บอร์ดทดลองการประมวลผลสัญญาณเชิงเลขและการประยุกต์ใช้งาน

DIGITAL SIGNAL PROCESSING BOARD

WITH

APPLICATION

โดย

นาย ธิติพงษ์	รัตภาสกร	35103227
นาย ธีรยศถ์	ธีรเทิดกิตติ	35103228
นาย บุญชัย	ฉัตร์ไกรเลิศ	35103231

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง บอร์ดทดลองการประมวลผลสัญญาณเชิงเลขและการประยุกต์ใช้งาน

DIGITAL SIGNAL PROCESSING BOARD WITH APPLICATION

ผู้จัดทำ

1. นายธิติพงษ์	รัตภาสกร	35103227
2. นายธีรยศ์	ธีรเทิดกิตติ	35103228
3. นายบุญชัย	ฉัตรไกรเลิศ	35103231

..... อาจารย์ที่ปรึกษา

(อาจารย์ ประทีป บัญญัตินพรัตน์)

บอร์ดทดลองการประมวลผลสัญญาณเชิงและการประยุกต์ใช้งาน

โดย นาย ธิติพงษ์ รัตภาสกร 35103227
นาย ธีรยศ์ ธีรเทิดกิตติ 35103228
นาย บุญชัย ฉัตรไกรเลิศ 35103231

อาจารย์ที่ปรึกษา อาจารย์ ประทีป บัญญัตินพรัตน์

บทคัดย่อ

ปัจจุบันสาขาการประมวลผลสัญญาณเชิงเลข (Digital Signal Processing) ได้เข้ามามีบทบาทต่อเทคโนโลยีด้านอิเล็กทรอนิกส์และคอมพิวเตอร์ โดยการประมวลผลสัญญาณเชิงเลขนี้ได้พัฒนาขึ้นมาทดแทนการประมวลผลสัญญาณเชิงอุปมาน เพื่อเพิ่มประสิทธิภาพทางด้านความแม่นยำ, ความละเอียด และความสะดวก ผลของการประยุกต์ใช้งานการประมวลผลสัญญาณเชิงเลข ได้ถูกนำไปใช้กับสาขาอื่น ๆ มากมาย อาทิ เช่น ด้านการสื่อสารโทรคมนาคม, ด้านการประมวลผล และจดจำภาพและเสียง, ด้านเครื่องมือวัด และอื่น ๆ อีกมากมาย

สำหรับปริญญานิพนธ์ได้เสนอการจัดสร้างบอร์ดอเนกประสงค์ที่สามารถทดลองขั้นตอนวิธี (Algorithm) ของสาขาการประมวลผลสัญญาณเชิงเลข เนื่องจากเพื่อให้สะดวกในการทดลอง จึงได้ออกแบบบอร์ดทดลองให้สามารถที่จะติดต่อและรับคำสั่งการทำงานจากเครื่องไมโครคอมพิวเตอร์ และได้ทดลองประยุกต์บอร์ดทดลองกับระบบการออกเสียงจากตัวหนังสือ (Text To Speech TTS)

DIGITAL SIGNAL PROCESSING BOARD

WITH

APPLICATION

By Mr. Thitiphong Rattapaskorn 35103227

Mr. Teerayot Teeraterdkitti 35103228

Mr. Boonchai Chatkrailert 35103231

Advisor Pratheep Bunyatnoparat

ABSTRACT

Nowly, The science of Digital Signal Processing (DSP) has to role of electronics and computer technology. Development of DSP to instead Analog Signal Processing. It increase accuracy, convenience and etc. DSP has to applies of others science example Telecommunications, Graphics/Image Processing, Pattern Recognition, Instrumentation and etc.

In this thesis have to introduce of DSP Board. It can experiment of digital signal processing. Howevery, DSP Board has to design and provide of convenience. It can to contract and receive command from micro computer. The example of application it can applie DSP Board to Text To Speech system (TTS).

สารบัญ

สารบัญรูปภาพ

บทที่ 1
ทฤษฎีบทการประมวลผลสัญญาณเชิงเลข

ก่อนที่เราจะทำการกล่าวถึงทฤษฎีการประมวลผลสัญญาณเชิงเลข ขอให้ทำความเข้าใจกับคำนิยามของศัพท์เทคนิคที่เกี่ยวข้องกับระบบการประมวลผลสัญญาณเชิงเลข เพื่อให้การสื่อความหมายและความเข้าใจเป็นแบบอย่างเดียวกัน

1.1 นิยามของสัญญาณ

1. สัญญาณเชิงอุปมาน (Analog signal) ใช้กับการกล่าวถึงสัญญาณที่มีรูปคลื่น (waveform) ที่แปรไปอย่างต่อเนื่องกับพิสัยเวลา โดยที่แอมพลิจูด (amplitude) หรือค่าขนาดของสัญญาณก็มีการแปรไปอย่างต่อเนื่องด้วย เช่น สัญญาณไซน์ ($\sin \omega t$)

2. สัญญาณเชิงเวลาต่อเนื่อง (Continuous-time signal) สัญญาณแบบนี้รูปคลื่นของสัญญาณแปรค่าไปอย่างต่อเนื่องกับพิสัยเวลา แต่แอมพลิจูดไม่ได้เจาะจงว่าต้องแปรไปอย่างต่อเนื่อง ซึ่งก็หมายถึงว่าอาจแปรอย่างไม่ต่อเนื่องก็ได้ เพราะฉะนั้นอาจกล่าวได้ว่า สัญญาณเชิงอุปมานเป็นสัญญาณเชิงเวลาต่อเนื่องได้

3. สัญญาณเชิงเวลาเต็มหน่วย (Discrete-time signal) เป็นสัญญาณ $x(t)$ ที่ค่าของฟังก์ชันกำหนดเฉพาะเซตของเวลาที่แน่นอนอันหนึ่งเท่านั้น สัญญาณแบบนี้อาจแบ่งตามลักษณะของแอมพลิจูดได้เป็น 2 แบบ คือ

3.1 สัญญาณเชิงข้อมูลเต็มหน่วย (Discrete data signal) สัญญาณแบบนี้แอมพลิจูดจะต่อเนื่อง หรือกล่าวได้ว่าแอมพลิจูดมีค่าเท่ากันทุกประการกับสัญญาณเชิงอุปมานที่เป็นตัวต้นแบบในการสุ่มตัวอย่าง (sampling) ตัวอย่างเช่น สัญญาณออกของ วงจรสุ่มและคงค่าสัญญาณ จะจัดเป็นสัญญาณเชิงเต็มหน่วย สัญญาณแบบนี้บางที่อาจเรียกว่า ข้อมูล หรือ สัญญาณเชิงเต็มหน่วย

3.2 สัญญาณเชิงเลข (Digital signal) สัญญาณแบบนี้แอมพลิจูดของสัญญาณมีค่าเฉพาะเซตของค่าที่แน่นอนเซตหนึ่งเท่านั้น เช่นสัญญาณที่ออกจากวงจร A/D

4. ระบบเวลาจริง (Real-time system) โดยทั่วไปคำนี้ใช้กับระบบการประมวลผลสัญญาณที่การคำนวณการประมวลผลทำได้เสร็จสิ้นก่อนที่จะมีการสุ่มตัวอย่างสัญญาณลำดับใหม่ อย่างไรก็ตามคาบเวลาในการสุ่มตัวอย่างสัญญาณควรมีค่าน้อย

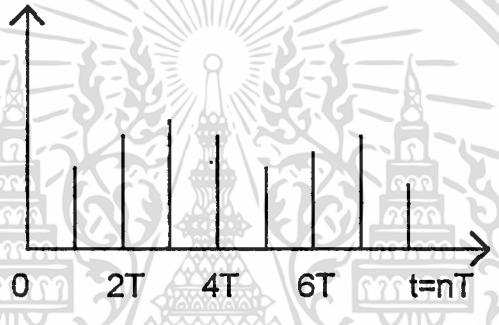
5. ลำดับ (Sequence) หนังสือบางเล่มอาจเรียกว่าเป็น ลำดับข้อมูล หรือ ลำดับสัญญาณ หรือ ลำดับ ก็ได้ ซึ่งคำว่า ลำดับ อาจใช้แทนสัญญาณเชิงเต็มหน่วย หรือ สัญญาณเชิงเลขก็ได้

1.2 สัญญาณพื้นฐานของการประมวลผลสัญญาณเชิงเลข

1. ลำดับสัญญาณ (sequence) ถ้าหากเรามีสัญญาณเชิงอุปมาน $x(t)$ แล้วทำการสุ่มตัวอย่างสัญญาณนี้ด้วยช่วงการสุ่มตัวอย่าง (sampling interval) T วินาทีเท่านั้นและเท่ากันโดยตลอด สิ่งที่ได้ต่อไปนี้เรียกว่าลำดับ ซึ่งเขียนแทนได้ดังสมการ

$$x(n) = x(nT) = x(t)\Big|_{t=nT} \quad ; \; - < n <$$

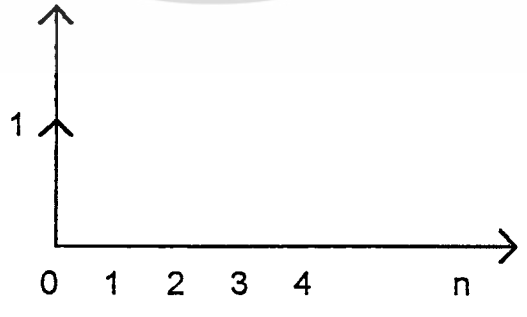
โดยที่ $n = 0, 1, 2, \ldots$ ซึ่งเป็นค่าคงตัวเต็มหน่วย และโดยทั่วไปใช้ $T = 1$ วินาที สมการนี้แสดงว่าลำดับ $x(n)$ ถูกสุ่มตัวอย่างมาจากสัญญาณเชิงอุปมาน $x(t)$ ทุกคาบเวลา nT วินาที



รูปที่ 1-1 ลำดับที่ได้จากการสุ่มตัวอย่างฟังก์ชันต่อเนื่อง

2. อิมพัลส์เชิงเลข (Digital impulse) ซึ่งนิยามได้เป็น

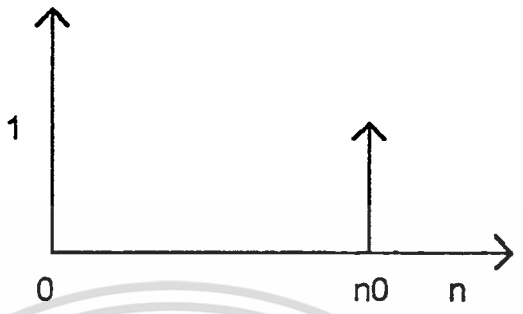
$$\delta(n) = 1 \quad ; \quad n = 0$$
$$= 0 \quad ; \quad n \neq 0$$



รูปที่ 1-2 อิมพัลส์เชิงเลข

3. อิมพัลส์เชิงเลขที่หน่วงเวลาไป n_0 วินาที นิยามได้เป็น

$$\delta(n-n_0) = 1 \quad ; \quad n = n_0$$



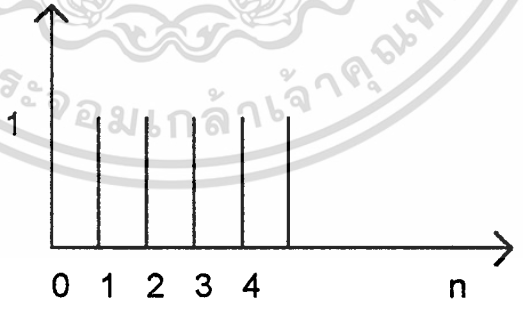
รูปที่ 1-3 อิมพัลส์เชิงเลขที่หน่วงไป n0 วินาที

4. ลำดับหนึ่งหน่วย (Unit step sequence) นิยามได้เป็น

$$u(n) = 1 \quad ; \quad n \geq 0$$
$$0 \quad ; \quad n \leq 0$$

ซึ่งลำดับหนึ่งหน่วยนี้อาจเขียนอยู่ในรูปแบบของผลบวกของอิมพัลส์เชิงเลข ได้คือ

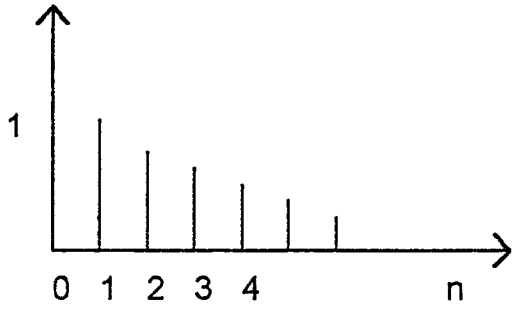
$$u(n) = \sum_{m=-\infty}^{\infty} x(m)\delta(n-m)$$



รูปที่ 1-4 ลำดับหนึ่งหน่วย

5. ลำดับชี้กำลังลดลง (Decay exponential) เมื่อใช้ a < 1 ลำดับแบบนี้สามารถนิยามได้ คือ

$$g(n) = a^n \quad ; \quad n \geq 0$$
$$= 0 \quad ; \quad n \leq 0$$



รูปที่ 1-5 ลำดับชี้กำลังลดลง

ลักษณะรูปร่างของลำดับสัญญาณที่กล่าวถึงนี้ ได้แสดงรวมกันไว้ในรูป 1-2 ..1-5 จากนิยามเหล่านี้ ถ้าหากเรามีลำดับสัญญาณเชิงเลขอยู่ชุดหนึ่งมีลำดับเป็น $x(0), x(1), x(2), \ldots, x(n)$ เราอาจเขียนแทนลำดับสัญญาณเหล่านี้ใหม่ ให้อยู่ในพจน์ของอิมพัลส์เชิงเลขที่หน่วงออกไปได้คือ

$$x(n) = [x(0), x(1), x(2), \ldots, x(m)]$$

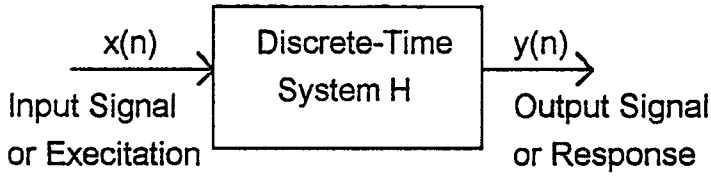
$$x(n) = \sum_{m=-\infty}^{\infty} x(m)\delta(n-m)$$

1.3 ระบบสัญญาณไม่ต่อเนื่อง (Discrete-time system)

ในการใช้งานของระบบประมวลผลสัญญาณเชิงเลข ระบบนี้จะมีอินพุตเป็นสัญญาณไม่ต่อเนื่อง และประมวลผลสัญญาณได้เอาต์พุตเป็นสัญญาณไม่ต่อเนื่องเช่นกัน โดยปรกติระบบจะประมวลผลสัญญาณ $x(n)$ เพื่อให้ได้สัญญาณเอาต์พุต $y(n)$ หรืออาจกล่าวได้อีกอย่างว่า คือ การแปลงสัญญาณ $x(n)$ ของระบบให้เป็นสัญญาณ $y(n)$ สมการที่ใช้แสดงความสัมพันธ์ระหว่าง $x(n)$ กับ $y(n)$ คือ

$$x(n) = \mathrm{H}[x(n)]$$

เมื่อ H คือการแปลง (Transformation) ซึ่งเป็นการประมวลผลของระบบที่มีต่อสัญญาณ $x(n)$ นั่นเอง ความสัมพันธ์ทางคณิตศาสตร์ของสมการ สามารถแสดงได้ดังรูป 1-6



รูปที่ 1-6 ผังของระบบสัญญาณไม่ต่อเนื่อง

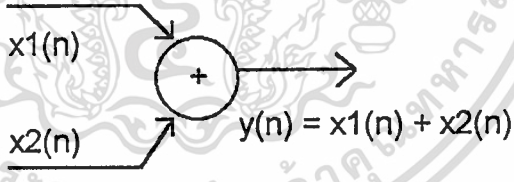
มีหลายวิธีที่ใช้อธิบายถึงคุณลักษณะของระบบซึ่งเป็นการประมวลผล $x(n)$ ให้ได้ $y(n)$ ในบทนี้เราจะกล่าวถึงลักษณะกว้าง ๆ ของระบบ หากต้องการศึกษารายละเอียดสามารถหาอ่านได้จากตำราหรือเอกสารที่อ้างอิงถึงท้ายเล่ม

1.4 องค์ประกอบของระบบการประมวลผลสัญญาณเชิงเลข

หลังจากทำความเข้าใจพื้นฐานของสัญญาณที่ใช้ในการประมวลผลแล้ว เราสนใจกับระบบของสัญญาณไม่ต่อเนื่อง ในระบบการประมวลผลสัญญาณไม่ต่อเนื่องจะมีองค์ประกอบพื้นฐานอันได้แก่ ตัวบวก, ตัวคูณค่าคงที่, ตัวคูณสัญญาณ, ตัวหน่วงหนึ่งหน่วยเวลา และตัวล่วงหน้าหนึ่งหน่วยเวลา ในที่นี้จะแสดงให้เห็นถึงบล็อกไดอะแกรมพื้นฐานที่จะถูกนำไปใช้ต่อเชื่อมกับระบบที่ซับซ้อนยิ่งขึ้น

1.4.1 ตัวบวก (ADDER)

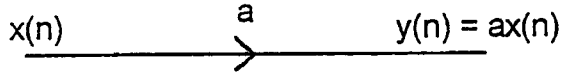
จากรูป 1-7 ข้างล่าง แสดงให้เห็นถึงระบบ (ตัวบวก) ซึ่งจะทำงานโดยการรวมลำดับของสัญญาณสองลำดับเข้าด้วยกัน ซึ่งผลการรวมของสัญญาณคือ $y(n)$ สังเกตว่าไม่จำเป็นต้องมีการเก็บจดจำลำดับสัญญาณใดสัญญาณหนึ่งเพื่อที่จะนำมาใช้ในการรวมสัญญาณ



รูปที่ 1-7 กราฟแสดงตัวบวก

1.4.2 ตัวคูณของค่าคงที่ (CONSTANT MULTIPLIER)

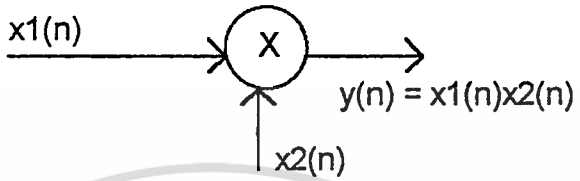
การทำงานของตัวคูณของค่าคงที่จะเป็นไปดังรูปที่ 1-8 ซึ่งแสดงให้เห็นง่าย ๆ ว่าเป็นการใช้ เฟคเตอร์ในการสเกลค่า Input $x(n)$ การทำงานนี้จะไม่ต้องการใช้หน่วยความจำเลย (memoryless)



รูปที่ 1-8 กราฟแสดงตัวคูณค่าคงที่

1.4.3 ตัวคูณสัญญาณ (SIGNAL MULTIPLIER)

รูปที่ 1-9 แสดงให้เห็นถึงการคูณของสองลำดับสัญญาณ ซึ่งจะให้ลำดับสัญญาณผลลัพธ์เป็น $y(n)$ เช่นเดียวกับสองขบวนการที่ผ่านมา จะพบว่าการทำงานของการคูณสัญญาณนี้ไม่ต้องใช้หน่วยความจำเช่นกัน

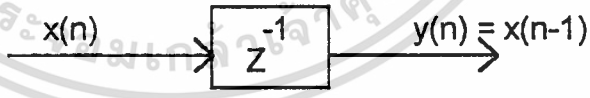


รูปที่ 1-9 กราฟแสดงการคูณของสัญญาณ

1.4.4 ตัวหน่วงหนึ่งหน่วยเวลา (UNIT DELAY ELEMENT)

ตัวหน่วงหนึ่งหน่วยเวลา เป็นระบบเฉพาะที่ใช้สำหรับหน่วงสัญญาณที่จะผ่านตัวมัน ให้สัญญาณภายหลังไปหนึ่งแซมเปิล รูปที่ 1-10 เป็นการแสดงถึงระบบดังกล่าวถ้าหาก input ของระบบเป็น $x(n)$ ก็จะได้ output เป็น $xd(n-1)$ โดยความเป็นจริงแล้วแซมเปิลที่ $x(n-1)$ จะถูกเก็บไว้ในหน่วยความจำ ณ เวลาที่ $(n-1)$ และสัญญาณดังกล่าวจะถูกเรียกออกจากหน่วยความจำที่เวลา (n)

ตัวหน่วงนี้จำเป็นต้องใช้หน่วยความจำ โดยสัญญลักษณ์ Z^{-1} จะหมายถึงการหน่วงสัญญาณไปหนึ่งหน่วยเวลา

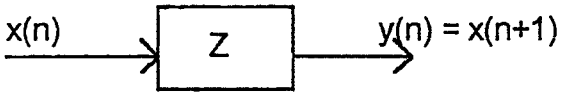


รูปที่ 1-10 กราฟแสดงตัวหน่วงเวลา

1.4.5 ตัวล่วงหน้าหนึ่งหน่วยเวลา (UNIT ADVANCE ELEMENT)

ตัวล่วงหน้าหนึ่งหน่วยเวลานี้จะทำงานตรงข้ามกับตัวหน่วงหนึ่งหน่วยเวลา ตัวล่วงเวลานี้จะเลื่อนสัญญาณ input $x(n)$ ให้เกิดก่อนหนึ่งแซมเปิล กล่าวคือให้ปรากฏเป็น $x(n-1)$ ณ เวลา (n) รูปที่ 1-11 แสดงให้เห็นถึงการทำงานดังกล่าวโดยตัว Z จะเป็นสัญญลักษณ์ของการเกิดล่วงหน้าหนึ่งหน่วยเวลา ในทางกายภาพคงไม่เกิดลักษณะการทำงาน

ดังกล่าว ทั้งนี้เพราะเป็นการมองเห็นสัญญาณที่เกิดล่วงหน้า แต่ถ้าเราเก็บสัญญาณลงในหน่วยความจำของเครื่องคอมพิวเตอร์ เราสามารถเรียกสัญญาณแซมเปิลใด ๆ ณ เวลาใดก็ได้



รูปที่ 1-11 กราฟแสดงการเกิดล่วงหน้าหนึ่งหน่วยเวลา

บทที่ 2

ฟูเรียส์ (Fourier)

บทนี้เราจะศึกษาถึงการวิเคราะห์สัญญาณด้วยอนุกรมฟูเรียส์ที่เราสามารถนำมาใช้แสดงสัญญาณเป็นคาบ โดยจะให้อยู่ในรูปการรวมเอาสัญญาณรูปคลื่นไซน์จำนวนอนันต์เข้าด้วยกัน ต่อจากนั้นก็จะมีการพัฒนาการแปลงฟูเรียส์ที่ทำงานในลักษณะเดียวกันแต่จะใช้วิเคราะห์สัญญาณที่ไม่เป็นคาบ ผลลัพธ์จากการวิเคราะห์จะให้สเปคตรัม (Spectrum) หรือแถบความถี่ของสัญญาณที่ถูกวิเคราะห์ทำให้รู้ว่าสัญญาณนั้นประกอบด้วยความถี่อะไรบ้าง

2.1 อนุกรมฟูเรียส์

ให้สัญญาณ $f(t)$ เป็นสัญญาณที่ซ้ำ ๆ กันเป็นคาบทุก ๆ T วินาที ซึ่งสามารถเขียนฟังก์ชั่นของสัญญาณ $f(t)$ แทนได้ด้วยอนุกรมฟูเรียส์ กล่าวคือ

$$f(t) = a_0 + \sum_{n=1}^{N} a_n \cos n\omega t + \sum_{n=1}^{N} b_n \sin n\omega t \quad (2.1)$$

เมื่อ a_n, b_n เป็นสัมประสิทธิ์ที่สามารถคำนวณได้จากสมการข้างล่างนี้

$$a_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).\cos n\omega t dt \quad (2.2)$$

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).\sin n\omega t dt \quad (2.3)$$

เมื่อ T เป็นคาบเวลา และ $\omega = 2\pi f$ และ เทอม dc คือ a_0 ซึ่งให้ค่าเฉลี่ยของสัญญาณ $f(t)$ ในช่วงคาบ T คำนวณได้ดังนี้

$$a_0 = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)dt \quad (2.4)$$

ข้อสังเกต

1. ถ้า $f(t) = f(-t)$ ฟังก์ชั่น $f(t)$ เป็นฟังก์ชั่นคู่ จะเกิดความสมมาตรรอบจุดกำเนิด และจะให้เพียงเทอมของ COSINE ในสมการ (2.1)

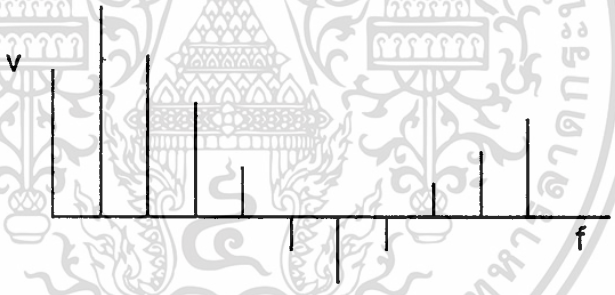
2. ถ้า $f(t) = -f(-t)$ ฟังก์ชั่น $f(t)$ เป็นฟังก์ชั่นคี่ จะปรากฏเพียงเทอมของ SINE เท่านั้นในสมการ (2.1)

3. ถ้า $f(t+T/2) = f(t)$ ในสมการ(2.1) จะปรากฏเพียงเทอมฮาร์โมนิคคู่ (EVEN HARMONICS)

4. ถ้า $f(t+T/2) = -f(t)$ ในสมการ(2.1)จะปรากฏเพียงเทอมฮาร์โมนิคคี่ (ODD HARMONICS)

2.2 แถบความถี่ที่ไม่ต่อเนื่อง (DISCRETE SPECTRUM)

อนุกรมฟูเรียส์ที่ใช้แทนฟังก์ชั่นในแกนเวลา $f(t)$ โดยเราจะแสดงให้เห็นองค์ประกอบทางความถี่ของสัญญาณนั้น องค์ประกอบความถี่เหล่านี้รวมกันเป็นแถบความถี่ที่ไม่ต่อเนื่อง โดยขนาดของแต่ละความถี่ถูกกำหนดโดยค่าสัมประสิทธิ์ a_n และ b_n ทุกความถี่จะเป็นฮาร์โมนิคของความถี่พื้นฐานที่เท่ากับ $/T$ และช่วงกว้างของความถี่ต่าง ๆ นี้จะเรียกว่าแบนด์วิดท์ของสัญญาณ



รูป 2.1 แถบความถี่ไม่ต่อเนื่อง

2.3 ตัวอย่างชนิดของอนุกรม (TYPICAL SERIES)

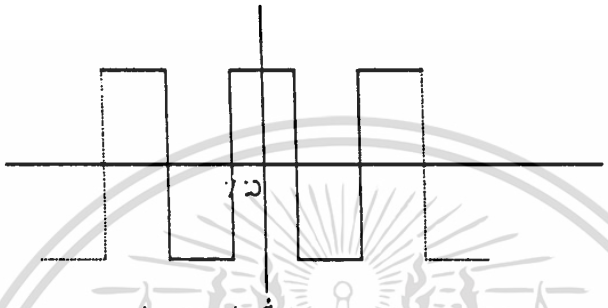
ถึงแม้แถบความถี่อาจจะประกอบด้วยความถี่ต่าง ๆ จำนวนอนันต์ก็ตาม ขนาดของความถี่จะลดลงเรื่อย ๆ เมื่อความถี่สูงขึ้น (ค่าเพิ่มขึ้น) สำหรับในทางปฏิบัติแล้วเราพิจารณาเพียงความถี่จำนวนจำกัดที่พอเพียงกับการติดต่อสื่อสาร

ข้อควรคำนึงในการประมวลผลสัญญาณเชิงเลข โดยเฉพาะในด้านการสื่อสารนั้น จุดสำคัญของการติดต่อสื่อสาร คือ การประหยัดในการใช้แถบความถี่ (band with) ในระบบการติดต่อสื่อสาร ถ้าหากเรารู้ถึงแถบของความถี่จะช่วยให้ระบบการส่งและการรับสัญญาณทำได้อย่างมีประสิทธิภาพและช่วยให้ประหยัดด้วย สำหรับลักษณะรูปคลื่นสัญญาณ

ที่ปรากฏในระบบการประมวลผลสัญญาณเชิงเลขมีหลายรูปแบบ ในที่นี้จะแสดงอนุกรมสัญญาณเฉพาะที่ใช้กันมาก

องค์ประกอบของฟูเรียร์สำหรับรูปคลื่นตัวอย่างชนิดต่าง ๆ พอจะยกตัวอย่างได้ดังนี้

1. รูปคลื่นสี่เหลี่ยมสมมาตร (SYMMETRICAL SQUARE WAVE)



