

การประยุกต์ใช้งานไมโครโปรเซสเซอร์ (DSP)

ในงานประมวลผลภาพ

(MicroprocessOr Application for Image Processing)



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2536

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

033219

ปริญญาโทสำหรับภาคการศึกษาที่ 2 ปีการศึกษา 2536

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งานไมโครโปรเซสเซอร์ (DSP) ในงานประมวลผลภาพ
(Microprocessor Application for Image Processing)

ผู้จัดทำ

นาย บวร ไชยสุขทักษิณ เลขประจำตัว 33100184

นาย บุญชัย งามวงศ์วัฒนา เลขประจำตัว 33100187

นาย สุรชัย สุวรรณพาณิชย์ เลขประจำตัว 33100447

อาจารย์ที่ปรึกษา

อ. เทอดศักดิ์ ลีวาททอง

(..........)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประยุกต์ใช้งานไมโครโปรเซสเซอร์ (DSP) ในงานประมวลผลภาพ
(Microprocessor Application for Image Processing)

โดย นาย บวร ไชยสุชกัณีน 33100184

นาย บุญชัย งามวงศ์วัฒนา 33100187

นาย สุรชัย ชูพรพาณิชย์ 33100447

อาจารย์ที่ปรึกษา อ. เทอดศักดิ์ ลีวาททอง

บทคัดย่อ

โดยทั่วไปการประมวลผลสัญญาณภาพ (Digital Image Processing) จะกระทำบนเครื่องไมโครคอมพิวเตอร์ซึ่งใช้เวลาในการประมวลผลค่อนข้างมาก ปรึญยานิพนธ์นี้มีจุดประสงค์ที่จะลดข้อเสียดังกล่าว คือใช้ไมโครโปรเซสเซอร์ทางด้านดิจิทัลซิกแนลโปรเซสซิ่ง (TMS320C25) มาประมวลผลแทนซีพียูหลักของเครื่องคอมพิวเตอร์ (IBM PC ตั้งแต่รุ่น AT ขึ้นไป) ซึ่งการส่งผ่านข้อมูลจะใช้หลักการ Map Memory ของ TMS320C25 ไปยังหน่วยความจำที่ว่างบน IBM PC เมื่อจะประมวลผลทางด้านดิจิทัลซิกแนลโปรเซสซิ่ง ซีพียูหลักจะให้ TMS320C25 ประมวลผลแทน และเมื่อประมวลผลเสร็จซีพียูหลักจะนำข้อมูลที่ได้ไปใช้งานต่อไป ซึ่งการทำงานของ TMS320C25 จะถูกควบคุมโดยเครื่องคอมพิวเตอร์ สำหรับโปรแกรมการประยุกต์ใช้งานในปรึญยานิพนธ์นี้เป็น การแปลง Fast Fourier Transform (FFT) 2 มิติ ของภาพขนาด 128*128 จุด

ABSTRACT

Digital signal processing generally operates on microcomputer that takes long process time. The objective of this project is to reduce these failure by using digital signal processing microprocessor (TMS320C25) to process instead of main CPU of microcomputer (IBM PC AT or over). Principle of map memory of TMS320C25 to the unused memory of IBM PC is used for this. When main CPU processes about DSP, it passes data to TMS320C25 for processing and return the result when ready. All of these processes are controled by microcomputer. For this applicatoon program is fast fourier transform of 128*128 image point.

สารบัญ

<u>บทที่</u>	<u>ชื่อเรื่อง</u>	<u>หน้า</u>
	บทนำ	1
1	IMAGE TRANSFORMS	2
2	FAST FOURIER TRANSFORM	14
3	TMS 320C25	18
4	IBM PC กับการอินเตอร์เฟส	32
5	รายละเอียดโครงการ	41
6	การประยุกต์ใช้งานและการทดลอง	65
7	ผลการทดลองและสรุป	72

ภาคผนวก

ตัวอย่างโปรแกรมการแปลง FFT ของ TMS 320C25
โปรแกรมควบคุมการทำงานของการ์ด

บทนำ

การประมวลผลสัญญาณเชิงเลข หรือที่เรียกว่า ดิจิตอลซิกแนลโปรเซสซิง (DSP) เป็นขบวนการที่ใช้ความรู้ทางคณิตศาสตร์เกี่ยวกับตัวเลขมาทำการจัดการกับสัญญาณต่างๆ ข้อมูลภาพก็คือสัญญาณชนิดหนึ่ง ซึ่งมีการนำมาประมวลผลแบบต่างๆ เพื่อใช้งาน ซึ่งงานด้านการประมวลผลเกี่ยวกับภาพจะมีชื่อเรียกเฉพาะว่า อิมเมจโปรเซสซิง (Image Processing) หรือดิจิตอลอิมเมจโปรเซสซิง (Digital Image Processing) โดยที่จะขอล่าถึงเฉพาะ ดิจิตอลอิมเมจโปรเซสซิง เท่านั้นเนื่องจากเกี่ยวกับปัญหานี้ขั้นต้น

ในงานประมวลผลสัญญาณภาพ ข้อมูลภาพจะประกอบด้วยจุดหลายๆ จุดประกอบกัน ขึ้นอยู่กับขนาดและความละเอียดของภาพนั้นๆ โดยจุดแต่ละจุดนั้นจะแสดงถึงค่าความสว่างของภาพ ในการประมวลผลภาพก็จะใช้ค่าของแต่ละจุดนี้มาทำการคำนวณทางคณิตศาสตร์ โดยวิธีในการคำนวณจะแตกต่างกันไปขึ้นอยู่กับความต้องการให้ภาพที่ได้เป็นเช่นไร สำหรับตัวอย่างของงานประมวลผล ได้แก่ การย่อและการขยายภาพ การลดสัญญาณรบกวนในภาพ การทำ Filter ภาพ เป็นต้น ซึ่งการประมวลผลแต่ละแบบก็จะมีรูปแบบในการคำนวณเฉพาะแบบ แต่ทุกแบบก็ยังคงเป็นการ บวก ลบ คูณและหารตัวเลขทั้งสิ้น ทำให้การประมวลผลแต่ละครั้งต้องมีการคำนวณเป็นจำนวนมาก สำหรับในปัญหานี้ขั้นต้นจะประยุกต์ใช้กับการแปลง FFT 2 มิติของภาพขาวดำ

