลา
7	$\overline{\text{HLD}}$	0	ตัวนับจะคงค่าสัญญาณ เมื่อบิตนี้เป็น 0 โดยที่ตัวนับ จะไม่ทำงานและถูกคงค่าในสถานะปัจจุบัน ถ้าตัวจับเวลากำลังกำหนด TCLK แล้ว สถานะของ TCLK ถูกคงค่าด้วยตัวนับภายในที่ถูกหารด้วย 2 ถูกคงค่าด้วย ดังนั้นตัวนับจะสามารถดำเนินต่อไปได้เมื่อ $\overline{\text{HLD}}$ ถูกตั้งค่าเป็น 1 สามารถอ่านและแก้ไขรีจิสเตอร์ของตัวจับเวลา ในขณะที่ตัวจับเวลากำลังถูกคงค่าได้ แต่ $\overline{\text{RESET}}$ มีลำดับความสำคัญ (priority) มากกว่า $\overline{\text{HLD}}$ ตาราง 2-3 แสดงผลของการเขียน GO และ $\overline{\text{HLD}}$
8	$\overline{\text{C/P}}$	0	ควบคุมโหมดสัญญาณนาฬิกา/สัญญาณพัลซ์ เมื่อ $\overline{\text{C/P}}$ = 1 โหมดสัญญาณนาฬิกา จะถูกเลือกและสัญญาณแฟลกของ TSTAT และเอาท์พุตภายนอกจะมีค่ารอบการทำงาน (duty cycle) 50% เมื่อ $\overline{\text{C/P}}$ = 0 แฟลกสถานะและเอาท์พุตภายนอกจะถูกกระตุ้น (active) สำหรับ 1 รอบของ H1 ระหว่างแต่ละคาบเวลาของตัวจับเวลา (ดูรูปที่ 2-10)

9	CLKSRC	0	แหล่งกำเนิด (Source) เฉพาะของสัญญาณนาฬิกาของตัวจับเวลา เมื่อ CLKSRC = 1, สัญญาณนาฬิกาภายใน เกี่ยวข้องกับความถี่ที่เท่ากับ 1.5 ของความถี่ของ H1 ถูกใช้ในการเพิ่มของตัวนับ โดยที่บิต INV ไม่มีผลกับแหล่งกำเนิดสัญญาณนาฬิกาภายใน เมื่อ CLKSRC = 0 คุณสามารถใช้สัญญาณภายนอก จากขา TCLK เพื่อเพิ่มตัวนับ, สัญญาณนาฬิกาภายนอกจะถูกทำให้สอดคล้องกัน (synchronize) กับภายใน ดังนั้นการอนุญาตให้แหล่งกำเนิดสัญญาณนาฬิกาที่ไม่สอดคล้องจากภายนอกที่ซึ่งไม่เกินค่าสูงสุดที่กำหนด เป็นความถี่สัญญาณนาฬิกาภายนอกจะน้อยกว่า f(H1) / 2 (ดูรูปที่ 2-9 สำหรับความสัมพันธ์ FUNC และ CLKSRC)
10	INV	0	บิตควบคุมอินเวอร์เตอร์ (Inverter) ถ้าสัญญาณนาฬิกาภายนอกถูกใช้และ INV = 1 แล้วสัญญาณนาฬิกาภายนอกจะถูกเปลี่ยน (invert) เสมือนเป็นตัวนับ ถ้าเอาท์พุตของตัวกำเนิดสัญญาณพัลซ์ถูกกำหนดเส้นทางไปสู่ TCLK และ INV = 1 แล้วจะเอาท์พุตจะถูกเปลี่ยนก่อนถูกส่งไป TCLK ถ้า INV = 0 แล้วจะไม่มีการเปลี่ยนแสดงบนอินพุตหรือเอาท์พุตของตัวจับเวลา โดยที่บิต INV จะมีผลเมื่อ TCLK ถูกใช้ในโหมดพอร์ตอินพุต/เอาท์พุต
11	TSTAT	0	บิตแสดงสถานะของตัวจับเวลา โดยจะติดตามเอาท์พุตของขา TCLK ที่ไม่ถูกเปลี่ยน แฟลกนี้จะตั้งค่าอินเทอร์รัพต์ โดยการส่งผ่านจาก 0 เป็น 1 ,ในการเขียนไม่มีผล
31-12	Reserved	0-0	สำรอง

X^{t}=0 or 1



รูปที่ 2-9 โหมดของตัวจับเวลาที่กำหนดโดย CLKSRC และ FUNC



รูปที่ 2-10 ช่วงเวลาของตัวจับเวลา (Timer Timing)

อัตราของการส่งสัญญาณของตัวจับเวลากำหนดโดยความถี่ของสัญญาณนาฬิกาอินพุตของตัวจับเวลาและรีจิสเตอร์คาบเวลาของตัวจับเวลา ดังสมการข้างล่างโดยมีผลทั้งสัญญาณนาฬิกาของตัวจับเวลาภายในหรือภายนอกก็ได้

f (pulse mode) = f (timer clock) / period register

f (clock mode) = f (timer clock) / (2xperiod register)

ตาราง 2-3 แสดงผลลัพธ์ของการเขียนโดยใช้ระบุค่าของบิต GO และ $\overline{\text{HLD}}$ ในรีจิสเตอร์ควบคุมส่วนกลาง

ตาราง 2-3 ผลลัพธ์ของการเขียนโดยระบุค่าของ GO และ $\overline{\text{HLD}}$

GO	$\overline{\text{HLD}}$	RESULT
0	0	ทุกๆ การทำงานของตัวจับเวลาจะถูกคงค่าไว้ และไม่มีการแสดงค่าตั้งใหม่
0	1	ตัวจับเวลา เกิดจากสภาวะก่อนการเขียน
1	0	ทุกๆ การทำงานของตัวจับเวลาจะถูกคงค่า รวมถึงการทำให้เป็น 0 ของ ตัวนับ และบิต GO จะไม่ถูกทำให้ว่างจนกว่าตัวจับเวลาจะไม่ถูกคงค่าไว้
1	1	ตัวจับเวลา ตั้งค่าใหม่ และเริ่มการทำงาน

2.1.4.1.2 การอินเทอร์รัพต์ตัวจับเวลา (Timer interrupts)

การอินเทอร์รัพต์ตัวจับเวลา จะถูกสร้างเมื่อใดก็ตามที่บิต TSTAT ของรีจิสเตอร์ควบคุมตัวจับเวลาเปลี่ยนจาก 0 เป็น 1 โดยที่ความถี่ของการอินเทอร์รัพต์ จะขึ้นอยู่กับว่าจะตั้งค่าให้ตัวจับเวลาอยู่ในโหมดใด

- ในโหมดสัญญาณพัลซ์, ความถี่อินเทอร์รัพต์ จะเป็นตามสมการ

  f (interrupt) = f (timer clock) / period register

  โดยที่ f (interrupt) = timer frequency

  f (timer clock) = interrupt frequency

- ในโหมดสัญญาณนาฬิกา, ความถี่อินเทอร์รัพต์ จะเป็นตามสมการ

  f (interrupt) = f (timer clock) / (2xperiod register),

  โดยที่ f (interrupt) = timer frequency

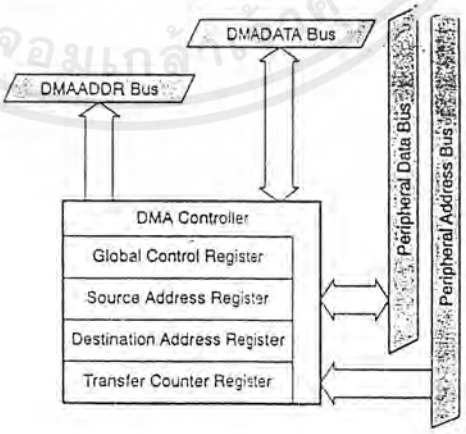
  f (timer clock) = interrupt frequency

2.1.4.2 พอร์ตอนุกรม (Serial Port)

พอร์ตอนุกรม 2 ทิศทางทั้ง 2 ตัวจะไม่ขึ้นต่อกัน โดยมีส่วนประกอบของรีจิสเตอร์ควบคุม ที่ควบคุมแต่ละพอร์ตเหมือนกัน แต่ละพอร์ตอนุกรมสามารถกำหนดให้ส่งผ่านข้อมูลต่อเวิร์ด 8, 16, 24 หรือ 32 บิต ส่วนสัญญาณนาฬิกาสำหรับแต่ละพอร์ตอนุกรม เกิดได้จากทั้งภายในและภายนอก ขาของพอร์ตอนุกรมสามารถกำหนดให้เป็นตัวจับเวลาได้ด้วย ส่วนโหมดพิเศษ : แฮนด์เชค (handshake) จะอนุญาตให้ TMS320C3x ทำการติดต่อพอร์ตอนุกรมได้อย่างไม่มีการผิดพลาด

2.1.4.3 การเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access : DMA)

ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรงบนชิป สามารถอ่านหรือเขียนลงตำแหน่งต่างๆ ใน แผนที่หน่วยความจำ โดยไม่เข้าไปแทรกแซงการทำงานของหน่วยประมวลผลกลาง, ดังนั้น TMS320C3x สามารถอินเตอร์เฟสกับหน่วยความจำภายนอกและอุปกรณ์ภายนอก โดยผ่านทางหน่วยประมวลผลหลักซึ่งข้อมูลนั้นจะไม่ถูกลดทอนลง ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง จะบรรจุตัวสร้างตำแหน่ง (address generator) ของตัวเอง, รีจิสเตอร์ต้นกำเนิดและจุดหมาย และตัวนับการส่งย้ายข้อมูล (transfer counter) บัสของตำแหน่งและบัสของข้อมูลของการเข้าถึงหน่วยความจำหน้าที่พิเศษ จะลดขนาดความขัดแย้งระหว่างหน่วยประมวลผลกลาง และตัวควบคุมการเข้าถึงหน่วยความจำให้น้อยลง การทำงานของการเข้าถึงหน่วยความจำประกอบด้วยบล็อก หรือเวิร์ดเดียว ส่งย้ายไปยังหรือไปจากหน่วยความจำ รูปที่ 2-11 แสดง ตัวควบคุมการเข้าถึงหน่วยความจำกับบัสที่สัมพันธ์กัน



รูปที่ 2-11 ตัวควบคุมการเข้าถึงหน่วยความจำ

2.2 โครงสร้างของ TMS320C3x DSP Starter Kit

2.2.1 โครงสร้างของ DSK

- TMS329C31 เป็น DSP อิงครรชนี
- คำสั่งมีเวลาครบรอบ (cycle time) 40 นาโนวินาที, 50 ล้านคำสั่งอิงครรชนีต่อวินาที (MFLOPS), 25 ล้านคำสั่งต่อวินาที (MIPS)
- อินเตอร์เฟสพอร์ตเครื่องพิมพ์แบบขนาน (parallel printer port interface) ที่เป็นมาตรฐานหรือพัฒนาแล้ว เพื่อต่อกับเครื่องคอมพิวเตอร์แม่ (host PC^{TW}) และ TMS320C31 ติดต่อกับโปรแกรมของเครื่องคอมพิวเตอร์ได้
- รับข้อมูลอนาลอกโดยผ่านวงจรอินเตอร์เฟสแบบอนาลอก (AIC) TLC32040
  - อัตราตัวแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (ADC) และตัวแปลงสัญญาณสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (DAC) สามารถเปลี่ยนแปลงพิสัยแบบพลวัตได้ 14 บิตใน 20000 ตัวอย่างต่อวินาที
  - มีฟิลเตอร์สร้างเอาท์พุตใหม่ และสามารถบายพาส (bypass) ได้, ตัวเก็บประจุแบบสวิตช์ (switched-capacitor) ป้องกันสัญญาณรบกวน เป็นอินพุตฟิลเตอร์
- ใช้คอนเนคเตอร์ปลั๊กแบบ RCA มาตรฐาน สำหรับอนาลอกอินพุตและเอาท์พุต
- คอนเนคเตอร์อีมูเลเตอร์ XDS510
- คอนเนคเตอร์เพิ่มเติมที่กำหนดขา TMS320C31 ทั้งหมดสำหรับใช้กับบอร์ด DSK ตัวลูก

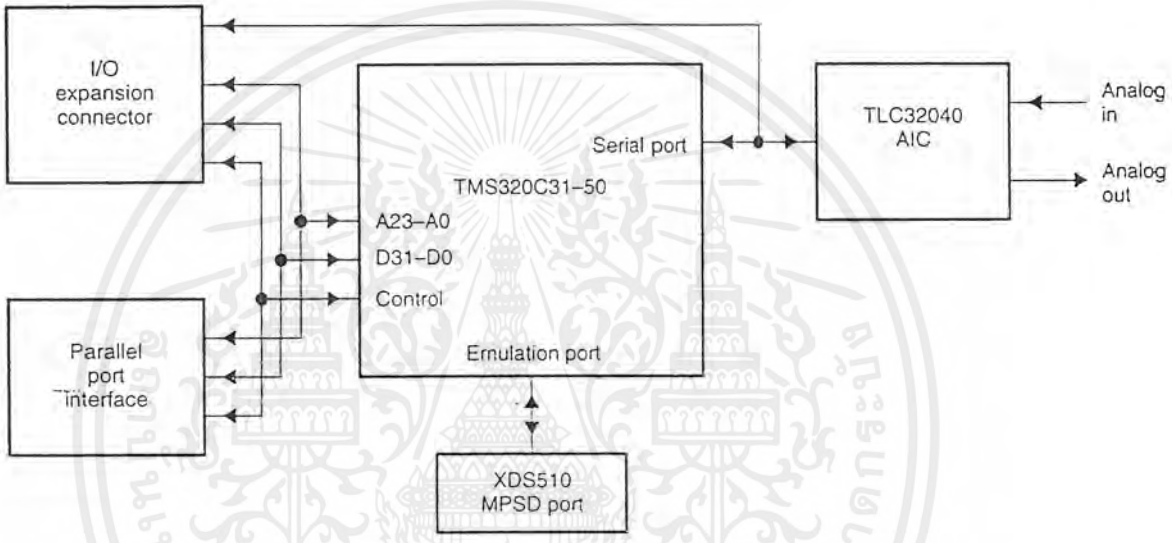
2.2.2 ภาพรวมของ DSK

รูปที่ 2-12 แสดงบล็อกไดอะแกรมของฮาร์ดแวร์ TMS320C3x DSK, รูปที่ 2-13 แสดง ส่วนประกอบพื้นฐานของ DSK ดังนี้