รูป 2-2 รูปคลื่นแบบสมมาตร

เนื่องจาก $f(t)$ มีลักษณะสมมาตรทางแนวนอน ค่าเฉลี่ยของพื้นที่จึงเป็นศูนย์ ทำให้เทอม dc มีค่าเท่ากับศูนย์ ($a_0 = 0$) และโดยที่ $f(t) = f(-t)$ จึงทำให้สัญญาณดังกล่าวจะมีเฉพาะเทอมของ COSINE เท่านั้น กล่าวคือ $b_n = 0$ ส่วนค่า a_n คำนวณได้ดังนี้

$$a_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).\cos n\omega t dt$$

$f(t)$ = -1 จาก -T/2 ถึง -T/4

= 1 จาก -T/4 ถึง T/4

= -1 จาก T/4 ถึง T/2

$$a_n = \frac{2}{T}\{\int_{-\frac{T}{2}}^{-\frac{T}{4}}(-1)\cos n\omega t dt + \int_{-\frac{T}{4}}^{\frac{T}{4}}(1)\cos n\omega t dt + \int_{-\frac{T}{4}}^{\frac{T}{2}}(-1)\cos n\omega t dt\}$$

$$= \frac{2}{T}\{-\frac{1}{n\omega}\int_{-\frac{T}{2}}^{-\frac{T}{4}}\cos n\omega t dt + \frac{1}{n\omega}\int_{-\frac{T}{4}}^{\frac{T}{4}}\cos n\omega t dt - \frac{1}{n\omega}\int_{-\frac{T}{4}}^{\frac{T}{2}}\cos n\omega t dt\}$$

$$= \frac{2}{n\omega T}\{-[\sin n\omega t\Big|_{-\frac{T}{2}}^{-\frac{T}{4}}] + [\sin n\omega t\Big|_{-\frac{T}{4}}^{\frac{T}{4}}] - \sin n\omega t\Big|_{\frac{T}{4}}^{\frac{T}{2}}]\}$$

$$= \frac{2}{n\omega T}\{\sin\frac{n\omega T}{4} - \sin\frac{n\omega T}{2} + \sin\frac{n\omega T}{4} + \sin\frac{n\omega T}{4} - \sin\frac{n\omega T}{2}$$

$$+ \sin\frac{n\omega T}{4}\}$$

$$= \frac{2}{n\omega T}\{4\sin\frac{n\omega T}{4} - 2\sin\frac{n\omega T}{2}\}$$

เนื่องจาก $\omega T = (2\pi f).T = 2\pi$ ดังนั้นจะได้

$$a_n = \frac{1}{n\pi}\{4\sin\frac{nT}{2} - 2\sin n\pi\}$$

แต่ $\sin n\pi = 0$ เมื่อ n = 0, 1, 2, ... $\propto$

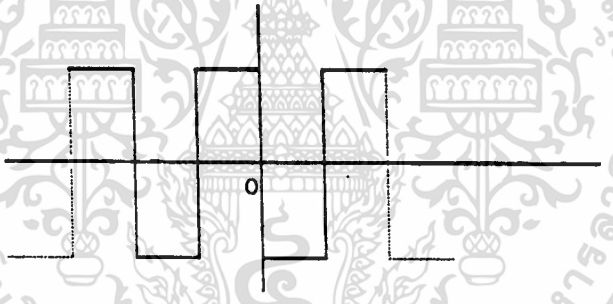
ดังนั้น $a_n = \frac{4}{n\pi}\sin\frac{n\pi}{2}$

$$a_1 = \frac{4}{\pi}\sin\frac{\pi}{2} = \frac{4}{2} \qquad a_2 = \frac{2}{\pi}\sin\pi = 0$$

$$a_3 = \frac{4}{3\pi}\sin\frac{3\pi}{2} = -\frac{4}{3\pi} \qquad a_4 = \frac{1}{\pi}\sin 2\pi = 0$$

$$f(t) = \frac{4}{\pi}(\cos\omega t - \frac{1}{3}\cos 3\omega t + \cos 5\omega t....)$$

2. รูปคลื่นสี่เหลี่ยมแบบไม่สมมาตร (ASYMTRICAL SQUARE WAVE)



รูป 2-3 รูปคลื่นแบบไม่สมมาตร

ฟังก์ชั่น $f(t)$ เป็นรูปคลื่นที่สมมาตรทางแนวนอน ดังนั้นค่าเฉลี่ยใต้พื้นที่จึงเป็นศูนย์ ทำให้เทอม $a_0 = 0$ และเนื่องจาก $f(t) = f(-t)$ จะได้เทอม $a_n = 0$ เหลือเฉพาะเทอม b_n ที่ต้องคำนวณ

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).\sin n\omega t dt$$

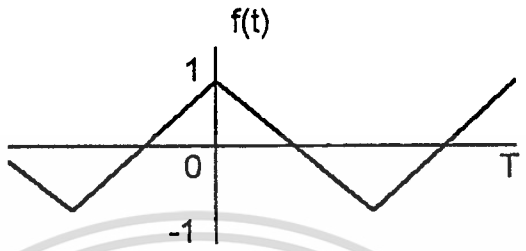
$f(t) = -1$ จาก $-T/2$ ถึง 0

$= 1$ จาก 0 ถึง $T/2$

จะได้

$$f(t) = \frac{4}{\pi}[\sin \omega t + \frac{1}{3}\sin 3\omega t + \frac{1}{5}\sin 5\omega t + ...]$$

3. รูปคลื่นสามเหลี่ยม (TRIANGULAR WAVE)

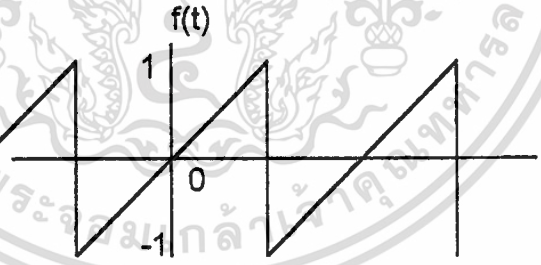


รูป 2-4 รูปคลื่นสามเหลี่ยม

จากรูปข้างบน จะได้ว่า

$$f(t) = \frac{8}{\pi^2}(\cos \omega t + \frac{1}{9}\cos 3\omega t + \frac{1}{25}\cos 5\omega t + ...)$$

4. รูปคลื่นฟันปลา (SAWTOOTH WAVE)



รูปที่ 2-5 รูปคลื่นฟันปลา

จากรูปข้างบน จะได้ว่า

$$f(t) = \frac{2}{\pi}(\sin \omega t - \frac{1}{2}\sin 2\omega t + \frac{1}{3}\sin 3\omega t + ...)$$

2.4 รูปคอมเพล็กซ์ (COPMPLEX FORM)

ฟังก์ชั่น $f(t)$ นี้สามารถเขียนแทนได้ด้วยปริมาตรคอมเพล็กซ์ อีกวิธีหนึ่ง โดยจากเทอมของสัญญาณรูปคอมเพล็กซ์ ซึ่งจะให้ว่า

$$\cos n\omega t = \frac{e^{jn\omega t} + e^{-jn\omega t}}{2}$$

$$\sin n\omega t = \frac{e^{jn\omega t} + e^{-jn\omega t}}{2j}$$

ทำการแทนค่าของ $\cos n\omega t$ และ $\sin n\omega t$ ลงในอนุกรมของฟูเรียร์ ก็จะได้

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \left(\frac{e^{jn\omega t} + e^{-jn\omega t}}{2}\right) + \sum_{n=1}^{\infty} b_n \left(\frac{e^{jn\omega t} + e^{-jn\omega t}}{2j}\right)$$

$$= a_0 + \sum_{n=1}^{\infty} \left\{\frac{(a_n - jb_n)e^{jn\omega t}}{2} + \frac{(a_n + jb_n)e^{-jn\omega t}}{2}\right\}$$

กำหนดให้

$$C_n = \frac{1}{2}(a_n - jb_n)$$

$$C_{-n} = \frac{1}{2}(a_n + jb_n)$$

$$C_0 = a_0 \frac{1}{T}$$

โดย C_{-n} เป็น Complex conjugate ของ C_n ดังนั้นการคำนวณหาค่า C_n ทำได้โดย

$$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)[\cos n\omega t - j\sin n\omega t]dt$$

$$C_{-n} = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)[\cos n\omega t + j\sin n\omega t]dt$$

$$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).e^{-jn\omega t}dt$$

$$C_{-n} = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).e^{jn\omega t}dt$$

ในที่สุดจะได้

$$f(t) = C_0 + \sum_{n=1}^{\infty} C_n e^{jn\omega t} + \sum_{n=-\infty}^{-1} C_n e^{jn\omega t}$$

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{jn\omega t}$$

2.5 การแปลงฟูเรียร์ (FOURIER TRANSFORM)

เทคนิคของอนุกรมฟูเรียร์สามารถปรับปรุงมาใช้กับรูปคลื่นที่ไม่เป็นคาบ อย่างเช่นลูกพัลซ์เดี่ยว ๆ (simple pulse) โดยการให้ค่า T เป็นค่าอนันต์ (∞)

สมมติว่า $f(t)$ เดิมเป็นสัญญาณที่เป็นคาบจะได้ว่า

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{jn\omega t}$$

เมื่อ $$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t).e^{-jn\omega t}dt$$

จากเงื่อนไขที่ว่าสัญญาณนั้นเป็นพัลซ์เดี่ยว ๆ ทำให้

$$T \to \infty \quad ; \quad \omega = \frac{2\pi}{T} \to d\omega$$

ดังนั้น $$\frac{1}{T} = \frac{\omega}{2\pi} \to \frac{d\omega}{2\pi}$$

ยิ่งไปกว่านั้น ฮาร์โมนิคส์ที่ n ของอนุกรมฟูเรียร์ $n\omega$ จะกลายเป็น $nd\omega$ ซึ่งถือว่ามีค่าเป็น ω นั่นเอง ในที่สุดเครื่องหมายซิกม่า (Sigma) จะกลายเป็นเครื่องหมายอินทริกรัล (Integral) ดังนั้น

$$C_n = \frac{d\omega}{2\pi}\int_{-\infty}^{\infty} f(t).e^{-j\omega t}dt$$

$$f(t) = \int_{-\infty}^{\infty}\frac{d\omega}{2\pi}[\int_{-\infty}^{\infty} f(t).e^{-j\omega t}dt]e^{j\omega t}$$

เทอมที่อยู่ในเครื่องหมายก้ามปู เป็นเทอมของความถี่เพียงอย่างเดียว ซึ่งให้เป็น $g(\omega)$ โดย

$$g(\omega) = \int_{-\infty}^{\infty} f(t).e^{-j\omega t}dt$$

ซึ่ง $g(\omega)$ เรียกว่าการแปลงฟูเรียร์ของฟังก์ชั่น $f(t)$ ดังนั้นสมการของ $f(t)$ จะกลายเป็น

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} g(\omega).e^{-j\omega t}d\omega$$

2.5.1 ความต่อเนื่องของสเปคตรัม (CONTINUOUS SPECTRUM)

สัญญาณที่มีลักษณะเป็นพัลซ์เดี่ยว ๆ สามาถเขียนแทนด้วยผลรวมของความถี่ต่าง ๆ จำนวนอนันต์ ทั้งนี้เนื่องจาก $g(\omega)$ มี ω เป็นเทอมของความถี่ จากการอินทริเกรทนี้เองทำให้ได้ว่าสเปคตรัมที่ได้มีความต่อเนื่องตลอด ซึ่งตรงกันข้ามกับกรณีของรูปคลื่น

ที่เป็นคาบที่ให้สเปคตรัมไม่ต่อเนื่อง โดยความจริงแล้วทุกความถี่จะอยู่ใกล้กันมาก ทั้งนี้เพราะช่องว่างระหว่างความถี่นั้น คือ $1/T$ จะมีค่าเป็นศูนย์ เมื่อ T เป็น $\propto$

โดยทั่วไปแล้ว $g(\omega)$ เป็นเทอมคอมเพล็กซ์ โดยที่ขนาดและเฟสสามารถนำมาพล็อตเป็นแถบความถี่ของสัญญาณ $f(t)$ ขนาด $|g(\omega)|$ จะแปรไปตามคาบและ $|g(\omega)|$ เป็นพื้นที่ใต้กราฟภายในช่วง $d\omega$ ถูกเรียกว่า สเปคตรัลเดนซิตี้ (SPECTRAL DENSITY) แต่เนื่องจาก $d\omega/2\pi$มีค่าเกือบเป็นหนึ่ง ดังนั้นพื้นที่ของ $|g(\omega)|d\omega/2\pi$ จึงมีค่าเพียงขนาดของ $|g(\omega)|$ ซึ่งเป็นแอมพลิจูด เดนซิตี้ (AMPLITUDE DENSITY)

2.5.2 การแปลงฟูเรียส์ของสัญญาณไม่ต่อเนื่อง (Discrete Fourier Transform)

จากความเจริญก้าวหน้าทางเทคโนโลยี ทำให้ปัจจุบันมีเครื่องคอมพิวเตอร์ใช้กันอย่างทั่วถึง ด้วยเหตุนี้เองการคำนวณการแปลงฟูเรียส์ด้วยการอินทริเกรทจึงไม่นิยมกันเพราะยุ่งยากและเสียเวลาทำให้มีการคิดโครปแกรมการแปลงฟูเรียส์ขึ้น ดังนั้นสัญญาณต่อเนื่องที่จะนำแปลงด้วยฟูเรียส์จึงต้องมีการสุ่ม (SAMPLING) ให้เป็นสัญญาณที่ไม่ต่อเนื่องเพื่อนำเอาขนาดของแซมเปิ้ลไปคำนวณนี้

สูตรในการแปลงฟูเรียส์ของสัญญาณต่อเนื่องคือ

$$G(f) = \int_{-\infty}^{\infty} g(t).e^{-j2\pi f} dt$$

$$g(\omega) = \int_{-\infty}^{\infty} f(t).e^{-j2\pi f} dt$$

เมื่อแปลงให้เป็นการแปลงฟูเรียส์ของสัญญาณที่ไม่ต่อเนื่องจะได้

$$G(u) = \frac{1}{N}\sum_{x=0}^{N-1} g(x)e^{-\frac{j2\pi ux}{N}} \qquad ; u = 0, 1, 2, ..., N-1$$

เมื่อ N เป็นจำนวนแซมเปิ้ล ในการแปลงกลับฟูเรียส์ของสัญญาณที่ไม่ต่อเนื่อง คือ

$$g(x) = \sum_{u=0}^{N-1} G(u)e^{\frac{j2\pi ux}{N}} \qquad ; x = 0, 1, 2, ..., N-1$$

ตัวอย่าง จากสัญญาณไม่ต่อเนื่องในรูปข้างบนจะให้ N = 4 โดย $g(x) = \{2, 3, 4, 4\}$ สเปคตรัลของสัญญาณดังกล่าวคำนวณได้จากสูตรการแปลงฟูเรียส์ของสัญญาณที่ไม่ต่อเนื่อง คือ

$$G(u) = \frac{1}{4}\sum_{x=0}^{3} g(x)e^{-\frac{j2\pi ux}{4}}$$

$$G(0) = \frac{1}{4}\sum_{x=0}^{3} g(x) = \frac{1}{4}[g(0)+g(1)+g(2)+g(3)]$$
$$= \frac{1}{4}[2+3+4+4]$$
$$= 3.25$$

สูตรการแปลงฟูเรียส์ของสัญญาณต่อเนื่อง

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft}dt$$

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(f)e^{j\omega t}d\omega$$

สูตรการแปลงฟูเรียส์ของสัญญาณที่ไม่ต่อเนื่อง

$$H(k) = \frac{1}{N}\sum_{m=0}^{N-1} h(m)e^{-\frac{j2\pi km}{N}} \quad ; \quad k = 0, 1, 2, \ldots, N-1$$

$$h(m) = \sum_{m=0}^{N-1} h(m)e^{\frac{j2\pi km}{N}} \quad ; \quad m = 0, 1, 2, \ldots, N-1$$

จาก sequence ของสัญญาณ $h(m) = \{2, 3, 4, 4\}$ ให้หา spectrum ของสัญญาณนั้นคือ N=4

$$H(k) = \frac{1}{4}\sum_{m=0}^{3} h(m)e^{-\frac{j2\pi km}{4}} = \frac{1}{4}\sum_{m=0}^{3} h(m)e^{-\frac{j\pi km}{2}}$$

$$H(0) = \frac{1}{4}\sum_{m=0}^{3} h(m)$$
$$= \frac{1}{4}[h(0)+h(1)+h(2)+h(3)]$$
$$= \frac{1}{4}[2+3+4+4] = 3.25$$

$$H(1) = \frac{1}{4}\sum_{m=0}^{3} h(m)e^{-\frac{j\pi km}{2}}$$
$$= \frac{1}{4}[h(0)e^{-j0}+h(1)e^{-\frac{j\pi}{2}}+h(2)e^{-j\pi}+h(3)e^{-\frac{j3\pi}{2}}]$$
$$= \frac{1}{4}[2.1+3.(-j)+4.(-1)+4(j)]$$
$$= \frac{1}{4}[2-3j-4+4j]$$
$$= \frac{1}{4}[-2+j]$$

$$H(2) = \frac{1}{4}\sum_{m=0}^{3} h(m)e^{-jm\pi}$$
$$= \frac{1}{4}[h(0)e^{-j0}+h(1)e^{-j\pi}+h(2)e^{-j2\pi}+h(3)e^{-j3\pi}]$$

$$= \frac{1}{4}[2.1+3.(-1)+4.(1)+4(-1)]$$

$$= \frac{1}{4}[2-3+4-4]$$

$$= -0.25$$

$$H(3) = \frac{1}{4}\sum_{m=0}^{3} h(m)e^{-\frac{j3m\pi}{2}}$$

$$= \frac{1}{4}[h(0)e^{-j0}+h(1)e^{-\frac{j3\pi}{2}}+h(2)e^{-j3\pi}+h(3)e^{-\frac{j9\pi}{2}}]$$

$$= \frac{1}{4}[2.1+3.(j)+4.(-1)+4(-j)]$$

$$= \frac{1}{4}[2+3j-4-4j]$$

$$= -\frac{1}{4}[2+j]$$

เพราะฉะนั้น สเปคตรัม คือ

$$|H(0)| = 3.25$$

$$|H(1)| = \frac{1}{4}\left|\sqrt{(-2)^2+(1)^2}\right| = \frac{\sqrt{5}}{4}$$

$$|H(2)| = |0.25| = 0.25$$

$$|H(3)| = \frac{1}{4}\left|\sqrt{(-2)^2+(1)^2}\right| = \frac{\sqrt{5}}{4}$$

2.5.3 การแปลงฟูเรียส์อย่างเร็ว (Fast Fourier Transform)

ขั้นตอนวิธี หรือ ลำดับการในการคำนวณฟูเรียส์ให้เร็วมีชื่อเรียกว่า **การแปลงฟูเรียส์อย่างเร็ว** คิดค้นโดย คูลีย์ (J.W. Cooley) กับ ทูคีย์ (J.W. Tukey) ซึ่งได้เสนอไว้ในปี ค.ศ. 1965 หลังจากนั้นทำให้เกิดการพัฒนาวิธีหลายวิธี แต่ในวิธีของ คูลีย์ และ ทูคีย์ ช่วยให้การคำนวณเชิงซ้อนเพียง $N\log_2 N$ หรือลดจำนวนครั้งในการคูณตัวเลขลดลงไปถึง $N/\log_2 N$ เท่า ผลดีอีกประการหนึ่งก็คือทำให้การสร้างวงจรเฉพาะเพื่อการคำนวณ DFT ทำให้ง่าย และคำนวณได้เร็วขึ้น

จากหัวข้อที่ผ่านมาแสดงถึงการแปลงฟูเรียส์สำหรับสัญญาณไม่ต่อเนื่อง จะสังเกตได้ว่าจะมีส่วนที่เราต้องคำนวณซ้ำ ๆ คือ การคูณค่าจำนวนเชิงซ้อนซึ่งมักเป็นเลขเทศนิยมความละเอียด 1-2 เท่า กันมากทำให้เสียเวลาในการคำนวณ ดังในตารางแสดงให้เห็นถึงจำนวนที่ลดลงเมื่อมีการใช้การแปลงฟูเรียส์อย่างเร็ว เช่น N ขนาด 1024 จุด เมื่อใช้ DFT จะต้องคูณจำนวนเชิงซ้อน 1048576 แต่เมื่อใช้ FFT จะเหลือเพียง 10240 เท่านั้น

จากหัวข้อที่แล้วเราได้สูตรการแปลงฟูเรียส์ คือ

$$H(k) = \frac{1}{N}\sum_{m=0}^{N-1} h(m)W^{km} \qquad k = 0, 1, ..., N-1$$

โดยที่

$$= e^{\frac{j2\pi}{N}} \qquad \text{โดย} \qquad j = \sqrt{-1}$$

Length of Transform (N)	DFT Operations (N^2)	FFT Operations $NLOG_2(N)$
8	64	24
16	256	64
32	1024	160
64	4096	384
128	16384	896
256	65536	1024
512	262144	4096
1024	1048576	10240
2048	4194304	22528

ตารางที่ 2-1 เปรียบเทียบจำนวนการคูณค่าเชิงซ้อนระหว่าง DFT กับ FFT

คุณสมบัติและเทคนิคช่วยแปลงฟูเรียส์อย่างเร็วมีดังนี้

1. คุณสมบัติของการแปลงฟูเรียส์ (Motivation to search for an algorithm)

$$H(\frac{N}{2}+l) = \overline{H}(\frac{N}{2}-l) \qquad l = 1, 2, ..., N/2 - 1$$

โดย $\overline{H}(k)$ เป็น Complex Conjugate ของ $H(k)$

2. เทคนิคสำหรับการลดการคำนวณ (Key to Developing the algorithm)

$$m = m_{n-1}2^{n-1} + m_{n-2}2^{n-2} + ... + m_1 2^1 + m_0 2^0$$

โดย $m_v = 0$ หรือ 1 เมื่อ v = 0, 1, ..., n-1 โดย $n = \log_2 N$

ทำนองเดียวกับ

$$k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \ldots + k_1 2^1 + k_0 2^0$$

โดย $k_v = 0$ หรือ 1 เมื่อ v = 0, 1, ..., n-1 โดย $n = \log_2 N$

ถ้าเราให้ $\tilde{h}(m)$ เป็น sequence ของข้อมูลที่ตัวแปรเป็นเลขฐานสองที่ใช้แทน $h(m)$ ชี้เป็นเลขฐานสิบเราจะได้ว่า

$$\begin{aligned} h(m) &= \tilde{h}(m_{n-1}2^{n-1} + m_{n-2}2^{n-2} + \ldots + m_1 2^1 + m_0 2^0) \\ &= \tilde{h}(m_{n-1}, m_{n-2}, \ldots, m_1, m_0) \end{aligned} \quad (2.5)$$

จากสมการที่ (2.5) จะได้ว่า

$$\sum_{m=0}^{N-1} h(m)W^{km} = \sum_{m_0=0}^{1}\sum_{m_1=0}^{1}\cdots\sum_{m_{n-1}=0}^{1} \tilde{h}(m_{n-1}, m_{n-2}, \ldots, m_0)W^{k[m_{n-1}2^{n-1} + m_{n-2}2^{n-2} + \ldots m_0 2^0]} \quad (2.6)$$

ตัวอย่างในกรณีที่ N = 4 หรือ $n = \log_2 4 = 2$ ดังนั้นสมการที่ (2) จะเป็น

$$\begin{aligned} \sum_{m_0}\sum_{m_1} \tilde{h}(m_1, m_0)W^{k[2m_1 + m_0]} &= \sum_{m_0} \{\tilde{h}(0, m_0)W^{km_0} + \tilde{h}(1, m_0)W^{k(2+m_0)}\} \\ &= \sum_{m_0=0}^{1} \tilde{h}(0, m_0)W^{km_0} + \sum_{m_0=0}^{1} \tilde{h}(1, m_0)W^{k[2+m_0]} \\ &= \tilde{h}(0,0) + \tilde{h}(0,1)W^k + \tilde{h}(1,0)W^{2k} + \tilde{h}(1,1)W^{3k} \end{aligned} \quad (2.7)$$

ดังนั้นสมการที่ (2.7) จะได้ว่า

$$\sum_{m_0}\sum_{m_1} \tilde{h}(m_1, m_0)W^{k[2m_1 + m_0]} = \sum_{m=0}^{3} h(m)W^{km}$$ นั่นเอง

ตัวอย่างการนำไปใช้งาน

เราเอากรณีที่ N = 8 เป็นพื้นฐานเบื้องต้นในการอธิบายขบวนการของ FFT นั้นคือ

$$H(k) = \frac{1}{8}\sum_{m=0}^{7} h(m)W^{km} \quad \text{โดย } k = 0, 1, \ldots, 7 \quad (2.8)$$

เมื่อ $W = e^{-\frac{j2\pi}{8}} = e^{-\frac{j\pi}{4}}$ และเราให้ m เขียนอยู่ในรูป binary คือ

$$m = m_2 2^2 + m_1 2^1 + m_0 2^0 \quad (2.9)$$

จากสมการ (2.8) เอา 8 คูณตลอดจะได้

$$8H(k) = \sum_{m=0}^{7} h(m)W^{km} \quad (2.10)$$

เมื่อแปลงเทอมทางขวามือของสมการที่ (6) ให้อยู่ในรูปของเลขฐานสอง จะได้ว่า

$$\begin{aligned} 8H(k) &= \sum_{m_0=0}^{1}\sum_{m_1=0}^{1}\sum_{m_2=0}^{1} \tilde{h}(m_2, m_1, m_0)W^{k[4m_2 + 2m_1 + m_0]} \\ &= \sum_{m_0=0}^{1}\sum_{m_1=0}^{1}\sum_{m_2=0}^{1} \tilde{h}(m_2, m_1, m_0)W^{4km_2}W^{2km_1}W^{km_0} \end{aligned} \quad (2.11)$$

จากสมการที่ (2.11) เราให้ Summation ในสุดเป็น M_2 นั่นคือ

$$M_2 = \sum_{m_2=0}^{1} \tilde{h}(m_2,m_1,m_0)W^{4km_2}$$

แต่เนื่องจาก $W^4 = -1$ และแทน k ด้วยเลขฐานสองจะได้

$$M_2 = \sum_{m_2=0}^{1} \tilde{h}(m_2,m_1,m_0)(-1)^{m_2[4k_2+2k_1+k_0]}$$

แต่เนื่องจาก $(-1)^{m_2[4k_2+2k_1]} = 1$, M_2 สามารถเขียนใหม่เป็น

$$M_2 = \sum_{m_2=0}^{1} \tilde{h}(m_2,m_1,m_0)(-1)^{m_2[4k_2+2k_1+k_0]} \quad (2.12)$$

Summation ของสมการที่ (2.12) เป็นการแทนค่า m_2 ด้วย 0 และ 1 ดังนั้นสมการที่ (2.12) จะมีตัวแปรที่ไม่ทราบค่าคือ k_0, m_1 และ m_0 ดังนั้น M_2 เขียนใหม่เป็น

$$M_2 = \sum_{m_2=0}^{1} \tilde{h}(m_2,m_1,m_0)(-1)^{m_2[4k_2+2k_1+k_0]} = \tilde{h}_1(k_0,m_1,m_0) \quad (2.13)$$

แทนค่าสมการ (2.13) ลงในสมการ (2.11)

$$8H(k) = \sum_{m_0=0}^{1}\sum_{m_1=0}^{1} \tilde{h}_1(k_0,m_1,m_2)W^{2km_1}W^{km_0} \quad (2.14)$$

Summation ชุดในของสมการที่ (2.14) เราสมมติให้เป็น M_1 นั่นคือ

$$M_1 = \sum_{m_1=0}^{1} \tilde{h}_1(k_0,m_1,m_0)W^{2km_1}$$

$$= \sum_{m_1=0}^{1} \tilde{h}_1(k_0,m_1,m_0)(-j)^{(4k_2+2k_1+k_0)m_1} \quad (2.15)$$

โดยที่จากสมการที่ (2.15) นั้น $W^2 = e^{-\frac{j2\pi 2}{4}} = -j$ และเทอม $(-j)^{4k_2m_1} = 1$

ดังนั้น M_1 เขียนได้ใหม่เป็น

$$M_1 = \sum_{m_1=0}^{1} \tilde{h}_1(k_0,m_1,m_0)(-j)^{(+2k_1+k_0)m_1} \quad (2.16)$$

จากสมการที่ (2.16) เมื่อทำการแปรค่า M_1 เป็น 0 และ 1 ดังนั้นเทอม M_1 จะเป็นตัวแปรของ k_0, k_1 และ m_0 นั้นคือสมการ (2.16) จึงกลายเป็น

$$M_1 = \tilde{h}_2(k_0,k_1,m_0) \quad (2.17)$$

แทน (2.17) ลงใน (2.13) จะได้

$$8H(k) = \sum_{m_0=0}^{1} \tilde{h}_2(k_0,k_1,m_0)W^{m_0k_1}$$

ในทำนองเดียวกับ M_2 และ M_1 เมื่อให้ M_0 เป็นผลการ summation ของค่า M_0 นั่นคือ

$$M_0 = \sum_{m_0=0}^{1} \tilde{h}_2(k_0,k_1,m_0)W^{m_0[4k_2+2k_1+k_0]}$$

$$= \sum_{m_0=0}^{1} \tilde{h}_2(k_0,k_1,m_0)(\frac{1-j}{\sqrt{2}})^{[4k_2+2k_1+k_0]m_0} \quad (2.18)$$

หลังจากทำการแปรค่า M_0 ไปแล้วจะพบว่า M_0 เป็นฟังก์ชั่นของ k_0, k_1 และ k_2 ซึ่งเราจะแทนด้วย

$$M_0 = \tilde{h}_3(k_0,k_1,k_2) \quad (2.19)$$

ดังนั้นจากสมการที่ (2.11) เราจะได้คำตอบว่า

$$8H(k) = 8\tilde{H}(k_2,k_1,k_0) = \tilde{h}_3(k_0,k_1,k_2) \quad (2.20)$$

จากสัมประสิทธิ์ (-1), $(-j)$ และ $(1-j)/\sqrt{2}$ ค่าเหล่านี้จะเป็นรากของ $e^{-j2\pi}$ นั่นเอง กล่าวคือ (-1), $(-j)$ และ $(1-j)/\sqrt{2}$ เป็นรากที่ 2, 4 และที่ 8 ของ unity ซึ่งเราจะใช้สัญลักษณ์แทนด้วย

$$A_{2^r} = e^{-\frac{j2\pi}{2^r}} \quad \text{เมื่อ } r = 1, 2, ..., \log_2 N \quad (2.21)$$

โดย A_{2^r} จะมีคุณสมบัติคือ

(i) $A_{2^r} = W^{\frac{N}{2^r}}$ $\quad W = e^{-\frac{j2\pi}{N}}$ (2.22a)

(ii) $(A_{2^r})^{\lambda+l} = -(A_{2^r})^l$ เมื่อ $r = 1, 2, ..., \log_2 N$, $l = 0, 1, ..., 2^{r-1}$

$\lambda = 2^{r-1}$ (2.22b)