เนื่องจากการประมวลผลเชิงเลขต้องอาศัยการคำนวณค่อนข้างมาก ผลที่ตามมาก็คือเวลาที่ใช้ก็มากขึ้นตาม ซึ่งในปัจจุบันงานด้านการประมวลผลภาพส่วนใหญ่จะใช้คอมพิวเตอร์เป็นตัวประมวลผล ถึงแม้ว่าคอมพิวเตอร์จะเป็นอุปกรณ์อิเล็กทรอนิกส์สามารถโปรแกรมให้ทำอะไรก็ได้มาก และยังทำงานได้รวดเร็วพอสมควร แต่อย่างไรก็ตามเมื่อนำคอมพิวเตอร์มาใช้งานประมวลผลเชิงเลขก็ยังมีข้อจำกัดในด้านความเร็วอยู่ เนื่องจากคอมพิวเตอร์ที่ใช้ในลักษณะออกมาสำหรับใช้งานทั่วไป เมื่อนำมาใช้คำนวณตัวเลขมาก ๆ คอมพิวเตอร์จึงไม่สามารถทำงานได้อย่างมีประสิทธิภาพนักคือยังทำงานได้ช้าอยู่ ยิ่งถ้าเป็นข้อมูลที่ใหญ่ขึ้นเวลาก็จะนานเพิ่มขึ้นอีก

เมื่อคอมพิวเตอร์ซึ่งทำงานได้เร็วในระดับหนึ่งแต่ยังไม่พอกับความต้องการ ดังนั้นจึงเกิดความคิดที่จะเพิ่มประสิทธิภาพด้านการคำนวณตัวเลขหลายๆ ให้แก่คอมพิวเตอร์ เพื่อช่วยให้คอมพิวเตอร์ทำการประมวลผลได้เร็วขึ้น โดยความคิดนี้เป็นที่มาของปัญหานี้ขั้นต้น คือ "การประยุกต์ใช้ไมโครโปรเซสเซอร์ (DSP) ในงานอิมเมจโปรเซสซิง" หลักการคร่าว ๆ ก็คือการนำเอาไมโครโปรเซสเซอร์เฉพาะทางมาช่วย CPU บนคอมพิวเตอร์ประมวลผล ซึ่งไมโครโปรเซสเซอร์ที่นำมาใช้ก็คือ TMS 320c25 ของบริษัท TEXAS INSTRUMENT สำหรับรายละเอียดอื่นจะขอกล่าวในส่วนต่อไป

บทที่ 1

FOURIER TRANSFORM

1.1 FOURIER TRANSFORM

ถ้า $f(x)$ เป็นฟังก์ชันต่อเนื่องของตัวแปรจริง x fourier tranform ของ $f(x)$ หรือ $\{f(x)\}$ สามารถนิยามได้ดังสมการ

$$\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x) \exp[-j2\pi ux] dx \quad (1.1-1)$$

และ $f(x)$ สามารถหาได้โดยการใช้ inverse fourier transform ดังสมการ

$$\begin{aligned} \{F(u)\}^{-1} &= f(x) \\ &= \int_{-\infty}^{\infty} F(x) \exp[j2\pi ux] du \end{aligned} \quad (1.1-2)$$

สมการ (1.1-1) และ (1.1-2) เรียกว่า คู่สมการ fourier transform และจะเป็นจริงเมื่อ $f(x)$ เป็นฟังก์ชันต่อเนื่อง, สามารถอินทิเกรตได้ และ $F(u)$ สามารถอินทิเกรตได้

โดยทั่วไป ถ้าฟังก์ชัน $f(x)$ เป็นจำนวนจริง fourier tranform ของ $f(x)$ จะเป็นจำนวนเชิงซ้อน และสามารถเขียนได้เป็น

$$F(u) = R(u) + jI(u) \quad (1.1-3)$$

เมื่อ $R(u)$ และ $I(u)$ เป็นส่วนจริงและส่วนจินตภาพของ $F(u)$ ตามลำดับ สมการ(1.1-3) สามารถแสดงในรูปของ exponential ได้เป็น

$$F(u) = |F(u)| \exp[j\phi(u)] \quad (1.1-4)$$

เมื่อ

$$|F(u)| = [R^2(u) + I^2(u)]^{1/2} \quad (1.1-5)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และ

$$\phi(u) = \tan^{-1} \frac{|I(u)|}{|R(u)|} \quad (1.1-6)$$

ขนาดของฟังก์ชัน $|F(u)|$ เรียกว่า fourier spectrum ของ $f(x)$ และ $\phi(u)$ เรียกว่า phase angle ส่วนกำลังสองของ spectrum

$$\begin{aligned} P(u) &= |F(u)|^2 \\ &= R^2(u) + I^2(u) \end{aligned} \quad (1.1-7)$$

เรียกว่า power spectrum ของ $f(x)$

ตัวแปร u จาก fourier transform โดยทั่วไปเรียกว่า ตัวแปรทางความถี่ ซึ่งจะเห็นได้จากเทอม exponential $\exp[-j2\pi ux]$ เมื่อใช้ Euler's formula จะได้

$$\exp[-j2\pi ux] = \cos 2\pi ux - j \sin 2\pi ux \quad (1.1-8)$$

และความหมายของสมการ (1.1-1) ก็คือ $F(u)$ จะประกอบด้วย ผลบวกจำนวนอนันต์เทอมของ sin ของ cosine ที่ความถี่ u ต่างๆกัน