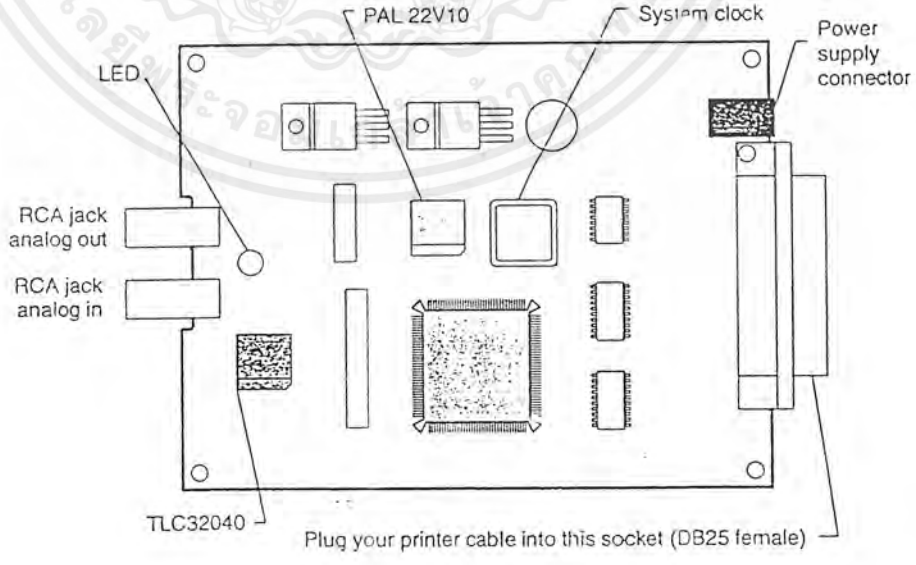
- DSP รุ่น TMS320C31
- วงจรอินเตอร์เฟสแบบอนาลอก TLC32040
- คอนเนคเตอร์เพิ่มเติม
- สัญญาณนาฬิกาของระบบ (System clock)
- อินเตอร์เฟสพอร์ตของเครื่องพิมพ์แบบขนาน
- ไฟแสดงผลสามสี (Tri-color LED)

เส้นทางสัญญาณทั้งหมดของ TMS320C3x จะไปยังคอนเนคเตอร์เพิ่มเติม โดยคอนเนคเตอร์เพิ่มเติมจะรวมทั้ง เฮดเดอร์ (header) 32 ขา 4 ตัว , บล็อกจัมเพอร์ (jumper block) 11 ขา 1 ตัว และเฮดเดอร์ XDS510 12 ขา

วงจรอินเตอร์เฟสแบบอนาลอก TLC32040 อินเตอร์เฟสกับพอร์ตอนุกรมของ TMS320C3x โดยบล็อกจัมเพอร์จะอนุญาตให้เคลื่อนย้ายการติดต่อไปตามพอร์ตอนุกรมเพื่อไปยังการ์ด DSK ตัวลูกที่จะจ่ายไฟ ใช้ คอนเนคเตอร์ RCA 2 ตัวเป็นอนาลอกอินพุตและเอาท์พุตของบอร์ด



รูปที่ 2-12 บล็อกไดอะแกรมของ TMS320C3x



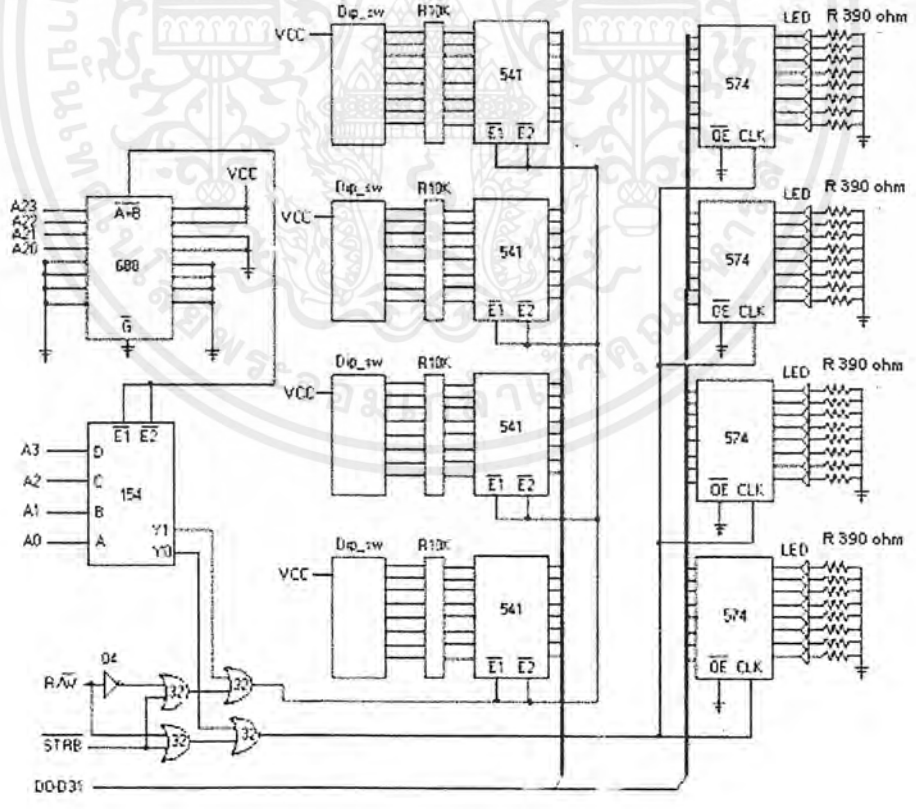
รูปที่ 2-13 แสดงส่วนประกอบพื้นฐานของ DSK

บทที่ 3
การออกแบบการทดลอง

ในการออกแบบการทดลองได้แยกการออกแบบเป็น 2 ส่วนคือ ทางด้านฮาร์ดแวร์ และซอฟท์แวร์ เพื่อทำการทดลองเกี่ยวกับการติดต่อกันระหว่างเครื่องคอมพิวเตอร์, บอร์ด DSP และบอร์ดอินเตอร์เฟสสำหรับรับค่าและแสดงผลแบบดิจิตอล ซึ่งทางด้านฮาร์ดแวร์ จะกล่าวถึงการสร้างบอร์ดอินเตอร์เฟส และการใช้บอร์ด DSP ส่วนทางด้านซอฟท์แวร์ จะอธิบายเกี่ยวกับคำสั่งที่สำคัญและแนวทางการเขียนโปรแกรม

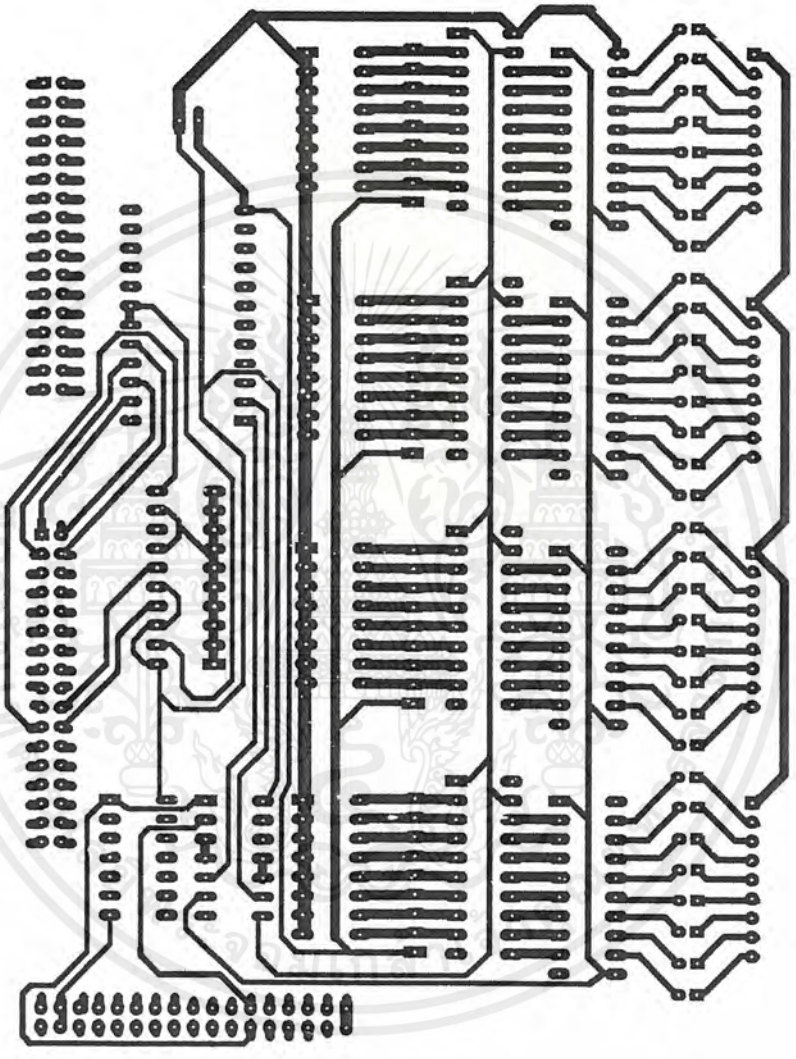
3.1 ฮาร์ดแวร์ (Hardware)

ในการทดลองนี้เราได้สร้างฮาร์ดแวร์เพื่อทำการแสดงผล โดยจัดทำหลอดไฟ LED ในการแสดงผล และใช้ดิพสวิตช์เป็นตัวส่งค่าเข้าไปประมวลผลใน DSP ดังแสดงในรูปที่ 3-1

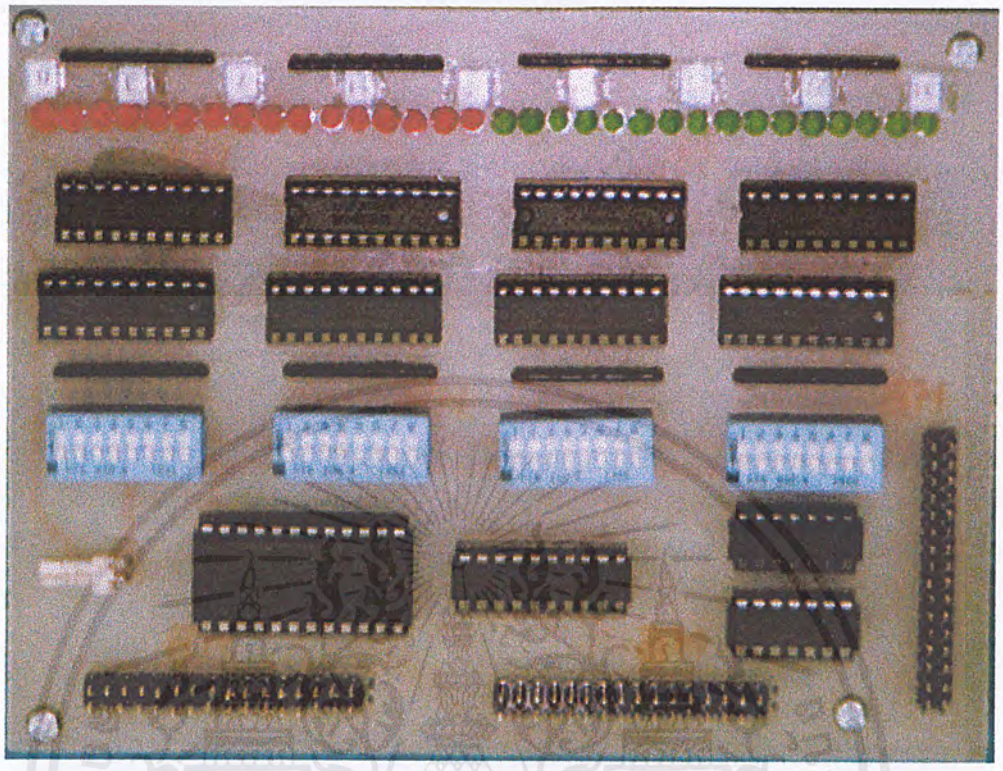


รูปที่ 3-1 แสดงวงจรรับค่าและแสดงผล

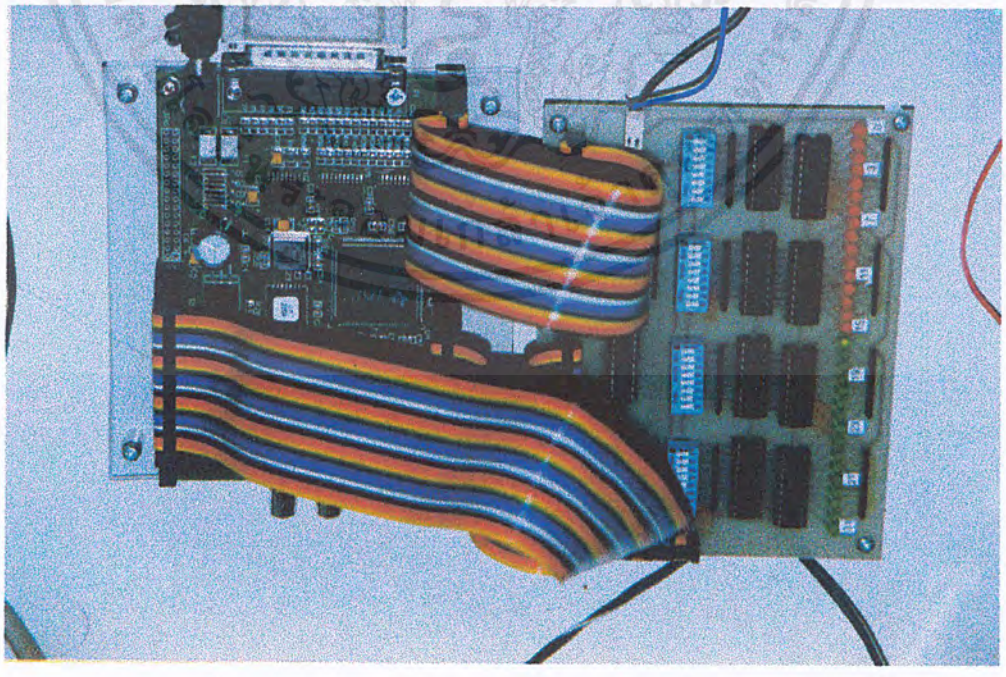
ในการทำบอร์ดอินเตอร์เฟสนี้ได้ทำทั้งแบบเชื่อม (wire) สายเอง และแบบเขียนลายวงจรโดยใช้โปรแกรมโพรเทล (Protel) ขึ้นมา แต่เพื่อความสะดวกจะเลือกใช้แบบที่ 2 ในการสร้างบอร์ด ดังรูปที่ 3-2



รูปที่ 3-2 แสดงลายวงจรของบอร์ดอินเตอร์เฟส



รูปที่ 3-3 บอร์ดอินเตอร์เฟส



รูปที่ 3-4 แสดงการอินเตอร์เฟสระหว่างบอร์ด DSP กับบอร์ดอินเตอร์เฟส

จากรูปที่ 3-1 ซึ่งแสดงวงจรรับค่าและแสดงผลได้มีการติดต่อกับพอร์ทของ DSP อยู่ 3 พอร์ทคือ

พอร์ทที่ 1 พอร์ทข้อมูล (Data Port) ทำหน้าที่รับและส่งข้อมูลระหว่างบอร์ด DSP และบอร์ดอินเตอร์เฟสโดยมีชิปเบอร์ 541 (74LS541) เป็นบัฟเฟอร์ (buffer) สำหรับอินพุตของบอร์ดอินเตอร์เฟส และชิปเบอร์ 574 (74LS574) เป็นดีฟลิบฟล็อบ (D-flipflop) ทำหน้าที่เป็นบัฟเฟอร์สำหรับเอาท์พุตของบอร์ดอินเตอร์เฟส โดยข้อมูลทั้งด้านอินพุตและเอาท์พุตมีขนาด 32 บิต

พอร์ทที่ 2 คือ พอร์ทตำแหน่ง (Address Port) มีสัญญาณ A0 ถึง A3 และ A20 ถึง A23 ซึ่งเป็นสัญญาณที่กำหนดค่าตำแหน่งของบอร์ดอินเตอร์เฟสในการเขียนโปรแกรมโดยบอร์ด DSP จะต่ออยู่กับชิปเบอร์ 688 (74LS688) ทำหน้าที่เป็นตัวเปรียบเทียบค่า (comparator) โดยเปรียบเทียบค่าที่ได้รับเข้ามากับค่าที่กำหนดไว้ซึ่งกำหนดตำแหน่งโดยทางฮาร์ดแวร์ มีชิปเบอร์ 154 (74LS154) เป็นตัวถอดรหัส (decoder)