(iii) $(A_N)^{\frac{N}{2}} = -1$ (2.22c)

จากสมการที่ (2.13) เราเขียนอยู่ในเทอม จะได้ว่า

$$\tilde{h}_1(k_0,m_1,m_0) = \sum_{m_2} \tilde{h}(m_2,m_1,m_0)A_2^{k_0}$$

นั่นคือ

$$\tilde{h}_1(k_0,m_1,m_0) = \tilde{h}(0,m_1,m_0) + \tilde{h}(1,m_1,m_0)A_2^{k_0} \quad (2.23)$$

สมการที่ (2.23) นี้ จะมีค่าไม่เป็น 0 ก็เป็น 1 จะได้ว่าค่า แต่ละค่าให้ 4 สมการ คือ

<u>case ที่ 1 k_0=0</u> (2.24a)

$$\tilde{h}_1(0,0,0) = \tilde{h}(0,0,0) + \tilde{h}(1,0,0) \Rightarrow h_1(0) = h(0) + h(4)$$

$$\tilde{h}_1(0,0,1) = \tilde{h}(0,0,1) + \tilde{h}(1,0,1) \Rightarrow h_1(1) = h(1) + h(5)$$

$$\tilde{h}_1(0,1,0) = \tilde{h}(0,1,0) + \tilde{h}(1,1,0) \Rightarrow h_1(2) = h(2) + h(6)$$

$$\tilde{h}_1(0,1,1) = \tilde{h}(0,1,1) + \tilde{h}(1,1,1) \Rightarrow h_1(3) = h(3) + h(7)$$

<u>case ที่ 2 k_0=1</u> (2.24b)

$$\tilde{h}_1(1,0,0) = \tilde{h}(0,0,0) + A_2\tilde{h}(1,0,0) \Rightarrow h_1(4) = h(0) - h(4)$$
$$\tilde{h}_1(1,0,1) = \tilde{h}(0,0,1) + A_2\tilde{h}(1,0,1) \Rightarrow h_1(5) = h(1) - h(5)$$
$$\tilde{h}_1(1,1,0) = \tilde{h}(0,1,0) + A_2\tilde{h}(1,1,0) \Rightarrow h_1(6) = h(2) - h(6)$$
$$\tilde{h}_1(1,1,1) = \tilde{h}(0,1,1) + A_2\tilde{h}(1,1,1) \Rightarrow h_1(7) = h(3) - h(7)$$

สมการที่ (2.16) เมื่อแทน $(-j)$ ด้วยเทอม A_4 จะได้ว่า

$$\tilde{h}_2(k_0,k_1,m_0) = \tilde{h}_1(k_0,0,m_0) + \tilde{h}_1(k_0,1,m_0)A_4^{(2k_1+k_0)}$$

จากการ fixed ค่า k_0,k_1 แล้วทำการแปร m_0 ไปแต่ละครั้งจะได้ 2 สมการดังนี้ คือ

case 1 $(k_1,k_0) = (0,0)$ (2.25a)

$$\tilde{h}_2(0,0,0) = \tilde{h}_1(0,0,0) + \tilde{h}_1(0,1,0) \Rightarrow h_2(0) = h_1(0) + h_1(2)$$
$$\tilde{h}_2(0,0,1) = \tilde{h}_1(0,0,1) + \tilde{h}_1(0,1,1) \Rightarrow h_2(1) = h_1(1) + h_1(3)$$

case 2 $(k_1,k_0) = (0,1)$ (2.25b)

$$\tilde{h}_2(1,0,0) = \tilde{h}_1(1,0,0) + A_4\tilde{h}_1(1,1,0) \Rightarrow h_2(4) = h_1(4) + A_4h_1(6)$$
$$\tilde{h}_2(1,0,1) = \tilde{h}_1(1,0,1) + A_4\tilde{h}_1(1,1,1) \Rightarrow h_2(5) = h_1(5) + A_4h_1(7)$$

case 3 $(k_1,k_0) = (1,0)$ (2.25c)

$$\tilde{h}_2(0,1,0) = \tilde{h}_1(0,0,0) + A_4^2\tilde{h}_1(0,1,0) \Rightarrow h_2(2) = h_1(0) - h_1(2)$$
$$\tilde{h}_2(0,1,1) = \tilde{h}_1(0,0,1) + A_4^2\tilde{h}_1(0,1,1) \Rightarrow h_2(3) = h_1(1) - h_1(3)$$

case 4 $(k_1,k_0) = (1,1)$ (2.25d)

$$\tilde{h}_2(1,1,0) = \tilde{h}_1(1,0,0) + A_4^3\tilde{h}_1(1,1,0) \Rightarrow h_2(6) = h_1(4) - A_4h_1(6)$$
$$\tilde{h}_2(1,1,1) = \tilde{h}_1(1,0,1) + A_4^3\tilde{h}_1(1,1,1) \Rightarrow h_2(7) = h_1(5) - A_4h_1(7)$$

สมการที่ (2.25.a) ถึง (2.25.d) เป็นการทำ iteration ครั้งที่ 2 ดังรูปที่ 1 โดยการทำ iteration ครั้งที่ 2 นี้คือ ในสมการที่ (2.21) นั้นเอง

ในที่สุดสมการที่ (2.19) เราเขียนค่าสัมประสิทธิ์ในเทอมของ A_8 จะได้ว่า

$$\tilde{h}_3(k_0,k_1,k_2) = \tilde{h}_2(k_0,k_1,0) + \tilde{h}_2(k_0,k_1,1)A_8^{[4k_2+2k_1+k_0]}$$

ซึ่งจะได้ว่า

case 1 $(k_2,k_1,k_0) = (0,0,0)$ (2.26a)

$$\tilde{h}_3(0,0,0) = \tilde{h}_2(0,0,0) + \tilde{h}_2(0,0,1) \Rightarrow h_3(0) = h_2(0) + h_2(1)$$

case 2 $(k_2,k_1,k_0) = (0,0,1)$ (2.26b)

$$\tilde{h}_3(1,0,0) = \tilde{h}_2(1,0,0) + A_8\tilde{h}_2(1,0,1) \Rightarrow h_3(4) = h_2(4) + A_8h_2(5)$$

case 3 $(k_2,k_1,k_0) = (0,1,0)$ (2.26c)

$$\tilde{h}_3(0,1,0) = \tilde{h}_2(0,1,0) + A_8^2\,\tilde{h}_2(0,1,1) \Rightarrow h_3(2) = h_2(2) + A_8^2\,h_2(3) \quad (2.26d)$$

case 4 $(k_2,k_1,k_0) = (0,1,1)$

$$\tilde{h}_3(1,1,0) = \tilde{h}_2(1,1,0) + A_8^3\,\tilde{h}_2(1,1,1) \Rightarrow h_3(6) = h_2(6) + A_8^3\,h_2(7) \quad (2.26e)$$

case 5 $(k_2,k_1,k_0) = (1,0,0)$

$$\tilde{h}_3(0,0,1) = \tilde{h}_2(0,0,0) + A_8^4\,\tilde{h}_2(0,0,1) \Rightarrow h_3(1) = h_2(0) - A_8^4\,h_2(1)$$

สมการที่เหลือเราใช้คุณสมบัติของ $H(\frac{N}{2}+l) = \overline{H}(\frac{N}{2}-l)$นั่นเอง จากกลุ่มสมการที่ (23) นี้เป็นการทำ iteration ครั้งที่ 3 นั้นคือ r = 3 ในสมการที่ (17) นั่นเอง

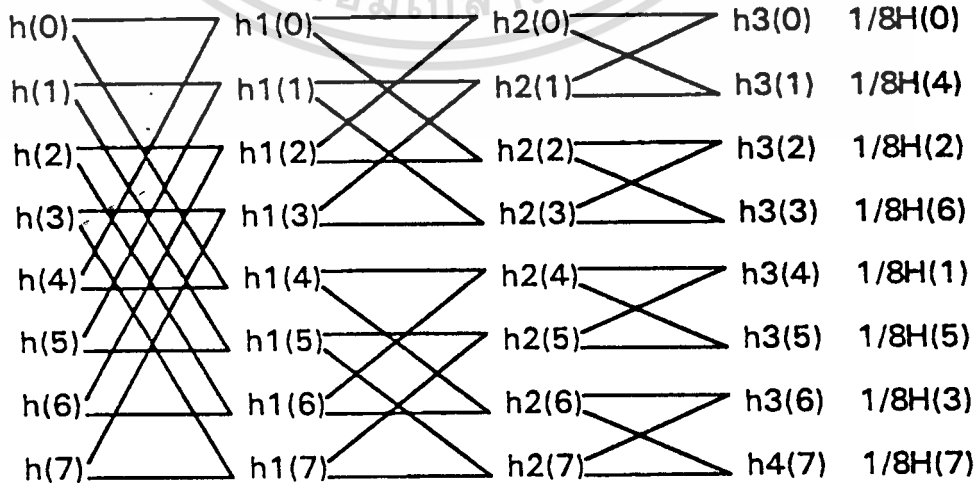
ในที่สุดจะได้ว่า

$$h_3(0) = 8H(0)$$
$$h_3(4) = 8H(1)$$
$$h_3(2) = 8H(2)$$
$$h_3(6) = 8H(3)$$
$$h_3(1) = 8H(4)$$

ส่วน $H(5), H(6)$ และ $H(7)$ ได้จากคุณสมบัติทาง Complex Conjugate หรือถ้าคำนวณจะพบว่า

$$8H(5) = h_3(5) = h_2(4) - A_8\,h_2(5)$$
$$8H(6) = h_3(3) = h_2(2) - A_8^2\,h_2(3)$$
$$8H(7) = h_3(7) = h_2(6) - A_8^3\,h_2(7)$$

จากสมการที่ (20), (21) และ (23) จะได้ว่า signal flow graph N = 8 คือ



ถ้า $\{h(m)\} = \{1, 2, 1, 1, 3, 2, 1, 2\}$ ให้หา $H(k)$

$h(0) = 1 \quad h_1(0) = 4 \quad h_2(0) = 13 \quad \rightarrow 1/8 \rightarrow \quad H(0)$

$h(1) = 2 \quad h_1(1) = 4 \quad h_2(1) = -1 \quad \rightarrow 1/8 \rightarrow \quad H(4)$

$h(2) = 1 \quad h_1(2) = 2 \quad h_2(2) = 2-j \quad \rightarrow 1/8 \rightarrow \quad H(2)$

$h(3) = 1 \quad h_1(3) = 3 \quad h_2(3) = 2+j \quad \rightarrow 1/8 \rightarrow \quad H(6)$

$h(4) = 3 \quad h_1(4) = -2 \quad h_2(4) = -1.293+j0.707 \quad \rightarrow 1/8 \rightarrow \quad H(1)$

$h(5) = 2 \quad h_1(5) = 0 \quad h_2(5) = -2.707-j0.707 \quad \rightarrow 1/8 \rightarrow \quad H(5)$

$h(6) = 1 \quad h_1(6) = 0 \quad h_2(6) = -2.707+j0.707 \quad \rightarrow 1/8 \rightarrow \quad H(3)$

$h(7) = 2 \quad h_1(7) = -1 \quad h_2(7) = -1.293-j0.707 \quad \rightarrow 1/8 \rightarrow \quad H(7)$

2.5 เดซิเมชั่น (DECIMATION)

เดซิเมชั่น คือ การลดทอนจำนวนของสัญญาณลง ตัวอย่างเช่น สัญญาณอินพุตถูกสุ่มที่ความถี่ 8000 Hz เพราะฉะนั้นใน 1 วินาที จะมีพัลส์ที่ถูกสุ่มจำนวน 8000 ค่า เราสามารถลดจำนวนสัญญาณลงได้โดยไม่ทำให้สัญญาณสูญหายไป จากทฤษฎีการสุ่มสัญญาณจะต้องมีจำนวนมากกว่าขนาดความถี่เป็นจำนวนสองเท่า แต่เราสามารถลดสัญญาณลงมาได้อีก เช่นจากสัญญาณอินพุตขนาด 8000 Hz เราทำการเดซิเมชั่นลง 2 : 1 จะได้สัญญาณเอาต์พุตใหม่ขนาด 4000 Hz เพื่อมิให้สัญญาณสูญหายไป เมื่อต้องการคืนรูปสัญญาณจำเป็นต้องอาศัยขั้นตอนวิธีอินเตอร์โปเลชั่น

2.6 อินเตอร์โปเลชั่น (INTERPOATION)

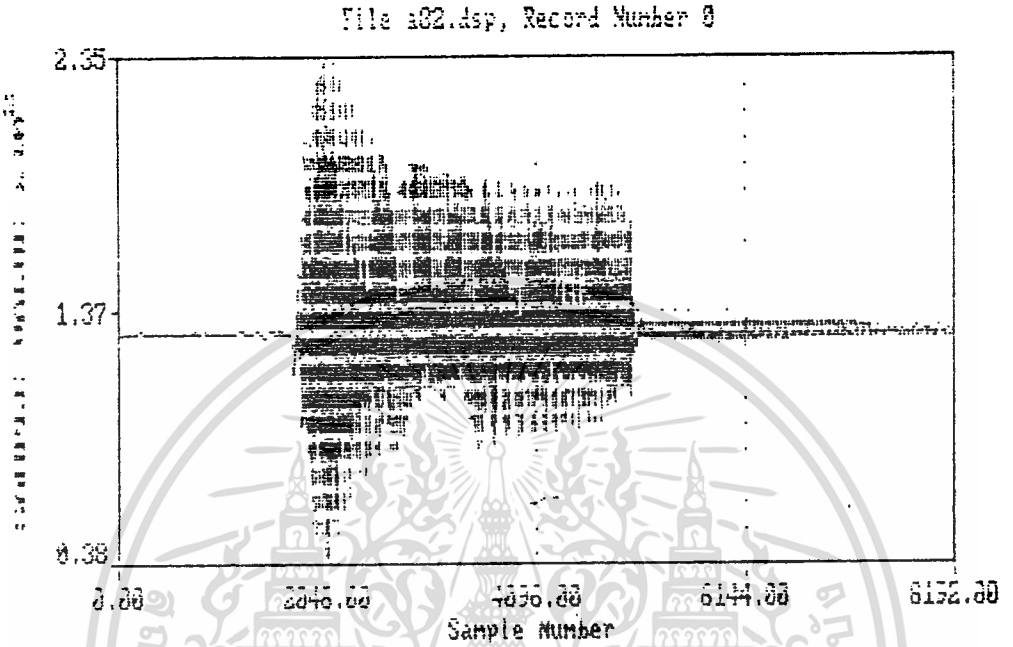
สืบเนื่องจากเมื่อมีการเดซิเมชั่นสัญญาณ เมื่อต้องการคืนรูปสัญญาณจำเป็นต้องการศัยขั้นตอนวิธีอินเตอร์โปเลชั่น การอินเตอร์โปเลชั่นจะต้องใช้ให้ถูกกับการเดซิเมชั่น ตัวอย่างเช่น เดซิเมชั่นขนาด 2 : 1 จำเป็นต้องอินเตอร์โปเลชั่น 1 : 2 เหมือนเดิม สำหรับขั้นตอนการอินเตอร์โปเลชั่นมีหลายวิธี เช่น ใช้ IIR FILTER, หรือ ฟูเรียส์ทรานฟอร์มก็ได้ ในที่นี้จะแสดงเฉพาะวิธี ฟูเรียส์

สำหรับขั้นตอนการอินเตอร์โปเลชั่นขนาด 1 : 2 ด้วยฟูเรียส์ มีดังนี้

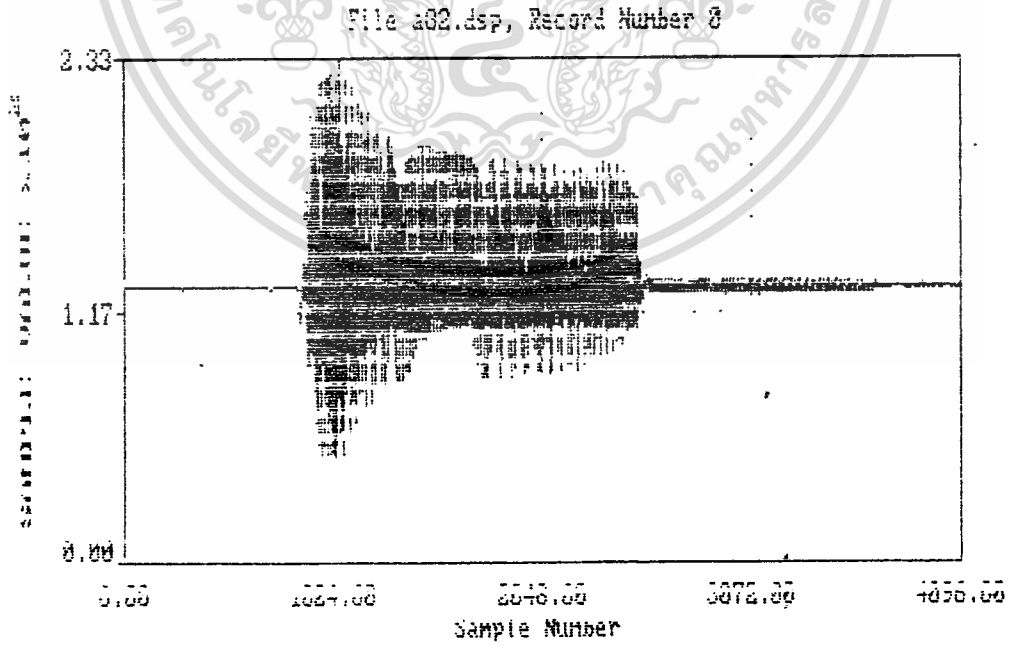
1. แปลงฟูเรียส์ทรานฟอร์มตามขนาดกำลัง 2 ของสัญญาณ
2. เติมค่า 0 ลงในโดเมนความถี่ที่ทำการแปลงฟูเรียส์จำนวน N-1 ตัว โดยใส่ระหว่างครึ่งบวกและครึ่งลบของสเปกตรัม และทำการหารค่ากึ่งกลางระหว่างสเปกตรัมด้วย 2 ตามทฤษฎีของไนควิท

3. ทำการแปลงส่วนกลับของฟูเรียส์

4. คูณค่าที่แปลงส่วนกลับของฟูเรียส์ทั้งหมดด้วย 2



รูปที่ 2.2a แสดงลำดับสัญญาณเสียงคำว่า "การ" ก่อนเดซิเมชั่น

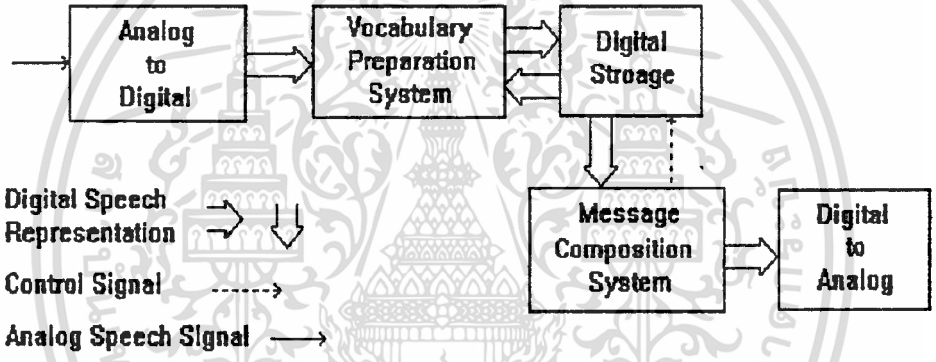


รูปที่ 2.2b แสดงลำดับสัญญาณเสียงคำว่า "การ" หลังเดซิเมชั่น 2:1

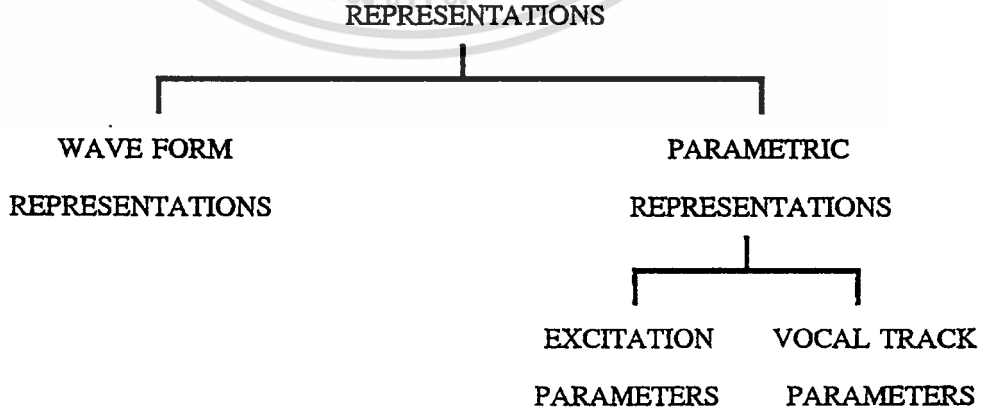
บทที่ 3
การประมวลผลสัญญาณเสียง

การประมวลผลสัญญาณเสียงเชิงเลข (Digital Speech Processing) เราจะนำสัญญาณเสียงมาผ่านวงจรแปลงสัญญาณเชิงอุปมานเป็นสัญญาณเชิงเลข (Analog to Digital) จากนั้นนำไปประมวลผลตามอัลกอริทึมดังรูป 3-1

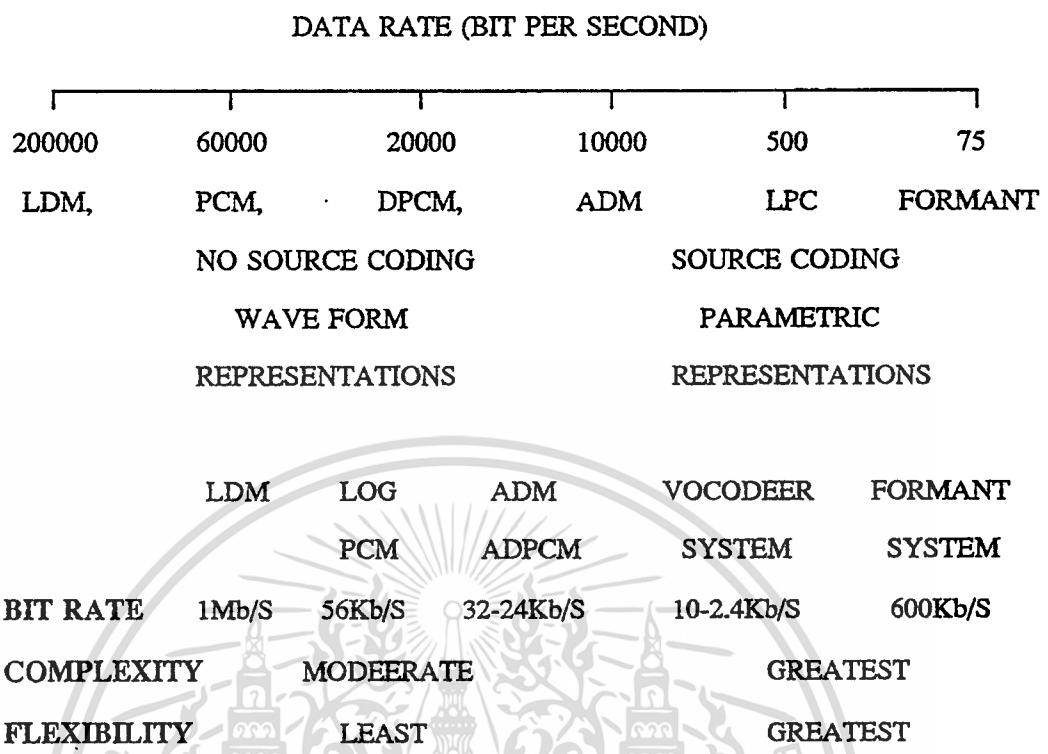
3.1 โครงสร้างระบบการประมวลผลสัญญาณเสียง



รูปที่ 3-1 ระบบการประมวลผลสัญญาณเสียงโดยทั่วไป



รูปที่ 3-2 แสดงการประมวลผลสัญญาณเสียง



รูปที่ 3-3 แสดงขนาดของข้อมูลสำหรับการ encoding ตามวิธีต่าง ๆ

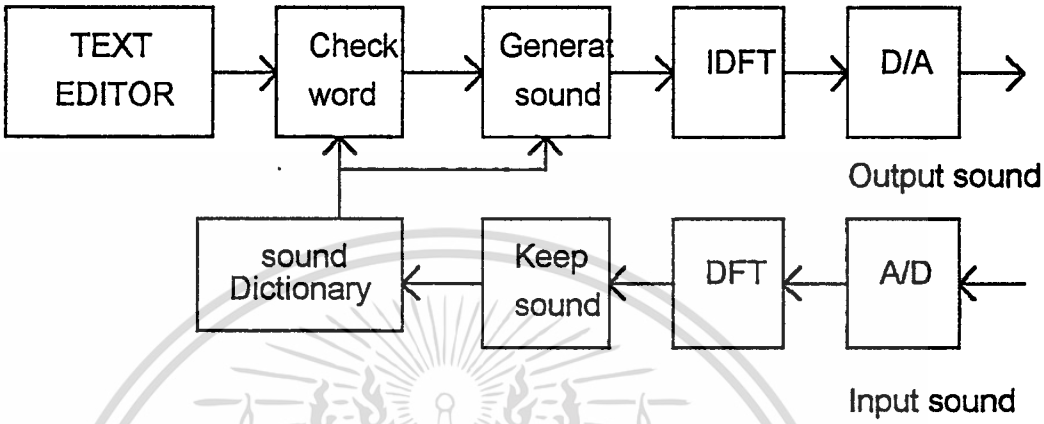
โครงงานนี้เรานำทั้งสองวิธีมารวมเข้าด้วยกัน กล่าวคือ ขั้นตอนแรก เราจะนำสัญญาณเสียงมาทำการแปลงเป็นสัญญาณเชิงเลขโดยอาศัยวิธีการแบบ PCM (Pulse Code Moduration) สัญญาณที่ได้จากการแปลงเรายังไม่นำไปใช้งาน เราจะเข้าลดขนาดสัญญาณด้วยการเดซิเมชั่นจากนั้นอาศัยทฤษฎีทางคณิตศาสตร์ด้านฟูเรียส์ทำการแปลงสัญญาณเชิงเลขที่อยู่ในโดเมนเวลาให้เปลี่ยนไปอยู่ในโดเมนความถี่ เพื่อสะดวกในการประยุกต์ใช้งาน

จากคุณสมบัติของฟูเรียส์ทรานฟอร์มสัญญาณเชิงเลขถูกแปลงเป็นคาบซึ่งมีคุณลักษณะที่ซ้ำกันของครึ่งคาบ เราสามารถจัดเก็บข้อมูลเฉพาะครึ่งคาบ เนื่องจากลักษณะของสัญญาณที่ถูกแปลงด้วยฟูเรียส์จะมีค่าเหมือนกัน เมื่อต้องการที่จะแปลงสัญญาณกลับจำเป็นต้องอาศัย การอินเตอร์โปเลชั่น

3.2 ระบบ TEXT TO SPEECH (TTS)

ระบบ TTS คือระบบที่รับตัวหนังสือที่พิมพ์เข้าไปแล้วออกเสียงตามตัวหนังสือนั้น เนื่องจากโครงงานนี้แสดงถึงการนำบอร์ดทดลองไปประยุกต์ใช้งาน โดยระบบ

TTS ในโครงงานนี้ได้ออกแบบเป็นระบบอย่างง่าย ๆ โดยการเน้นถึงการประมวลผลสัญญาณเสียง โดยใช้ทฤษฎีของการประมวลผลสัญญาณเชิงเลข



รูปที่ 3-4 ระบบ TTS เบื้องต้น

ระบบการทำงาน TTS อย่างง่าย ๆ ที่ได้พัฒนาขึ้นมาแสดงการนำระบบการประมวลผลสัญญาณเสียงเชิงเลข เริ่มต้นจากการเก็บสัญญาณเสียงที่ผ่านวงจรอนาล็อก-ดิจิตอล จากนั้นผ่านขบวนการแปลงฟูเรียส์เพื่อเก็บสเปกตรัมของสัญญาณ โดยจะเก็บไว้ในพจนานุกรมที่ได้กำหนดไว้แล้ว

สำหรับส่วนของ Editor จะรับคำภาษาไทยที่พิมพ์ไว้ จากนั้นทำการแยกคำออกแต่ละคำ คำทุกคำที่ได้จะถูกส่งไปค้นหาคำศัพท์เสียง หากหาเจอก็จะทำการแปลงเป็นข้อมูลเสียง โดยการแปลงส่วนกลับของฟูเรียส์แล้วส่งไปยังวงจรดิจิตอล-อนาล็อก และออกลำโพงต่อไปตามลำดับ

บทที่ 4
การตัดคำ

เนื่องจากการใช้ระบบภาษาไทยในเครื่องไมโครคอมพิวเตอร์มีมากมายหลายแบบ แต่ละแบบมีความแตกต่างบ้าง ในที่นี้จึงขออธิบายระบบภาษาไทยที่จำเป็นต้องนำมาเกี่ยวข้องกับโครงงาน โดยการใช้ในระบบ TTS

4.1 รหัสภาษาไทย

รหัสของตัวอักษรภาษาไทยนั้นมีอยู่ด้วยกันหลายแบบ เนื่องจากช่วงแรกที่มีการพัฒนาระบบภาษาไทยขึ้นมานั้นยังไม่มีมาตรฐานที่แน่นอน รหัสต่าง ๆ จะขึ้นอยู่กับผู้ออกแบบ และวิธีการออกแบบ ภายหลังจึงมีการกำหนดมาตราฐานขึ้นมา คือ รหัส สมอ. ในปัจจุบันบนเครื่องไมโครคอมพิวเตอร์ส่วนใหญ่จะเป็นรหัส สมอ. กับ เกษตร ในโครงการนี้จะเลือกใช้รหัสของ สมอ. เป็นหลักเนื่องจากเป็นมาตรฐาน ข้อแตกต่างที่เห็นได้ชัดของรหัส สมอ. กับ เกษตร คือ รหัส สมอ. จะมีการรวมตัว ขอขวด (ฃ) กับ คอคน (ค) ไว้ในตารางตามลำดับตัวอักษรเลย

สำหรับการแสดงผลภาษาไทยนั้นมีความยุ่งยากมากกว่าภาษาอังกฤษมาก การดัดแปลงภาษาไทยเพื่อใช้กับระบบไมโครคอมพิวเตอร์ จะมีหลายลักษณะแต่เริ่มตั้งแต่ระดับเดียว, สามระดับ, สี่ระดับนอกจากนั้นแต่ละค่ายยังมีรหัสไม่เหมือนกันดังข้างต้น ดังนั้นการแสดงผลของภาษาไทยจึงใช้ระบบกราฟิกทำให้ไม่ต้องใช้การ์ดภาษาไทย

สำหรับในโครงงานนี้เลือกใช้การแสดงอักษรภาษาไทยนั้นจะมีทั้งหมด 3 ระดับ คือ พยัญชนะ สระบน สระล่างและวรรณยุกต์ ในการแสดงอักษรบางลักษณะ เพื่อความสวยงามจึงมีการรวมสระกับวรรณยุกต์ ไว้เป็น อักษรอีกตัวหนึ่งในตารางรหัส สาเหตุอีกอย่างน่าจะเนื่องมาจากว่า ในแป้นพิมพ์นั้นจะรวม สระบน กับวรรณยุกต์บางตัวไว้แล้ว เพื่อให้แป้นพิมพ์บนเครื่องไมโครคอมพิวเตอร์เหมือนกับแป้นพิมพ์ของเครื่องพิมพ์ดีด จึงมีการรวมสระ เช่นกันโดยจะกำหนดสระที่รวมกันไว้ที่ตำแหน่งว่าง ในตารางรหัส สมอ.