ตัวอย่างของการแปลง fourier transform ของฟังก์ชัน $f(x) = A$, $0 < x < X$ แสดงได้ดังรูปที่ 1.1(a) จากสมการ (1.1-1) จะได้

$$\begin{aligned} F(u) &= \int_{-\infty}^{\infty} f(x) \exp[-2j\pi ux] dx \\ &= \int_0^X A \exp[-2j\pi ux] dx \\ &= \frac{-A}{j2\pi u} (\exp[-2j\pi ux])_0^X = \frac{-A}{j2\pi u} (\exp[-j2\pi uX] - 1) \end{aligned}$$

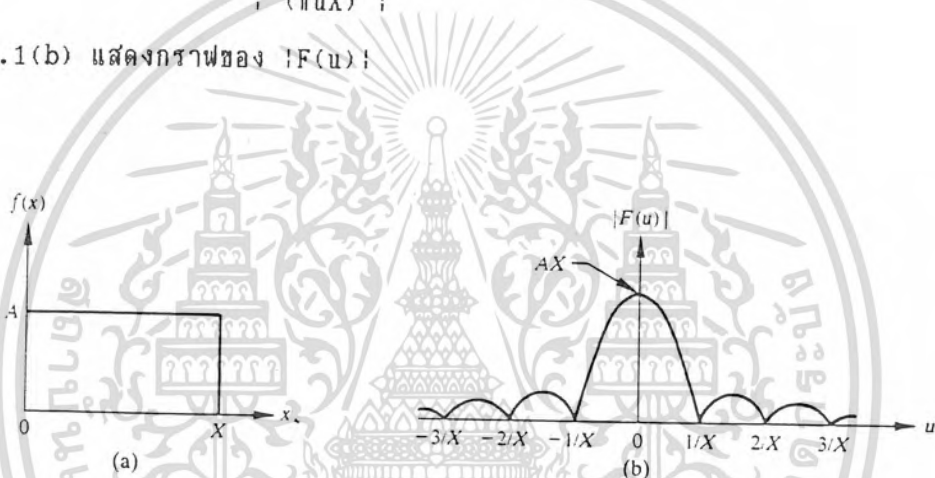
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\begin{aligned}
 &= \frac{A}{j2\pi u} (\exp[j\pi u X] - \exp[-j\pi u X]) \exp[-j\pi u X] \\
 &= \frac{A}{\pi u} \sin(\pi u X) \exp[-j\pi u X]
 \end{aligned}$$

และสามารถหา fourier spectrum ได้เป็น

$$\begin{aligned}
 |F(u)| &= |A| |\sin(\pi u X)| |\exp[-j\pi u X]| \\
 &= AX |\sin(\pi u X)| \\
 &= AX \text{sinc}(\pi u X)
 \end{aligned}$$

รูปที่ 1.1(b) แสดงกราฟของ $|F(u)|$



รูปที่ 1.1 แสดงการแปลง fourier transform ของฟังก์ชัน $f(x)$ และ power spectrum ของการแปลง

fourier transform สามารถใช้กับฟังก์ชัน $f(x,y)$ ซึ่งเป็นฟังก์ชัน 2 ตัวแปร ได้ในลักษณะเดียวกัน โดยมีคู่สมการ fourier transform เป็น

$$\{f(x,y)\} = F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \exp[-j2\pi(ux + vy)] dx dy \quad (1.1-9)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และ

$$F^{-1}\{F(u,v)\} = f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) \exp[j2\pi(ux + vy)] du dv \quad (1.1-10)$$

เมื่อ u และ v เป็นตัวแปรทางความถี่

สำหรับ fourier spectrum , phase และ power spectrum จะเหมือนกับกรณี 1 มิติ คือ

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2} \quad (1.1-11)$$

$$\phi(u,v) = \tan^{-1} \frac{|I(u,v)|}{|R(u,v)|} \quad (1.1-12)$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \quad (1.1-13)$$

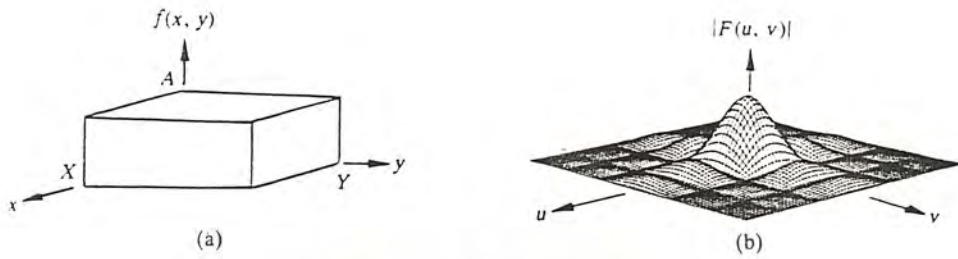
ตัวอย่างของการแปลง fourier transform 2 มิติ แสดงดังรูปที่ 1.2(a) และจะได้

$$\begin{aligned} F(u,v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \exp[-j2\pi(ux + vy)] dx dy \\ &= A \int_{-\infty}^{\infty} \exp[-2j\pi ux] dx \int_{-\infty}^{\infty} \exp[-2j\pi vy] dy \\ &= A \frac{\exp[-j2\pi ux]}{-j2\pi u} \Big|_{-\infty}^{\infty} \frac{\exp[-j2\pi vy]}{-j2\pi v} \Big|_{-\infty}^{\infty} \\ &= \frac{A}{-2j\pi u} (\exp[-2j\pi uX] - 1) \frac{1}{-j2\pi v} (\exp[-2j\pi vY] - 1) \\ &= \frac{AXY \sin(\pi uX) \exp[-j2\pi uX]}{(\pi uX)} \frac{\sin(\pi vY) \exp[-j2\pi vY]}{(\pi vY)} \end{aligned}$$