พอร์ทที่ 3 คือพอร์ทควบคุม (Control Port) มีอยู่ 2 สัญญาณคือ สัญญาณ $R/\overline{W}$ และ $\overline{STRB}$ ซึ่งเป็นสัญญาณที่ส่งมาจากบอร์ด DSP เพื่อทำการควบคุมการส่งหรือรับค่า โดย $R/\overline{W}$ ต่ออยู่กับชิปเบอร์ 04 (74LS04) ซึ่งทำหน้าที่เป็นอินเวอร์เตอร์ (invertor) และชิปเบอร์ 32 (74LS32) เป็นออร์เกตท์ (OR-gate) ส่วน $\overline{STRB}$ ต่ออยู่กับออร์เกตท์ 2 ตัว

การทำงานของบอร์ดอินเตอร์เฟสนี้จะเริ่มต้นเมื่อ บอร์ด DSP ได้ส่งค่าตำแหน่งเข้าที่ชิปเบอร์ 688 ซึ่งในการทดลองนี้ได้ทำการกำหนดค่าตำแหน่งขึ้นเองซึ่งจะเกี่ยวข้องในการเขียนโปรแกรม โดยในที่นี้จะต้องเป็นค่า XXCXXXXX (X เป็นเลขฐาน 16 ซึ่งจะเป็นอะไรก็ได้) เท่านั้นจึงจะสามารถทำให้ 688 ทำงานได้ ซึ่งก็คือ A20 และ A21 ต้องเป็น 0 ส่วน A22 และ A23 ต้องเป็น 1 ดังนั้น 688 จะส่งสัญญาณให้กับ 154 ทำงานเมื่อสัญญาณ A3, A2, A1 และ A0 เป็น 0 ทั้งหมด 154 จะทำให้ค่า Y0 เป็น 0 แต่หากเปลี่ยนให้ A0 เป็น 1 จะทำให้ค่า Y1 เป็น 0 แสดงได้ดังนี้

A23	A22	A21	A20	A3	A2	A1	A0	ค่าตำแหน่ง	Y0	Y1
1	1	0	0	0	0	0	0	00C00000	0	1
1	1	0	0	0	0	0	1	00C00001	1	0

ในบอร์ดอินเตอร์เฟสนี้ จะมีการทำงานระหว่างการรับค่าและส่งค่าออกมาคนละเวลา ในขณะที่ทำการรับค่าจะไม่สามารถส่งค่าได้ และในขณะส่งค่าก็จะไม่สามารถรับค่าได้เช่นเดียวกัน แต่ในการเขียนโปรแกรมจะเป็นว่าทำงานได้พร้อมกันนั้น เนื่องจากบอร์ด DSP มีการทำงานที่เร็วมากจึงเห็นเป็นเช่นนั้น เราจะใช้สัญญาณ $R/\overline{W}$, $\overline{STRB}$, Y0 และ Y1 ในการควบคุมว่าเมื่อใดที่

จะรับค่าอินพุตเข้ามา หรือเมื่อใดที่จะส่งค่าออกไปแสดงทาง LED ซึ่งต่ออยู่กับอาร์เพ็ค (R-pack) ขนาด 390 กิโลโอห์มเพื่อใช้ในการพูลดาวน์ (pull down) สัญญาณเพื่อจะได้ค่าทางดิจิตอลออกมาเป็น 1 และ 0 อย่างแท้จริง เมื่อค่า $R/\overline{W}$, $\overline{STRB}$ และ Y0 มีค่าเท่ากับ 0 ก็แสดงว่าช่วงนั้นกำลังทำการส่งค่าจากพอร์ทข้อมูล 32 บิตของบอร์ด DSP ไปสู่ LED โดยที่ LED จะติดตามค่าที่ได้รับจากพอร์ทข้อมูล และจะคงค่าอยู่อย่างนั้นจนกระทั่งเมื่อค่า $\overline{STRB}$ = 1 และจะนำค่าใหม่จากพอร์ทข้อมูลมาแสดงผล แต่ถ้า $\overline{STRB}$ และ Y1 เท่ากับ 0 แต่ $R/\overline{W}$ เป็น 1 แสดงว่ากำลังทำการรับค่าจากดิพสวิตช์ซึ่งต่ออยู่กับอาร์เพ็คขนาด 10 กิโลโอห์มเพื่อใช้สำหรับทำการพูลดาวน์ โดยรับค่าผ่านเข้ามาทางพอร์ทข้อมูล 32 บิต ของบอร์ด DSP โดยใช้พอร์ทข้อมูลร่วมกัน เมื่อทำการปิดสวิตช์ จะเป็นการจ่ายไฟให้โดยจะให้ค่าออกมาเป็น 1 และเมื่อเปิดสวิตช์ จะให้ค่าออกมาเป็น 0 ซึ่งค่าต่างๆ จะถูกส่งไปประมวลผลที่บอร์ด DSP

ในการกำหนดสถานะของอินพุท ถ้ากำหนดให้เข้ามา 1 แต่ให้แสดงออกมาเป็น 0 จะทำให้เกิดความยุ่งยากในการคำนวณค่าต่างๆ ภายหลัง ดังนั้นจึงควรทำให้การกำหนดค่าให้เข้ามาเป็น 0 และให้แสดงผลออกมาเป็น 0 จะทำให้เขียนโปรแกรมได้ง่ายขึ้น

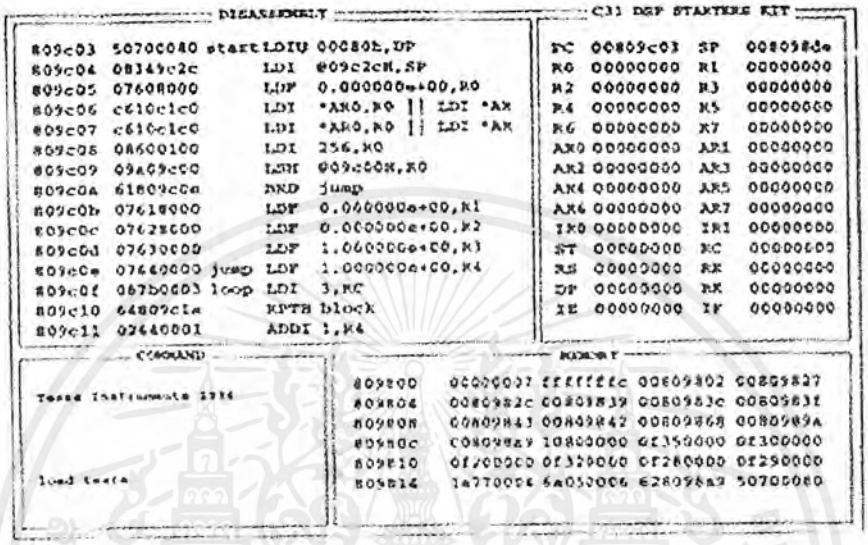
3.1.1 การใช้ DSP ในการควบคุมแบบดิจิตอล

บนบอร์ด DSP มีช่องทางให้ติดต่อกับภายนอกโดยผ่านทางพอร์ท JP2, JP3, JP5 และ JP6 ในการทดลองนี้ใช้เพียง JP2, JP5 และ JP6 โดยพอร์ท JP2 เป็นพอร์ทควบคุม ใช้สำหรับควบคุมสัญญาณ $R/\overline{W}$ และ $\overline{STRB}$ ส่วนพอร์ท JP3 และ JP5 เป็นพอร์ทตำแหน่งและพอร์ทข้อมูล ตามลำดับ (ดูรายละเอียดได้ใน TMS320C3x DSP Starter Kit User's Guide) แล้วจึงเขียนโปรแกรมควบคุมเพื่อส่งค่าออกมาตามพอร์ทต่างๆ ตามต้องการ

ในการติดต่อกันระหว่างเครื่องคอมพิวเตอร์และบอร์ด DSP เราจะติดต่อกันผ่านพอร์ทของเครื่องพิมพ์ของคอมพิวเตอร์โดยต่อกับคอนเนคเตอร์ DB25 บนบอร์ด DSP และจะทำการโหลดโปรแกรมจากคอมพิวเตอร์เข้าสู่ DSP ผ่านทางนี้ ส่วนไฟเลี้ยงที่ใช้กับบอร์ดจะใช้ไฟ 7-12 Vdc หรือ 6-9 Vac ต่อเข้าไปที่คอนเนคเตอร์ของแหล่งจ่ายกำลังไฟฟ้า (power supply connector) บนบอร์ด

การติดต่อกันนี้จำเป็นที่จะต้องใช้ซอฟท์แวร์ DSK ซึ่งเป็นซอฟท์แวร์ที่ให้มาพร้อมกับตัวบอร์ดมาเป็นตัวเชื่อมต่อ เพื่อให้เครื่องคอมพิวเตอร์รู้จักบอร์ด DSP โดยที่คอมพิวเตอร์จะต้องถูกตั้งค่า config.sys และ autoexec.bat ซึ่งเป็นโปรแกรมภายในของคอมพิวเตอร์ให้รองรับกับการใช้ซอฟท์แวร์ DSK นี้ (ดูรายละเอียดได้ใน TMS320C3x DSP Starter Kit User's Guide) เมื่อต้องการ

เชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับบอร์ด DSP เมื่อไรก็ให้เรียกคำสั่ง dsk3d เมื่อเรียกแล้วปรากฏภาพดังรูปที่ 3-5 ซึ่งจะสามารถทำการติดต่อได้



รูปที่ 3-5 หน้าต่างแสดงการติดต่อ

3.1.2 วิธีการติดตั้งโปรแกรมลงในเครื่องคอมพิวเตอร์

มีขั้นตอนในการติดตั้งดังต่อไปนี้

1. ปิดเครื่องคอมพิวเตอร์
2. ต่อสายของเครื่องพิมพ์แบบขนาน (parallel printer) เข้ากับพอร์ทของเครื่องพิมพ์ และต่อสายที่เป็นด้านตัวเมียของสายเข้ากับคอนเนคเตอร์ DB25 บนบอร์ด DSP
3. ต่อไฟเลี้ยงขนาด 7-12 Vdc หรือ 6-9 Vac เข้าที่ แหล่งจ่ายกำลังไฟฟ้าบนบอร์ด DSP
4. เปิดเครื่องแล้วสังเกต LED จะสว่างเป็นสีเขียวสลับแดง
5. เตรียมซอฟท์แวร์ที่ได้พร้อมกับบอร์ด DSP ใส่ลงไปในไดเรกทอรี (directory) ที่ชื่อว่า dsktools ซึ่งไดเรกทอรีนี้จะประกอบด้วยตัวแปลแอสเซมบลี (assembler) DSK และโปรแกรมตรวจสอบจุดบกพร่อง (debugger)
6. ทำการสำเนา (copy) แฟ้มที่ได้มาลงใน ไดเรกทอรี
7. แก้ไขแฟ้ม config.sys โดยเพิ่มเติมเข้าไปว่า FILES=20 เพื่อให้สามารถทำงานได้หลายหน้าต่างและสะดวกรวดเร็วมากขึ้น
8. แก้ไขแฟ้ม autoexec.bat โดยเพิ่ม PATH=C:\dsktools

9. เมื่อทำตามข้างบนเรียบร้อยแล้วพิมพ์คำสั่ง dsk3d จะปรากฏหน้าต่างดังรูปที่ 3-5 ซึ่งจะสามารถติดต่อกันได้ระหว่างบอร์ด DSP และ เครื่องคอมพิวเตอร์

ในการติดตั้งโปรแกรมซึ่งใช้ในการโหลดโปรแกรมจากเครื่องคอมพิวเตอร์ เข้าสู่ DSP จำเป็นต้องใช้เครื่องคอมพิวเตอร์ที่เราสามารถแก้ไข config.sys และ autoexec.bat ในคอมพิวเตอร์เครื่องนั้นได้ ดังนั้นจึงไม่สามารถใช้กับคอมพิวเตอร์ที่เป็นเครือข่าย (network) ได้

3.2 ซอฟท์แวร์ (Software)

เมื่อทำการติดต่อได้แล้วก็จะทำการโหลดโปรแกรมจากเครื่องคอมพิวเตอร์เข้าสู่บอร์ด DSP โดยที่โปรแกรมดังกล่าวจะต้องถูกแปลงให้บอร์ด DSP รู้จักเสียก่อนโดยใช้คำสั่ง dsk3a [namefile] เป็นตัวแปลงโปรแกรมแล้วจึงทำการโหลดโปรแกรมด้วยหน้าต่างที่เห็นในภาพด้านบน ขณะที่ dsk3a กำลังทำการแปลงนั้นมันยังสามารถเป็นตัวตรวจสอบจุดบกพร่อง หากเกิดจุดบกพร่องขึ้นในโปรแกรมได้

ในการเขียนโปรแกรมเราจำเป็นที่จะต้องเรียนรู้คำสั่งของบอร์ด DSP เสียก่อนเนื่องจากเป็นภาษาแอสเซมบลี (assembly) แบบเฉพาะตัวซึ่งเป็นคำสั่งพิเศษบางครั้งยังมีคำสั่งแบบขนาน เพิ่มเติมขึ้นมาอีกด้วย ส่วนแนวทางในการเขียนโปรแกรมก็เสมือนกับการเขียนโปรแกรมโดยทั่วไปคือมีการเขียนผังงาน (flowchart) ออกมาเป็นลักษณะลูป (loop) การทำงานแล้วจึงนำมาเขียนเป็นโปรแกรม ดังนั้นจะขออธิบายคำสั่งและยกตัวอย่างในการเขียนโปรแกรมง่ายๆ ดังต่อไปนี้

3.2.1 ตัวอย่างคำสั่ง (ดูคำสั่งทั้งหมดได้ใน TMS320C3x UESER'S GUIDE)

คำสั่งทั้งหมดมีอยู่ 113 คำสั่งซึ่งจะแบ่งเป็นกลุ่มได้ดังนี้

1. โหลด (LOAD) และ เก็บ (STORE) ค่า
2. การคำนวณแบบ 2 ตัวถูกดำเนินการ
3. การคำนวณแบบ 3 ตัวถูกดำเนินการ
4. โปรแกรมที่ใช้ในการควบคุม
5. การทำการอินเตอร์ล็อก (interlock)
6. คำสั่งแบบขนาน (parallel)

3.2.1.1 คำสั่ง LOAD และ STORE ค่า

เป็นคำสั่งที่ใช้สำหรับนำค่าออกมาจากรีจิสเตอร์หรืออื่นๆ และนำค่าไปเก็บไว้ในรีจิสเตอร์หรืออื่นๆ ตามที่ค่าตำแหน่งได้กำหนดไว้

คำสั่ง	คำอธิบาย
LDI	โหลดค่าจำนวนเต็ม
LDF	โหลดค่าอิงดรรชนี
LDP	โหลดค่าตัวชี้เพจของข้อมูล
POP	ดึงค่าจำนวนเต็มออกมาจาก สแตก