ในโครงงานนี้ใช้ตารางรหัส สมอ. ที่มีการดัดแปลงเพิ่มเติมลักษณะเดียวกับที่ใช้ในโปรแกรมเวิร์ดจุฬา.

4.1.1 การเก็บรหัส สมอ. ในไฟล์ตารางตัวอักษร

ในโครงงานนี้ใช้การแสดงผลแบบ BITMAP ซึ่งจะต้องเก็บข้อมูล BITMAP ของตัวอักษรไว้เพื่อใช้ในการแสดงผล เช่นโปรแกรมเวิร์ดจุฬา โดยจะเก็บข้อมูล BITMAP ของตัวอักษรในไฟล์ *.FON ซึ่งการเก็บจะเรียงลำดับตัวอักษรตามรหัส สมอ. ตัวอักษรมีขนาด 8*20 โดยจะเก็บข้อมูล 1 ไบต์ จำนวน 20 ตัวต่อหนึ่งตัวอักษร ตัวอักษรจำนวน 256 ตัว ดังนั้นไฟล์จะมีขนาด 5120 ไบต์ โดยที่หากเราต้องการตัวอักษรแบบใดก็จะต้องออกแบบตัวอักษรแบบนั้นๆ แล้วเก็บไว้ในไฟล์ หรือในหน่วยความจำเมื่อจะแสดงตัวอักษรตัวใดก็อ่านจากตารางแล้วนำมาแสดงผลทีละไบต์จนครบ 20 ไบต์

การเก็บตัวอักษรแบบ BITMAP นั้นมีข้อดี คือเป็นวิธีที่ง่าย และเร็วไม่ต้องอาศัยการคำนวนมากแต่ ข้อเสีย คือ หากเราต้องการตัวอักษรแบบใดก็จะต้องมีการเก็บตัวอักษรแบบนั้น ๆ ไว้ทำให้เปลืองหน่วยความจำและจะมีข้อจำกัดในการแสดงผลมาก และขนาดเมื่อมีการขยายจะมีลักษณะผิดเพี้ยนไปไม่ตรงตามที่ได้ออกแบบไว้ นอกจากนี้ยังมีลักษณะตัวอักษร อีกลักษณะหนึ่งคือ แบบ VECTOR โดยจะเก็บจุดเริ่มต้นและทิศทางของเส้นตัวอักษรว่ามีทิศทางใด จะมีข้อดี คือ ตัวอักษรจะมีได้หลายแบบ สามารถขยายหรือย่อได้ แต่ทุกครั้งที่แสดงผลจะต้องอาศัยการคำนวนเพื่อทำการวาดตัวอักษรทุกครั้ง ทำให้การแสดงผลทำได้ช้ากว่าแบบ BITMAP มาก สำหรับเวิร์ดจุฬานั้นแสดงผลแบบ BITMAP

สำหรับโครงงานนี้ใช้การเก็บข้อมูลแบบ BITMAP

ลักษณะการเก็บจะเป็นดังนี้ ตัวอักษร 'ก' จำนวน 20 ไบต์

	0 1 2 3 4 5 6 7	Address
0	0 0 0 0 0 0 0 0	00H
1	0 0 0 0 0 0 0 0	00H
2	0 0 0 0 0 0 0 0	00H
3	0 0 0 0 0 0 0 0	00H
4	0 0 0 0 0 0 0 0	00H
5	0 0 0 0 0 0 0 0	00H
6	0 0 0 0 0 0 0 0	00H
7	0 0 0 0 0 0 0 0	00H
8	0 0 1 1 1 1 0 0	3CH
9	0 1 0 0 0 0 1 0	42H
10	0 1 0 0 0 0 1 0	22H

11	0 1 0 0 0 0 1 0	22H
12	0 1 0 0 0 0 1 0	22H
13	0 1 0 0 0 0 1 0	22H
14	0 1 0 0 0 0 1 0	22H
15	0 1 0 0 0 0 1 0	22H
16	0 0 0 0 0 0 0 0	00H
17	0 0 0 0 0 0 0 0	00H
18	0 0 0 0 0 0 0 0	00H
19	0 0 0 0 0 0 0 0	00H

4.1.2 การแสดงผลโดยการใช้วิธี Direct Video RAM

การแสดงผลในโหมดกราฟิกนั้น ในปัจจุบัน ภาษาระดับสูงมีฟังก์ชั่นที่ช่วยในการแสดงผลกราฟิกอยู่แล้ว แต่หากต้องการความเร็วในการแสดงผลที่สูงขึ้น จำเป็นจะต้องมีการเขียนฟังก์ชั่นพิเศษที่ใช้ในการแสดงผล โดยจะติดต่อกับ Hardware โดยตรง ในการแสดงผลในเครื่องไมโครคอมพิวเตอร์นั้น จะมีการกำหนดหน่วยความจำขึ้นมาช่วงหนึ่งเพื่อใช้ในการแสดงผล คือ Video ram หากมีการนำข้อมูลไปไว้ที่ส่วนนี้ ก็จะเป็นการแสดงผลออกจอภาพโดยตรง ทำให้มีความเร็วในการแสดงผลที่ค่อนข้างสูง

ตำแหน่งของ Address ของ Video ram นั้น Address ของ Video Ram นั้นจะขึ้นอยู่กับชนิดของอุปกรณ์แสดงผลอุปกรณ์แสดงผลมี 2 ประเภท คือ Color กับ Monochrome ตำแหน่งหน่วยความจำดังนี้

1. Color ในโหมดกราฟิก จะมี Address เริ่มต้นที่ A000:0000 จะเก็บข้อมูลที่ละ 8 จุดหรือ 1 ไบท์ จากซ้ายไปขวา และจากบนลงล่างต่อเนื่องกันไป

2. MonoChrome Address เริ่มต้นที่ B000:0000 เก็บข้อมูลที่ละไบท์เช่นเดียวกันแต่ Address จะไม่ต่อเนื่องกัน โดยจะเรียงที่ละเส้น แล้วเว้น 4 เส้น แล้วจึงแสดงเส้นต่อไป

```
if( TypeScreen ) {
        v_ram = ( char far * )0xB0000000L;
        test_scan = y % 4;                  /* test No. scan */
        if( test_scan == 0 )                /* thefirst line */
                Row_v_ram += 0x6000 + ((int)(y /4) -1)*0x5A;
        else
```

```
                    Row_v_ram +=(test_scan -1)*0x2000+(int)(y/4))*0x5A;
} else {
        v-ram = (char far *)0xA0000000L;
        dis_p = (long)((maxx+7)/8)*(long)(y)+(long)(x);
}
```

โปรแกรมข้างต้นเป็นส่วนของการคำนวน Address ในการแสดงผลตัวอักษร การแสดงผลภาษาไทย สำหรับการแสดงตัวอักษรภาษาไทยในโครงการนี้ จะใช้การแสดงผลในโหมดกราฟิก โดยตัวอักษรแต่ละตัวจะเก็บเป็น Bitmap การแสดงผลก็จะนำ Bitmap นั้นมาแสดงตัวอักษรจะเก็บอยู่ในไฟล์ NORMAL.FON การแสดงผลจะทำโดยการติดต่อกับ Video RAM โดยตรงเพื่อให้มีความเร็วในการแสดงผลที่สูง ตัวอักษรแต่ละตัวจะมีขนาด 8*20 และเก็บเรียงตามรหัส สมอ. การแสดงผลตัวอักษรกับวรรณยุกต์จะมาทำงานเหมือนกับเอาตัวอักษรมาซ้อนกัน เนื่องจากวรรณยุกต์มีช่องว่างที่มีส่วนล่างของ Bitmap จึงเหมือนกับวรรณยุกต์อยู่บนตัวอักษร เนื่องจากเรามีการตรวจสอบว่าตัวอักษรนั้นเป็นสระ หรือพยัญชนะ หากเป็นสระบนหรือล่างก็จะไม่เลื่อนตำแหน่ง

ขั้นตอนการแสดงผล

1. ทำการอ่านแบบตัวอักษรจากไฟล์ มาไว้ในหน่วยความจำ ซึ่งมีโครงสร้างเป็น Array 2 มิติ

2. เมื่อจะแสดงผลตัวอักษรก็เลือกชุด Bitmap ของตัวอักษรนั้น แล้วแสดงผลทีละ Byte จนครบ 20 Byte (ขนาดตัวอักษร8*20) ทำการตรวจสอบว่าเป็นพยัญชนะหรือไม่ หากเป็นสระบนหรือล่างก็จะไม่มีการเลื่อนตำแหน่ง

3. การลบตัวอักษรจะใช้การ XOR ด้วยตัวอักษรเดิม ซึ่งการทำ XOR ถ้าเป็นข้อมูลเดียวกัน เช่น 1 กับ 1 ก็ จะกลายเป็น 0 คือ ไม่มีการแสดงผล

4.2 ความรู้เบื้องต้นเกี่ยวกับการตัดคำ

ในการเข้าสู่ระบบ text to speech เพื่อที่จะแสดงการทำงานของ DSP BOARD เราจะต้องมีข้อมูลที่จะให้เครื่องไมโครคอมพิวเตอร์อ่าน โดยข้อมูลเราจะมาจากสองแหล่ง คือ จากการคีย์ข้อมูลเข้าทาง keyboard และจากแฟ้มข้อมูลที่เราสร้างเก็บไว้ จากนั้นก็จะทำการตัดคำ โดยอาศัย กฎการตัดคำที่ได้สร้างขึ้นไว้ สำหรับการตัดคำเนื่องจากภาษาไทยนั้นระบบการตัดคำยุ่งยากมาก แต่กฎการตัดคำที่ได้จัดสร้างไว้ก็เพียงพอกับการใช้งานในโครง

งานนี้ สำหรับข้อมูลที่ได้จากการตัดคำแล้ว จึงทำการส่งข้อมูลเกี่ยวกับเสียงให้ในส่วนอื่นทำงานต่อไป

จากการทำงานที่กล่าวไว้แล้วข้างต้น ในส่วนนี้จะต้องประกอบด้วยการทำงานหลัก ๆ 3 ส่วน คือ

1. ส่วนรับข้อมูล เป็น Editor ภาษาไทย จะต้องสามารถพิมพ์ข้อความภาษาไทยได้ สามารถแก้ไขข้อความได้ง่าย และสามารถจัดเก็บข้อความหรือนำข้อความที่เก็บไว้แล้วมาแสดงผลใหม่อีกได้

2. ส่วนตัดคำ นำข้อมูลจาก Editor ภาษาไทย มาแยกออกให้ได้เป็นคำ ๆ ตามขั้นตอนวิธี (Algorithms) ที่ได้จัดสร้างขึ้น

3. ส่วนค้นหา นำคำที่ได้จากส่วนตัดคำมาเปรียบเทียบกับข้อมูลที่มีอยู่ จากนั้นจึงส่งข้อมูลเกี่ยวกับเสียง เช่น ชื่อแฟ้มข้อมูลของเสียง จำนวนของเสียง ไปยังส่วนอื่นต่อไป แต่ถ้าหากไม่สามารถค้นหาคำศัพท์จากพจนานุกรมได้ ก็จะส่งรหัส A0 ไปให้ยังส่วนกำเนิดเสียง

4.2.1 ส่วนรับข้อมูล

ทำการจัดสร้าง Editor ภาษาไทยแบบง่าย ๆ โดยการจัดโครงสร้างข้อมูลเป็นแบบ Array มีขนาดเท่ากับ MAXSTR สาเหตุที่ใช้แบบนี้เนื่องจากว่าในโครงงานนี้ไม่ได้มีการประมวลผลมากจนขนาดต้องสร้างเป็น word processor โครงงานนี้ใช้เพียง Editor แต่ก็สามารถทำงานได้หลายอย่างเช่นสามารถ ลบ และแก้ไขข้อความได้สามารถจัดเก็บและนำข้อความที่พิมพ์เก็บไว้ในแฟ้มข้อมูลออกมาแสดงผลใหม่ได้

4.2.2 ส่วนตัดคำ

หลังจากที่เราได้ทำการป้อนข้อมูลเข้าทาง Editor ภาษาไทย หรือทำการอ่านข้อมูลภาษาไทยจากแฟ้มข้อมูล โดยการกดคีย์ F2 ข้อมูลทั้งหมดจะถูกนำมาเก็บใน Array ชื่อ str[MAXSTR] ซึ่งจะมีตัวชี้ข้อมูล(pointer data) คอยนำตัวอักษรที่ pointer ตัวนี้ชี้ไปตรวจสอบในกฎ (rule) ที่กำหนดขึ้นมาจนหมด แล้วจึงนำคำที่ตัดได้แล้วไปแสดงผลที่หน้าจอ เพื่อให้ผู้ที่ใช้ได้เห็นว่าเครื่องคอมพิวเตอร์จะอ่านคำอะไรออกมาบ้าง ซึ่งเป็นที่ทราบกันดีอยู่แล้วว่าคำในภาษาไทยมีอยู่มากมาย หลากหลายรูปแบบมาก ในโครงงานนี้ได้กำหนดข้อจำกัดหรือขอบเขตของคำที่สามารถจะตัดได้นั้นก็คือ

1. จะต้องไม่ใช่คำทับศัพท์มาจากภาษาอื่น เช่น ซอล์ก
2. จะต้องไม่ใช่คำที่อ่านได้หลายแบบ เช่น ตากลม หรือ โคลง เป็นต้น
3. อื่น ๆ นอกจากกฎที่สร้างไว้

กฎที่ใช้ในการตัดคำ

การตัดคำในโครงงานนี้ได้สร้างกฎไว้หลายกฎด้วยกัน ซึ่งแต่ละกฎที่สร้างขึ้นจะอาศัยหลักการของภาษาไทยในการตรวจสอบ และจะทำการตรวจสอบจากกฎที่จะประกอบด้วยจำนวนตัวอักษรสูงสุดก่อน โดยเราสามารถแบ่งคำภาษาไทยออกเป็น 2 กรณีใหญ่ ๆ คือ

1. คำที่พยัญชนะนำหน้า เช่น การ, ภาพ, ขายของเป็นต้น
2. คำที่สระนำหน้า เช่น เสียง , โดย , แรง เป็นต้น

ซึ่งในแต่ละกรณีเราจะเริ่มตรวจสอบตั้งแต่กฎที่ประกอบด้วยจำนวนตัวอักษรมากที่สุดก่อน แล้วทำการตรวจสอบจนหมด Array ที่ใช้เก็บข้อมูล ในแต่ละครั้งที่ตรวจสอบจนได้หนึ่งคำ แล้วจึงนำคำที่ได้มาเก็บใน Array ชื่อ display[] เพื่อเก็บไว้ใช้ในการแสดงผลและใช้ในการค้นหาชื่อของแฟ้มข้อมูลที่เก็บเสียงอีกทีหนึ่ง

ในกฎแต่ละข้อจะประกอบด้วยการตรวจสอบต่าง ๆ ดังนี้

- ตรวจสอบว่าใช่พยัญชนะภาษาไทยหรือไม่ เพราะในโปรแกรมจะสามารถให้ผู้ใช้เลือกพิมพ์ข้อความภาษาอังกฤษได้ (AlphT(*p))
- ตรวจสอบว่าใช่สระหรือไม่ (Pond(*p))
- ตรวจสอบว่าใช่วรรณยุกต์หรือไม่ (Tonemark(*p))
- ตรวจสอบว่าใช่ตัวสะกดหรือไม่ (Spelling(*p))
- ตรวจสอบว่าใช่สระที่อยู่บนพยัญชนะหรือไม่ (UpPond(*p))
- ตรวจสอบว่าใช่สระที่อยู่ใต้พยัญชนะหรือไม่ (DownPond(*p))
- ตรวจสอบว่าใช่สระที่อยู่ระดับเดียวกับพยัญชนะหรือไม่ (FontPond(*p))
- ตรวจสอบว่าใช่ตัวควบกล้ำหรือไม่ (Mix(*(p))
- ตรวจสอบว่าใช่คำถัดไปหรือไม่ (NextWord(*p))

เช่นในกรณีที่เป็นคำที่ขึ้นต้นด้วยพยัญชนะ (จะต้องมีการตรวจสอบว่าใช่พยัญชนะหรือสระ) จากนั้นก็ทำการตรวจสอบตัวอักษรตัวที่สองว่าเป็นพยัญชนะหรือไม่ ถ้าใช่ก็ทำการตรวจสอบตัวอักษรตัวที่สามว่าใช่วรรณยุกต์หรือไม่ ถ้าใช่ก็ทำการตรวจสอบตัว

อักษรตัวที่สี่ว่าใช่สระอาหรือสระออหรือไม่ ถ้าใช่ก็ทำการตรวจสอบตัวอักษรตัวที่ห้าว่าใช่ตัวสะกดหรือไม่ ถ้าใช่ก็ทำการตัดคำออกมาเก็บใน Array display[] แต่ถ้าไม่ใช่ก็ทำการตรวจสอบในกฎถัดไป

```
if(AlphaT(*p))
        if((AlphaT(*(p+1)))&&(Tonemark(*(p+2)))  ; rule 1
if((*(p+3)==0xd2)||(*(p+3)==0xcd))
        if((Spelling(*(p+4)))&&(NextWord(*(p+5))))
StoreChar(5);                ; เก็บตัวอักษร 5 ตัว
```

ส่วนในกฎถัดไปก็จะมีลักษณะคล้าย ๆ กัน แต่จำนวนตัวอักษรที่จะทำการตรวจสอบก็จะค่อย ๆ ลดหลั่นลงมา ซึ่งมีในบางกรณีที่คำสั้น ๆ มาอยู่ติดกันแล้วไปตรงกับกรณีของคำยาว ๆ เข้า จึงจะต้องมีการเพิ่มการตรวจสอบเกี่ยวกับคำควบกล้ำเพื่อให้เกิดการผิดพลาดน้อยที่สุด

ซึ่งในการทำงานของส่วนตัดคำนี้ เราจะต้องเริ่มตรวจสอบจากกฎที่ใช้ตรวจสอบคำที่ยาว ๆ ก่อนดังนั้นเราจะมีการเพิ่มการตรวจสอบว่าตัดคำได้หรือยัง เพื่อที่จะสามารถกลับไปเริ่มต้นทำการตรวจสอบที่กฎแรกใหม่ เพราะฉะนั้นเมื่อตัดคำได้แล้วจะให้ค่าของ word=1 ถ้าไม่พบจะให้ word=0 สามารถแก้ไขได้ดังนี้

```
if(AlphaT(*p))
        if((AlphaT(*(p+1)))&&(Tonemark(*(p+2))&&word)  ; rule 1
if((*(p+3)==0xd2)||(*(p+3)==0xcd))
        if((Spelling(*(p+4)))&&(NextWord(*(p+5))))
StoreChar(5);                ; เก็บตัวอักษร 5 ตัว
```

ซึ่งการทำงานของโปรแกรมจริง ๆ แล้ว จะประกอบด้วยกฎต่าง ๆ มากมาย ถ้าจะให้อธิบายจนครบทุกกฎก็จะเป็นการสิ้นเปลืองเวลา เพราะแต่ละกฎก็จะมีลักษณะที่คล้าย ๆ กันถ้าผู้อ่านสนใจให้ดูได้ในคู่มือการใช้โปรแกรม

4.2.3 ส่วนค้นหา

เราจะนำแต่ละคำที่ได้มาค้นหาชื่อของแฟ้มข้อมูลที่เก็บเสียงของคำนั้น ๆ โดยชื่อของแฟ้มข้อมูลที่เก็บเสียงจะถูกเก็บไว้ที่แฟ้มข้อมูลชื่อ table.h หลังจากทำการค้นหาจนพบ

แล้ว ก็จะรวบรวมชื่อของแฟ้มข้อมูลที่เก็บเสียงไว้ที่ Array ชื่อ fdata[] และจำนวนชื่อแฟ้มข้อมูลที่เก็บเสียงไว้ที่ Array ชื่อ fnun ซึ่งจะทำการส่งค่าทั้งสองนี้ไปยังขั้นตอนการทำงานถัดไป

แสดงการเก็บชื่อของแฟ้มข้อมูล สำหรับโครงงานนี้ได้จัดโครงสร้างของข้อมูลเป็นแบบ Structure

```
struct sound_rec {
        char word[8];
        char fname[8];
};
type  struct sound_rec sd;
sd k1[130] = {{"ก็","A1"},
        {"กก","A2"},
        {"กง","A3"},
        |
        |
        |
{"END","A0"};
```

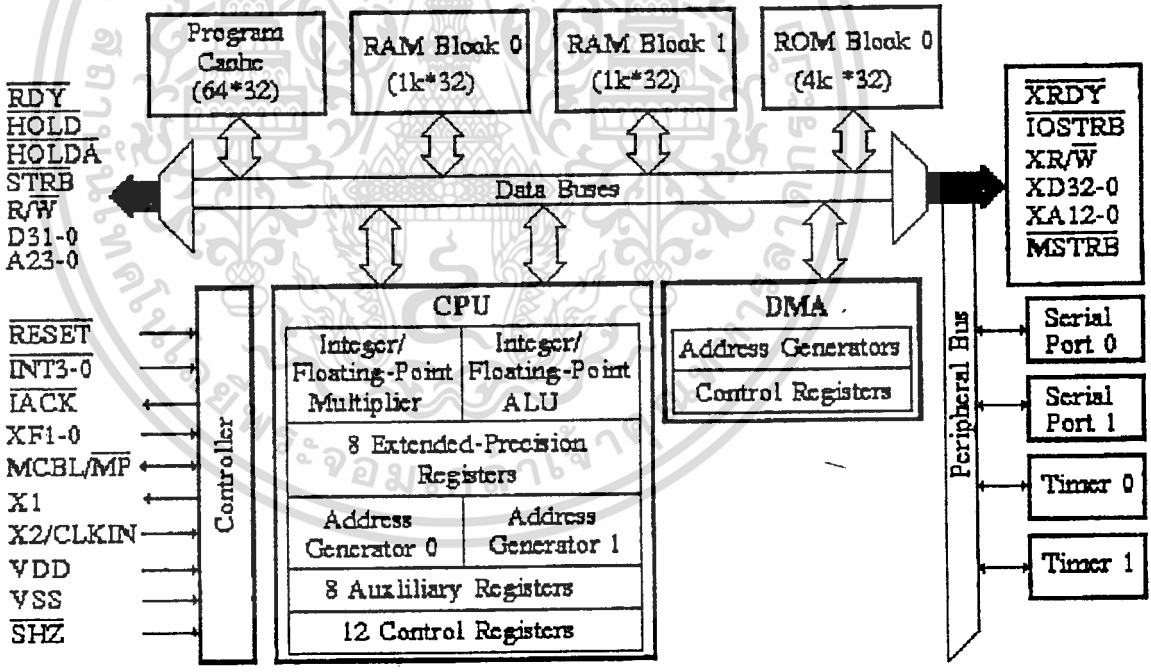
ในความเป็นจริงแล้วคำในภาษาไทยมีอยู่มากมาย เพราะฉะนั้นเราไม่สามารถทำการจัดเก็บไว้ได้ทั้งหมด ดังนั้นเมื่อไม่พบกับคำที่เก็บไว้เราจะส่งชื่อของแฟ้มชื่อ A0 ไปยังส่วนกำเนิดเสียงแทน

บทที่ 5.

TMS320C30

TMS320C30 ถูกผลิตขึ้นมาเพื่อ การประมวลผลสัญญาณเชิงตัวเลข (Digital Signal Processing) และมีประสิทธิภาพในการทำงานสูง เริ่มจากปี 1982 TMS32010 ได้ถูกผลิตออกสู่ท้องตลาด และได้รับความนิยมสูง TMS320 ได้มีการพัฒนาในรุ่นต่อ ๆ มาให้มีประสิทธิภาพในการทำงานสูงขึ้น

TMS230C30 เป็น CMOS ขนาด 32 บิท, มีชุดคำสั่งที่ สนับสนุนการทำงานทางด้าน DSP, ความเร็วสูง (33 MFLOPS, 16.7 MIPS), มีหน่วยความจำขนาดใหญ่บนชิพ, ชุดควบคุม DMA, และการติดต่ออุปกรณ์ภายนอกที่มีประสิทธิภาพ



รูปที่ 5-1 บล็อกไดอะแกรม ของ TMS320C30

5.1 ลักษณะของ TMS320C30

ลักษณะของ TMS320C30 มีดังต่อไปนี้ :-

- ประสิทธิภาพ (Performance)
- ประมวลผลภายใน 1 cycle (60 ns)
- ประมวลผลเลขจุดทศนิยม 33.3 ล้านคำต่อ 1 วินาที (33.3 MFLOPS)
- ประมวลผลคำสั่งทั่วไป 16.7 ล้านคำต่อ 1 วินาที (16.7 MIPS)
- หน่วยความจำแบบ ROM ภายใน ขนาด 4K x 32 บิท
- หน่วยความจำแบบ RAM ภายใน ขนาด 1K x 32 บิท
- หน่วยความจำ แค็ช (cache) ภายใน ขนาด 64 x 32 บิท
- คำสั่งและข้อมูลขนาด 32 บิท, แอดเดรสขนาด 24 บิท
- เลขจุดทศนิยมขนาด 40 บิท, เลขจำนวนเต็มขนาด 32 บิท
- Barrel shifter ขนาด 32 บิท
- รีจีสเตอร์ขนาด 40 บิท 8 ตัว (Extended-precision register) สำหรับประมวลผลเลขจุดทศนิยม และเลขจำนวนเต็ม
- มีตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง (Direct memory access) สำหรับการทำงานแบบขนาน ระหว่างอุปกรณ์ภายนอก และ การทำงานของซีพียู
- ระบบตัวเลขแบบจำนวนเต็ม (Integer), จุดทศนิยม (Floating point) และการทำงานแบบตรรกะ (Logic)
- คำสั่งแบบมีตัวที่ถูกกระทำ 2 และ 3 ตัว (two and three operand)
- คำสั่งที่ทำงานพร้อมกันระหว่าง ALU และ Multiplier ภายใน 1 clock cycle
- ความสามารถการทำงานซ้ำเป็นกลุ่ม (Block repeat)
- คำสั่งที่ใช้ในการวนรอบ ใช้เวลาเพียง 1 clock cycle
- เงื่อนไขสำหรับการเรียก (CALL) และการกลับ (RETURN)
- คำสั่งที่ประสานงานกันเพื่อสนับสนุนการทำงานแบบหลายงาน (Multiprocessing)
- บัสข้อมูลขนาด 32 บิท สองชุด (24 และ 13 bit address)
- พอร์ทอนุกรมสองพอร์ท สนับสนุนการส่งแบบ 8,16,24,32 บิท
- ตัวตั้งเวลา(Timer) สองชุด ขนาด 32 บิท
- การขัดจังหวะสี่ช่องจากภายนอก (4 Interrupt)
- 181 ขา (181-pin grid array)

5.3 สถาปัตยกรรมของ TMS320C30

สถาปัตยกรรมของ TMS320C30 แสดงในรูปที่ 5-1 ประกอบด้วยส่วนหลัก ๆ ดังต่อไปนี้

1. หน่วยประมวลผลกลาง (CPU)
2. หน่วยความจำ
3. ลักษณะบัสภายใน
4. ลักษณะบัสภายนอก
5. อุปกรณ์สนับสนุน
6. ชุดควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA)
7. การต่อเชื่อมระบบ

5.4 หน่วยประมวลผลกลาง (CPU)

สถาปัตยกรรมของ TMS320C30 ถูกสร้างแบบ Harvard Architechture ทำให้หน่วยประมวลผลกลางมีประสิทธิภาพเพิ่มขึ้นสามารถทำงานไปพร้อม ๆ กันได้ สำหรับหน่วยประมวลผลกลางจะประกอบด้วยส่วนประกอบย่อยต่าง ๆ ดังแสดงในรูปที่ 5-3 มีส่วนประกอบดังนี้

1. ตัวคูณ (Multiplier)
2. หน่วยคำนวณ (ALU)
3. 32-bit barrel shifter
4. บัสภายใน
5. รีจีสเตอร์ช่วย (Auxiliary register)
6. แฟ้มรีจีสเตอร์ของซีพียู (CUP register file)

5.4.1 ตัวคูณ (Multiplier)