และ fourier spectrum คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$|F(u,v)| = \frac{AXY}{\pi^2} \frac{|\sin(\pi uX)|}{(\pi uX)} \frac{|\sin(\pi vY)|}{(\pi vY)}$$



รูปที่ 1.2 (a) ฟังก์ชัน 2 มิติ $f(x,y)$, (b) fourier spectrum ของ $f(x,y)$ และ (c) การแสดง spectrum ในรูปของความเข้ม โดยความสว่างจะขึ้นอยู่กักับ แอมพลิจูดของ $|F(x)|$

1.2 DISCRETE FOURIER TRANSFORM

ถ้าฟังก์ชันต่อเนื่อง $f(x)$ ถูก sampling ด้วย Δx จำนวน N ส่วนออกจากกัน ดังแสดงในรูปที่ 1.3 จะสามารถเขียน $f(x)$ ได้เป็น

$$f(x) = f(x_0 + x\Delta x) \tag{1.2-1}$$

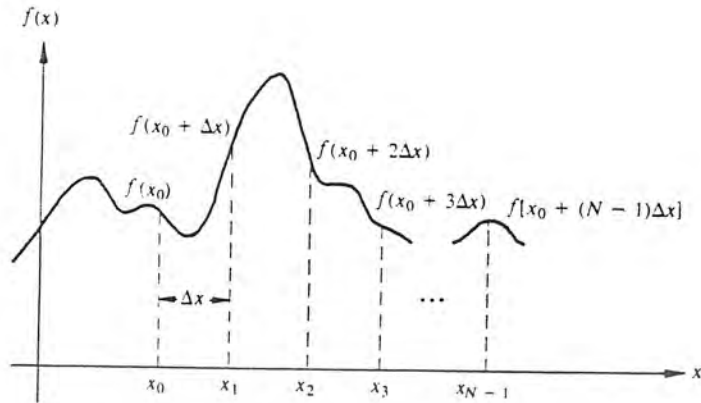
หรือเป็นลำดับ

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \dots, f(x_0 + [N-1]\Delta x)\}$$

และถ้าให้ x มีค่าเป็น $0, 1, 2, \dots, N-1$ จะเขียนลำดับได้เป็น

$$\{f(0), f(1), f(2), \dots, f(N-1)\}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.3 แสดงการ sampling ฟังก์ชันต่อเนื่อง

fourier transform ของ $f(x)$ ที่ถูก sampling เรียกว่า discrete fourier transform และมีคู่สมการเป็น

$$F(u) = \sum_{x=0}^{N-1} f(x) \exp[-j2\pi ux/N] \quad (1.2-2)$$

เมื่อ $u = 0, 1, 2, \dots, N-1$ และ

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \exp[j2\pi ux/N] \quad (1.2-3)$$

เมื่อ $x = 0, 1, 2, \dots, N-1$

ในกรณีของฟังก์ชัน 2 ตัวแปร คู่สมการ discrete fourier transform จะเป็น

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)] \quad (1.2-4)$$

เมื่อ $u = 0, 1, 2, \dots, M-1$, $v = 0, 1, 2, \dots, N-1$ และ

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux/M + vy/N)] \quad (1.2-5)$$

เมื่อ $x = 0, 1, 2, \dots, M-1$, $y = 0, 1, 2, \dots, N-1$

สำหรับ fourier spectrum , phase และ power spectrum ของ discrete fourier transform ทั้ง 1 มิติ และ 2 มิติ สามารถหาได้เช่นเดียวกับสมการ (1.1-5)-(1.1-7) และสมการ (1.1-11)-(1.1-13) ตามลำดับ

1.3 คุณสมบัติของ FOURIER TRANSFORM 2 มิติ

คุณสมบัติของ fourier transform มีทั้งทาง 1 มิติ และ 2 มิติ ซึ่งโดยทั่วไปแล้ว จะมีลักษณะคล้ายคลึงกัน สำหรับในส่วนนี้จะแสดงเฉพาะคุณสมบัติทาง 2 มิติ ซึ่งเกี่ยวข้องกับปริภูมิตวินพจน์เท่านั้น

1.3.1 separability

คู่สมการ discrete fourier transform ในสมการที่ (1.2-4) และ (1.2-5) สามารถเขียนแบบแยกตัวแปรได้เป็น

$$F(u,v) = \sum_{x=0}^{N-1} \exp[-j2\pi ux/N] \sum_{y=0}^{N-1} f(x,y) \exp[-j2\pi vy/N] \quad (1.3-1)$$

เมื่อ $u, v = 0, 1, \dots, N-1$ และ

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u,v) \exp[j2\pi uv/N] \quad (1.3-2)$$