คำสั่ง	คำอธิบาย
POPF	ดึงค่าอิงดรรชนีออกจากสแตก
PUSH	เก็บค่าจำนวนเต็มเข้าในสแตก
STI	นำค่าจำนวนเต็มไปเก็บ
STF	นำค่าอิงดรรชนีไปเก็บ

3.2.1.2 คำสั่งการคำนวณแบบ 2 ตัวถูกดำเนินการ

เป็นคำสั่งที่ใช้ในการคำนวณทางคณิตศาสตร์แบบ 2 ตัวถูกดำเนินการ ซึ่งเมื่อทำการคำนวณเสร็จก็จะนำค่าที่คำนวณแล้วไปเก็บไว้ที่รีจิสเตอร์ที่ถูกกระทำ

คำสั่ง	คำอธิบาย
ABSI	ทำให้ค่าจำนวนเต็มเป็นค่าบวก
ABSF	ทำให้ค่าอิงดรรชนีเป็นค่าบวก
ADDI	บวกค่าจำนวนเต็ม
ADDF	บวกค่าอิงดรรชนี
SUBI	ลบค่าจำนวนเต็ม
SUBF	ลบค่าอิงดรรชนี
MPYI	คูณค่าจำนวนเต็ม
MPYF	คูณค่าอิงดรรชนี
FIX	เปลี่ยนค่าจากอิงดรรชนีเป็นจำนวนเต็ม

คำสั่ง	คำอธิบาย
OR	เป็นการ or กันของเลข 2 จำนวน
AND	เป็นการ and กันของเลข 2 จำนวน
ROL	หมุนบิตไปทางซ้าย
ROLC	หมุนบิตไปทางซ้ายด้วยบิตทด
ROR	หมุนบิตไปทางขวา
RORC	หมุนบิตไปทางขวาด้วยบิตทด
CMPI	เปรียบเทียบค่ากันแบบจำนวนเต็ม
CMPF	เปรียบเทียบค่าแบบอิงดรรชนี
FLOAT	เปลี่ยนค่าจากแบบจำนวนเต็มเป็นอิงดรรชนี

3.2.1.3 การคำนวณแบบ 3 ตัวถูกดำเนินการ

เป็นคำสั่งที่ใช้ในการคำนวณทางคณิตศาสตร์แบบ 3 ตัวถูกดำเนินการ ซึ่งเมื่อทำการคำนวณเสร็จก็จะนำค่าที่คำนวณแล้วไปเก็บไว้ที่รีจิสเตอร์ที่ใดก็ได้ที่ระบุไว้ในคำสั่ง

คำสั่ง	คำอธิบาย
ADDI3	บวกค่าแบบจำนวนเต็ม
ADDF3	บวกค่าแบบอิงดรรชนี
SUBI3	ลบค่าแบบจำนวนเต็ม
SUBF3	ลบค่าแบบอิงดรรชนี
MPYI3	คูณค่าแบบจำนวนเต็ม

คำสั่ง	คำอธิบาย
MPYF3	คูณค่าแบบอิงดรรชนี
OR3	เป็นการ or กันของเลข 2 จำนวน
AND3	เป็นการ and กันของเลข 2 จำนวน
CMPI3	เปรียบเทียบค่ากันแบบจำนวนเต็ม
CMPF3	เปรียบเทียบค่ากันแบบอิงดรรชนี

3.2.1.4 คำสั่งโปรแกรมที่ใช้ในการควบคุม

เป็นคำสั่งที่ใช้ในการกระโดดข้ามคำสั่งหรือเป็นการสั่งให้ทำซ้ำหลายๆครั้ง

คำสั่ง	คำอธิบาย
Bcond	กระโดดไปที่ลูปคำสั่งอื่นตามสถานการณ์ที่ถูกกำหนด
CALL	กระโดดไปทำคำสั่งย่อย

คำสั่ง	คำอธิบาย
RETS	ย้อนกลับหลังจากทำคำสั่งย่อยเสร็จแล้ว
RETI	ย้อนกลับหลังจากทำตามคำสั่งเมื่อเกิดการอินเทอร์รัพต์

3.2.1.5 คำสั่งที่ใช้ในการอินเตอร์ล็อก (interlock)

ใช้สำหรับการป้องกันความสับสนในการใช้ไมโครโปรเซสเซอร์หลายตัวและการใช้สัญญาณต่างๆ จากภายนอก

คำสั่ง	คำอธิบาย
LDII	โหลดค่าจำนวนเต็มแบบมีการอินเตอร์ล็อก
LDFI	โหลดค่าอิงดรรชนีแบบมีการ อินเตอร์ล็อก

คำสั่ง	คำอธิบาย
STII	นำค่าจำนวนเต็มไปเก็บแบบมีการอินเตอร์ล็อก
STFI	นำค่าอิงดรรชนีไปเก็บแบบมีการอินเตอร์ล็อก

3.2.1.6 คำสั่งแบบขนาน (parallel)

เป็นคำสั่งที่ใช้เหมือน 2 ตัวถูกดำเนินการ และ 3 ตัวถูกดำเนินการ แต่จะทำได้เร็วกว่าโดยการใช้เพียง 1 คำสั่งเทียบเท่ากับ 2 คำสั่ง

คำสั่ง	คำอธิบาย
ADDI || STI	เป็นคำสั่งที่บวกค่าจำนวนเต็ม แล้วจึงนำค่าไปเก็บไว้ในที่ที่คำสั่งได้ระบุไว้
LDI || STI	เป็นคำสั่งที่โหลดค่าจำนวนเต็ม มาจากตัวที่ระบุไว้ในคำสั่งแล้วนำค่าที่ได้ไปเก็บไว้ในที่ที่คำสั่งได้ระบุไว้

3.2.2 ตัวอย่างโปรแกรม

ตัวอย่างโปรแกรมจะอยู่ในโปรแกรมที่ 2 ของบทที่ 4 ในภาคผนวกโดยใช้คำสั่งด้านบนจะอธิบายเกี่ยวกับการใช้อินเตอร์รัพต์ โดยที่ตอนเริ่มต้นของโปรแกรมจะเป็นการตั้งค่าของรีจิสเตอร์ต่างๆ ที่เกี่ยวข้อง ซึ่งมีดังนี้ (ดูรายละเอียดได้ในบทที่ 8 ของ TMS320C3x User's Guide)

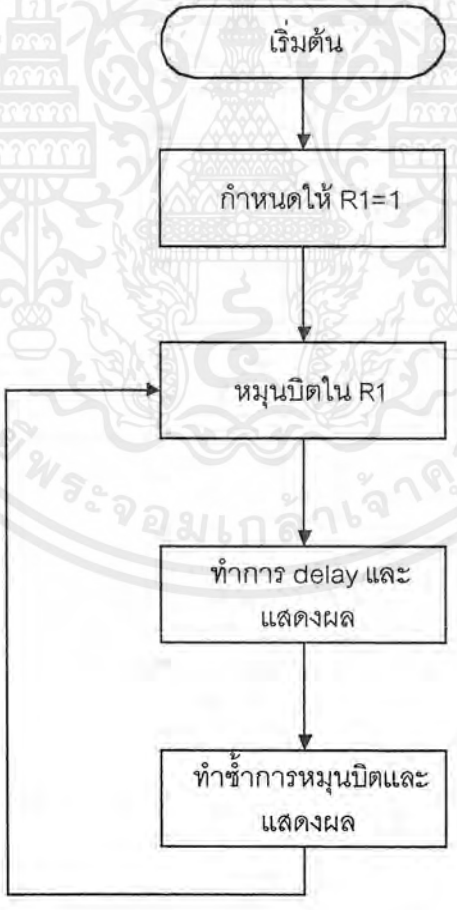
```
TIM_GCTRL0  .set 0x808020  เป็นการกำหนดค่าตำแหน่งของตัวควบคุมทั้งหมดของตัวจับ
                           เวลาของตัวจับเวลา 0 (Timer 0)
TIM_COUNT0  .set 0x808024  เป็นการกำหนดค่าตำแหน่งของตัวนับของตัวจับเวลา (Timer
                           Counter) ของตัวจับเวลา 0
TIM_PER0    .set 0x808028  เป็นการกำหนดค่าตำแหน่งของคาบเวลาของตัวจับเวลา
                           (Timer Period) ของตัวจับเวลา 0
USRIOW      .set 0xc00000  เป็นการกำหนดค่าตำแหน่งโดยตัวบอร์ดภายนอก
USRIOR      .set 0xc00001  เป็นการกำหนดค่าตำแหน่งโดยตัวบอร์ดภายนอก
```

เริ่มต้นโปรแกรมหลัก (Main Program) โดยการกำหนดค่าให้กับรีจิสเตอร์ 1 (R1) ให้เป็น 1 แล้วส่งออกไปแสดง ต่อมาทำการกำหนดค่าให้กับตัวควบคุมทั้งหมดของตัวจับเวลาให้อยู่ในสภาวะโหมดสัญญาณนาฬิกา และกำหนดค่าของคาบเวลาของตัวจับเวลา ให้มีค่าเท่ากับ 1 วินาที แล้วจึงเริ่มลูปการทำงาน โดยทำการส่งค่าที่อยู่ใน R1 ออกไปแสดง แล้วรอจนกระทั่งเกิดการอินเทอร์รัพต์ เมื่อมีการอินเทอร์รัพต์เกิดขึ้นจะทำการหมุนบิตใน R1 ทีละบิต เสร็จแล้วจะวนกลับไปเริ่มต้นที่ลูปทุกครั้ง ผลที่ได้จะเห็นว่า LED จะติดไล่กันไปทีละดวงจนครบแล้วจึงดับแล้วเริ่มติดใหม่อีกครั้งไปตามลำดับ

บทที่ 4
การทดลองและผลการทดลอง

สามารถติดต่อกับบอร์ดดิจิตอลภายนอกได้โดยทำการอินเตอร์เฟส ระหว่างบอร์ด DSP กับบอร์ดอินเตอร์เฟส โดยมีการเขียนโปรแกรมด้วยภาษาแอสเซมบลี ของ DSP ให้แสดงผลตามที่ต้องการดังนี้

ตัวอย่างที่ 1 เป็นโปรแกรมแสดงการทำให้ LED ติดทีละหนึ่งบิตเรียงกันไปแบบไม่ใช้อินเทอร์รัพต์ ในการติดเรียงไปทีละบิตนั้นหากโปรแกรมนี้ไม่มีการหน่วงเวลา (delay) จะทำให้ LED ติดเรียงกันไปเรื่อยๆอย่างรวดเร็ว ดังนั้นเพื่อให้แสดงผลได้ดีขึ้น ณ ที่นี้จึงทำการแสดงผลแบบมีการหน่วงเวลา โดยแสดงตามผังงาน ดังรูปที่ 4-1



รูปที่ 4-1 ผังงานแสดงการหมุนทีละบิตโดยไม่ใช้อินเทอร์รัพต์

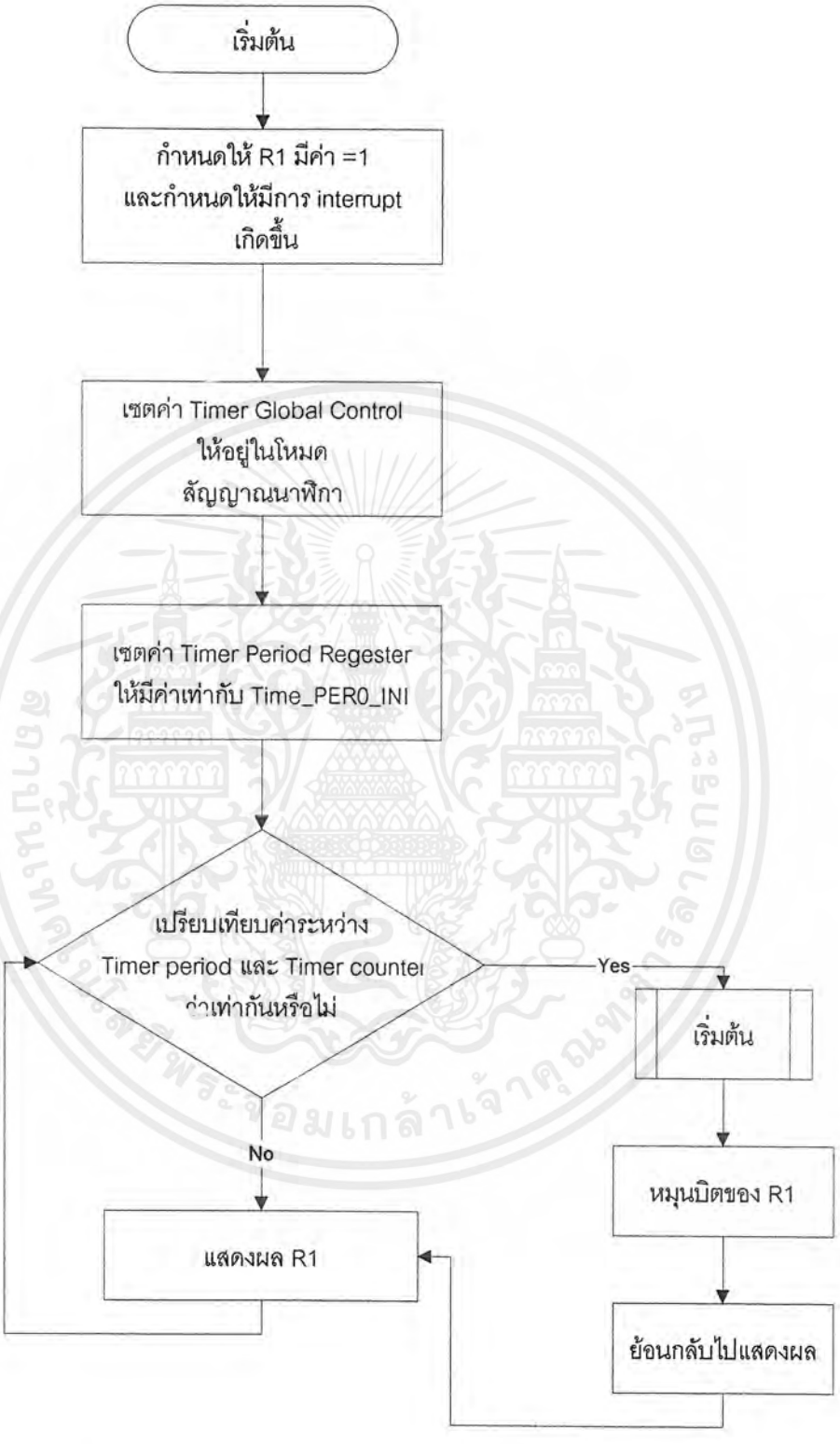
```
;=============================================================
;          โปรแกรม LED ติดเรียงทีละหนึ่งบิต
;=============================================================
                    .start    "mysect",0x809820
                    .sect     "mysect"
USRIOW              .set      0xc00000
;-------------------------------------
LOOP                rol       R1              ;หมุนบิตใน R1
                    ldi       32000,R2        ;การแสดงผลและการหน่วงเวลา

LOOP1               subi      0x1,R2
                    bnz       LOOP1
                    ldi       32000,R2

LOOP2               subi      0x1,R2
                    ldp       USRIOW
                    sti       R1,@USRIOW
                    bnz       LOOP2

                    ldp       USRIOW
                    sti       R1,@USRIOW
                    b         LOOP
;-------------------------------------
                    .entry    START
START               ldp       USRIOW
                    ldi       0x1,R1
                    sti       R1,@USRIOW
                    b         LOOP
;-------------------------------------
```