ตัวคูณจะทำงานภายใน 1 ไซเคิล ซึ่งจัดการเกี่ยวกับตัวเลขจำนวนเต็ม (integer) ขนาด 24, 32บิท และตัวเลขแบบจุดทศนิยม (floating point) สามารถทำงานแบบขนานไปพร้อมกับการทำงานของ ALU ซึ่งจะมีคำสั่งการทำงานแบบขนาน

เมื่อตัวคูณทำการคูณข้อมูลขนาด 32 บิทแบบจุดทศนิยม จะถูกนำมาทำการคำนวณจะได้ผลลัพธ์เป็นขนาด 40 บิท (Extend precission) หรือถ้าเป็น 24 บิทจะได้ผลลัพธ์เป็นขนาด 32 บิท

5.4.2 ALU

ALU จะทำงานภายใน 1 ไซเคิล บนตัวเลขจำนวนเต็มขนาด 32 บิท และตัวเลขแบบจุดทศนิยมขนาด 40 บิท สามารถแปลงตัวเลขระหว่างเลขจำนวนเต็มกับตัวเลขแบบจุดทศนิยมผลลัพธ์ของ ALU จะเป็น 32 และ 40 บิทเสมอ ภายใน ALU มีวงจรเลื่อน (barrel shifter) ขนาด32บิท ระบบบัสภายใน, CPU1/CPU2 และ REG1/REG2 ซึ่งสามารถนำโอเปอร์แรน สองชุดจากหน่วยความจำ และ โอเปอร์แรนสองชุดจากแฟ้มรีจีสเตอร์ ซึ่งการทำเช่นนี้จะเห็นว่าสามารถทำงานแบบขนานระหว่าง ALU ไปพร้อมกับการทำงานของตัวคูณ (multiplier) ได้ภายใน 1 ไซเคิล

5.4.3 Auxiliary Register Arithmetic Units (ARAUs)

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

5.4.4 แฟ้มรีจีสเตอร์ (CPU Register file)

TMS320C30 มี 28 รีจีสเตอร์ที่สนับสนุนการทำงานของตัวคูณ และ ALU สามารถใช้รีจีสเตอร์เหล่านี้ในลักษณะงานทั่ว ๆ ไปตามต้องการ ซึ่งอย่างไรก็ตามรีจีสเตอร์จะถูกแบ่งเพื่อให้ทำงานเฉพาะอย่าง เช่น รีจีสเตอร์ R0-R7 (Extended precision register) ถูกออกแบบมาเพื่อให้เก็บผลลัพธ์ที่เป็นเลขแบบจุดทศนิยม รีจีสเตอร์ช่วย (Auxiliary register AR0-AR7) ถูกออกแบบมาเพื่อใช้งานทั่ว ๆ ไปการอ้างแอดเดรสเป็นแบบตรง (direct addressing mode) ใช้เก็บที่เป็นตัวเลขจำนวนเต็มขนาด 32 บิท และการทำงานทางตรรกะ (logical) รีจีสเตอร์ที่เหลือจะทำหน้าที่เกี่ยวกับระบบเช่น การอ้างแอดเดรส, การจัดการสแตก (stack), การจัดการสถานะ (status), การขัดจังหวะ (interrupt), การทำงานซ้ำ ๆ เป็นกลุ่ม (block repeat) ซึ่งจะได้กล่าวให้ละเอียดต่อไป

รีจีสเตอร์ต่าง ๆ และความหมายอยู่ในรูปที่ 5-2

ชื่อรีจีเตอร์	หน้าที่ การทำงาน
R0 - R7	Extended-precision egister0- 7
AR0 - AR7	Auxiliary register0 - 7

DP	Data-page pointer
IR0	Index register0
IR1	Index register1
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

รูปที่ 5-2 CPU รีจิสเตอร์

5.4.5 รีจีสเตอร์ขนาดพิเศษ (Extended-precision registersR0-R7)

เป็นรีจีสเตอร์ขนาด 40 บิท สนับสนุนการทำงานของตัวเลขจำนวนเต็มขนาด 32 บิทและ 40 บิทสำหรับเลขจุดทศนิยม คำสั่งใด ๆ จะถือว่าเป็นเลขจุดทศนิยม บิทที่ 0-39 ถ้าคำสั่งเป็นการทำงานของ เลขจำนวนเต็ม บิทที่ 0-31 จะถูกใช้ ส่วนบิทที่ 32-39 จะยังคงเหมือนเดิม รูปแบบการเก็บข้อมูล จะกล่าวให้ละเอียดต่อไป

5.4.6 รีจีสเตอร์ช่วย (Auxiliary register AR0-AR7)

สามารถที่จะเข้าถึงโดย CPU และสามารถถูกเปลี่ยนแปลงแก้ไขโดย ARAUs ซึ่งเป็นหน่วยคำนวณสำหรับรีจีสเตอร์ช่วย (มีสองตัว) ซึ่งมีหน้าที่พื้นฐานของรีจีสเตอร์ช่วย คือ กำเนิดสัญญาณแอดเดรสขนาด 24 บิท สามารถที่จะใช้งานในลักษณะที่เป็นตัวนับจำนวนรอบ (loop counter) และยังคงสามารถเก็บข้อมูลทั่วๆไปขนาด 32 บิท ซึ่งสามารถถูกแก้ไขเปลี่ยนแปลงโดย ตัวคูณ(multiplier) และ ALU

5.4.7 Data Page Pointer (DP)

เป็นรีจีสเตอร์ขนาด 32 บิท บิท 0 - 7 (8 bit LSB) ของรีจีสเตอร์ DP ถูกใช้โดยโหมด การอ้างแอดเดรสแบบตรง ทำหน้าที่เสมือนกับตัวชี้เพจ (Page) ของข้อมูล 1 เพจมีขนาด 64 กิโลเวิร์ด (64K words) ซึ่งมีทั้งหมด 256 เพจ

5.4.8 Index Registers (IRQ,RR1)

เป็นรีจีสเตอร์ขนาด 32 บิทสำหรับเก็บค่าข้อมูลโดย ARAU (Auxiliary Register Arithmetic Unit) เพื่อทำการประมวลแอดเดรสแบบ อินเด็ก (index addressing)

5.4.9 System Stock Pointer (SP)

เป็นรีจีสเตอร์ขนาด 32 บิท เป็นตัวเก็บค่าของแอดเดรสของสแตค ส่วนบนสุดของคำสั่ง พุช (Push) จะมีผลทำให้ SP มีค่าเพิ่มขึ้น 1 และ POP มีผลทำให้ SP มีค่าลดลง 1 นอกจากคำสั่ง Push และ Pop แล้ว คำสั่ง TRAP, CALLS, RETURN ก็มีผลต่อรีจีสเตอร์ SP ด้วย รายละเอียดจะได้กล่าวต่อไป

5.4.10 Status Register (ST)

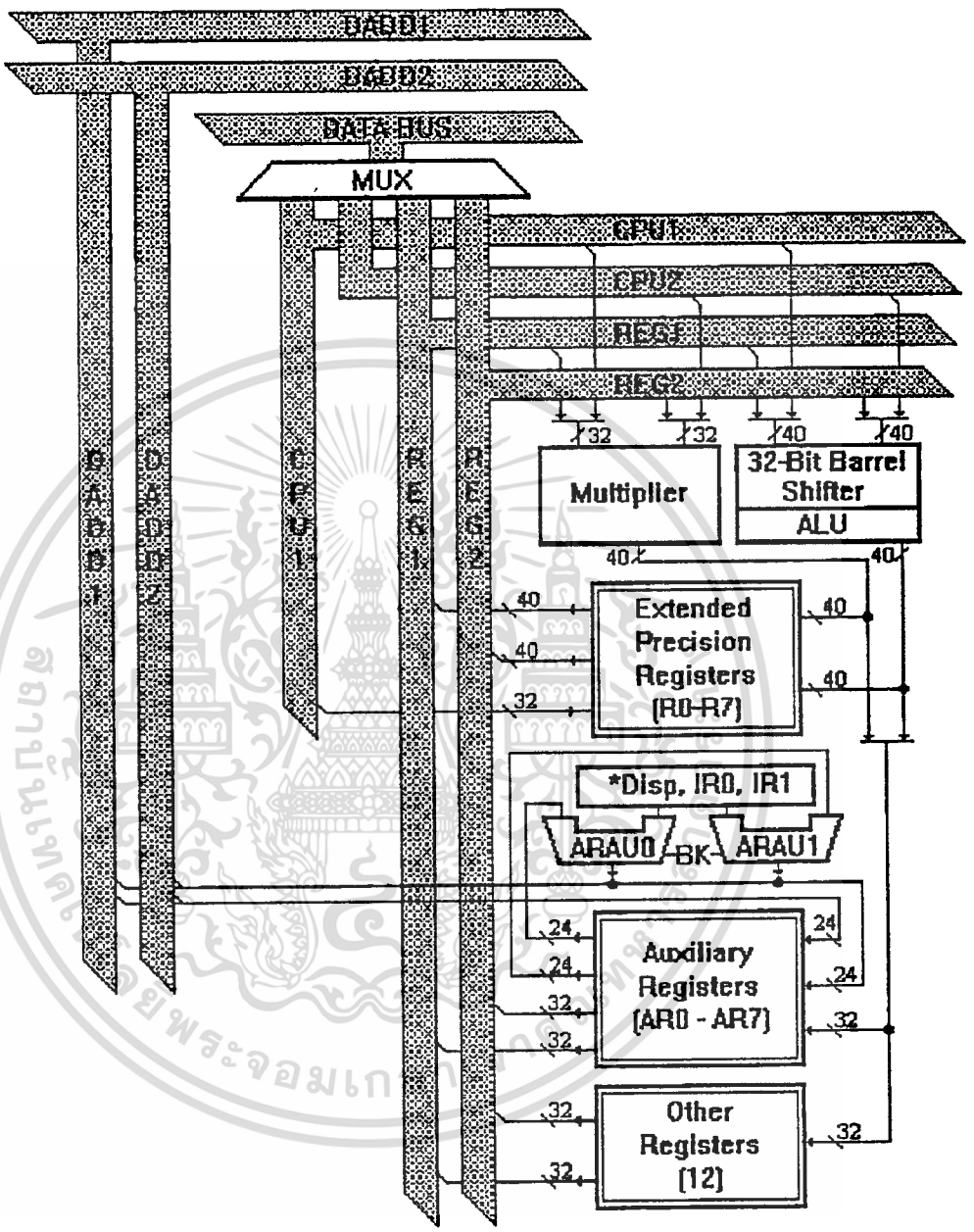
เป็นตัวเก็บข้อมูล ซึ่งจะแสดงถึงสถานะต่าง ๆ ของ CPU เช่น ผลจากการกระทำคำสั่งต่าง ๆ สถานะของ Status Register จะเปลี่ยนแปลงไป เช่น ผลลัพธ์เป็น 0, เป็นค่าลบเป็นต้น ซึ่งจะได้กล่าวถึงโดยละเอียดต่อไป

5.4.11 CPU/DMA Interrupt Enable Register (IE)

เป็นรีจีสเตอร์ขนาด 32บิท ที่บิท 0-10 จะเป็นการให้อนุญาตให้ทำการอินเตอร์รัพท์ได้หรือไม่ การอนุญาตให้ทำการเข้าถึงหน่วยความจำโดยตรง (DMA) จะอยู่ที่ตำแหน่งบิท 16-26 รายละเอียดของรีจีสเตอร์ IE จะอยู่บทต่อไป

5.4.12 CPU Interrupt Flag Register (IF)

เป็นรีจีสเตอร์ขนาด 32 บิท บิท 0-10 จะบอกสถานะของการ อินเตอร์รัพท์ และบิท 16-26 จะบอกสถานะของการทำ ดีเอ็มเอ



* Disp = an 8-bit integer displacement carried in a program control instruction

รูปที่ 5-3 หน่วยประมวลผลกลาง (CPU)

5.4.13 I/O Flags Register (IOF)

ใช้ควบคุมการทำงานของขา XFO และขา XF1 ซึ่งขาเหล่านี้จะถูกติดตั้งให้เป็นอินพุต และ เอาท์พุต รายละเอียดจะอยู่บทต่อไป

5.4.14 Repeat Counter (RC)

เป็นรีจีสเตอร์ขนาด 32 บิท สำหรับเก็บค่าจำนวนครั้งในการกระทำคำสั่งแบบซ้ำ ๆ กันเป็นกลุ่ม (Block Repeat) ถ้าCPU ทำงานในโหมดกระทำซ้ำ (Repeat Mode) CPUจะกระทำซ้ำตั้งแต่ตำแหน่งที่ถูกเก็บอยู่ในรีจีสเตอร์ Repeat Start Address (RS) จนถึงตำแหน่งที่ถูกเก็บในรีจีสเตอร์ Repeat End Address (RE)

5.4.15 Program Counter(PC)

เป็นรีจีสเตอร์ขนาด 32 บิท ใช้เก็บตำแหน่งแอดเดรสของคำสั่งต่อไปที่จะนำเข้ามาทำงาน (fetch)

5.5 การจัดหน่วยความจำ (Memory Organization)

ขนาดหน่วยความจำที่ TMS320C30 สามารถอ้างได้คือ 16 M 32 บิทเวิร์ด ในส่วนของโปรแกรม, ข้อมูล, พอร์ท อินพุต และเอาท์พุต ก็อยู่ใน 16 M นี้

5.5.1 RAM, ROM, และ Cache

รูป 5-4 แสดงถึงหน่วยความจำที่มีอยู่ภายใน TMS 320C30 RAM บล็อก 0 และ 1 แต่ละบล็อกมีขนาด 1K 32 บิท ROM มีขนาด 4K 32 บิท ระบบบัสจะถูกแบ่งเป็นโปรแกรมบัส (Program buses), บัสข้อมูล (Data buses), บัสสำหรับ DMA (DMA buses) เพื่อให้สามารถทำงานแบบขนานได้ เช่น CPU สามารถเข้าถึงข้อมูลที่อยู่ใน RAM Block และสามารถ fetch โปรแกรมจากหน่วยความจำภายนอกพร้อมกับการนำข้อมูลจาก Ram บล็อกอื่นเข้ามา (DMA) ซึ่งทำงานภายใน 1 ไซเคิล

หน่วยความจำแคชขนาด 64x32 บิท เก็บคำสั่งที่มีการใช้งานบ่อย ๆ

5.5.2 การจัดแบ่งหน่วยความจำ (Memory Maps)

การจัดแบ่งหน่วยความจำจะขึ้นอยู่กับว่า โปรเซสเซอร์ทำงานอยู่ในโหมดใด ไมโครโปรเซสเซอร์โหมด (MC/-MP =0) หรือ ไมโครคอมพิวเตอร์โหมด (MC/-MP =1) ทั้ง

สองโหมดจะมีส่วนที่เหมือนกันคือ ที่ตำแหน่งแอดเดรส 800000h ถึง 801FFh ต้องถูกต่อใช้งานโดย Expansion bus เมื่อมีการอ้างอิงหน่วยความจำในส่วนนี้ขาสัญญาณ -MSTRB จะเป็น "0" (MSTRB active) ที่ตำแหน่งแอดเดรส 802000h ถึง 803FFF ถูกสำรองไว้ (ห้ามใช้) ตำ แหน่งแอดเดรส 804000h 805FFFh จะต้องถูกต่อใช้งานโดย Expansion bus เมื่อมีการอ้างอิงหน่วยความจำส่วนนี้ขาสัญญาณ -IOSTRB จะเป็น "0" ตำแหน่งแอดเดรส 806000h ถึง 807FFFh ถูกสำรองใช้ (ห้ามใช้)

ในโหมด ไมโครคอมพิวเตอร์โหมด จะมีหน่วยความจำแบบรอม ได้ถูกกำหนดไว้ที่ตำแหน่ง 000000h - 000FFFh ซึ่งตำแหน่ง 00 - 0Bh(192ตำแหน่ง) จะใช้เก็บ อินเตอร์รัพท์ เวคเตอร์(interrupt vector), แท็รปเวคเตอร์ (Trap vector), และพื้นที่สงวนไว้สำหรับระบบ ในการเข้าถึงหน่วยความจำตำแหน่ง 001000h - 7FFFFFh จะต้องมีหน่วยความจำภายนอก (RAM) ต่อร่วมอยู่ด้วย ในโหมดไมโครโปรเซสเซอร์โหมด จะไม่มีหน่วยความจำแบบรอม ดังนั้นจึงต้องมีหน่วย ความจำภายนอก(External RAM) ต่อร่วมอยู่ด้วย

5.5.3 Memory addressing mode

TMS320C30 สนับสนุนการทำคำสั่งทั่ว ๆ ไป และเหมาะสมอย่างยิ่งสำหรับงานทางด้านประมวลผลสัญญาณเชิงเลขในการเขียนโปรแกรม จึงจำเป็นที่จะต้องทราบโหมดการอ้างแอดเดรสแบบต่างๆ สำหรับ TMS320C30 จัดแบ่งเป็น 5 กลุ่มดังนี้

1. General addressing mode
   - Register. แบบนี้ตัวที่ถูกกระทำ (operand) คือรีจิสเตอร์ของซีพียู
   - Short immediate. แบบนี้ตัวที่ถูกกระทำ (operand) คือข้อมูลตัวเลขขนาด 16บิท (ข้อมูลตัวเลขขนาด 16 บิท อยู่ใน instruction code)
   - Direct. แบบนี้ตัวที่ถูกกระทำ (operand) คือแอดเดรสขนาด 24 บิท
   - Indirect. ตำแหน่ง (address) ของตัวที่ถูกกระทำ (operand) อยู่ใน AR (Auxiliary register)
2. Three operand addressing mode
   - Register. เหมือนกับในแบบ General addressing mode
   - Indirect. เหมือนกับในแบบ General addressing mode
3. Parallel addressing mode
   - Register. ตัวที่ถูกกระทำ (operand) คือ R0-R7 (Extended precision register)

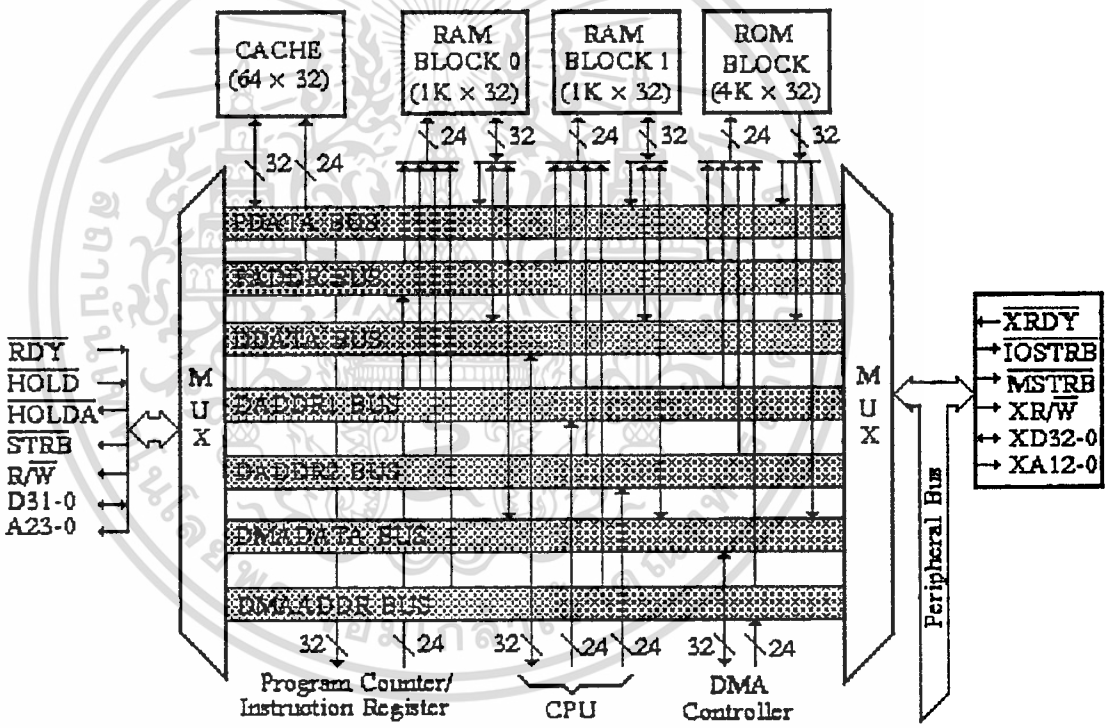
- Indirect. เหมือนกับในแบบ General addressing mode

4. Long immediate addressing mode

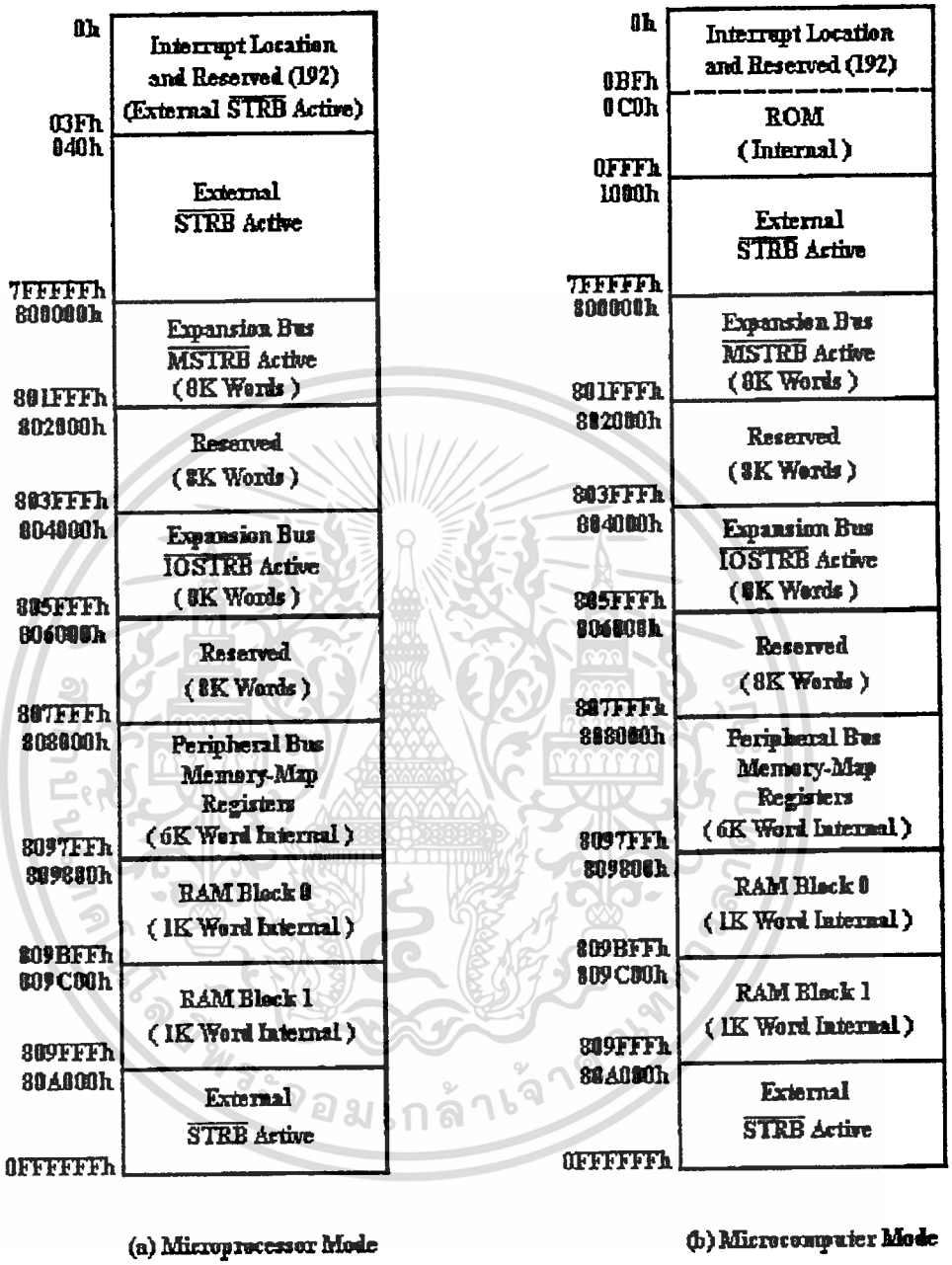
- Long immediate addressing mode แบบนี้ตัวที่ถูกกระทำ(operand) คือ ข้อมูลตัวเลขขนาด 24 บิท

5. Conditional branch addressing mode

- Register. เหมือนกับในแบบ General addressing mode
- PC-relative. ข้อมูลตัวเลขขนาด 16 บิท แบบมีเครื่องหมายจะถูกบวกเข้ากับ รีจีสเตอร์ PC (Program counter)



รูปที่ 5-4 แสดงการจัดหน่วยความจำ



รูปที่ 5-5 แสดงส่วนต่าง ๆ ในหน่วยความจำหลัก (Memory Map)

5.5.4 หน่วยความจำส่วนของ การรีเซ็ท,อินเตอร์รัพท์,แท็รป

(Reset/Interrupt/Trap Vector)

แอดเดรสในส่วนที่ใช้จัดการกับการรีเซ็ท, การอินเตอร์รัพท์, และคำสั่งแท็รป คือตำแหน่งที่ 00h - 3Fh ซึ่งแอดเดรสในส่วนนี้ จะใช้เก็บค่าตำแหน่งของโปรแกรมที่ต้องการจะให้กำงาน เมื่อเกิดการีเซ็ท, การอินเตอร์รัพท์, หรือการทำคำสั่งแท็รปโดยแอดเดรส 00h ถูกใช้เมื่อเกิดรีเซ็ท, แอดเดรส 01h - 0Bh ถูกใช้เมื่อเกิดการอินเตอร์รัพท์ และแอดเดรส 20h - 3Bh ถูกใช้เมื่อเกิดคำสั่งแท็รป ส่วน แอดเดรสที่ 0Ch - 1Fh และ 3Ch - 3Fh ถูกสงวนไว้สำหรับระบบ ห้ามใช้ ดังแสดงในรูปที่ 5-6

00h	RESET
01h	INT0
02h	INT1
03h	INT2
04h	INT3
05h	XINT0
06h	RINT0
07h	XINT1
08h	RINT1
09h	TINT0
0Ah	TINT1
0Bh	DINT
0Ch	Reserved
1Fh	
20h	TRAP 0
	:
3Bh	TRAP 27
3Ch	TRAP 28 (Reserved)
	: (Reserved)
3Fh	TRAP 31 (Reserved)

รูปที่ 5-6 รีเซ็ทเวคเตอร์, อินเตอร์รัพท์เวคเตอร์, และแท็รปเวคเตอร์

5.5.5 สรุปชุดคำสั่ง (Instruction set summary)

สำหรับชุดคำสั่งเรียงลำดับตามตัวอักษรซึ่งจะแสดง ชื่อย่อคำสั่ง, ความหมาย และการทำงาน การทำงานของคำสั่งต่างๆโดยละเอียด จะได้กล่าวถึงในบทต่อไป ส่วนคำสั่งต่าง ๆ แสดงไว้ในคู่มือการใช้

5.6 การทำงานของบัสภายใน (Internal Bus Operation)

การที่ TMS320C30 มีประสิทธิภาพความสามารถสูง เพราะระบบบัสภายในและการทำงานแบบขนานระบบบัสถูกแบ่งออกเป็น โปรแกรมบัส (PADDR และ PDATA) และดาต้าบัส (DADD1, DADDR2, และ DDATA) เพื่อให้การเฟ็ช(fetch), การเข้าถึงข้อมูล, และการทำ DMA ได้พร้อมกัน ระบบบัสเหล่านี้ได้ถูกต่อเข้ากับอุปกรณ์และหน่วยความจำภายในและภายนอก ดังแสดงในรูป 5-4

โปรแกรมเคาท์เตอร์ (Program Counter) 24บิท ถูกต่อเข้ากับโปรแกรมแอดเดรสบัส (PADDR) รีจีสเตอร์ IR ถูกต่อ เข้ากับ โปรแกรมดาต้าบัส (PDATA) บัสเหล่านี้ใช้สำหรับเฟ็ชคำสั่งจากหน่วยความจำ ซึ่งคำสั่งมี ขนาด 32 บิทเวิร์ด ดังแสดงในรูป 5-3

ดาต้าแอดเดรสบัสขนาด 24บิท (DADDR1 และ DADDR2) และดาต้าดาต้าบัส (DDATA) สนับสนุนการเข้าถึงข้อมูล (data) สองข้อมูล ดาต้าดาต้าบัสเป็นตัวส่งข้อมูลให้กับบัส CPU1 และบัสCPU2

ซึ่งบัส CPU1 และบัส CPU2 นี้เป็นตัวส่งข้อมูล (2 data memory operand) ให้กับตัวคูณ (Multiplier), ALU, และรีจีสเตอร์ต่าง ๆ ทุก ๆ รอบคำสั่ง(machine cycle) ตัวควบคุม DMA สนับสนุนการทำงานกับแอดเดรสขนาด 24 บิท (DMAADDR) และ 32บิท ดาต้าบัส (DMADATA) บัสเหล่านี้ใช้สำหรับการเข้าถึงหน่วยความจำโดยตรง (DMA) เพื่อให้การทำงานเป็น แบบขนานกับ การเข้าถึงหน่วยความจำของดาต้าบัสและโปรแกรมบัส

5.7 การทำงานของบัสภายนอก (External Bus Operation)

TMS320C30 มีระบบบัสที่ทำการติดต่อกับภายนอก สองแบบคือ Primary bus และ Expansion bus ทั้งสองแบบมีบัสข้อมูลขนาด 32 บิท และชุดสัญญาณควบคุม Primary bus มีแอดเดรสขนาด 24 เส้นในขณะที่ Expansion bus มีขนาด 13 เส้น ระบบบัสทั้งสองนี้ใช้สำหรับ ติดต่อกับ โปรแกรมภายนอก, ข้อมูลจากหน่วยความจำภายนอก หรือ ติดต่ออุปกรณ์ภายนอก ระบบบัสเหล่านี้จะมีขาสัญญาณ -RDY ซึ่งเป็นขาอินพุท สำหรับการทำสถานะรอ (wait state)