เมื่อ $x, y = 0, 1, \dots, N-1$

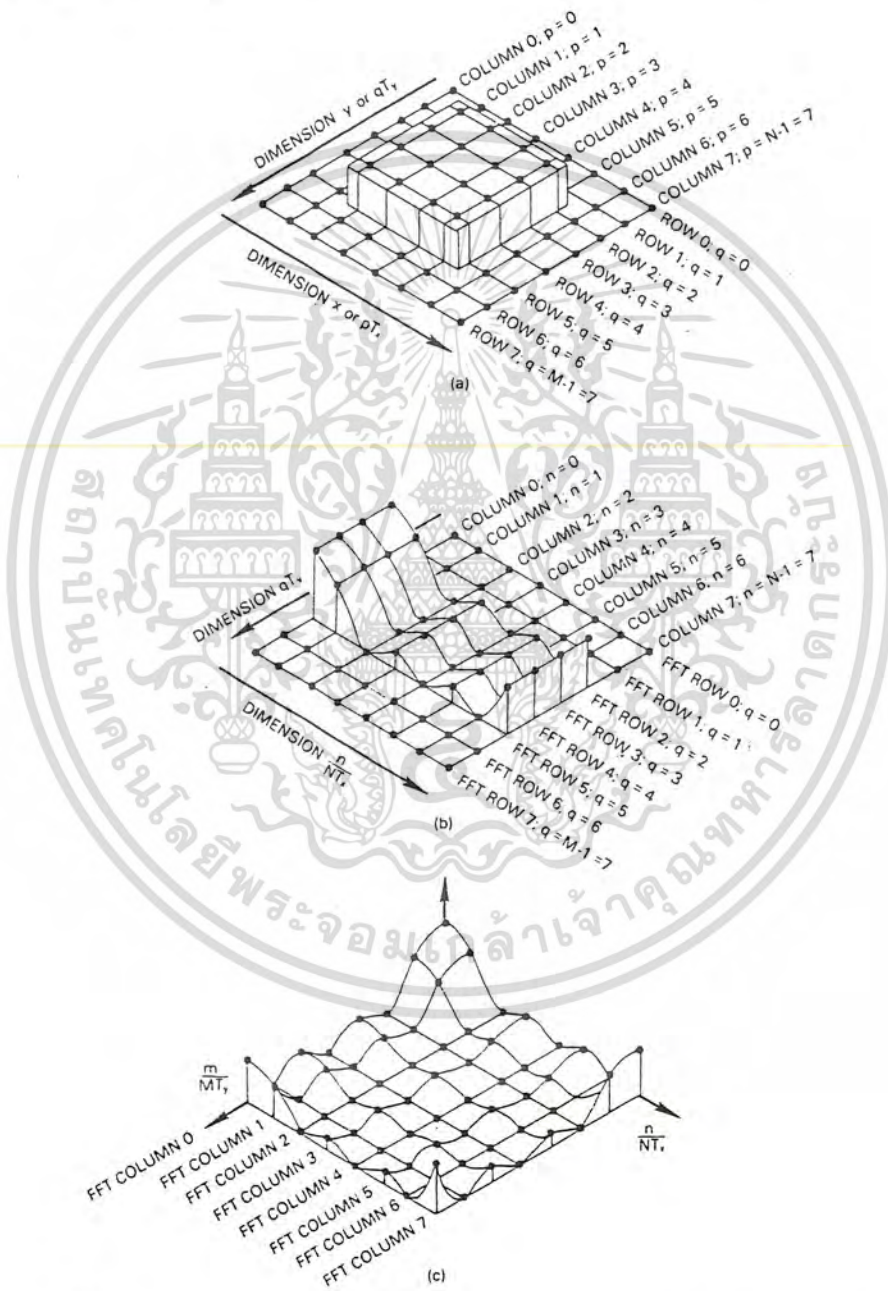
ประโยชน์ของคุณสมบัตินี้ จะทำให้การหา $F(u,v)$ (หรือ $f(x,y)$) สามารถทำได้โดยการแปลง fourier transform (หรือ inverse) 1 มิติ 2 ครั้ง โดยการหา $F(x,v)$ ซึ่งเป็นการแปลงในแต่ละแถวของ $f(x,y)$ และ หา $F(u,v)$ ซึ่งการแปลงในแต่ละหลักของ $F(x,v)$ ซึ่งสามารถแสดงได้ดังสมการ

$$F(u,v) = \sum_{x=0}^{N-1} F(x,v) \exp[-j2\pi ux/N] \quad (1.3-3)$$



$$F(u, v) = \sum_{v=0}^{N-1} f(x, y) \exp[-j2\pi v y / N] \tag{1.3-4}$$

รูปที่ 1.4 แสดงการแปลง fourier transform แบบ 2 มิติ โดยใช้คุณสมบัติ separability และโดยวิธีเดียวกันนี้ การแปลงทางด้านหลักก่อน แล้วจึงแปลงทางด้านแถว ก็สามารถทำได้เช่นเดียวกัน



รูปที่ 1.4 แสดงการแปลง fourier transform 2 มิติ โดยใช้คุณสมบัติ separability

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการอ้างถึง

1.3.2 translation

คุณสมบัติข้อนี้สามารถแสดง คู่สมการ fourier transform ได้เป็น

$$f(x,y)\exp[j2\pi(u_0x + v_0y)/N] \leftrightarrow F(u-u_0, v-v_0) \quad (1.3-5)$$

และ

$$f(x-x_0, y-y_0) \leftrightarrow F(u, v)\exp[-j2\pi(ux_0 + vy_0)/N] \quad (1.3-6)$$

สมการ (1.3-6) แสดงให้เห็นว่าการแปลง fourier transform ของ $f(x,y)$ ที่ถูกคูณด้วยเทอม exponential จะทำให้ระนาบความถี่มีการ shift จากจุด origin มาถึงจุด (x_0, y_0) และในทำนองเดียวกัน การแปลง inverse ของ $F(u,v)$ ที่ถูกคูณด้วยเทอม exponential จะทำให้ระนาบจริง shift ไปยังจุด (x_0, y_0)

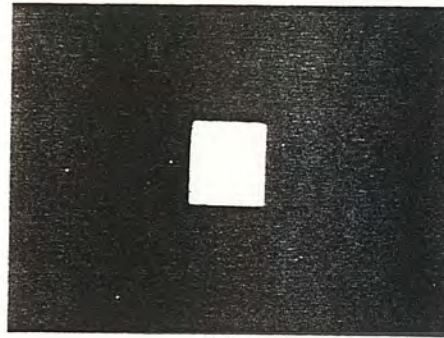
ในกรณีที่ $u_0 = v_0 = N/2$ หรือ

$$\begin{aligned} \exp[j2\pi(u_0 + v_0)/N] &= \exp[j\pi(x + y)] \\ &= (-1)^{x+y} \end{aligned}$$