ในโปรแกรมที่ 1 ที่ให้ LED ติดทีละบิตโดยไม่ใช้อินเทอร์รัพต์นั้นบิตข้างล่างจะเห็นเสมือนว่า LED ติดอยู่ตลอดเวลา ซึ่งแท้จริงแล้วมีการเปลี่ยนแปลงเกิดขึ้นแต่ตัว DSP นี้มีการทำงานที่เร็วมากจึงเสมือนว่าไฟติดอยู่ เราอาจใช้การหน่วงเวลา ในการทำให้การแสดงผลช้าลงได้บ้าง แต่ไม่สามารถกำหนดเวลาที่แน่นอนได้ว่า จะให้กี่วินาทีจึงแสดงผล 1 ครั้ง และก็ไม่สามารถหน่วงเวลาได้มากนัก ดังนั้นเราควรใช้การแสดงผลแบบมีอินเทอร์รัพต์ จะทำให้ผลดีกว่าและกำหนดเวลาได้

ตัวอย่างที่ 2 โปรแกรมแสดงการทำให้ LED ติดทีละหนึ่งบิตโดยติดไล่ไปทีละดวงและใช้อินเทอร์รัพต์ จะทำให้ LED ติดแบบกำหนดเวลาได้ ซึ่งอาจมีคาบเวลาให้เลือกได้หลายค่าตามที่ต้องการในโปรแกรมตัวอย่างก็มีตัวอย่างคาบเวลาต่างๆให้เลือกใช้ โดยโปรแกรมนี้จะมีการทำงานตามผังงาน ดังรูปที่ 4-2

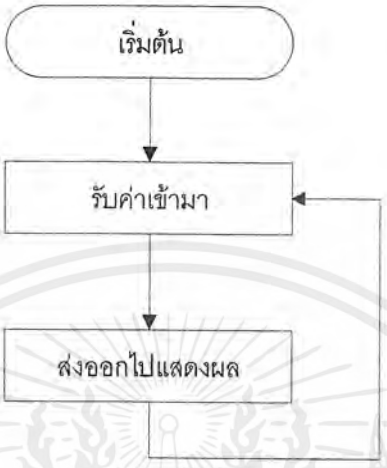
```
;=============================================
;        โปรแกรม แสดงการติดทีละหนึ่งบิตแบบอินเทอร์รัพต์
;=============================================
                    .start   "mysect",0x809820
                    .sect    "mysect"
;
TIM_GCTRL0          .set     0x808020          ;กำหนดค่าตัวจับเวลา 0
TIM_COUNT0          .set     0x808024
TIM_PER0            .set     0x808028
GIE                 .set     0x2000            ;บิต ST GIE จะ ON
;
TIM_GCTRL_INI       .word    0x3c3             ;โหมดสัญญาณนาฬิกา ,I/O=1
;TIM_PER0_INI       .word    0x00000271        ;625=0.1[msec]
;TIM_PER0_INI       .word    0x0000186a        ;6250=1[msec]
;TIM_PER0_INI       .word    0x0000f424        ;62500=10[msec]
;TIM_PER0_INI       .word    0x00098968        ;625000=100[msec]
;TIM_PER0_INI       .word    0x002faf08        ;3125000
TIM_PER0_INI        .word    0x005f5e10        ;6250000=1.0[sec]
;TIM_PER0_INI       .word    0x00bebc20        ;12500000=2.0[sec]
;TIM_PER0_INI       .word    0x017d7840        ;25000000=4.0[sec]
;TIM_PER0_INI       .word    0x07735940        ;125000000=10.0[sec]
;
USRIOW              .set     0xc00000
USRIOR              .set     0xc00001
;-------------------------------------
LOOP                or       0x2000,ST         ;เซตค่าให้เกิดการอินเทอร์รัพต์
                    ldi      0x100,IE
                    ldp      USRIOW            ;แสดงผล
                    sti      R1,@USRIOW
                    b        LOOP
```

```
TINT0           push    ST              ;อินเทอร์รัพต์
                rol     0x1,R1
                pop     ST
                reti
;-----------------------------------
                .entry  START
;       IO INIT
START           ldp     USRIOW
                ldi     0x1,R1
                sti     R1,@USRIOW
;       Timer INIt
                ldp     TIM_GCTRL0
                ldi     0,R6
                ldi     1,R7
                sti     R6,@TIM_GCTRL0
                ldi     @TIM_PER0_INI,R5
                sti     R5,@TIM_PER0
                ldi     @TIM_PER0,R6
                sti     R6,@TIM_COUNT0
                ldi     @TIM_GCTRL_INI,R7
                sti     R7,@TIM_GCTRL0
                b       LOOP
;-----------------------------------
;       interrupt vector
;-----------------------------------
                .start  "interrupt",0x809fc9
                .sect   "interrupt"
                b       TINT0           ;TINT0 0x809fc9
;-----------------------------------
```



รูปที่ 4-2 ผังงานแสดงการหมุนทีละบิตโดยใช้อินเทอร์รัพต์

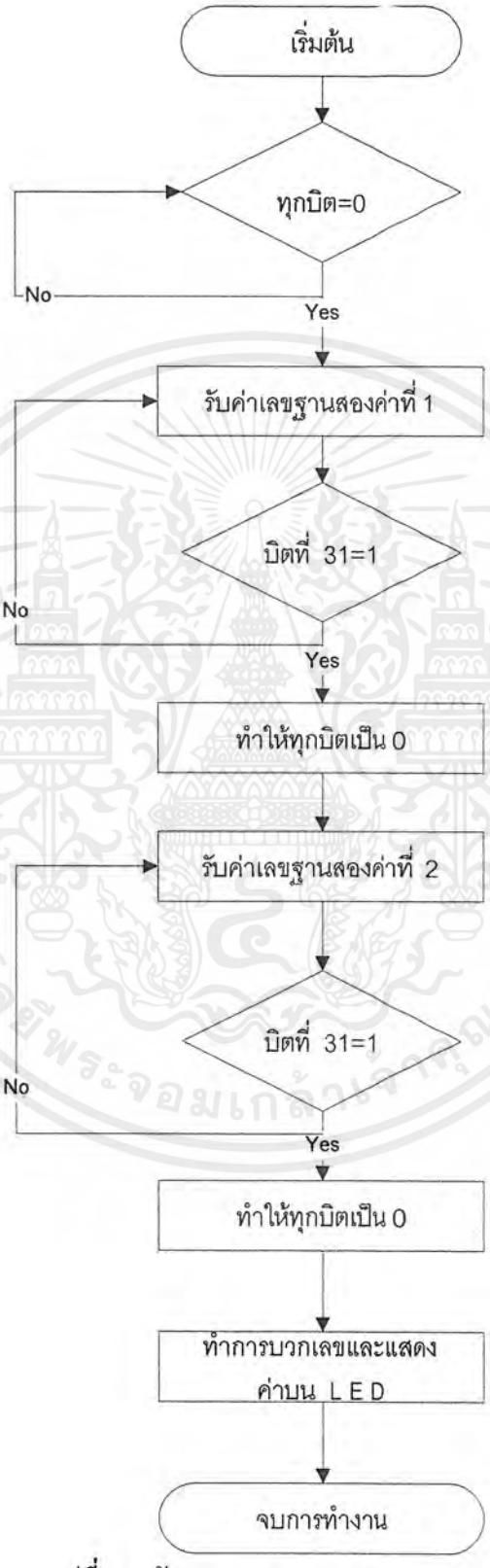
ตัวอย่างที่ 3 โปรแกรมแสดงการรับค่าเข้ามาจากดิพสวิตช์ และส่งค่าออกไปที่ LED โดยที่เมื่อเปลี่ยนค่าที่อินพุต ค่าของ LED ก็จะเปลี่ยนแปลงตาม โดยแสดงการทำงานดังรูปที่ 4-3



รูปที่ 4-3 ผังงานแสดงการรับค่าและส่งออกไปแสดงผล

```
;=============================================
;       โปรแกรมแสดงการรับค่าและส่งออกไปแสดงผล
;=============================================
                .start  "mysect",0x809820
                .sect   "mysect"
USRIOW          .set    0xc00000
USRIOR          .set    0xc00001
;---------------------------------
LOOP            ldp     USRIOR          ;รับค่าเข้ามา
                ldi     @USRIOR,R1
                ldp     USRIOW          ;ส่งค่าออกไป
                sti     R1,@USRIOW
                b       LOOP
                .entry  START
START           ldp     USRIOW
                ldi     0,R1
                sti     R1,@USRIOW
                b       LOOP
;---------------------------------
```

ตัวอย่างที่ 4 โปรแกรมการบวกเลขฐานสอง โดยมีผังงานดังรูปที่ 4-4



รูปที่ 4-4 ผังงานแสดงการบวกเลขฐานสอง

```
;=============================================
;          โปรแกรม การบวกเลขแสดงผล
;=============================================
                    .start    "mysect",0x809820
                    .sect     "mysect"
USRIOW              .set      0xc00000
USRIOR              .set      0xc00001
;-----------------------------------------
LOOPa2              ldp       USRIOR
                    ldi       @USRIOR,R1
                    cmpi      R1,R6             ;ทุกบิตต้อง = 0
                    bnz       LOOPa2
LOOP                ldp       USRIOR            ;รับค่าที่ 1
                    ldi       @USRIOR,R1
                    ldi       R1,R3
                    ldi       0xfffffffe,R2
                    ror       R2
                    and       R2,R3
                    ldp       USRIOW
                    sti       R3,@USRIOW
                    ldp       USRIOR
                    ldi       @USRIOR,R1
                    ldi       R7,R5
                    and       R1,R5
                    cmpi      R5,R7             ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
                    bnz       LOOP
LOOPa               ldp       USRIOR
                    ldi       @USRIOR,R1
                    cmpi      R1,R6             ;ทุกบิตต้อง = 0
                    bnz       LOOPa
LOOP3               ldp       USRIOR            ;รับค่าที่ 2
                    ldi       @USRIOR,R1
                    ldi       R1,R4
                    ldi       0xfffffffe,R2
```

```
                ror     R2
                and     R2,R4
                ldp     USRIOW
                sti     R4,@USRIOW
                ldp     USRIOR
                ldi     @USRIOR,R1
                ldi     R7,R5
                and     R1,R5
                cmpi    R5,R7                   ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
                bnz     LOOP3
LOOPa1          ldp     USRIOR
                ldi     @USRIOR,R1
                cmpi    R1,R6                   ; ทุกบิตต้อง = 0
                bnz     LOOPa1
                addi    R3,R4                   ;บวก
Dis             ldp     USRIOW                  ;แสดงผล
                sti     R4,@USRIOW
                b       Dis
;-------------------------------------
                .entry  START
;       IO INIT
START           ldp     USRIOW
                ldi     0,R1
                sti     R1,@USRIOW
                ldi     1,R7
                ldi     0,R6
                ror     R7
                b       LOOP
;-------------------------------------------------
```

ซึ่งโปรแกรมนี้จะสามารถบวกเลขฐานสอง และแสดงได้ 32 บิต โดยที่โปรแกรมการลบ, คูณและการหาร จะมีการทำงานคล้ายคลึงกันโดยเปลี่ยนจากการทำการบวกเป็นการลบ, การคูณและการหารตามลำดับ โดยที่การคูณนั้นจะเปลี่ยนจากคำสั่งบวกเป็นคำสั่งคูณเท่านั้น ส่วนคำสั่งในการหารจำเป็นต้องใช้ขั้นตอนวิธีหารเลขฐานสองเข้ามาช่วย เพราะในตัวบอร์ด DSP ไม่มีคำสั่งที่ใช้สำหรับการหาร (ได้เขียนโปรแกรมการหารซึ่งดูได้ในภาคผนวก โปรแกรมที่ 6)

4.1 ข้อจำกัดในการเขียนโปรแกรม

1. ในการรับค่าจากในโปรแกรมบวก, การคูณ และการหารจะสามารถรับค่าได้เพียง 31 บิตเท่านั้นเนื่องจากจำเป็นต้องใช้บิตที่ 32 เพื่อเป็นตัวยืนยันคำสั่งในการรับค่า
2. ในโปรแกรมการบวกเลขฐานสองเนื่องจากสามารถรับค่าได้เพียง 31 บิตเท่านั้น เราจึงสามารถบวกเลขได้เพียง 31 บิตเท่านั้นแต่การแสดงผลก็สามารถแสดงเป็น 32 บิตได้ ส่วนโปรแกรมการลบมีลักษณะในการทำงานคล้ายคลึงกับการบวกจึงไม่นำเสนอไว้ ณ ที่นี้
3. ในโปรแกรมการคูณเลขฐานสองจะสามารถคูณได้เพียง 16 บิตเท่านั้นเพราะตัวแสดงผล มีการแสดงค่าเพียง 32 บิต ดังนั้นหากต้องการคูณเลข 32 บิตจำเป็นจะต้องใช้โปรแกรมอีกโปรแกรมหนึ่งซึ่งไม่ได้แสดงไว้ ณ ที่นี้โดยต้องมีการแสดงผลสองครั้งครั้งละ 32 บิต
4. ในโปรแกรมการหารเลขฐานสองเมื่อทำการหารแล้วจะได้ผลออกที่มาจะมีเพียงผลหารเท่านั้น ไม่ได้แสดงเศษที่เหลือจากการหารไว้ในโปรแกรม และจำเป็นต้องทำให้ตัวหารเป็นบวกทั้ง 2 ค่าด้วยเนื่องจากถ้าไม่ทำดังนี้จะทำให้เกิดปัญหาในการคำนวณเกิดขึ้น โดยตัวเศษจะต้องมีค่ามากกว่าตัวส่วนหากตัวส่วนมากกว่าจำเป็นจะต้องใช้โปรแกรมอื่นในการหาค่า

4.2 การนำไปประยุกต์ใช้งาน (Application)

ในที่นี้จะนำ DSP Board ไปประยุกต์ใช้กับการแปลงฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่องอย่างรวดเร็ว (FFT) โดยใส่สัญญาณอินพุตเข้าไปทาง RCA jack analog input แล้วบอร์ด DSP จะทำการชักตัวอย่าง (sampling) และประมวลผลสัญญาณใน TMS320C31-50 แล้วเอาท์พุตที่ได้จะเป็นสเปกตรัมของความถี่ของสัญญาณอินพุตที่ป้อนเข้าไป

หลักการแปลงฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่องอย่างรวดเร็ว (Fast Fourier Tranform : FFT)

เป็นขั้นตอนวิธีการคำนวณฟูเรียร์ทรานฟอร์มแบบไม่ต่อเนื่อง (Discrete Fourier Transform : DFT) โดยสามารถคำนวณได้รวดเร็วกว่าการแปลงฟูเรียร์ทรานฟอร์มแบบไม่ต่อเนื่องแบบธรรมดา