5.7.1 การขออินเตอร์รัพท์จากภายนอก (External Interrupts)

การขออินเตอร์รัพท์จากภายนอก มีทั้งหมด 4 อินเตอร์รัพท์ (INT0-INT3) และขาสัญญาณรีเซ็ท (RESET) ซึ่งเป็นการอินเตอร์รัพท์แบบปฏิเสธไม่ได้ (nonmaskable) การทำ DMA ก็เป็นการอินเตอร์รัพท์อีกแบบหนึ่ง ขาสัญญาณ -IACK เป็นขาสัญญาณที่ใช้ตอบรับการขออินเตอร์รัพท์

5.7.2 สัญญาณการทำงานแบบประสานงานกัน (Interrupt-Instruction Signaling)

ขาสัญญาณภายนอกสองขา XF0 และ XF1 สามารถที่จะกำหนดให้เป็น อินพุต (input) หรือ เอาท์พุต (output) ได้โดยการควบคุมทางซอฟทแวร์ขาสัญญาณทั้งสองนี้ถูกใช้โดยการทำงานแบบประสานกัน (Interlock operation) ของ TMS320C30 การทำงานแบบประสานงานกันนี้ ก็จะมีกลุ่มคำสั่งทางด้านนี้ (Interlock-operation instruction group) เพื่อสนับสนุนการทำงานแบบโปรเซสเซอร์หลายตัว (Multiprocessor communication)

5.8 อุปกรณ์สนับสนุนภายในชิพ (Peripherals)

อุปกรณ์สนับสนุนภายทั้งหมดใน TMS320C30 ถูกควบคุมในลักษณะที่เสมือนกับเป็นหน่วยความจำ โดยการส่งผ่านข้อมูลผ่านทางบัสอุปกรณ์สนับสนุน (Peripheral Bus) บัสข้อมูลอุปกรณ์สนับสนุน (Peripheral Data Bus) มีขนาด 32 บิท บัสแอดเดรสอุปกรณ์สนับสนุน (Peripheral Address Bus) มีขนาด 24 บิท อุปกรณ์สนับสนุนภายใน TMS320C30 มีตัวจับเวลา (Timer) 2 ตัว และพอร์ทอนุกรม (Serial port) จำนวน 2 พอร์ท ดังในรูป 5-7

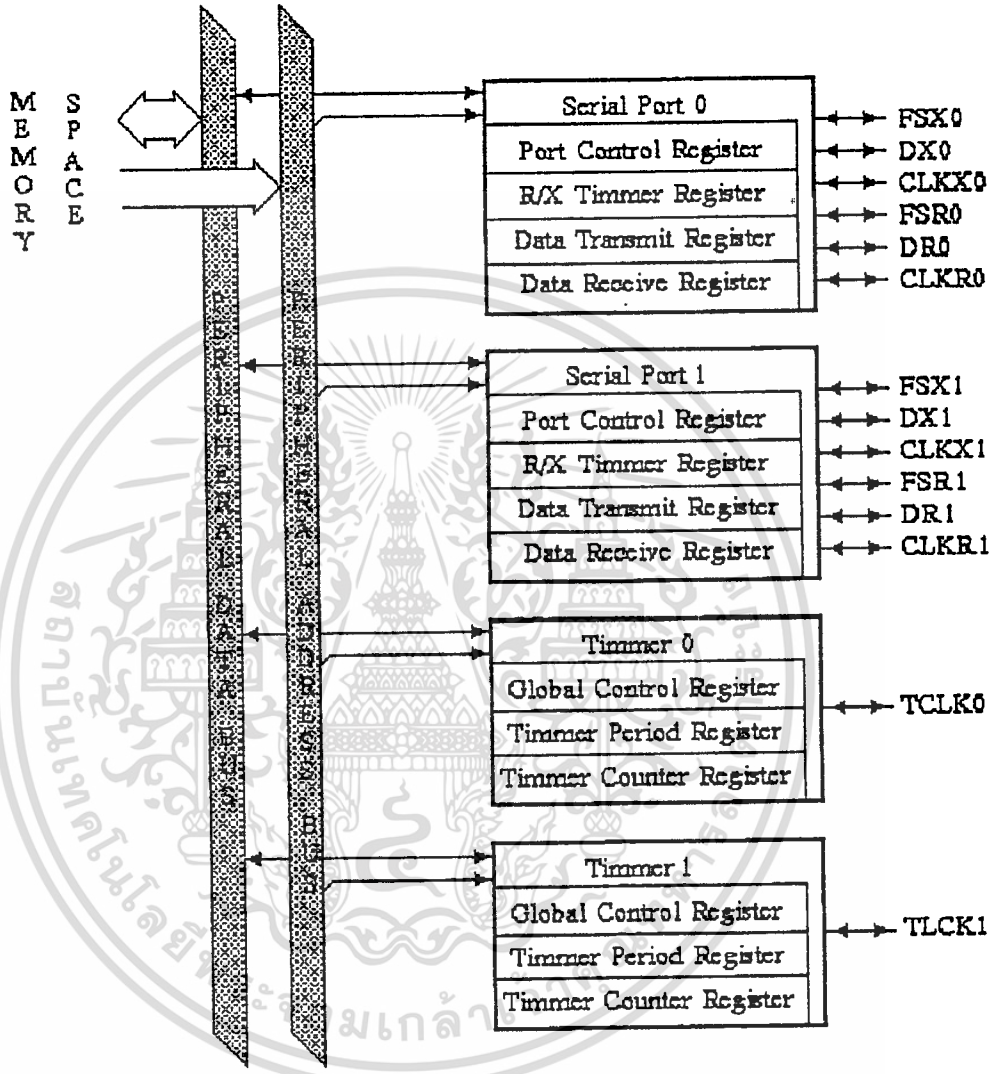
5.8.1 ตัวจับเวลา (Timers)

ตัวจับเวลาทั้งสองใช้งานคล้ายกับตัวจับเวลาทั่ว ๆ ไปซึ่งมีขนาด 32 บิท มีสองโหมดคือ ใช้สัญญาณที่ใช้ในการนับ (Clock) สามารถเลือกได้ว่าจะใช้สัญญาณจากภายนอก หรือ จากสัญญาณนาฬิกาจากภายใน

5.8.2 พอร์ทสื่อสารอนุกรม (Serial Port)

พอร์ทอนุกรมทั้งสองพอร์ทนั้นเป็นอิสระต่อกัน โดยจะมีรีจีสเตอร์ควบคุมเป็นของใครของมัน แต่ละพอร์ทสามารถกำหนดให้ส่งข้อมูลแบบ 8, 16, 24, หรือ 32 บิท สัญญาณนาฬิกาของแต่ละพอร์ท (Input clock) สามารถกำหนดได้ว่าจะใช้สัญญาณจากภาย

นอก หรือสัญญาณนาฬิกาจากภายใน สัญญาณนาฬิกาจากภายในสามารถที่จะถูกหารความถี่ลงได้

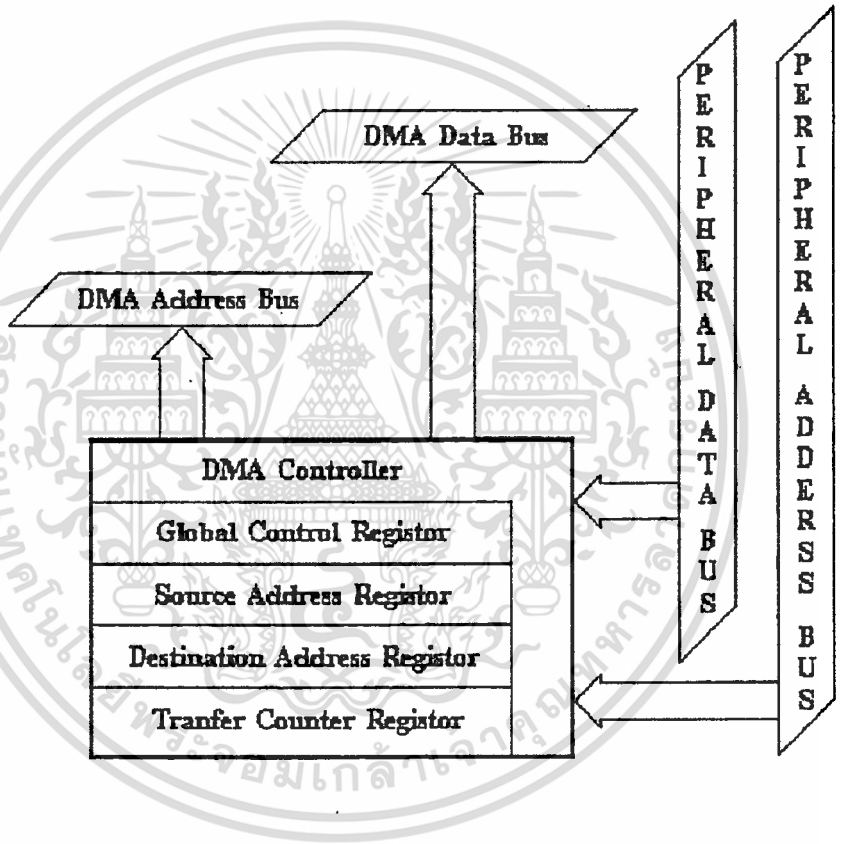


รูปที่ 5-7 อุปกรณ์สนับสนุนภายใน (Peripheral Module)

5.9 การเข้าถึงหน่วยความจำโดยตรง (Direct Memory Access)

ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA) สามารถ อ่านหรือเขียนข้อมูลตำแหน่งใด ๆ ในหน่วยความจำโดยไม่จำเป็นต้องผ่านการทำงานโดย CPU ดังนั้น

TMS320C30 จึงสามารถที่จะติดต่อกับหน่วยความจำหรือ อุปกรณ์ภายนอกที่มีความเร็วต่ำ โดยไม่ทำให้ปริมาณงานต่อหนึ่งหน่วยเวลาของ CPU ลดลง (Throughput) ตัวควบคุมการเข้าถึงหน่วยความจำโดยตรง จะมีตัวสร้างสัญญาณแอดเดรส, รีจีสเตอร์ต้นทาง (Source Register), รีจีสเตอร์ปลายทาง (Destination Register), และตัวนับจำนวนการส่งข้อมูล (Transfer Counter Register) ดังในรูปที่ 5-8



รูปที่ 5-8 ชุดควบคุมการเข้าถึงหน่วยความจำโดยตรง (DMA Controller)

บทที่ 6
ฮาร์ดแวร์บอร์ดทดลอง

6.1 การ์ดอินเตอร์เฟส (Interface Card)

ลักษณะโดยทั่วไปของการ์ดอินเตอร์เฟส (Interface Card) ตัวนี้การติดต่อกับอุปกรณ์ภายนอกจะทำการรับส่งข้อมูลครั้งละ16 บิท เพื่อความเร็วในการรับส่งข้อมูล การ์ดอินเตอร์เฟสที่ออกแบบนี้ ดูในรูปจะเห็นว่าประกอบด้วย ดาต้าบัสถูกต่อออกจากสล็อต PC ออกมาที่เอาท์พุตโดยตรง, เอาท์พุตพอร์ทขนาด 8 บิท, สัญญาณควบคุมจากสล็อต, และสัญญาณ CS_0-CS_7 ซึ่งได้จากการ โค้ดสัญญาณแอดเดรส, และสัญญาณควบคุม (Control Bus) การที่อินเตอร์เฟสการ์ด ต้องออกแบบเช่นนี้ก็เพื่อต้องการที่จะลดจำนวนสายสัญญาณที่จะต้องต่อระหว่างอินเตอร์เฟสการ์ดกับ DSP บอร์ด ซึ่งอินพุตพอร์ท และเอาท์พุตพอร์ทจะอยู่บน DSP บอร์ดจากรูปที่ 6-1 จะเห็นว่ามีจำนวนสายบัสข้อมูลเท่ากับ 16 เส้น, CS_0-CS_6 เท่ากับ 7 เส้น, Control เท่ากับ 8 เส้น และ A_0 , A_1 , IOW, IOR, IRQ, Vcc, GND รวม 7 เส้น รวมทั้งหมด 38 เส้น ซึ่งสามารถใช้คอนเน็คเตอร์ขนาด 40 ขาได้

6.1.1 การดีโค้ดสัญญาณแอดเดรส (Decode Address)

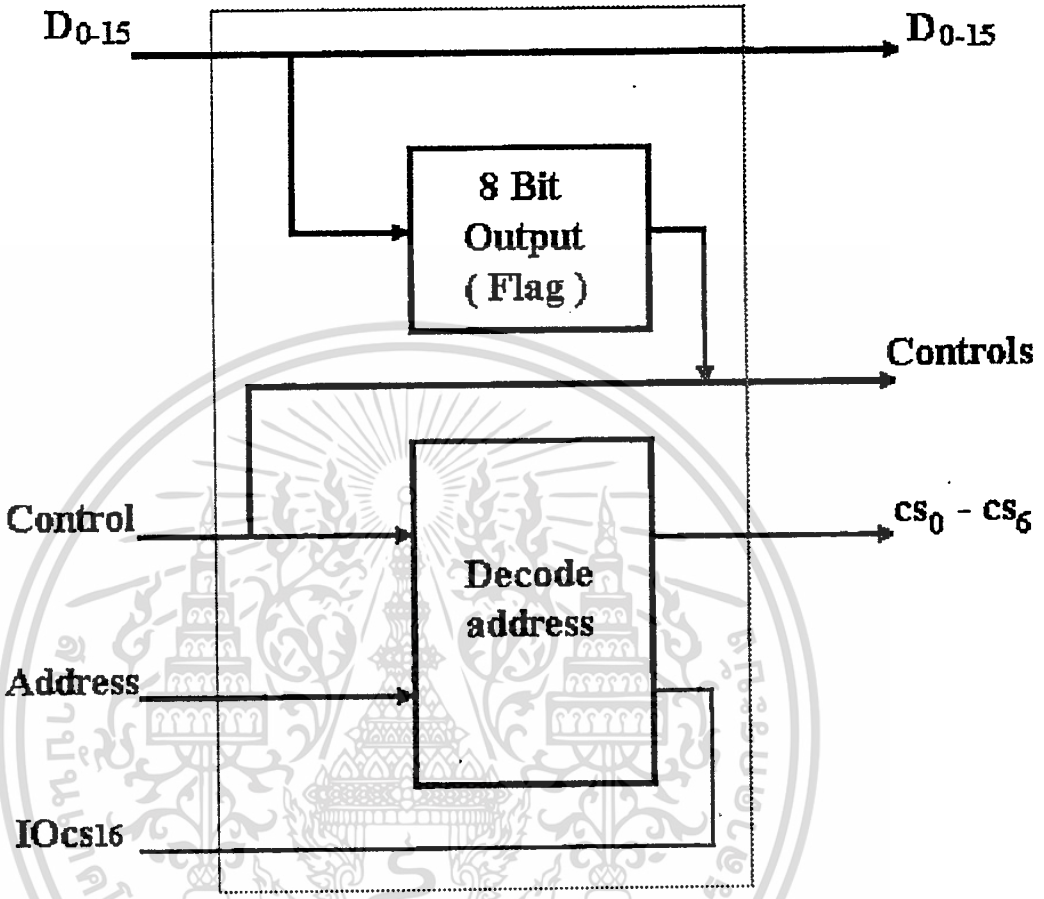
หัวข้อนี้จะอธิบายถึงการดีโค้ด แอดเดรสดูจากรูปที่ 6-2 ไอซีเบอร์ 74 ALS688 เป็นตัวเปรียบเทียบสัญญาณแอดเดรส A_8 - A_{16} (Comparator) ถ้าสัญญาณแอดเดรสมีค่าตรงกับค่าดิพสวิทช์ (Dip Switch) ที่ได้ตั้งไว้ไอซีเบอร์ 74ALS688 ก็จะให้สัญญาณเอาท์พุตเป็น"0"และสัญญาณนี้จะถูกนำไป ออร์(OR)กับสัญญาณ AEN (Address Enable) แล้วถูกส่งไปควบคุมไอซี เบอร์ 74ALS138 ซึ่งหมายความว่าเงื่อนไขประการหนึ่งที่จะทำให้ ไอซี74 ALS138 ทำการดีโค้ดแอดเดรสได้ คือ A_8- A_{16} มีค่าตรงกับดิพสวิทช์และสัญญาณ AEN (AEN มีค่า= 0) สัญญาณ A_7 A_6 , A_5 ถูกนำเข้าที่นอร์เกท (NOR) ซึ่งหมายความว่าสัญญาณ A_7 A_6 , A_5 ต้องเป็น "1" จึงจะได้เอาท์พุทเป็น "0" แล้วถูกนำไปเป็นส่วนหนึ่งของการควบคุมการดีโค้ดแอดเดรสโดยไอซีเบอร์ 74ALS138 ,สัญญาณ IOR และIOW ถูกต่อเข้ากับนอร์เกท ซึ่งเอาท์พุตจะได้เป็น IORQ (IO Request) แล้วนำไปต่อกับขา G1 (ขา Gate)ของ ไอซีเบอร์ 74ALS138 เพื่อควบคุมการทำงานส่วนA_4, A_3, A_2 ถูกนำเข้ามาดีโค้ดโดยไอซีเบอร์ 74 ALS138 และ A_1, A_0 ถูกต่อออกไปเป็นเอาท์พุตของอินเตอร์เฟสการ์ด ดังนั้นสัญญาณเอาท์พุตของ74ALS138 แต่ละเส้น สามารถนำไปควบคุมพอร์ทได้ 4 พอร์ท (4 Address Port Number) โดยใช้สัญญาณ $\bar{C}S$ (Chip Select) ร่วมกับสัญญาณ A_0 , A_1 สัญญาณเอาท์พุตท

Y_0-Y_7 ครอบคลุมแอดเดรสใดบ้างสามารถ ดูได้จากตารางที่ 6-1 ในตารางที่ 1 XX คือค่าที่กำหนดโดยดิพ สวิทช์

สัญญาณ ˜CS	หมายเลขพอร์ท (Hex)
˜CS0 (Y_0)	XXE0 - XXE3
˜CS1 (Y_1)	XXE4 - XXE7
˜CS2 (Y_2)	XXE8 - XXEB
˜CS3 (Y_3)	XXEC - XXEF
˜CS4 (Y_4)	XXF0 - XXF3
˜CS5 (Y_5)	XXF4 - XXF7
˜CS6 (Y_6)	XXF8 - XXFB
˜CS7 (Y_7)	XXFC - XXFF

ตารางที่ 6-1 แสดงหมายเลขที่ดีโค้ดแอดเดรสโดยไอซีเบอร์ 74ALS138

ในการอินเตอร์เฟสครั้งละ 16 บิทนั้น ในขณะที่ทำการรับส่งข้อมูลต้องสร้างลอจิก "0" ให้กับขา ˜I/0CS16 เพื่อบอกว่าต้องการติดต่อครั้งละ 16 บิท ดังนั้นจึงต้องนำขาสัญญาณที่ควบคุมไอซีเบอร์ 74ALS138 คือ ขา G2A และ G2B ผ่าน OR gate แล้วนำเอาท์พุทต่อเข้ากับ I/OCS16 ซึ่งการทำเช่นนี้หมายความว่า การที่จะได้สัญญาณลอจิก "0" ส่งให้กับของ ˜I/0CS16 คือ หมายเลขพอร์ท ต้องอยู่ระหว่าง XXE0 - XXFF เท่านั้น

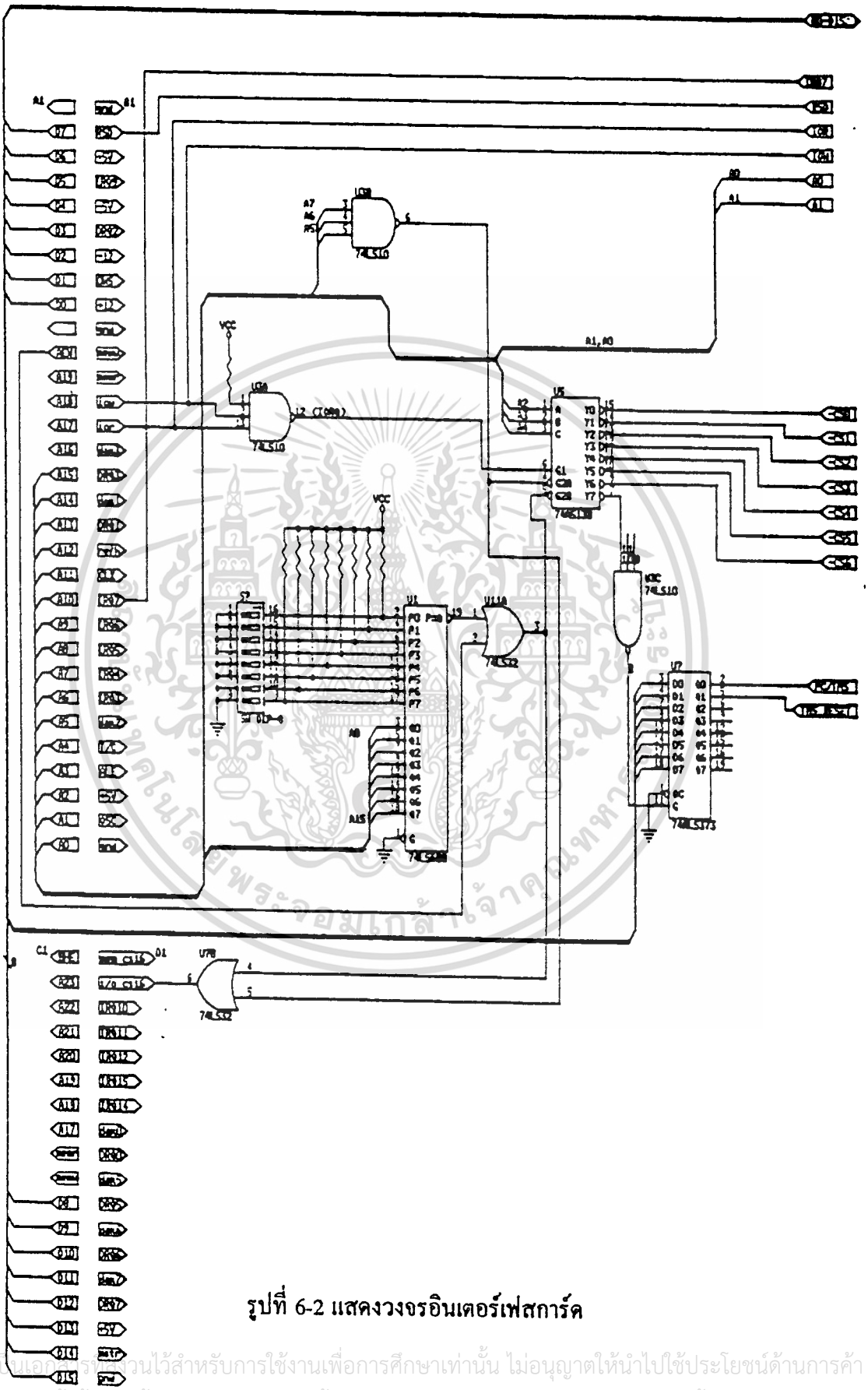


รูปที่ 6-1 บล็อคไดอะแกรม ของ อินเตอร์เฟสการ์ด

6.1.2 สัญญาณควบคุม (Control Signal)

สัญญาณควบคุมต่าง ๆ ที่ต่อ ออกจากอินเตอร์เฟสการ์ดทั้งหมดมีดังนี้ :-

- ˜CS0 - ˜CS7 ได้กล่าวถึง และอธิบายการใช้สัญญาณนี้แล้วในหัวข้อ 1.1
- A0 , A1 สำหรับทำการดีโค้ด แอดเดรสในกรณีที่ DSP บอร์ดต้องการพอร์ท จำนวนมากกว่า 7 พอร์ท หรือ นำไปต่อร่วมกับ ไอซีเบอร์ 8255
- RESET DRVใช้สำหรับนำไปรีเซ็ทอุปกรณ์ภายนอก เมื่อเริ่มเปิดเตรื่องคอมพิวเตอร์ หรือรีเซ็ท เตรื่องคอมพิวเตอร์
- ˜IOR , ˜IOW สำหรับควบคุมการอ่าน (IOR) หรือการเขียน (IOW) ข้อมูลของอินพุต, และเอาท์พุตพอร์ท



รูปที่ 6-2 แสดงวงจรอินเตอร์เฟสการ์ด

- IRQ7 เป็นอินพุตสำหรับทำการ ขัดจังหวะของ PC (Interupt) จากอุปกรณ์ภายนอกที่ร้องขอเข้ามา
- พอร์ทควบคุม(8 Bit Latch) ใช้ในการควบคุม DSPซิงเกิลบอร์ด เป็นเอาท์พุตพอร์ทขนาดขนาด 8 บิทโดยใช้ไอซีเบอร์ 74ALS373 ที่ใช้ในการควบคุม DSP ซิงเกิลบอร์ด นั้นมีสองสัญญาณคือ สัญญาณ ¯PC/TMS (D0) สัญญาณนี้จะถูกใช้เป็นสัญญาณเลือกว่าจะให้ TMS320C30 หรือ PC ทำการติดต่อกับหน่วยความจำ หรือวงจรแปลงสัญญาณ ดิจิตอล เป็นอนาล็อก หรือ วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลและสามารถหยุดการทำงาน TMS320C30 ชั่วคราว (เมื่อสัญญาณ ¯PC/TMS = "0") สัญญาณ TMS_RESET (D1) สัญญาณนี้ถูกต่ออยู่กับ ขารีเซ็ทของ TMS320C30 เพื่อให้สามารถทำการรีเซ็ทTMS320C30 ได้ตามที่ต้องการ สามารถทำได้โดยการทำให้เกิด การเปลี่ยนแปลงจากลอจิก"1" ไปเป็นลอจิก "0" (แอคทีฟที่ขอบขาลง) ตัวอย่างการสร้างสัญญาณ เปลี่ยนแปลงจากลอจิก"1" ไปเป็นลอจิก "0" สามารถทำได้ดังนี้

ตัวอย่างโดยใช้ภาษาซี

```
OUTPORT (CONTROL, Status | 0x02);
OUTPORT (CONTROL, Status & 0x02);
OUTPORT (CONTROL, Status | 0x02);
```

6.1.3 รายละเอียดของสัญญาณต่าง ๆ บนสล็อต

(I),(O) และ (I/O) หมายถึงทิศทางของขาสัญญาณเมื่อเทียบกับเมนบอร์ดโดย

(I) หมายถึง ขาสัญญาณอินพุต

(O) หมายถึง ขาสัญญาณเอาต์พุต

(I/O) หมายถึง ขาสัญญาณที่เป็นได้ทั้งอินพุตและเอาต์พุต

(*I/O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณเอาต์พุตแต่จะเป็นอินพุตในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้า หมายถึงขาสัญญาณที่แอคทีฟที่ลอจิก "0" และขาสัญญาณที่ไม่มีหรือมีเครื่องหมายบวกอยู่หน้า หมายถึงขาสัญญาณที่แอคทีฟที่ลอจิก"1" สัญญาณที่ต่ออยู่บนสล็อตนี้สามารถขับ ICTTL ชนิด Low Powerได้สองตัว โดยไม่ทำให้เกิดการโหลดหรือการเพี้ยนของสัญญาณ ขาสัญญาณต่าง ๆ บนสล็อตของ XT และ AT สามารถแบ่งออกเป็นกลุ่ม ๆ ได้ดังนี้

Power Supply

Ground ขาสัญญาณนี้ต่ออยู่กับกราวด์ของระบบเรกูเลเตอร์

+ 5 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ + 5 โวลต์

- 5 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ - 5 โวลต์

+12 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ + 12โวลต์

-12 V ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ - 12โวลต์

แอดเดรสบัส และสัญญาณต่างๆที่เกี่ยวข้อง

SAO-SA19 (I) เป็นแอดเดรสบิตที่ 0 ถึง 19 โดยที่ SA0 มีนัยสำคัญต่ำที่สุด ขาสัญญาณนี้จะแอคทีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็น "1" และจะถูกแลตช์ไว้ตอนขอบขาลงของขาสัญญาณ BALE แอดเดรสทั้ง 20 บิตนี้สามารถอ้างหน่วยความจำได้ถึง 1 เมกะไบต์ XT และ สำหรับ AT เมื่อใช้ร่วมกับ LA17-LA23 จะอ้างถึง 16 เมกกะไบต์

LA17-LA23 (*I/O) (เฉพาะรุ่น AT) ขาสัญญาณนี้จะแอคทีฟเมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก "1" แต่จะไม่มีการแลตช์ไว้ตอนขอบขาลงของขาสัญญาณ BALE ดังนั้นถ้าอุปกรณ์ I/O ไม่มีการอ้างแอดเดรสเกิน 1 เมกะไบต์ ขาสัญญาณนี้ก็ไม่จำเป็นจะต้องใช้ แต่ถ้ามีการอ้างแอดเดรสเกินอุปกรณ์ I/O จะต้องทำการแลตช์สัญญาณนี้ โดยใช้ขอบขาลงของขาสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR

AEN (O) (Address Enable) ขาสัญญาณนี้จะแอคทีฟเมื่อตัวควบคุม DMA ได้ทำการควบคุมบัสต่าง ๆ ของระบบแล้ว ดังนั้นการอ้างพอร์ตของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ในการดีโค้ดด้วย เพื่อที่จะไม่ทำให้เกิดการติดต่อระหว่างระบบกับอุปกรณ์ I/O ตัวอื่นยกเว้นตัวที่ทำขบวนการ DMA อยู่

BALE (O) (Address Latch Enable) ขาสัญญาณนี้ใช้ในการแสดงการเริ่มต้นของขบวนการต่าง ๆ ที่มีการติดต่อกับหน่วยความจำโดยจะแอคทีฟเมื่อค่าแอดเดรสที่ CPU ต้องการติดต่อด้วยอยู่บนแอดเดรสบัสเรียบร้อยแล้ว ตามปกติขอบขาลงของสัญญาณนี้จะทำให้เกิดการแลตช์สัญญาณ SAO-SA19 และถ้ามีการอ้างแอดเดรสเกิน 1 เมกกะไบต์ใน AT จะใช้ขอบขาลงสัญญาณนี้ในการแลตช์สัญญาณ LA17-LA23 ด้วยเช่นกันแต่สำหรับในขบวนการ DMA สัญญาณนี้จะมีสถานะเป็น "1" ตลอด

SBHE (*I/O) (เฉพาะรุ่น AT) (Bus High Enable) เป็นขาสัญญาณที่ใช้แสดงว่ามีการรับส่งข้อมูลในบิตที่ SD8-SD15.