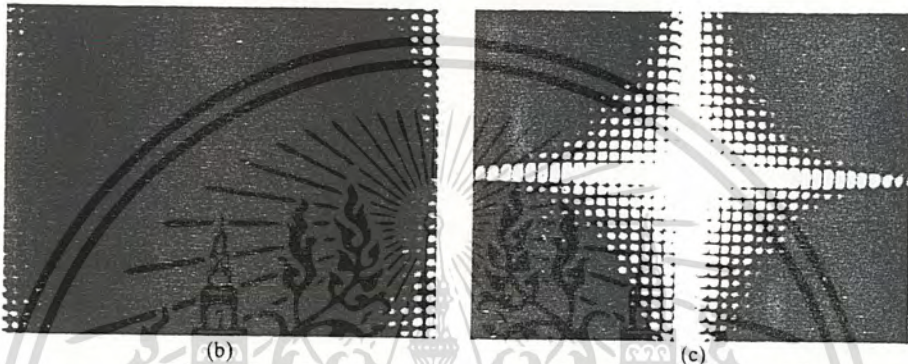
และ

$$f(x,y)(-1)^{x+y} = F(u - N/2, v - N/2) \quad (1.3-7)$$

ดังนั้น เมื่อทำการแปลง fourier transform $f(x,y)$ จะสามารถทำให้จุด origin เลื่อนไปยังจุดศูนย์กลางของระนาบความถี่ $N \times N$ ได้โดยการคูณ $f(x,y)$ ด้วยเทอม $(-1)^{x+y}$ คุณสมบัติข้อนี้แสดงได้ดังรูปที่ 1.5 และ 1.6



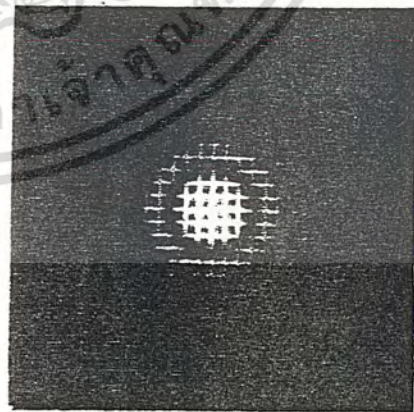
(a)



(b)

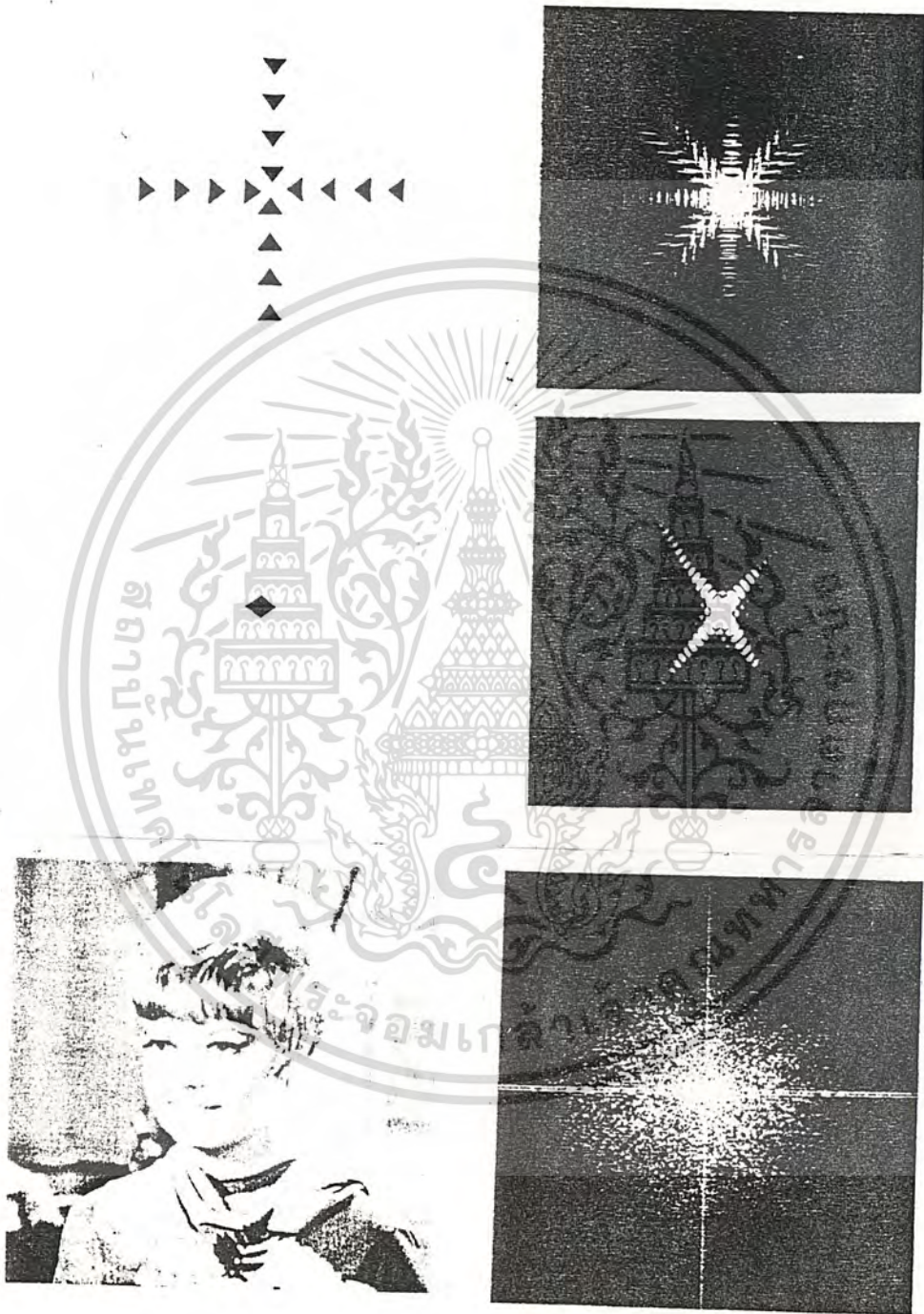
(c)