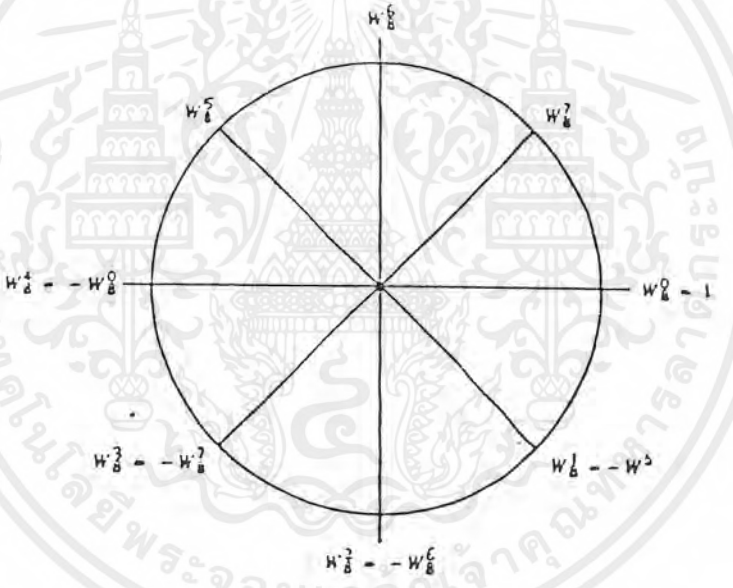
มีจุดประสงค์เพื่อแปลงข้อมูลจากโดเมนของเวลาให้อยู่ในโดเมนของความถี่ และสามารถแปลงข้อมูลโดเมนของความถี่กลับมาอยู่ในรูปโดเมนของเวลาได้เช่นเดียวกัน

FFT นั้นถูกคิดค้นขึ้นหลายวิธีด้วยกัน แต่ในที่นี้ใช้วิธีกำลังสอง (power of 2) เนื่องจากเป็นวิธีที่ง่ายและเป็นที่นิยมใช้กันมากที่สุด โดยอาศัยหลักการลดรูปเพื่อลดจำนวนการคูณ ซึ่งเป็นส่วนที่เสียเวลาในการคำนวณมากให้น้อยลง สามารถเขียนอธิบายออกมาด้วยแผนภาพผีเสื้อ (butterfly diagram)

พิจารณาสูตรฟูเรียร์ทรานฟอร์มแบบไม่ต่อเนื่อง จะเห็นได้ว่า $e^{-j2\pi nk/N}$ นั้นจะมีลักษณะเป็นคาบและมีคาบเท่ากับ N เพื่อความสะดวกเราจะใช้ W^k แทน $e^{-j2\pi nk/N}$ นั่นคือ

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

โดยที่ W_N^{nk} จะถูกเรียกว่า Twiddle factor ซึ่งมีค่าซ้ำๆ กันดังรูปที่ 4-5



รูปที่ 4-5 แสดง Twiddle factor

สูตรฟูเรียร์ทรานส์ฟอร์มดังกล่าวสามารถจัดรูปใหม่ได้ดังนี้

$$X(k) = x(0)W_N^0 + x(1)W_N^k + x(2)W_N^{2k} + \ldots + x(N-1)W_N^{(N-1)k}$$

จากสมการ $X(k)$ ต้องการการคูณเลขเชิงซ้อน N ครั้ง และการบวกเชิงซ้อน $N-1$ ครั้ง จึงประมาณเป็น $2N$ ของการคำนวณทางคณิตศาสตร์สำหรับ $X(k)$ แต่ละตัว แต่เนื่องจาก $X(k)$ มีทั้งหมด N ค่า ดังนั้นจึงต้องการการคำนวณทั้งหมด $2N^2$ ครั้ง แบ่งเป็น N^2 ครั้งสำหรับการคูณ

เชิงซ้อน และ N^2 ครั้งสำหรับการบวกเชิงซ้อน และวิธีการทำให้วิธีนี้ลดลง คือ วิธีการของการแปลงฟูเรียร์แบบเร็ว (FFT) ซึ่งหลักการพื้นฐานของการลดการคำนวณ คือการแบ่งลำดับที่ได้มาให้เป็นลำดับย่อยๆ โดยที่ความยาวของลำดับน้อยลง แล้วค่อยนำมารวมกันอีกทีในแต่ละ DFT ย่อยๆ ที่เหมาะสมเพื่อที่จะได้การแปลงฟูเรียร์ของการแปลงลำดับเดิม

ในการแปลงฟูเรียร์ไม่ต่อเนื่องทางเวลาแบบเร็ว ในที่นี้จะมีสองลักษณะ คือการสลับทางเวลากับการสลับทางความถี่ ซึ่งในทั้งสองกรณีลำดับที่จะทำการแปลงนั้นจะต้องมีความยาว N เป็นค่าของสองยกกำลัง ดังนั้น N จะอยู่ในรูปของ $N = 2^p$ เมื่อ p เป็นจำนวนเต็มบวก ทำให้ N มีค่าเป็น 2, 4, 8, 16, 32,... ซึ่งถูกเรียกว่า ฐานสอง (radix-2)

1. การสลับทางเวลาของ FFT (Decimation in time : DIT)

วิธีการนี้จะแตกลำดับอินพุตในโดเมนเวลาออกเป็นลำดับย่อยๆ (subsequence) ในขั้นแรกจะแบ่งลำดับออกเป็นชุดย่อยๆ จำนวน 2 ชุด ให้ $x_1(m)$ เป็นลำดับคู่และ $x_2(m)$ เป็นลำดับคี่ โดยจัดลำดับอินพุตใหม่

$$x_1(m) = x(2m)$$

$$x_2(m) = x(2m+1)$$

เมื่อ $m = 0,1,2,3,...(N/2)-1$

จากสมการ DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \text{เมื่อ} \quad 0 \le k \le N-1$$

$$= \sum_{m=0}^{(N/2)-1} x_1(m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x_2(m) W_N^{(2m+1)k}$$

$$= \sum_{m=0}^{(N/2)-1} x_1(m) W_{N/2}^{mk} + W_N^k \sum_{m=0}^{(N/2)-1} x_2(m) W_{N/2}^{mk} \quad (W_N^{2n} = W_{N/2}^n)$$

$$= x_1(k) + W_N^k x_2(k) \quad \text{เมื่อ} \quad 0 \le k \le (N/2)-1$$

$$- x_1(k - N/2) + W_N^k x_2(k - N/2) \quad \text{เมื่อ} \quad (N/2) \le k \le N-1$$

โดยแสดงแผนภาพผีเสื้อ 16 จุดของ DIT FFT ดังรูปที่ 4-6

2. การสลับทางความถี่ของ FFT (Decimation in frequency : DIF)

วิธีการคล้ายกับแบบแรก แต่จะจัดรูปใหม่ให้มีการสลับเอาท์พุตแทนที่จะสลับอินพุต

$x_1(m) = x(m)$

$x_2(m) = x(m + N/2)$

เมื่อ $m = 0,1,2,3,...(N/2)-1$

จากสมการ DFT

$$X(k) = \sum_{m=0}^{(N/2)-1} x_1(m)W_N^{mk} + \sum_{m=0}^{(N/2)-1} x_2(m)W_N^{(m+N/2)k}$$

$$= \sum_{m=0}^{(N/2)-1} x(m)W_N^{mk} + \sum_{m=0}^{(N/2)-1} x(m+N/2)W_N^{(m+N/2)k}$$

$$= \sum_{m=0}^{(N/2)-1} [x(m)W_N^{(N/2)k} + x(m+N/2)]W_N^{nk}$$

แยกเทอม $x(k)$ ออกเป็นเทอมคู่และคี่

$$x(2h) = \sum_{m=0}^{(N/2)-1} [x(m) + x(m+N/2)]W_{N/2}^{mh}$$

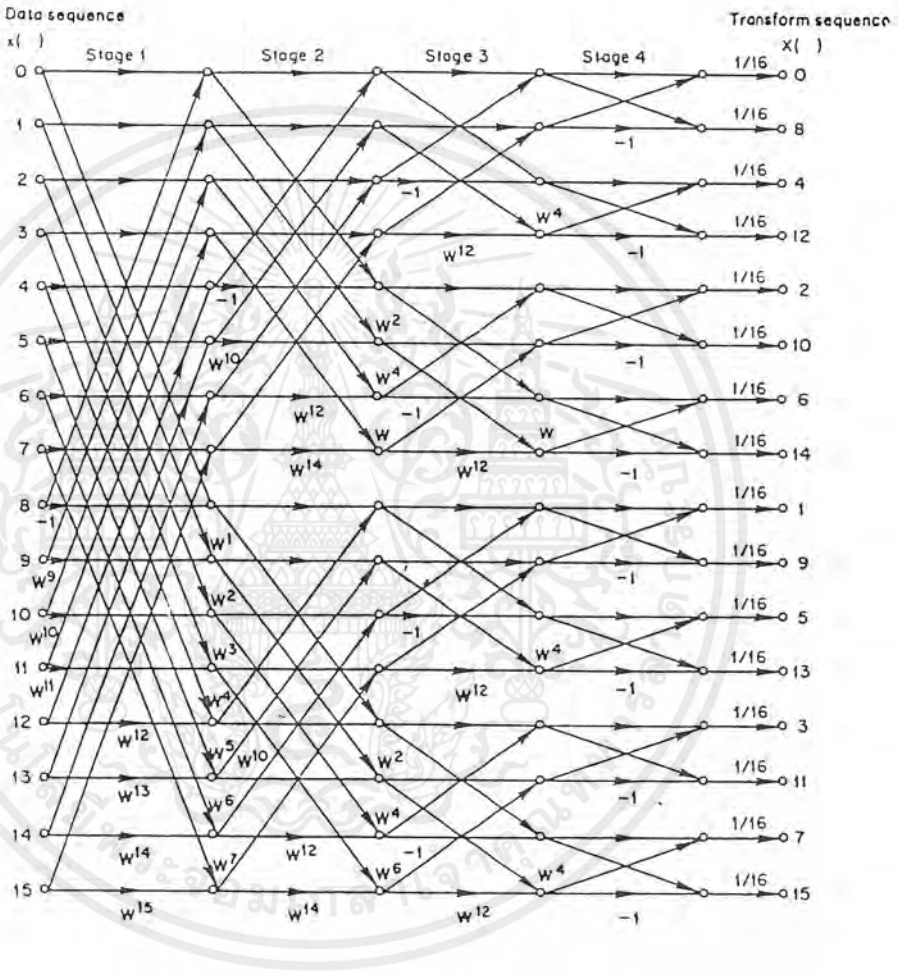
$$x(2h+1) = \sum_{m=0}^{(N/2)-1} [x(m) + x(m+N/2)]W_N^m W_{N/2}^{mh}$$

โดยแสดงแผนภาพผีเสื้อ 16 จุดของ DIF FFT ดังรูปที่ 4-7

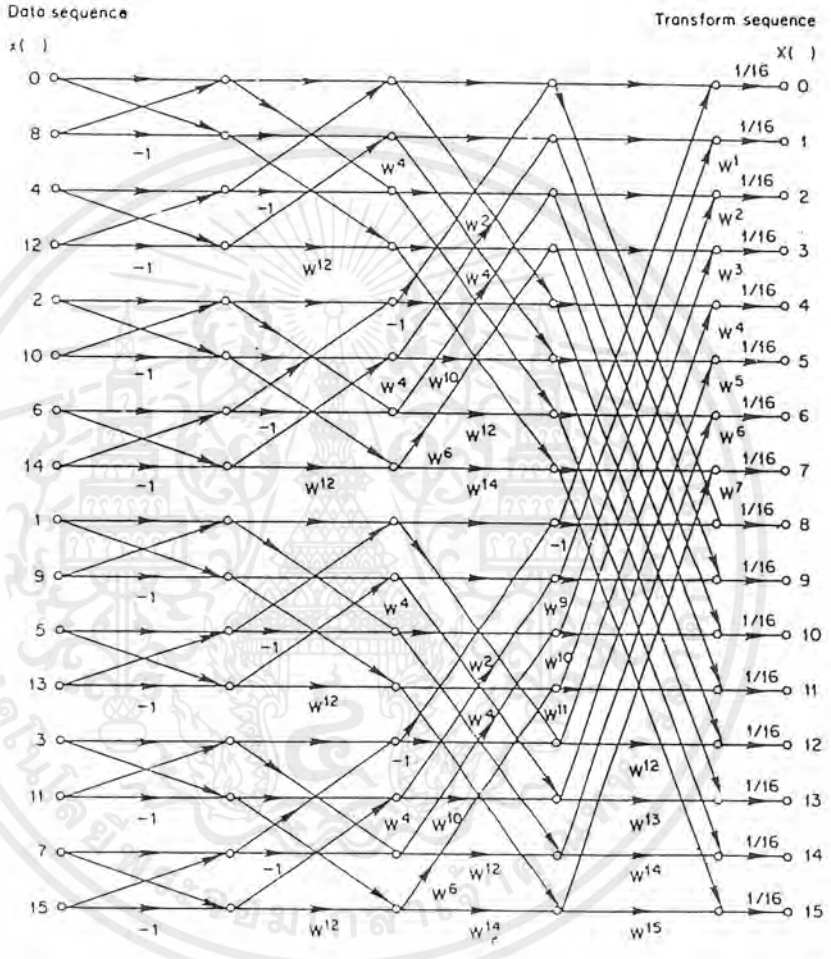
3. การสลับตำแหน่งใน FFT

การสลับตำแหน่งใน FFT จะใช้วิธีการสลับบิตตัวเลขลำดับที่อยู่ในรูปของตัวเลขฐานสอง โดยตัวอย่างแสดงการกลับบิตเมื่อ $N = 8$ เป็นดังนี้

ลำดับอินพุตฐาน 10	0	1	2	3	4	5	6	7
ลำดับอินพุตฐาน 2	000	001	010	011	100	101	110	111
การสลับบิต	000	100	010	110	001	101	011	111
ลำดับอินพุตเมื่อทำการสลับบิตแล้ว	0	4	2	6	1	5	2	7



รูปที่ 4-6 แสดงแผนภาพผีเสื้อ 16 จุดของ DIT FFT



รูปที่ 4-7 แสดงแผนภาพผีเสื้อ 16 จุดของ DIF FFT

บทที่ 5
สรุปผลการทดลอง

เราสามารถทำการติดต่ออินเตอร์เฟสระหว่างบอร์ด DSP และอุปกรณ์ภายนอกที่เป็นดิจิตอลได้ โดยทำการโหลดโปรแกรมจากเครื่องคอมพิวเตอร์โดยผ่านพอร์ทขนาน ในการติดต่อกับอุปกรณ์ภายนอกนี้เราได้มีโปรแกรมหลายรูปแบบในการติดต่อดังนี้

-โปรแกรมที่ 1 เป็นโปรแกรมบวกเพิ่มค่าขึ้นมาทีละ 1 โดยไม่ใช้อินเทอร์รัพต์

-โปรแกรมที่ 2 เป็นโปรแกรมบวกเพิ่มค่าขึ้นทีละ 1 เช่นเดียวกันแต่ใช้อินเทอร์รัพต์ ซึ่งในการใช้อินเตอร์รัพต์นี้เราสามารถกำหนดเวลาที่แน่นอนให้กับตัวแสดงผล (display) ได้ว่าจะให้กี่วินาทีจึงจะให้ LED ติดไฟแต่ละครั้ง

-โปรแกรมที่ 3 เป็นการติดต่อระหว่างอินพุต และการแสดงผล โดยเมื่อส่งค่าใดเข้าไปก็จะแสดงผลเป็นค่านั้นออกมาทางตัวแสดงผล

-โปรแกรมที่ 4 เป็นโปรแกรมที่ใช้ในการบวกค่าของเลขฐานสอง โดยที่จะรับค่าสองค่าจากอินพุต นำไปคำนวณใน DSP แล้วจึงส่งออกมาแสดงผลที่ตัวแสดงผล ส่วนโปรแกรมการลบทำคล้ายๆ การบวกจึงไม่แสดงไว้ ณ ที่นี้