ดาต้าบัส

SDO-SD7 (I/O) สำหรับรุ่น AT จะมี SD0-SD15 เพิ่มขึ้นมาด้วยคือ ดาต้าบิต 0 ถึง 7 สำหรับรุ่น XT และ สำหรับรุ่น AT คือ ดาต้าบิต 0 ถึง 15โดยที่ SDO มีนัยสำคัญต่ำที่สุด สำหรับ AT ถ้ามีการติดต่อกับบิตที่ SD8-SD15 สามารถตรวจสอบได้จากขาสัญญาณ SBHE

สัญญาณอินเตอร์รัพต์

IRQ2-IRQ7 (I) (Interrupt Request) (สำหรับรุ่น AT จะเป็น IRQ3-7,9-12,14,15) เป็นขาสัญญาณอินเตอร์รัพต์ CPU สำหรับรุ่น AT ลำดับความสำคัญของสัญญาณ IRQ เป็นดังนี้คือ 9,10,11,12,14,15,3,4,5,6 และ 7 โดย IRQ9 มีลำดับความสำคัญมากที่สุดและIRQ7 มีลำดับความสำคัญน้อยที่สุด สำหรับ XT IRQ2 มีลำดับความสำคัญมากที่สุดและรอง ๆ ลงไปคือ IRQ3,4,5,6,7 สำหรับรายละเอียดในการใช้งานของแต่ละอินเตอร์รัพต์ให้ดูจากตารางการจัดลำดับอินเตอร์รัพต์ในหัวข้อการอินเตอร์รัพต์โดยปกติสัญญาณนี้จะมีสถานะเป็น "0" เสมอถ้าต้องการอินเตอร์รัพท์ CPU ให้ส่งพัลส์ที่เป็นลอจิก "1" ให้กับมัน โดยไม่จำเป็นต้องคำนึงถึงคาบเวลาของพัลส์ทั้งนี้เพราะระบบของ IBM ตัวอินเตอร์รัพต์คอนโทรลเลอร์ (8259 Interrupt Controller) จะถูกโปรแกรมให้ทำการตรวจสอบสัญญาณอินเตอร์รัพต์โดยใช้ขอบขาลงของสัญญาณนี้

I/O CH CK (I/O Channel Check) เป็นขาสัญญาณที่บอกถึงความผิดพลาด ในการรับส่งข้อมูล ซึ่งตรวจสอบจากพาริตี้บิต ถ้าพาริตี้บิตที่อ่านจากหน่วยความจำกับพาริตี้บิตที่สร้างขึ้นจากขบวนการรับส่งข้อมูลมีค่าไม่เท่ากันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูลสัญญาณนี้จะเกิดการอินเตอร์รัพต์ CPU แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิดความผิดผลาดทางพาริตี้ (Parity Error) ขึ้น CPU จะแสดงข้อความบอกความผิดพลาดขึ้นและจะหยุดการทำงาน (Halt) เพื่อให้ ผู้ใช้ตรวจสอบหาสาเหตุของความผิดพลาด

สัญญาณที่ใช้ในขบวนการ DMA

DRQ1-DRQ3 (DMA Request) (สำหรับรุ่น AT จะเป็น DRQ0-3,5-7) เป็นขาสัญญาณใช้ในการขอทำขบวนการ DMA โดยที่ DRQ0 มีลำดับความสำคัญมากที่สุด และ

DRQ3 มีลำดับความสำคัญน้อยที่สุด สำหรับรุ่น XT และสำหรับรุ่น AT ขา DRQ7 จะมีลำดับความ สำคัญน้อยที่สุด บน XT DRQ0 ใช้สำหรับการรีเฟรชหน่วยความจำแบบ ไดนามิค จึงไม่มีขาสัญญาณ DRQ0 ต่อออกมาที่สล็อต แต่สำหรับ AT แล้วจะมีวงจรโดยเฉพาะสำหรับใช้ในการรีเฟรชหน่วยความจำแบบไดนามิคอยู่แล้ว ดังนั้นขา DRQ0 จึงว่างลงและนำมาต่อที่สล็อต เพื่อให้อุปกรณ์ I/O ได้ใช้งานได้บางครั้งอาจเรียก DRQ0 เป็น DRQ4 ก็ได้ เพื่อป้องกันความสับสนกับส่วนที่ใช้รีเฟรชไดนามิคแรมบน XT (เช่นเดียวกัน ก็จะเรียก DACK0 เป็น DACK4) การขอทำ DMA ทำได้โดยทำให้ขาสัญญาณนี้มีสถานะเป็น "1" แล้วรอจนกระทั่งได้รับตอบสนองการทำ DMA จาก CPU โดยการตรวจสอบสัญญาณ DACK ที่ส่งออกมา

-DACK0-3 (O) (DMA Acknowledge) (สำหรับรุ่น AT จะเป็น -DACK 0-3, 5-7) เป็นสัญญาณตอบสนองการขอทำ DMA ของอุปกรณ์ I/O เพื่อให้อุปกรณ์ I/O ทราบว่าการขอทำขบวนการ DMA นั้นได้รับการตอบสนองแล้ว เช่นถ้ามีการขอทำ DMA ผ่านทาง DRQ2 และเมื่อ CPU รับรู้แล้วจะทำให้สัญญาณ DACK2 แอคทีฟ

ถึงแม้ว่าบน XT จะมีการนำเอา DRQ0 ไปใช้ในการรีเฟรชไดนามิคแรมก็ตาม แต่สัญญาณ -DACK0 ก็จะถูกต่อออกมาที่สล็อตด้วย เพื่อแสดงถึงขบวนการรีเฟรชไดนามิคแรม และ อุปกรณ์ I/O สามารถนำสัญญาณนี้ไปใช้ในการรีเฟรชหน่วยความจำแบบไดนามิคที่อยู่ในตัวมันได้

Refresh (O) (เฉพาะรุ่นAT) (Memory Refresh) มีหน้าที่เหมือนกับขาสัญญาณ DACK0 ในรุ่น XT คือ ใช้แสดงขบวนการรีเฟรชหน่วยความจำ เพราะว่าในรุ่น AT จะมีวงจรที่ใช้ในการรีเฟรช หน่วยความจำโดยตรงอยู่แล้ว ดังนั้นจึงไม่จำเป็นต้องใช้ขาสัญญาณ DRQ0 และ DACK0

-Master (I) (เฉพาะรุ่นAT) (Master) ขาสัญญาณนี้จะใช้ร่วมกับ DMA Request ในการเข้าควบคุมระบบบัสในขบวนการ DMA โดยที่ตัว DMAคอนโทรลเลอร์จะส่งสัญญาณ DMA Request แล้วรอจนกระทั่งได้รับการตอบสนองโดยสัญญาณ DACK เกิดการแอคทีฟขึ้น แล้วจึงจะส่งสัญญาณนี้ให้กับ CPU จะทำให้แอดเดรสบัส, ดาต้าบัส และคอนโทรลบัสเข้าสู่สถานะไตรสเตต หรือ ไฮอิมพิแดนซ์ หลังจากนั้นตัว DMA คอนโทรลเลอร์จะต้องรออีกหนึ่งคาบสัญญาณคล็อกก่อนที่จะเข้าควบคุมดาต้าบัสต่าง ๆ และจะต้องรออีก 2 ไซเคิล

ก่อนที่จะทำการอ่านหรือเขียนข้อมูล ช่วงเวลาที่สัญญาณนี้แอคทีฟไม่ควรเกิน 15 ไมโครวินาที มิฉะนั้น ข้อมูลภายในหน่วยความจำจะสูญหายไปได้ เนื่องจากขาดสัญญาณรีเฟรชหน่วยความจำ

T/C (O) (Terminal Count) เป็นขาสัญญาณที่บอกอุปกรณ์ I/O ที่ทำDMA ให้ทราบว่าจำนวนข้อมูลที่รับส่งในขบวนการ DMA นี้ครบจำนวนแล้ว โดยจะส่งสัญญาณนี้เป็นพัลส์ให้กับอุปกรณ์ I/O

สัญญาณควบคุมต่างๆ

-MEMR (*I/O) (Memory Read) (สำหรับรุ่น AT คือขาสัญญาณ -SMEMR (System Memory Read)) ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำส่งข้อมูลออกมาที่ดาต้าบัสแต่สำหรับ AT สัญญาณ-SMEMR จะแอคทีฟเมื่อเกิดการอ่านข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 เมกกะไบต์ แรกเท่านั้น

MEMR (O) (เฉพาะรุ่น AT) (Memory Read) ขาสัญญาณนี้ไม่ใช่สัญญาณเดียวกันกับสัญญาณ -MEMR ใน XT มันจะแอคทีฟก็ในทุก ๆ ขบวนการอ่านข้อมูลที่เกิดขึ้นไม่ว่าอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์แรกหรือไม่

-MEMW (*I/O) (Memory Write) (สำหรับรุ่น AT คือขาสัญญาณ -SMEMW (System Memory Write)) ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำเก็บข้อมูลจากดาต้าเบสแต่สำหรับ AT สัญญาณ -SMEMW จะแอคทีฟ เมื่อเกิดการเก็บข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 เมกกะไบต์แรกเท่านั้น

MEMW (O) (เฉพาะรุ่น AT) (Memory Write) ขาสัญญาณนี้ไม่ใช่สัญญาณเดียวกันกับสัญญาณ -MEMW ใน XT มันจะแอคทีฟก็ในทุก ๆ ขบวนการเก็บข้อมูลที่เกิดขึ้นไม่ว่าอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์แรกหรือไม่

-IOR (*I/O) (I/O Read) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ทำการส่งข้อมูลลงมาที่ดาต้าบัส

-IOW (*I/O) (I/O Write) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ทำการเก็บข้อมูลจากคาต้าบัสเข้าไป

RESET DRV (O) (Reset Driver) เป็นขาสัญญาณที่แอคทีฟตอนช่วงที่เราเริ่มจ่ายไฟให้กับระบบเพื่อใช้ในการรีเซต CPU และอุปกรณ์ต่างๆในระบบคอมพิวเตอร์ รวมทั้งอุปกรณ์ I/O ที่ต่ออยู่ด้วย

-MEM CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่าต้องการรับส่งข้อมูลกับหน่วยความจำทีละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้การรับส่งข้อมูลก็จะทำเหมือน XT คือ ทำการรับส่งข้อมูลทีละ 8 บิต สองครั้งเพื่อให้ได้ข้อมูลขนาด 16 บิต

-I/O CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่าต้องการรับส่งข้อมูลกับอุปกรณ์ I/O ทีละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลก็จะทำเหมือน XT คือ ทำการรับส่งข้อมูลทีละ 8 บิต สองครั้งเพื่อให้ได้ข้อมูลขนาด 16 บิต

สัญญาณที่ใช้สร้างสถานะรอ (Wait States)

I/O CH RDY (I) (I/O Channel Ready) ขาสัญญาณนี้จะถูกทำให้แอคทีฟโดยอุปกรณ์ I/O หรือหน่วยความจำที่ไม่สามารถทำงานได้ทันกับระบบ ดังนั้นจะต้องทำการหน่วงระบบให้ทำงานช้าลงด้วยการเพิ่ม สถานะรอ โดยการทำให้สัญญาณแอคทีฟในช่วงเวลาที่ I/O ได้รับสัญญาณจากการ ดีโค้ดแอดเดรส, สัญญาณ -MEMR , -MRMW,สัญญาณ -IOR, สัญญาณ -IOW

0WS (I) (เฉพาะรุ่น AT) (Zero Wait State) การแอคทีฟของขาสัญญาณนี้จะบังคับไม่ให้เกิดการสร้าง สถานะรอ (Wait States) โดยอัตโนมัตินั่นคือการที่จะเกิดสถานะรอขึ้นได้นั้นต้องขึ้นอยู่กับสัญญาณนี้ เช่น การทำงานในขบวนการอ่านเขียนข้อมูลขนาด16 บิต โดยไม่ใช้ สถานะรอ ทำได้โดยการสร้างสัญญาณOWS จากสัญญาณการดีโค้ดแอดเดรสและสัญญาณที่ใช้ในการอ่านหรือเขียน หรือการลด สถานะรอในขบวนการอ่านเขียนข้อมูลขนาด 8 บิต ให้เหลือเพียง 2 Wait States ทำได้โดยให้สัญญาณ OWS แอคทีฟหลังจากสัญญาณอ่าน

หรือเขียนไปแล้ว คล็อกโดยปกติการขับสัญญาณนี้ควรใช้เกตที่มีเอาท์พุตเป็นแบบ โอเพ่นคอลเล็กเตอร์ (Open Collector) ที่ทนกระแสได้ 20 มิลลิแอมป์ (Sinking Current)

สัญญาณนาฬิกา

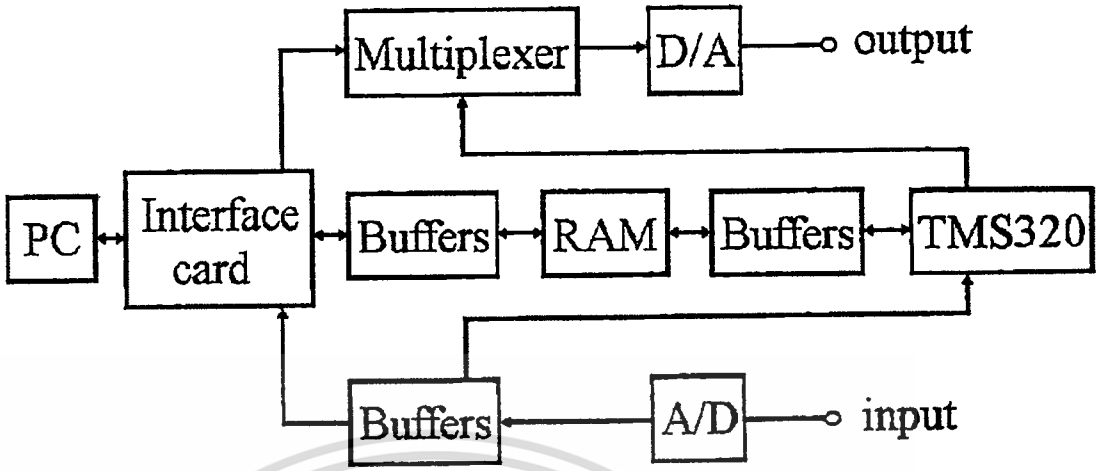
CLK (O) (System Clock) สำหรับ XT ขาสัญญาณนี้จะมีความถี่ประมาณ 4.77 MHz หรืออาจจะสูงกว่านี้ก็ได้สำหรับรุ่นใหม่ ๆ และสำหรับ AT จะมีความถี่ประมาณ 6 MHz หรือในรุ่นใหม่ ๆ อาจจะมีความถี่สูงถึงประมาณ 15 MHzโดยปกติขาสัญญาณนี้มีดิวตี้ไซเคิล 50 % สำหรับ CPU เบอร์ 80286 ตัวกำเนิดสัญญาณนาฬิกาที่ป้อนให้จะมีความถี่เป็น 2 เท่า ของความถี่ CPU ทำงาน แต่ขาสัญญาณนี้ก็จะยังคงมีความถี่เป็น 2 เท่า ของความถี่ที่ CPU ทำงานแต่ขาสัญญาณนี้ก็จะยังคงมีความถี่เท่ากับความถี่ที่ CPU ทำงานอยู่เสมอ

OSC (O) (Oscillator) เป็นขาสัญญาณที่มีความถี่สูงคือ 14.3181 MHz ความถี่ของสัญญาณนี้จะคงที่เสมอและจะไม่ซิงโครนัสกับ สัญญาณอื่น ๆ ในระบบ ดังนั้นจึงไม่ควรนำสัญญาณนี้ไปใช้เป็นสัญญาณคล็อกของอุปกรณ์ I/O ที่ต่ออยู่กับระบบ

6.2. บอร์ดประมวลผลสัญญาณดิจิตอล (DSP Single Board)

ตัว DSP Board สามารถทำงานร่วมกับ PC โดยต่อกับ การ์ดอินเตอร์เฟส หรือในกรณีที่ PC โหลดโปรแกรมลงบนบอร์ดแล้วสามารถแยกออกมาจาก PC ได้ ส่วนประกอบต่างๆบน Board แบ่งเป็น 4 ส่วนใหญ่ๆได้ดังนี้

1. ชิพประมวลผลสัญญาณดิจิตอล (DSP Chip) ใช้เบอร์ TMS320C30
2. Memory และ Buffers PC สามารถส่งโปรแกรมหรือข้อมูลให้ TMS320C30 ทำงาน(Run)
3. วงจรแปลงสัญญาณ ดิจิตอล เป็นอนาล็อก (Digital to Analog)
4. วงจรแปลงสัญญาณ อนาล็อก เป็นดิจิตอล(Analog to Digital)



รูปที่ 6-3 แสดงบล็อกไดอะแกรมของระบบ

6.2.1 ชิพประมวลผลสัญญาณดิจิตอล (DSP Chip)

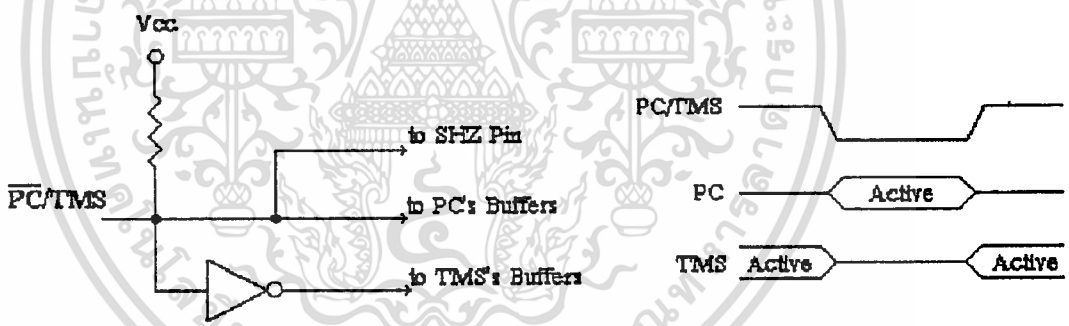
TMS320C30 เป็นชิพที่ถูกผลิตขึ้นมาสำหรับ การประมวลผลสัญญาณเชิงเลข (Digital Signal Processing) เป็น CPU ขนาด 32 บิท มีชุดคำสั่งที่สนับสนุนการทำงานทางด้านDSP และมีความเร็วสูง (33 MFLOPS ,16.7 MIPS) ระบบบัสที่ใช้ในการติดต่ออุปกรณ์ภายนอกถูกแยกออกมา (Expansion Bus) ซึ่งสะดวกในการใช้งาน , หน่วยความจำที่จะทำการติดต่อต้องมีขนาด 32 บิทเวิร์ด (32 Bit word) รายละเอียดเกี่ยวกับตัวชิพได้ อธิบายในบทที่ 6 จากในรูปที่ 7-3 จะเห็นว่า TMS320C30 สามารถติดกับหน่วยความจำ(External RAM) โดยผ่านบัฟเฟอร์ (Buffers) และติดต่อกับอุปกรณ์อินพุท ,เอาท์พุท (I/O Device) ซึ่งมีวงจรแปลงสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอล (Analog to Digital) และวงจรแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอนาล็อก (Digital to Analog)

6.2.2 หน่วยความจำและ บัฟเฟอร์ (Share Memory & Buffers)

การติดต่อหน่วยความจำโดย PC และ TMS

หน่วยความจำบน DSPบอร์ดที่ได้ออกแบบและสร้างขึ้นมานี้ สามารถที่จะได้รับการติดต่อจาก PC และชิพDSP การควบคุมหน่วยความจำนี้จะถูกควบคุมโดย PC ที่สัญญาณ ¯PC/TMS เป็นตัวเลือก ซึ่งสัญญาณนี้จะเป็นสัญญาณที่ควบคุมการทำงานของบัฟเฟอร์ ว่าจะอนุญาตให้บัฟเฟอร์ทางด้านใดทำงาน คือถ้าสัญญาณนี้เป็น “0” บัฟเฟอร์ทางด้านที่ติดต่อกับ PC ก็จะได้รับอนุญาตให้ทำงานได้ (Enable) บัฟเฟอร์ทางด้านที่ติดต่อกับชิพ DSP จะไม่ได้รับอนุญาตให้ทำงาน(Disable) ในทางกลับกันถ้าสัญญาณ ¯PC/TMS เป็น “1”

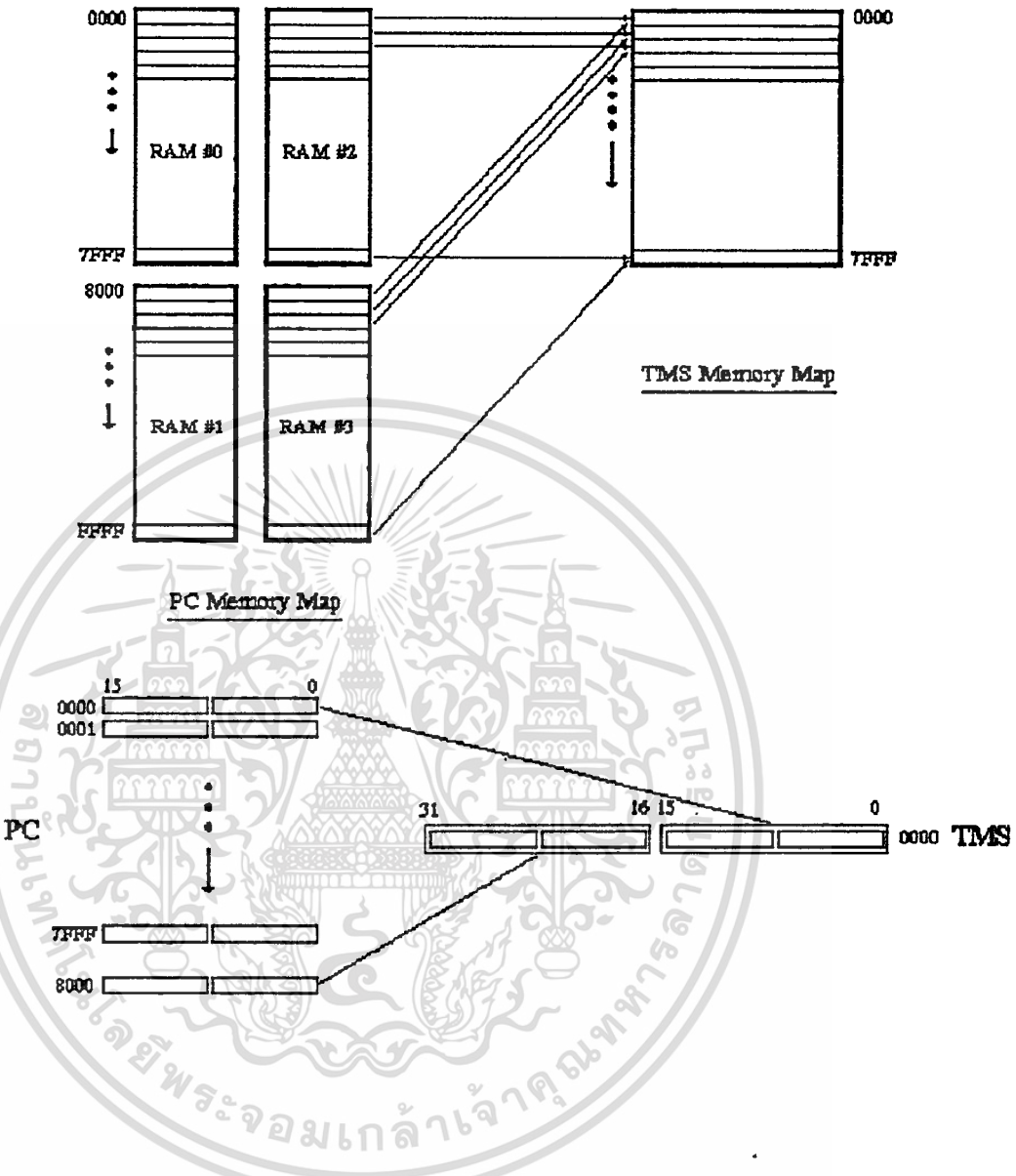
บัฟเฟอร์ทางด้านที่ติดต่อกับชิพ DSP ก็จะได้รับอนุญาตให้ทำงานได้ ส่วนบัฟเฟอร์ทางด้านที่ติดต่อกับ PC จะไม่ได้รับอนุญาตให้ทำงาน และสัญญาณ ¯PC/TMS นี้ยังใช้ในการควบคุมการทำงานของTMS320C30 ด้วยคือ สัญญาณนี้จะถูกต่อเข้ากับขา ¯SHZ (Shut down) ดังรูปที่ 7-4 ที่ขา¯SHZ ของTMS320C30 นี้ถ้าเป็นลอจิก "0" TMS320C30 จะหยุดทำงานทั้งหมด และทุกขาของ TMS320C30 จะอยู่ในสภาวะขาลอย (High Imperdance) ดังนั้นเราจะเห็นว่า เมื่อ PCกำลังติดต่ออยู่กับหน่วยความจำ TMS320C30 จะหยุดทำงานและ จะเริ่มทำงานใหม่เมื่อสัญญาณ ¯PC/TMS เป็น"1" ซึ่งนั่นก็คือที่ขาสัญญาณ ¯SHZ ได้รับลอจิก "1" และคืนหน่วยความจำกลับมาให้กับ TMS320C30 ในกรณีที่ DSP บอร์ด ถูกปลดสายที่ต่ออยู่กับการ์ดอินเตอร์เฟส สัญญาณ¯PC/TMS จะเป็น"1" เพราะมีตัวต้านทานต่อสัญญาณนี้อยู่กับ Vcc หน่วยความจำจะถูกต่อเข้ากับ TMS320C30 หมายความว่าหน่วยความจำจะเป็นของ TMS320C30 เมื่อถูกปลดสายที่ต่ออยู่กับการ์ดอินเตอร์เฟส



รูปที่ 6-4 สัญญาณควบคุมทิศทางการใช้หน่วยความจำ

การเชื่อมต่อหน่วยความจำ

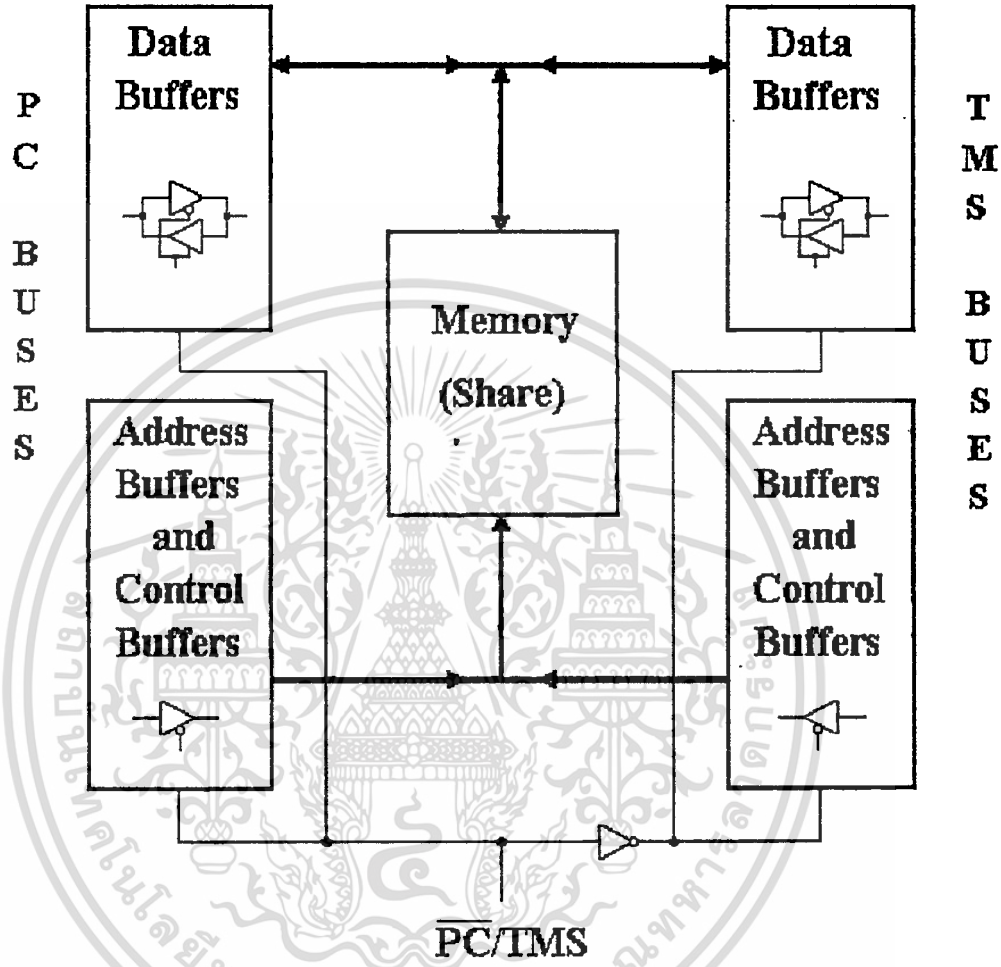
ในการเข้าถึงหน่วยความจำของ TMS320C30 จะทำครั้งละ 32 บิท (แอดเดรสละ 32 บิท) แต่ทางด้าน PC ดูจากสลอต (Slot) มีบัสข้อมูลเพียง 16 บิท ดังนั้นในการที่ PC จะเขียนข้อมูลหรือ ดาวน์โหลด (¨Down Load) โปรแกรมเพื่อใช้กับ TMS320C30 นั้นต้องมีการเขียนข้อมูลถึง 2 ครั้ง สำหรับ 1 แอดเดรส ของ TMS320C30