รูปที่ 1.5 (a) แสดงภาพตัวอย่าง (b) แสดง fourier spectrum ของภาพ
เมื่อไม่มีการ shift และ (c) แสดง fourier spectrum ของ
ภาพที่มีการ shift ไปยังจุดศูนย์กลางของความถี่



รูปที่ 1.6 แสดงตัวอย่างการแปลง fourier transform 2 มิติ ของรูปต่าง ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.6 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.6 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

FAST FOURIER TRANSFORM

โดยทั่วไป การแปลง discrete fourier transform หรือ DFT เป็นการคำนวณที่ใช้เวลาค่อนข้างมาก เช่นการคำนวณ DFT ของข้อมูล N ลำดับ จะต้องมีการคำนวณจำนวนเชิงซ้อนถึง $N*N$ ครั้ง และมีการบวกจำนวนเชิงซ้อนอีก $N(N-1)$ ครั้ง จะเห็นได้ว่าความเร็ว และจำนวนครั้งในการคำนวณเป็นสิ่งสำคัญที่จะทำให้ผลการคำนวณ DFT ช้าหรือเร็ว

ลำดับขั้นตอนในการคำนวณ DFT ให้เร็วขึ้นเรียกว่า fast fourier transform หรือ FFT วิธีการนี้จะทำให้การคำนวณ DFT ใช้การคูณจำนวนเชิงซ้อนเพียง $N \log_2 N$ ครั้งเท่านั้น ซึ่งจะทำให้การคำนวณ DFT เร็วขึ้นมาก และจะทำให้มีลักษณะเป็น real time มากขึ้น

การแปลง FFT แบ่งได้เป็น 2 ชนิดใหญ่ ๆ คือ ชนิดลดทอนทางเวลา (decimation in time หรือ DIT) และ ชนิดลดทอนทางความถี่ (decimation in frequency หรือ DIF) สำหรับในส่วนนี้จะแสดงเฉพาะชนิดลดทอนทางเวลา ซึ่งเกี่ยวข้องกับ project นี้เท่านั้น

2.1 ขั้นตอนวิธีลดทอนทางเวลา (decimation in time หรือ DIT)

วิธีนี้เป็นการจัดแบ่งกลุ่มลำดับสัญญาณในโดเมนเวลา $x(n)$ ขนาด N จุด ออกเป็น 2 ลำดับสัญญาณขนาด $N/2$ จุดเท่ากัน คือ ลำดับคู่ และลำดับคี่ โดยที่ลำดับคู่เกิดจากการเอาลำดับในตำแหน่งคู่มาเรียงกัน ที่เหลือเป็นลำดับคี่ ดังนั้นจะได้

$$x_o(m) = x(2m) \quad ; m = 0, 1, \dots, (N/2)-1 \quad (2.1-1)$$

$$x_o(m) = x(2m + 1) \quad ; m = 0, 1, \dots, (N/2)-1$$

ถ้าให้ W_N เท่ากับ $\exp[-j2\pi/N]$ จะทำให้การคำนวณ DFT ของลำดับ $x(n)$ ที่ยาว N จุด สามารถเขียนใหม่ได้เป็น

$$\begin{aligned} X(k) &= \sum_{m=0}^{N-1} x_o(m) (W_N)^{km} + \sum_{m=0}^{N-1} x_o(m) (W_N)^{k+m} \\ &= \sum_{m=0}^{N/2-1} x(2m) (W_N)^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) (W_N)^{(2m+1)k} \end{aligned} \quad (2.1-2)$$

โดยที่

$$(W_N)^2 = \{\exp[j2\pi/N]\}^2 = \exp[j2\pi/N/2] = W_{N/2}$$

ซึ่ง $W_{N/2}$ เป็นค่า W ของลำดับยาว $N/2$ จุด และ (2.1-2) สามารถเขียนใหม่เป็น

$$X(k) = \sum_{m=0}^{N/2-1} x_o(m) (W_{N/2})^{km} + (W_N)^k \sum_{m=0}^{N/2-1} x_o(m) (W_{N/2})^{km}$$

$$X(k) = X_o(k) + (W_N)^k X_o(k) \quad (2.1-3)$$

โดยที่ $X_o(k)$ และ $X_o(k)$ แทนผลการแปลง DFT ขนาด $N/2$ จุด ของลำดับ $x_o(m)$ และ $x_o(m)$ ตามลำดับ จากวิธีการนี้ จะเห็นได้ว่าการคำนวณ DFT ขนาด N จุด สามารถแบ่งย่อยเป็นการคำนวณ DFT ขนาด $N/2$ จุด สองอันได้ โดยหลักการเดียวกันนี้ ถ้าแบ่งลำดับ $x_o(m)$ และ $x_o(m)$ ออกเป็นลำดับคู่ และคี่ จนเหลือลำดับขนาด 2 จุด จะทำให้การคำนวณ DFT ขนาด N จุด สามารถทำได้โดยการแปลง DFT ขนาด 2 จุดจำนวน $N/2$ ภาค ดังแสดงในรูปที่ 2.2