-โปรแกรมที่ 5 เป็นโปรแกรมที่ใช้ในการคูณค่าซึ่งจะสามารถทำการคูณโดยที่รับค่าแต่ละค่ามาได้เพียง 16 บิตเท่านั้น เนื่องจากว่ามีตัวแสดงผลเพียง 32 บิต

-โปรแกรมที่ 6 เป็นโปรแกรมที่ใช้ในการหารเลขฐานสอง โดยที่จะต้องทำเลขฐานสองเหล่านี้ให้เป็นเลขบวกอยู่เสมอ และจะแสดงผลแต่เพียงผลหารเท่านั้น ไม่ได้แสดงเศษที่เหลือจากการหาร

ปัญหาในการทดลองและแนวทางแก้ไข

ปัญหาที่พบในทางปฏิบัติของการคำนวณฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่องมีอยู่ 3 ประการด้วยกัน

1. Aliasing เป็นปัญหาที่เกิดขึ้นเนื่องจากความถี่ในการชักตัวอย่างน้อยกว่า 2 เท่าของความถี่สัญญาณอินพุต มีผลทำให้โดเมนความถี่เกิดการซ้อนทับกัน เมื่อทำการแปลงอินเวอร์สฟูเรียร์ทรานส์ฟอร์มกลับจะทำให้สัญญาณในโดเมนเวลาที่ได้เพี้ยนไปจากเดิม แสดงการเกิดการซ้อนทับของสัญญาณในโดเมนความถี่ดังรูปที่ 5-1

2. Leakage ปัญหานี้เป็นผลมาจากการชักสัญญาณไม่ครบ ซึ่งไม่ได้หมายความว่าการชักตัวอย่างนั้นมีการขาดหายไปบางจุด แต่เป็นเพราะการชักตัวอย่างตามความเป็นจริงนั้นไม่สามารถชักตัวอย่างให้เป็นตามฟังก์ชันทางคณิตศาสตร์ที่สมบูรณ์ได้ เสมือนว่าตัดส่วนของสัญญาณส่วนหนึ่งจากสัญญาณทั้งหมดมา ทำให้การคำนวณได้ผลที่เพี้ยนไป

การลดผลจาก leakage นี้จะใช้วินโดว์ฟังก์ชัน (window function) ซึ่งเป็นฟังก์ชันที่นำมาใช้ช่วยลดผลของ leakage ดังนี้

-Rectangle window

-Triangular window

-Gaussian window

-Hanning window

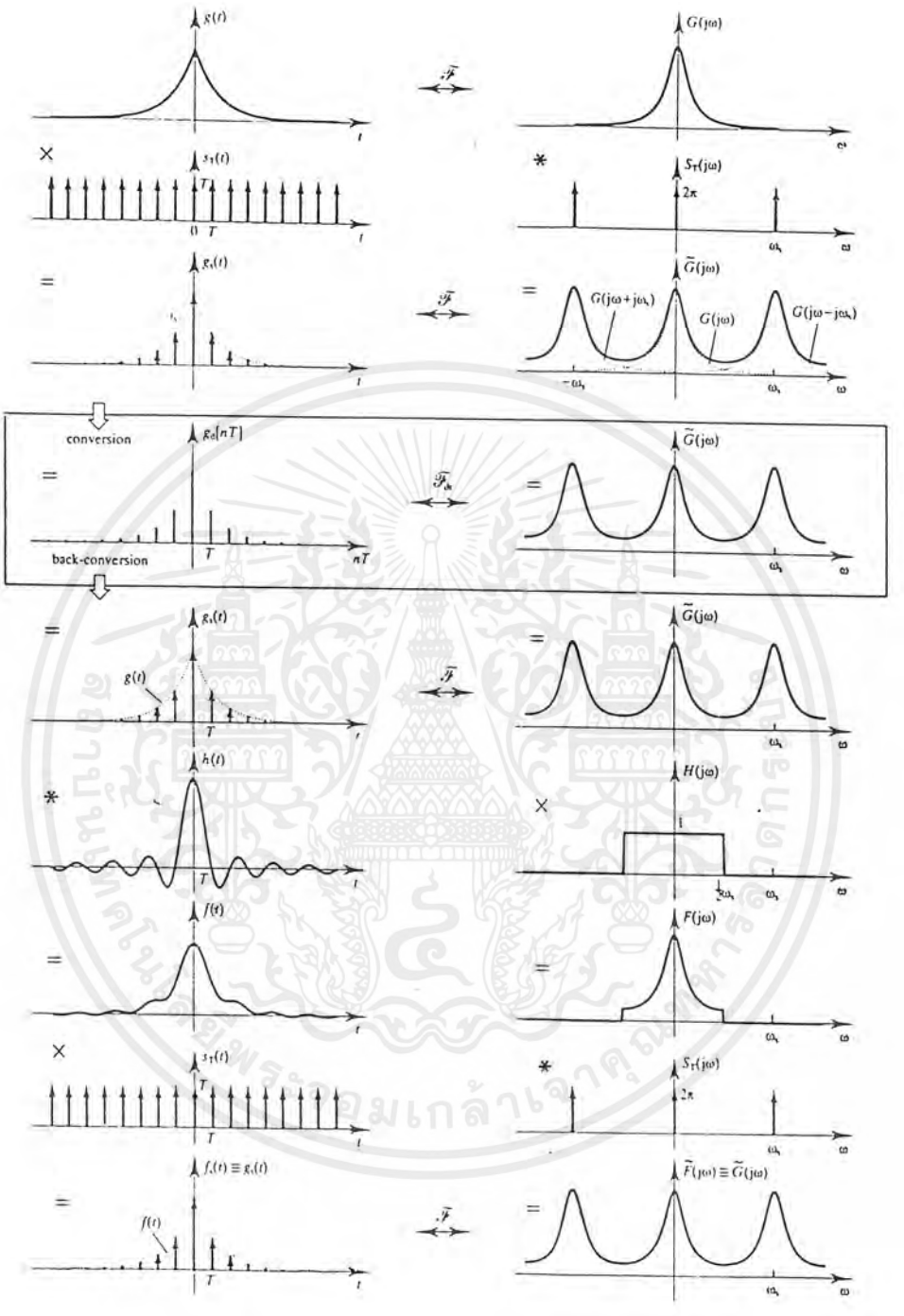
-Hamming window

-Blackman window

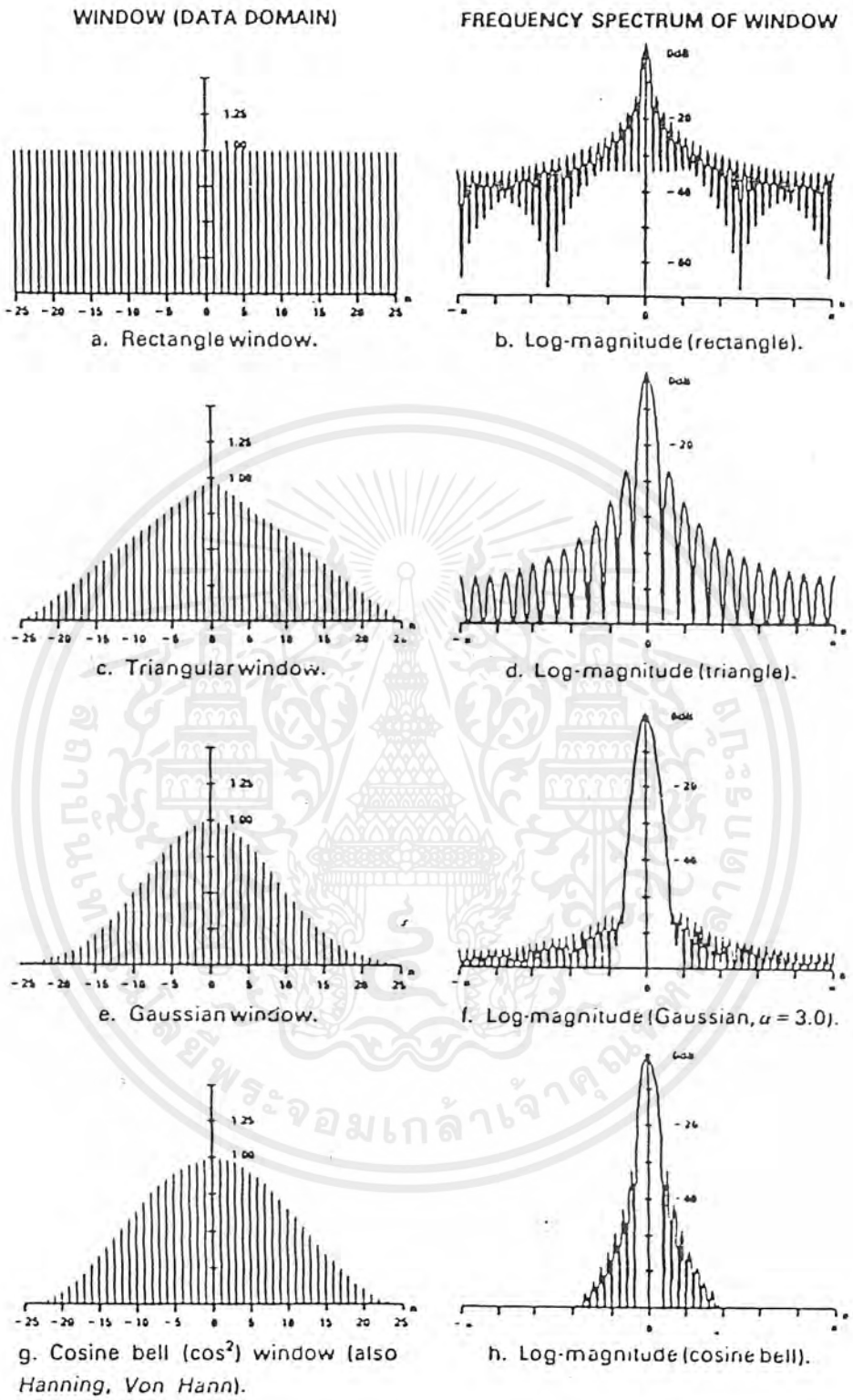
-Kaiser-Bessel window

แสดงในรูปที่ 5-2

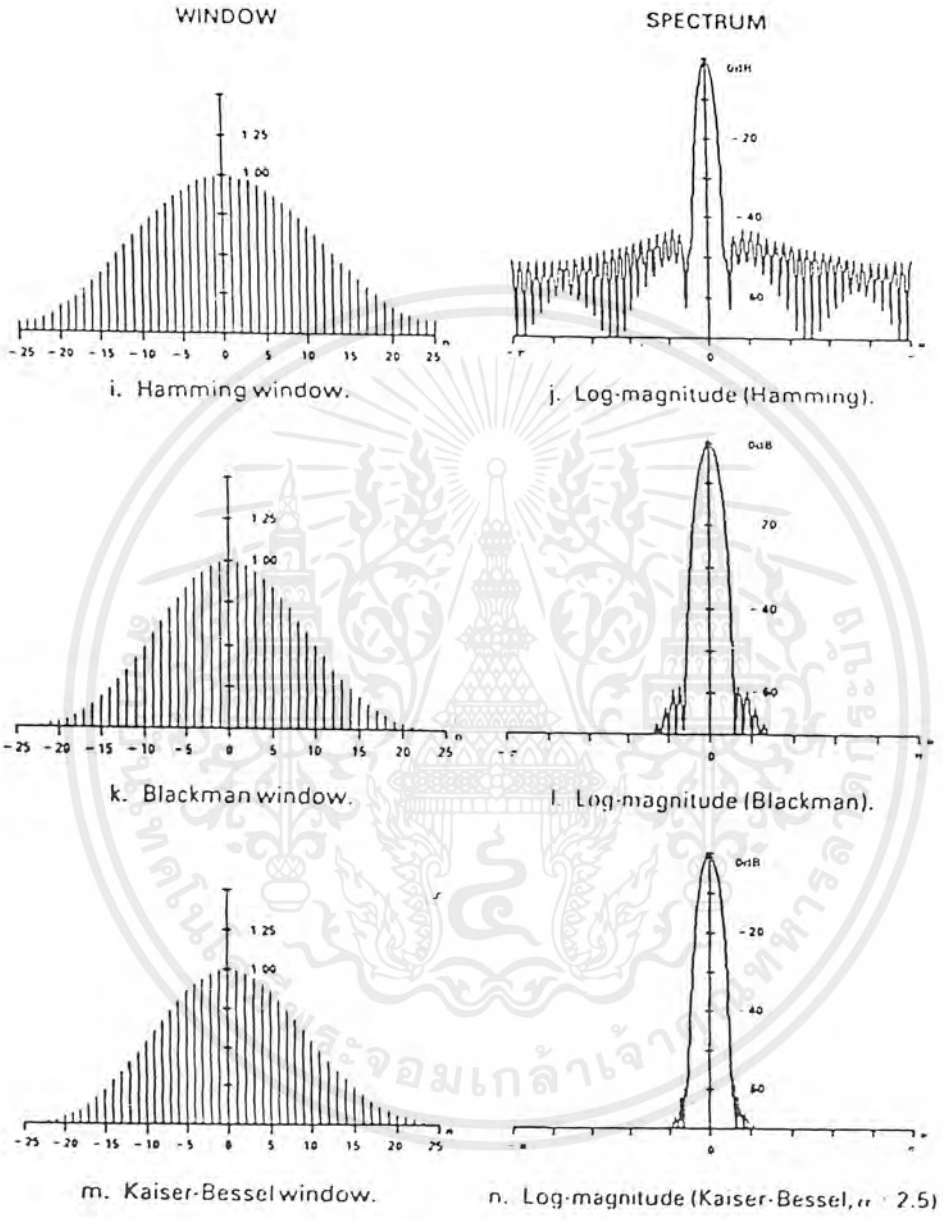
3. Picket fence effect เนื่องจากการแปลงฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่องนั้น เป็นการคำนวณจากการชักตัวอย่าง สัญญาณในโดเมนความถี่หลังจากที่คำนวณได้จึงอยู่ในรูปไม่ต่อเนื่องเช่นกัน ถ้าความถี่ในการชักตัวอย่างมีค่าค่อนข้างน้อย จะทำให้พิจารณาสัญญาณไม่ต่อเนื่องได้ยากขึ้น เพราะรูปลักษณ์ของสัญญาณจะไม่ชัดเจน ทางแก้ปัญหาคือ เพิ่มความถี่ในการชักตัวอย่าง และเพิ่มหน่วยความจำในการชักตัวอย่างที่เพิ่มขึ้นอีกด้วย โดยต้องกำหนด TA, TB, RA, และ RB ซึ่งเป็นรีจิสเตอร์ที่ใช้ในการกำหนดการชักตัวอย่างของวงจรอินเตอร์เฟสอนาลอก TLC32040 รูปที่ 5-3 แสดงผลการใช้ความถี่ในการชักตัวอย่างต่างกัน