รูปที่ 6-5 แสดงการจัดแอดเดรสหน่วยความจำให้กับ PC และ TMS320C30

จากรูปที่ 6-5 จะเห็นว่าเป็นหน่วยความจำขนาด 32 กิโลไบต์ 4 ตัว ถ้าทางด้านบัฟเฟอร์ ทางด้านที่ต่ออยู่กับ PC ได้รับอนุญาตให้ทำงาน(Active) จะเป็นเสมือนการต่อเชื่อมหน่วยความจำครั้งละ 16 บิท ขนาดหน่วยความจำเท่ากับ 64K * 16 บิท แต่ถ้าทางด้านบัฟเฟอร์ของ TMS320C30 ได้รับอนุญาตให้ทำงาน จะเป็นเสมือนกับการต่อเชื่อมหน่วยความจำครั้งละ 32 บิท ขนาดหน่วยความจำเท่ากับ 32 K * 32 บิท การ Map Memory หน่วยความจำ 1 ตำแหน่งของทาง TMS จะเท่ากับหน่วยความจำ 2 ตำแหน่งของทาง PC ตัวอย่างเช่นที่ตำแหน่ง 0000h ของ TMS320C30 เป็นตำแหน่งของทางด้าน PC คือ 0000h (D_{15}-D_0)

และตำแหน่ง 8000h (D_{31}-D_{16}) ลักษณะการต่อหน่วยความจำ ดูได้จากบล็อกไดอะแกรม รูปที่ 6-6



รูปที่ 6-6 บล็อกไดอะแกรมแสดงการเชื่อมต่อหน่วยความจำ

การเข้าถึงหน่วยความจำบน DSP บอร์ด

ในการรับและการส่งข้อมูลระหว่าง PC กับหน่วยความจำบน DSP บอร์ด ข้อมูลจากหน่วยความจำจะถูกส่งผ่านพอร์ตอินพุต/เอาท์พุต จากในวงจรรูปที่ 6-13 คือ U4 , U5 ,U6 ,U7 ซึ่งU4 และ U5 ใช้ ไอซี เบอร์ 74ALS373 เป็นเอาท์พุตพอร์ต เป็นตัวส่งข้อมูล และ U6 ,U7 ใช้ ไอซี เบอร์ 74ALS3244 เป็นอินพุทพอร์ต ซึ่งทั้งสองพอร์ตนี้จะใช้หมายเลขพอร์ตเดียวกันหรือเสมือนเป็นอินพุต/เอาท์พุต เพียงพอร์ทเดียว (หมายเลข **FO) ในการแยกแยะว่าจะให้อินพุตหรือเอาท์พุทพอร์ตทำงานนั้นใช้ ¯IOR และ ¯IOW ที่ต่อมาจากการ์ดอิน

เตอร์เฟส สัญญาณ ¯IOW จะแอคทีฟ (Active) คือเป็น "0" เมื่อมีการนำข้อมูลซึ่งจะนำสัญญาณนี้ไปควบคุมเอาท์พุตพอร์ท จากรูป 7-12 การควบคุมที่ขาเกต (Gate) ของไอซีเบอร์ 74ALS373 จะใช้สัญญาณ ¯CS4 ออร์ (OR) กับ ¯IOW แล้ว กลับสัญญาณ (NOT) แล้วนำสัญญาณไปควบคุมขาที่เกตของ U4 และ U5 การควบคุมอินพุทพอร์ทก็จะเป็นลักษณะเดียวกันคือใช้สัญญาณ ¯CS4 ออร์ กับ ¯IOW แล้วนำไปควบคุมที่ขา G1 และ G2 ของ U6 และ U7 นี้ เอาท์พุตพอร์ทของ อินพุต/เอาท์พุตพอร์ท (ขนาด 16 บิท) ถูกนำไปต่อเข้ากับ ไอซีเบอร์ 74ALS245

การที่ต้องต่อ ไอซี เบอร์ 74ALS245 ระหว่างหน่วยความจำและ I/O พอร์ท เพื่อป้องกันไม่ให้ขาข้อมูล D_0- D_7 ของ RAM 0 ต่อเข้ากับ D_0-D_7 ของ RAM 2 และ ขาข้อมูล D_0- D_7 ของRAM 1 ต่อเข้ากับขาข้อมูล D_0- D_7 ของRAM 3 ถ้าเรายอมปล่อยให้บัสข้อมูลจากเอาท์พุตพอร์ท ซึ่งมีขนาด 16 บิท ต่อเข้าโดยตรงนั้น เราจะไม่สามารถต่อบัสข้อมูลของ TMS320C30 ได้เพราะเราจะได้ หน่วยความจำแบบที่มีการเข้าถึงข้อมูลขนาด 16 บิท แต่ TMS320C30 จะต้องทำการเข้าถึงข้อมูล (access) ครั้งละ 32 บิท

ในการส่งสัญญาณอ้างอิงตำแหน่ง (Address) จาก PC จะส่งผ่านการ์ดอินเตอร์เฟส ไปให้กับพอร์ทหมายเลข **E4 และ **E8 (** คือค่า Dip Switch) ซึ่งที่พอร์ตหมายเลข **E4 จะเป็นสมือน แอดเดรส A_0 - A_{15} และที่พอร์ท **E8 จะเป็นเสมือนแอดเดรส A_{16} - A_{23} พอร์ททั้งสองนี้ถูกควบคุมสัญญาณเอาท์พุตของตัวมัน (OC) โดยสัญญาณ ¯PC/TMS เมื่อสัญญาณนี้เป็น "0" สัญญาณข้อมูลแอดเดรสจะถูกเชื่อมต่อเข้ากับ ขาสัญญาณแอดเดรสของไอซีหน่วยความจำ เบอร์ CY7B199 แต่แอดเดรสบัฟเฟอร์ ที่ต่ออยู่ทางด้าน TMS320C30 จะอยู่ในสภาวะขาลอย ถ้าสัญญาณ ¯PC/TMS เป็นลอจิก "1" เอาท์พุตของพอร์ท แอดเดรสทั้งสองพอร์ท ที่ต่ออยู่ทางด้าน PC จะอยู่ในสภาวะขาลอย ส่วนขาแอดเดรสทางด้าน TMS320C30 จะถูกต่อเข้ากับขาของแอดเดรสของไอซีหน่วยความจำ การทำเช่นนี้จะสังเกตว่าเป็นการป้องกันเพื่อไม่ให้มีการต่อเชื่อมขาแอดเดรสพร้อมกันทั้งสองด้านในเวลาเดียวกัน

การอ่านข้อมูล

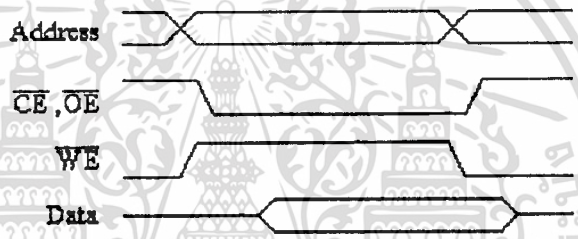
ในการที่ PC จะอ่านข้อมูลจากหน่วยความจำบน DSP บอร์ดนั้น ต้องมีการส่งข้อมูลแอดเดรสไปที่แอดเดรสพอร์ต จากนั้นก็ให้อ่านข้อมูลกลับมาโดยผลจากการอ่านข้อมูลนี้จะเป็นการสร้างไทม์มิ่งดังในรูปที่ 6-7 (Timing Diagram) สำหรับการอ่านข้อมูล จากในรูปสัญญาณ ¯CS และ ¯OE ได้จากการดีโค้ดแอดเดรสโดยใช้ไอซีเบอร์ 74ALS138 เอาท์

พุตที่ได้จากการดีโค้ดนี้ ก่อนที่จะต่อเข้ากับ ¯CS และ ¯OE จะต้องผ่านบัฟเฟอร์ก่อนเพื่อป้องกันไม่ให้ต่อเข้ากับขา ¯CS และ ¯OE ของไอซี หน่วยความจำโดยตรง

ตัวอย่างการอ่านข้อมูลโดยใช้ภาษาซี

```
OUTPORT (ADDR_H , 00H);
OUTPORT (ADDR_L , 1234H);
DATA = INPORT ( MEM_PORT );
```

จากตัวอย่างเป็นการอ่านข้อมูลจากตำแหน่งที่ 01234h



รูปที่ 6-7 ไทม์มิ่งไดอะแกรมของการอ่านข้อมูล

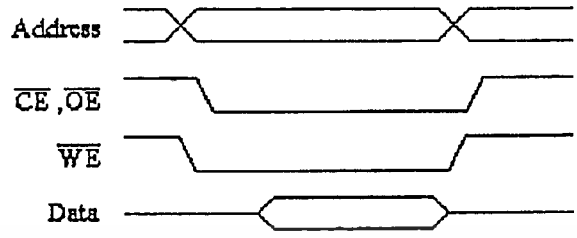
การเขียนข้อมูล

ในการที่ PC จะเขียนข้อมูลจากหน่วยความจำบน DSP บอร์ด นั้นต้องมีการส่งข้อมูลแอดเดรสไปที่แอดเดรสพอร์ท จากนั้น ก็ให้เขียนข้อมูลออกไปโดยผลของการเขียนข้อมูลนี้จะเป็นการสร้างไทม์มิ่ง สำหรับการเขียนข้อมูลให้กับไอซีหน่วยความจำ จากในรูปที่ 6-8 สัญญาณ ¯CS และ ¯OE ได้จากการดีโค้ดแอดเดรสโดยใช้ไอซีเบอร์ 74ALS138 เอาท์พุตที่ได้จากการดีโค้ดนี้ ก่อนที่จะต่อเข้ากับ ¯CS และ ¯OE จะต้องผ่านบัฟเฟอร์ก่อนเพื่อป้องกันไม่ให้ต่อเข้ากับขา ¯CS และ ¯OE ของไอซีหน่วยความจำโดยตรง

ตัวอย่างการเขียนข้อมูลโดยใช้ภาษาซี

```
OUTPORT (ADDR_H , 00H);
OUTPORT (ADDR_L , 1234H);
OUTPORT ( MEM_PORT , DATA );
```

จากตัวอย่างเป็นการเขียนข้อมูลลงไปที่ตำแหน่ง 01234h



รูปที่ 6-8 ไทม์มิ่งไดอะแกรมของการเขียนข้อมูล

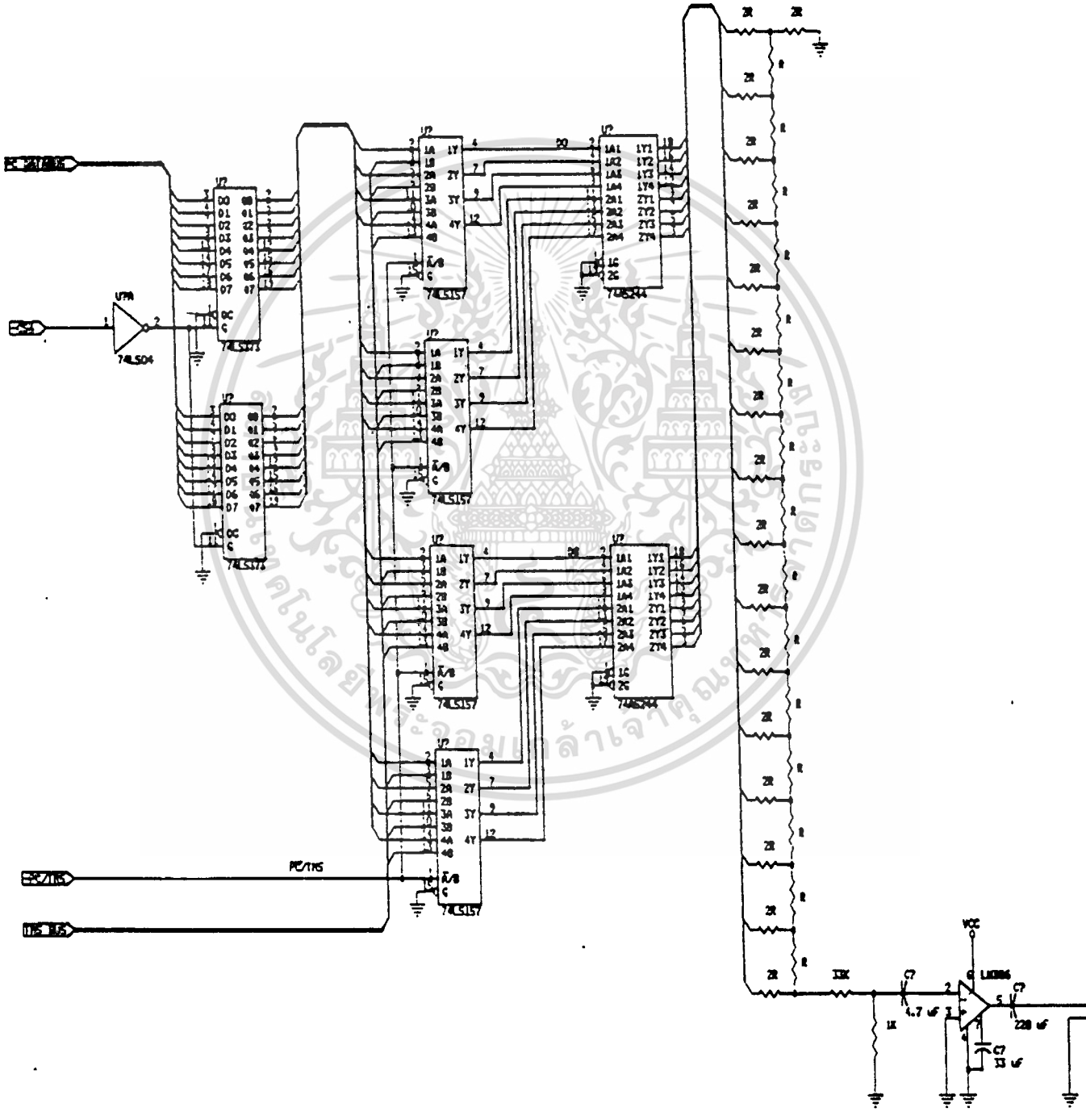
6.3 การแปลงสัญญาณดิจิตอล เป็นอนาล็อก

ในส่วนการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก นี้สามารถรับข้อมูลจากทางด้าน PC หรือจาก TMS320C30 ได้โดยมีวงจรเลือกข้อมูล (Multiplex หรือ Data selector) การแปลงสัญญาณ จะใช้วงจร R-2R แลดเดอร์ (R-2R Ladder) ในการแปลงสัญญาณดิจิตอลขนาด 16 บิทไปเป็นสัญญาณอนาล็อก

การรับข้อมูลจาก PC จากในรูปที่ 6-10 จะต้องส่งข้อมูลมาค้างไว้ (latch) ที่เอาท์พุตพอร์ท หมายเลข **F0 พอร์ทนี้ถูกควาบคุมโดยสัญญาณ ¯CS4 นำมากลับลอจิก (NOT) แล้วนำไปควบคุมการทำงานเอาท์พุตพอร์ท ซึ่งใช้ไอซีเบอร์ 74ALS373 จำนวนสองตัว เอาท์พุตของพอร์ท จะถูกส่งไปที่วงจรเลือกข้อมูล

วงจรเลือกข้อมูลจะเป็นตัวเลือกว่าจะใช้ข้อมูลจาก PC หรือ TMS320C30 โดยใช้ไอซีเบอร์ 74LS157 จำนวน 4 ตัวเป็นตัวเลือกข้อมูล คือสัญญาณ ¯PC/TMS เป็นตัวเลือกว่าจะใช้ข้อมูลจากแหล่งใด ซึ่งถูกส่งมาจาก PC ถ้าสัญญาณ ¯PC/TMS เป็น"0" ตัวเลือกจะใช้ข้อมูลที่ส่งมาจาก PC ถ้า เป็น"1" จะเป็นการเลือกใช้ข้อมูลจาก TMS320C30 ในกรณีที่ DSP บอร์ด ถูกปลดสายที่ต่ออยู่กับการ์ดอินเตอร์เฟส สัญญาณ ¯PC/TMS จะเป็น"1" ทันที เพราะมีตัวต้านทานต่อสัญญาณนี้อยู่กับ Vcc ดังนั้นจึงเป็นการเลือกใช้ข้อมูลจาก TMS320C30 ซึ่งหมายความว่าวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อกจะเป็นของ TMS320C30 ทันทีเมื่อปลดสายที่ต่ออยู่กับการ์ดอินเตอร์เฟส

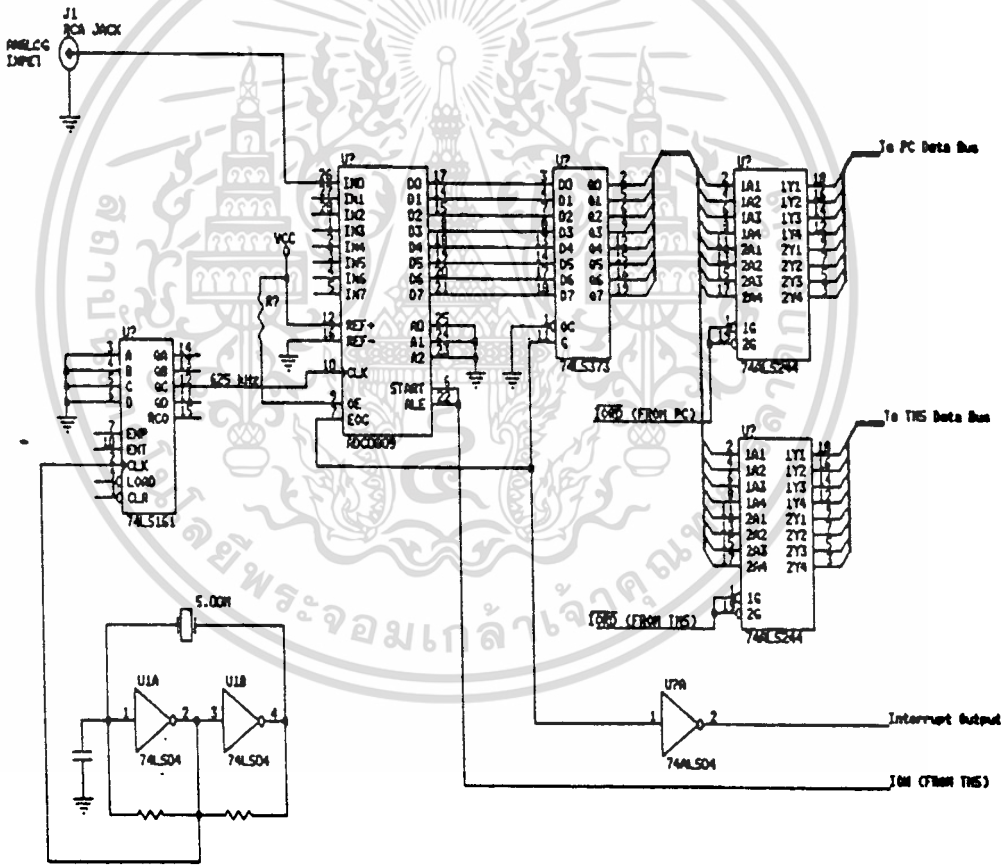
หลังจากที่ผ่านวงจรเลือกข้อมูลแล้วสัญญาณที่ได้ทั้ง 16 เส้นจะถูกนำมาต่อเข้ากับไอซีเบอร์ 74ALS244 เพื่อเป็นการขยายสัญญาณ แล้วนำไปต่อเข้ากับวงจร R-2R แลดเดอร์



รูปที่ 6-10 แสดงวงจรของการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อก

6.4 วงจรแปลงสัญญาณ อนาล็อก เป็นดิจิตอล

วงจรในส่วนนี้จะสามารถส่งข้อมูลให้ได้ทั้ง PC และ TMS320C30 ดังนั้น วงจรในส่วนนี้จึงไม่จำเป็นต้องมีสัญญาณ PC/TMS เข้ามาควบคุมเหมือนกับวงจรในส่วนอื่น ๆ วงจรที่ได้ออกแบบไว้ในส่วนนี้ การแปลงสัญญาณโดยใช้ไอซีเบอร์ ADC0809 ซึ่งเป็นไอซีแปลงสัญญาณอนาล็อกเป็นดิจิตอลขนาด 8 บิท สาเหตุที่ไม่ใช้ไอซีที่มีขนาด 16 บิท นั้นก็เพราะว่าไอซีที่มีขนาด 16 บิท ก็เพราะว่าไอซีแปลงสัญญาณอนาล็อกเป็นดิจิตอลขนาด 16 บิท มีราคาแพงมาก และอาจต้องสั่งซื้อจากต่างประเทศ



รูปที่ 6-11 แสดงวงจรแปลงสัญญาณอนาล็อก เป็นดิจิตอล

จากรูป ไอซีเบอร์ ADC0809 จะต้องมีสัญญาณนาฬิกา เพื่อใช้ในกระบวนการแปลงสัญญาณอนาล็อกเป็นดิจิตอล สำหรับสัญญาณนาฬิกาที่จะป้อนให้กับไอซีเบอร์ ADC0809 นี้ จะต้องมีความถี่ไม่เกิน 640KHz จากวงจรใช้คริสตัลผลิตความถี่ 5 MHz แล้วนำไปเข้าวงจรหาร 8 โดย ไอซีเบอร์ 74LS161 จะได้ความถี่เท่ากับ 625 KHz สำหรับไอซีเบอร์ ADC0809 นี้ จะใช้จำนวนสัญญาณนาฬิกาในการแปลงสัญญาณไม่เกิน 64 คล็อก ดังนั้นในการสุ่ม (Sampling) สัญญาณ ทำได้สูงสุดคือ 64*625 KHz = 9.765 Khz การควบคุมการทำงานจะถูกควบคุมโดย TMS320C30 ในการสั่งให้เริ่มแปลงสัญญาณใช้สัญญาณ IOW ซึ่งได้จากการดีโค้ดแอดเดรสและสัญญาณ IOSTRB ของ TMS320C30 เมื่อ ADC0809 แปลงสัญญาณเสร็จแล้วที่ขาสัญญาณ EOC ของ TMS320C30 จะเปลี่ยนจากลอจิก "0" ไปเป็น ลอจิก "1" สัญญาณนี้สามารถนำไปบอก TMS320C30 (โดยการขัดจังหวะ) เพื่อให้ TMS320C30 รับข้อมูลที่ได้แปลงเสร็จแล้วเข้าไป หรือเพื่อให้มีการเริ่มแปลงสัญญาณใหม่อีก

6.5 โปรแกรมสนับสนุน

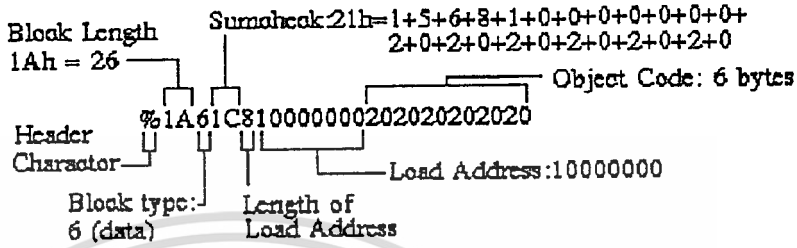
โปรแกรมสนับสนุน การใช้งาน DSPซิงเกิลบอร์ด นอกจากโปรแกรมการอ่าน, การเขียน ข้อมูลลงหน่วยความจำลงบน DSPซิงเกิลบอร์ด หรือการควบคุม ซึ่งได้กล่าวมาแล้ว ในหัวข้อก่อนๆ โครงงานนี้ยังมีการเขียนโปรแกรมที่ ใช้สนับสนุน การทำงานขึ้นมาอีก สองโปรแกรม คือ โปรแกรมดาวน์โหลด (Down load) และโปรแกรมขอดูข้อมูลในหน่วยความจำ (Dump)

6.5.1 โปรแกรมดาวน์โหลด

โปรแกรมดาวน์โหลดนี้ เป็นตัวทำหน้าที่นำโปรแกรม ที่ได้หลังจากการคอมไพล์ โปรแกรม ที่ได้เลือกรูปแบบของเอาท์พุตอยู่ในรูปแบบของ "เอ็กซ์เทน เทคโทรนิค เฮกซ ออบเจ็ก" (Extended Tektronix Hex Object Format) เราได้ไฟล์เอาท์พุตออกมา 4 ไฟล์ ซึ่งมีนามสกุล เป็น ".X0, .X1, .X2, .X3" โปรแกรมดาวน์โหลดนี้จะทำการอ่านข้อมูลจากไฟล์ทั้งสี่ แล้วปรับให้อยู่ในรูปแบบที่สามารถนำไปให้ TMS320C30 สามารถรันได้ จากนั้นก็นำข้อมูลที่ได้ไปเขียนลงในหน่วยความจำ บน DSPบอร์ด

ลักษณะของ "เอ็กซ์เทน เทคโทรนิค เฮกซ ออบเจ็ก" มีทั้งหมด 4 ไฟล์ เพราะว่าโค้ดคำสั่งของ TMS320C30 มีขนาด 32 บิท หรือ 4 ไบท์ สาเหตุที่ไม่รวมเข้าเป็นไฟล์เดียว เพื่อการทำอย่างอื่น เช่น การเขียนข้อมูลลงหน่วยความจำแบบรอม (Burn ROM) ไฟล์ ".X0" เก็บข้อมูลไบท์ที่มีนัยสำคัญต่ำสุด (D_0 - D_7) เรียงกันตามลำดับจนถึง ไฟล์ ".X3"

เก็บข้อมูลไบท์ที่มีนัยสำคัญสูงสุด รูปแบบการเก็บข้อมูลของ "เอ็กซ์เทน เทคโทรนิค เฮกซ ออบเจ็ก" ดูได้จากรูปตัวอย่าง



รูปที่ 6-12 รูปแบบของ เอ็กซ์เทน เทคโทรนิค เฮกซ ออบเจ็ก

การทำงานของโปรแกรมดาวน์โหลดจะเริ่มจากการ เปิดไฟล์ทั้งสี่พร้อม ๆ กันแต่ละไฟล์ จะถูกอ่านขึ้นมาครั้งละ 1 ไบท์ พร้อม ๆ กัน แล้วทำการวิเคราะห์ตามรูปแบบของ"เอ็กซ์เทน เทคโทรนิค เฮกซ ออ็บเจ็ก" เมื่อถึงในส่วนที่เป็นออบโค้ด ก็จะทำการจัดเรียงตามลำดับให้ถูกต้อง แล้วนำข้อมูลที่ได้ไปเขียนลงในหน่วยความจำบน DSPบอร์ด จนกระทั่งจบไฟล์ จากลักษณะรูปแบบของไฟล์ และการอ่านขึ้นมาพร้อมกันนี้สามารถที่จะทำการตรวจจับความผิดผลาดของไฟล์ได้ง่าย โดยหาจากความสัมพันธ์ของแต่ละไฟล์ เช่น ตัวแรกของทุก ๆ ไฟล์ ควรเป็นอักษรเริ่มต้น (Header Charcter) เหมือนกันหมด, หรือในกลุ่มข้อมูลถ้ามีเลขข้อมูลที่ไม่ได้อยู่ในย่านหรือขอบเขตที่กำหนด แสดงว่าในไฟล์เกิดความผิดปรกติ รายละเอียดตัวโปรแกรมสามารถดูได้จากคู่มือการใช้บอร์ดตัวนี้

6.5.2 โปรแกรมขอดูค่าข้อมูลในหน่วยความจำ

โปรแกรมขอดูค่าข้อมูลในหน่วยความจำ หรือเรียกว่าดัมพ์ (Dump) เป็นการอ่านค่าข้อมูลขึ้นมาจากหน่วยความจำของ DSP บอร์ด ในการทดลองโปรแกรมต่าง ๆ เรามักมีการขอดูค่าในหน่วยความจำว่าเป็นอย่างไร ถูกต้องหรือไม่ ลักษณะการแสดงผลแต่ละข้อมูล จะเป็นเสมือนกับ 1 แอดเดรสของ TMS320C30 ซึ่งมีขนาด 32 บิทดังนั้นในโปรแกรมดัมพ์ 1 แอดเดรสของ TMS320C30 จึงต้องมีการอ่านข้อมูลขึ้นมาถึง 2 ครั้งดังได้อธิบายแล้วในหัวข้อก่อนหน้านี้

เอกสารอ้างอิง

ฟูศักดิ์ ชีวสุวิทย์ "Digital", สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 123 หน้า, 2535

ยืน ภู่วรวรรณ, "เทคโนโลยีฮาร์ดแวร์ IBM PCW, กรุงเทพมหานคร 335 หน้า, 2533

วัลลภ สุรกำพลธร, "การประมวลผลสัญญาณเชิงเลข การกรองและการแปลง", สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 301 หน้า, 2533

Douglas O' Shaughnessy, "Speech Communication human and machine", Addsion-wesley, 425p., 1987

L.R. Rabiner/R.W. Schafer, "Digital Processing of Speech Signals", Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 458p., 1972.

Panos Papamichalis, "Digital Signal Processing Applications with the TMS320 Family", Prentice-Hall, Englewood Cliffs, New Jersey, 561p., 1990.

Paul M. Embree/Bruce Kimble, "C Language Algorithms for Digital Signal Processing", Prentice-Hall Englewood Cliffs, New Jersey, 456p., 1991.

Sonitech, "SPIRIT-30/ISA Application Note Rev 3.1", Sonitech Interational, Inc, 65p., 1990

Texas Instruments, "TMS320C3X User's Guide", Texas Instruments Incorporated, 1125p., 1992

Texas Instruments, "TMS320 Floating-Point DSP Optimizing C Compiler User's Guide", Texas Instruments Incorporated, 135p., 1991.

Texas Instruments, "TMS320 Floating-Point DSP Asembly Language Tools User's Guide", Texas Instruments Incorporated, 185p., 1991.