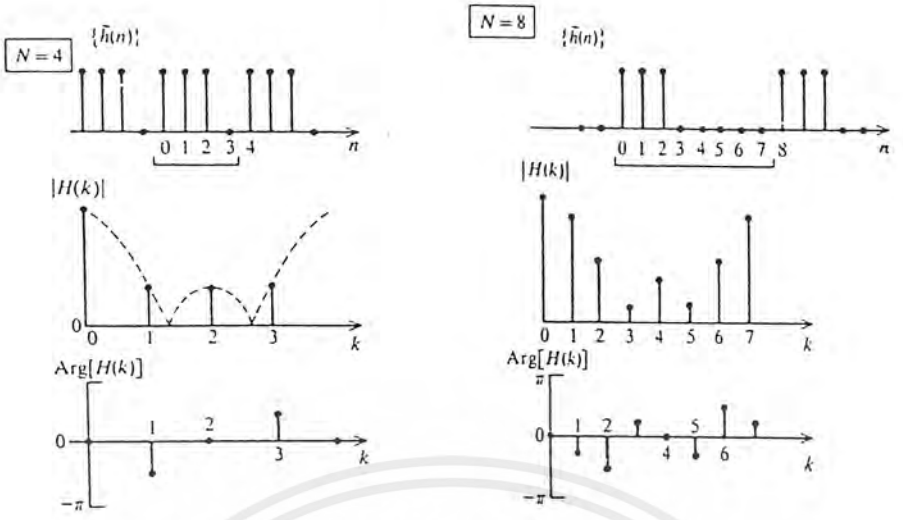
รูปที่ 5-1 แสดงการเกิดการซ้อนทับของสัญญาณในโดเมนความถี่ (aliasing)



รูปที่ 5-2 แสดงวินโดว์แบบต่างๆ



รูปที่ 5-2 (ต่อ) แสดงวินโดว์แบบต่างๆ



รูปที่ 5-3 แสดงผลของการทำฟูเรียร์ทรานส์ฟอร์มแบบไม่ต่อเนื่อง
โดยใช้ความถี่ในการชักตัวอย่างต่างกัน

ภาคผนวก

ตัวอย่างโปรแกรม

```
;=============================================================
;       โปรแกรมที่ 1 LED ติดทีละหนึ่งบิตแบบไม่ใช้อินเทอร์รัพต์
;=============================================================
                    .start   "mysect",0x809820
                    .sect    "mysect"
USRIOW              .set     0xc00000
;-------------------------------------------------------
LOOP                rol      R1
                    ldi      32000,R2          ;การหน่วงเวลา

LOOP1               subi     0x1,R2
                    bnz      LOOP1
                    ldi      32000,R2

LOOP2               subi     0x1,R2
                    ldp      USRIOW
                    sti      R1,@USRIOW
                    bnz      LOOP2
                    ldp      USRIOW
                    sti      R1,@USRIOW
                    b        LOOP
;-------------------------------------------------------
                    .entry   START
START               ldp      USRIOW
                    ldi      0x1,R1
                    sti      R1,@USRIOW
                    b        LOOP
;-------------------------------------------------------
```

```
;=============================================================
;        โปรแกรมที่ 2 LED ติดทีละหนึ่งบิตแบบใช้อินเทอร์รัพต์
;=============================================================
                    .start   "mysect",0x809820
                    .sect    "mysect"
TIM_GCTRL0          .set     0x808020
TIM_COUNT0          .set     0x808024
TIM_PER0            .set     0x808028
;
GIE                 .set     0x2000          ;บิต ST GIE  ON
TIM_GCTRL_INI       .word    0x3c3           ;โหมดสัญญาณนาฬิกา, I/O=1
;TIM_PER0_INI       .word    0x00000271      ;625=0.1[msec]
;TIM_PER0_INI       .word    0x0000186a      ;6250=1[msec]
;TIM_PER0_INI       .word    0x0000f424      ;62500=10[msec]
;TIM_PER0_INI       .word    0x00098968      ;625000=100[msec]
;TIM_PER0_INI       .word    0x002faf08      ;3125000
TIM_PER0_INI        .word    0x005f5e10      ;6250000=1.0[sec]
;TIM_PER0_INI       .word    0x00bebc20      ;12500000=2.0[sec]
;TIM_PER0_INI       .word    0x017d7840      ;25000000=4.0[sec]
;TIM_PER0_INI       .word    0x07735940      ;125000000=10.0[sec]
;
USRIOW              .set     0xc00000
USRIOR              .set     0xc00001
;-------------------------------------------------------------
LOOP                or       0x2000,ST       ;ตั้งค่าอินเทอร์รัพต์
                    ldi      0x100,IE
                    ldp      USRIOW          ;แสดงผล
                    sti      R1,@USRIOW
                    ldp      RAMBLK
                    b        LOOP
```

```
TINT0           push    ST              ;อินเทอร์รัพต์
                rol     R1
                pop     ST
                reti
;---------------------------------------------------
                .entry  START
START           ldp     USRIOW
                ldi     0x1,R1
                sti     R1,@USRIOW
;       Timer INIt
                ldp     TIM_GCTRL0
                ldi     0,R6
                ldi     1,R7
                sti     R6,@TIM_GCTRL0
                ldi     @TIM_PER0_INI,R5
                sti     R5,@TIM_PER0
                ldi     @TIM_PER0,R6
                sti     R6,@TIM_COUNT0
                ldi     @TIM_GCTRL_INI,R7
                sti     R7,@TIM_GCTRL0
                b       LOOP
;---------------------------------------------------
;       interrupt vector
;---------------------------------------------------
                .start  "interrupt",0x809fc9
                .sect   "interrupt"
                b       TINT0           ;TINT0 0x809fc9
;---------------------------------------------------
```

```
;=================================================
;       โปรแกรมที่ 3 การรับส่งค่าโดย LED จะเปลี่ยนแปลงตามอินพุท
;=================================================
                    .start    "mysect",0x809820
                    .sect     "mysect"
USRIOW              .set      0xc00000
USRIOR              .set      0xc00001
;-------------------------------------------------
LOOP                ldp       USRIOR
                    ldi       @USRIOR,R1
                    ldp       USRIOW
                    sti       R1,@USRIOW
                    b         LOOP
;-------------------------------------------------
                    .entry    START
;       IO INIT
START               ldp       USRIOW
                    ldi       0,R1
                    sti       R1,@USRIOW
                    b         LOOP
;-------------------------------------------------
```

```
;=============================================
;          โปรแกรมที่ 4 การบวกเลขแสดงผล
;=============================================
                    .start   "mysect",0x809820
                    .sect    "mysect"
USRIOW              .set     0xc00000
USRIOR              .set     0xc00001
;-----------------------------------------------
LOOPa2              ldp      USRIOR
                    ldi      @USRIOR,R1
                    cmpi     R1,R6              ;ทุกบิตต้อง=0
                    bnz      LOOPa2

LOOP                ldp      USRIOR             ;รับค่าที่ 1
                    ldi      @USRIOR,R1
                    ldi      R1,R3
                    ldi      0xfffffffe,R2
                    ror      R2
                    and      R2,R3
                    ldp      USRIOW             ;แสดงค่าที่รับมา
                    sti      R3,@USRIOW
                    ldp      USRIOR
                    ldi      @USRIOR,R1
                    ldi      R7,R5
                    and      R1,R5
                    cmpi     R5,R7              ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
                    bnz      LOOP

LOOPa               ldp      USRIOR
                    ldi      @USRIOR,R1
                    cmpi     R1,R6              ;ทุกบิตต้อง = 0
                    bnz      LOOPa
```

```
LOOP3       ldp     USRIOR          ;รับค่าที่ 2
            ldi     @USRIOR,R1
            ldi     R1,R4
            ldi     0xfffffffe,R2
            ror     R2
            and     R2,R4
            ldp     USRIOW          ;แสดงค่าที่รับเข้ามา
            sti     R4,@USRIOW
            ldp     USRIOR
            ldi     @USRIOR,R1
            ldi     R7,R5
            and     R1,R5
            cmpi    R5,R7           ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
            bnz     LOOP3
LOOPa1      ldp     USRIOR
            ldi     @USRIOR,R1
            cmpi    R1,R6
            bnz     LOOPa1
            addi    R3,R4           ;บวก
Dis         ldp     USRIOW          ;แสดงผล
            sti     R4,@USRIOW
            b       Dis
;-----------------------------------
            .entry  START
START       ldp     USRIOW
            ldi     0,R1
            sti     R1,@USRIOW
            ldi     1,R7
            ldi     0,R6
            ror     R7
            b       LOOP
;-----------------------------------
```

```
;=================================================
;        โปรแกรมที่ 5 การคูณเลขแสดงผล
;=================================================
                .start   "mysect",0x809820
                .sect    "mysect"
USRIOW          .set     0xc00000
USRIOR          .set     0xc00001
;-------------------------------------------------
LOOPa2          ldp      USRIOR
                ldi      @USRIOR,R1
                cmpi     R1,R6          ;ทุกบิตต้อง=0
                bnz      LOOPa2

LOOP            ldp      USRIOR         ;รับค่าที่ 1
                ldi      @USRIOR,R1
                ldi      R1,R3
                ldi      0xfffffffe,R2
                ror      R2
                and      R2,R3
                ldp      USRIOW
                sti      R3,@USRIOW
                ldp      USRIOR
                ldi      @USRIOR,R1
                ldi      R7,R5
                and      R1,R5
                cmpi     R5,R7          ;ตรวจสอบว่าบิตที่ 31 =1หรือยัง
                bnz      LOOP

LOOPa           ldp      USRIOR
                ldi      @USRIOR,R1
                cmpi     R1,R6          ;ทุกบิตต้อง = 0
                bnz      LOOPa
```

```
LOOP3       ldp     USRIOR          ;รับค่าที่ 2
            ldi     @USRIOR,R1
            ldi     R1,R4
            ldi     0xfffffffe,R2
            ror     R2
            and     R2,R4
            ldp     USRIOW
            sti     R4,@USRIOW
            ldp     USRIOR
            ldi     @USRIOR,R1
            ldi     R7,R5
            and     R1,R5
            cmpi    R5,R7           ;ตรวจสอบว่าเบิตที่ 31 =1 หรือยัง
            bnz     LOOP3
LOOPa1      ldp     USRIOR
            ldi     @USRIOR,R1
            cmpi    R1,R6           ;ทุกบิตต้อง = 0
            bnz     LOOPa1
            mpyi    R3,R4           ;คูณ
Dis         ldp     USRIOW          ;แสดงผล
            sti     R4,@USRIOW
            b       Dis
;-------------------------------------
            .entry  START
START       ldp     USRIOW
            ldi     0,R1
            sti     R1,@USRIOW
            ldi     1,R7
            ldi     0,R6
            ror     R7
            b       LOOP
;-------------------------------------
```

```
;=============================================
;        โปรแกรมที่ 6 การหารเลขแสดงผล
;=============================================
                    .start   "mysect",0x809820
                    .sect    "mysect"
USRIOW              .set     0xc00000
USRIOR              .set     0xc00001
;-----------------------------------------------
LOOPa2              ldp      USRIOR
                    ldi      @USRIOR,R1
                    cmpi     R1,R6          ;ทุกบิตต้อง = 0
                    bnz      LOOPa2

LOOP                ldp      USRIOR         ;รับค่าที่ 1
                    ldi      @USRIOR,R1
                    ldi      R1,R3
                    ldi      0xfffffffe,R2
                    ror      R2
                    and      R2,R3
                    ldp      USRIOW
                    sti      R3,@USRIOW
                    ldp      USRIOR
                    ldi      @USRIOR,R1
                    ldi      R7,R5
                    and      R1,R5
                    cmpi     R5,R7          ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
                    bnz      LOOP

LOOPa               ldp      USRIOR
                    ldi      @USRIOR,R1
                    cmpi     R1,R6          ;ทุกบิตต้อง =0
                    bnz      LOOPa
```

```
LOOP3       ldp     USRIOR          ;รับค่าที่ 2
            ldi     @USRIOR,R1
            ldi     R1,R4
            ldi     0xfffffffe,R2
            ror     R2
            and     R2,R4
            ldp     USRIOW
            sti     R4,@USRIOW
            ldp     USRIOR
            ldi     @USRIOR,R1
            ldi     R7,R5
            and     R1,R5
            cmpi    R5,R7           ;ตรวจสอบว่าบิตที่ 31 =1 หรือยัง
            bnz     LOOP3

LOOPa1      ldp     USRIOR
            ldi     @USRIOR,R1
            cmpi    R1,R6           ;ทุกบิตต้อง = 0
            bnz     LOOPa1

            absi    R3,R3           ;เริ่มต้นการหาร
            cmpi    R3,R6
            bz      ZERO

            absi    R4,R4
            cmpi    R4,R6
            bz      ZERO

            cmpi    R3,R4
            bz      Equl

            bhi     ZERO            ;ถ้าลบแล้วตัวตั้งน้อยกว่าตัวหาร แสดง=0
```

```
                ldi     R3,R2           ;หาร
                ldi     0x1,R7
                ror     R7
divi1           addi    0x1,R0
                rol     R3
                and3    R7,R3,R5
                cmpi    R7,R5
                bnz     divi1

                ldi     R4,R3
                ldi     0,R1
divi2           addi    0x1,R1
                rol     R4
                and3    R7,R4,R5
                cmpi    R7,R5
                bnz     divi2

                ldi     32,R4
                subi3   R0,R4,R0
                subi3   R1,R4,R1
                subi    R1,R0
                ldi     R0,R5
                addi    0x1,R0
                ldi     R5,R6
rep             subi    0x1,R5
                rol     R3
                ldi     0,R4
                cmpi    R4,R5
                bnz     rep

Sub             subi    0x1,R0
                subi3   R3,R2,R1
```

```
                ldi     0,R4
                cmpi    R1,R4
                bge     sub2

sub1            ldi     R1,R2
                rol     R2
                addi    0x1,R2
                cmpi    R0,R4
                bnz     Sub

                ldi     R2,R4
                ldi     0,R7
                b       Dis1

sub2            rol     R2
                cmpi    R0,R4
                bnz     Sub

                ldi     R2,R4
                ldi     0,R7
                b       Dis1

Dis1            subi    0x1,R6
                addi    1,R7
                rol     R7
                ldi     0,R1
                cmpi    R6,R1
                bnz     Dis1

                addi    1,R7
                and     R7,R4
                b       Dis
```

```
Equl        ldi     1,R4
            b       Dis

ZERO        ldi     0,R4
            b       Dis             ;จบลูปการหาร

Dis         ldi     0x1,R7          ;แสดงผล
            ror     R7
            ldi     0,R6

LOOP7       ldp     USRIOW
            sti     R4,@USRIOW
            b       LOOP7
;-------------------------------------
            .entry  START
START       ldp     USRIOW
            ldi     0,R1
            sti     R1,@USRIOW
            ldi     1,R7
            ldi     0,R6
            ror     R7
            b       LOOP
;-------------------------------------
```

หนังสืออ้างอิง

1. "TMS320C3x User's Guide", Texas Instrument, 1996
2. "TMS320C3x DSP Starter Kit User's Guide", Texas Instrument, 1996
3. Peter Kraniauskas, "Transforms In Signals and Systems", Addison-Wesley Publishers Ltd., Second Edition, 1993
4. ยืน ภู่วรวรรณ, "การประมวลผลสัญญาณดิจิตอล", วารสารเซมิคอนดัคเตอร์อิเล็กทรอนิกส์ ฉบับที่ 76, 2530, หน้า 148-154
5. ยอดเยี่ยม ทิพย์สุวรรณ์, การประยุกต์ใช้ดิจิตอลซิกแนลโปรเซสซิ่งในงานมัลติมีเดีย, โครงงานวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์, 2538-2539