การนำผลการแปลง DFT ขนาด 2 จุด จำนวน $N/2$ ภาคมารวมกันเพื่อเป็นการคำนวณ DFT ขนาด N จุด จะต้องมีหลักเกณฑ์ที่ต้องด้วย เพราะจาก (2.1-3) ทั้ง $X_1(k)$ และ $X_2(k)$ เป็น DFT ขนาด $N/2$ จุดที่นิยามเฉพาะช่วง $0 \leq k < N/2$ เท่านั้น ดังนั้น เมื่อนิยามในช่วง $k > N/2$ ด้วย จะได้

$$X(k) = X_o(k) + (W_N)^k X_o(k) \quad ; 0 < k < N/2-1$$

$$= X_o(k-N/2) + (W_N)^k X_o(k-N/2) \quad ; N/2 < k < N-1 \quad (2.1-4)$$

เทอม $(W_N)^k$ เรียกว่า ตัวประกอบหมุน (twiddle factor) ซึ่งใช้ร่วมกับ DFT ขนาด 2 จุด หรือขนาด $N/2$ จุด ในการนำมาประกอบเป็น DFT ขนาด N จุดได้เหมือนเดิม

และจากความสัมพันธ์ $(W_N)^{k-N/2} = -(W_N)^k$ จะได้

$$X(k) = x_1(k) + (W_N)^k x_2(k) \quad ; 0 < k < N/2-1$$

$$= x_1(k-N/2) + (W_N)^{k-N/2} x_2(k-N/2) \quad ; N/2 < k < N/2-1 \quad (2.1-5)$$

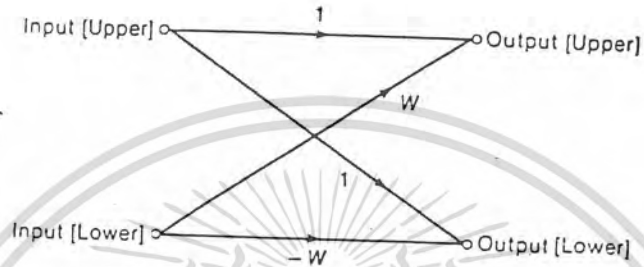
ผลจากสมการนี้ สามารถนำไปใช้สร้างหน่วยคำนวณที่ เรียกว่า หน่วยผีเสื้อ (butterfly unit) โดยมีข้อมูลเข้า คือ A และ B และข้อมูลออกคือ X และ Y เป็น

$$X = A + (W_N)^k B$$

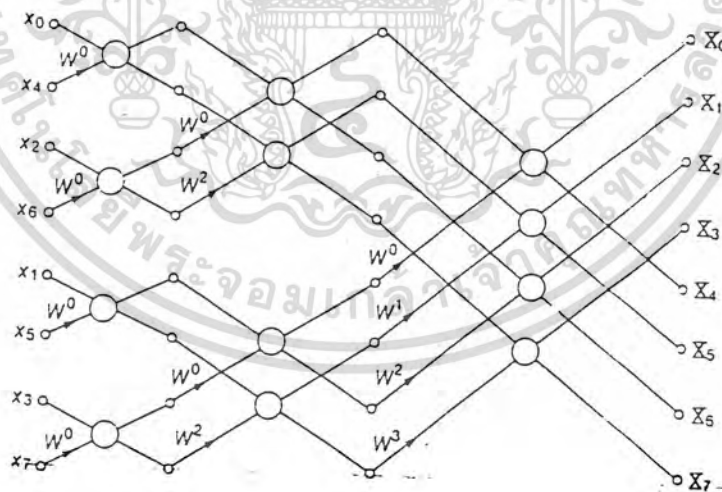
(2.1-6)

$$Y = A - (W_N)^k B$$

ซึ่งสามารถเขียนอธิบายแทนด้วยกราฟการไหล ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 แสดงหน่วยเซลล์ของการคำนวณตามขั้นตอนวิธีลดทอนทางเวลา สำหรับตัวอย่างการคำนวณ DFT โดยวิธี FFT แสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 แสดงวิธีการของ FFT แบบลดทอนทางเวลา (DIT) สำหรับข้อมูล ขนาด 8 จุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะสังเกตเห็นได้ว่า เมื่อมีการแบ่งข้อมูลเข้าเป็นลำดับคู่ และคู่ ไปเรื่อย ๆ จนได้ข้อมูล 2 จุด จะทำให้ลำดับข้อมูลเข้ามีการเปลี่ยนแปลงไป ซึ่งการเปลี่ยนแปลงนี้ เป็นการเปลี่ยนอย่างมีหลักเกณฑ์ ที่เรียกว่า การผันบิตกลับ (bit reverse) และสามารถแสดงได้ดังรูปที่ 2.3

INDEX	BIT PATTERN	BIT-REVERSED PATTERN	BIT-REVERSED INDEX
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

รูปที่ 2.3 แสดงการสลับตำแหน่งข้อมูลเข้าด้วยการผันบิตกลับ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยญาติให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

TMS320C25

TMS320C25 เป็นการปรับปรุงขึ้นจากรุ่นเดิมคือ TMS32020 จึงทำให้ขีดความสามารถของ TMS320C25 นี้เพิ่มขึ้นอีกมากคือ

- คำสั่งแต่ละคำสั่งที่ทำงานจะใช้เพียง 100 nS และคำสั่งส่วนใหญ่จะใช้เพียงหนึ่งไซเคิลเท่านั้น ดังนั้น TMS320C25 จึงมีความเร็วในการทำงานได้สูงกว่า 10 ล้านคำสั่งในเวลา 1 วินาที

- มีคำสั่งที่จะโอนย้ายข้อมูล ระหว่างหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้

- ในการทำงานแต่ละคำสั่ง TMS320C25 จะทำงานสามจังหวะแบบ pipeline คือ เฟดท์ , ดีโค้ด และ เอ็กcute โดยการทำงานนี้จะไม่เห็นได้โดยผู้ใช้งาน เพราะเป็นการทำงานที่เกิดขึ้นพร้อมกันเพื่อเพิ่มความเร็ว

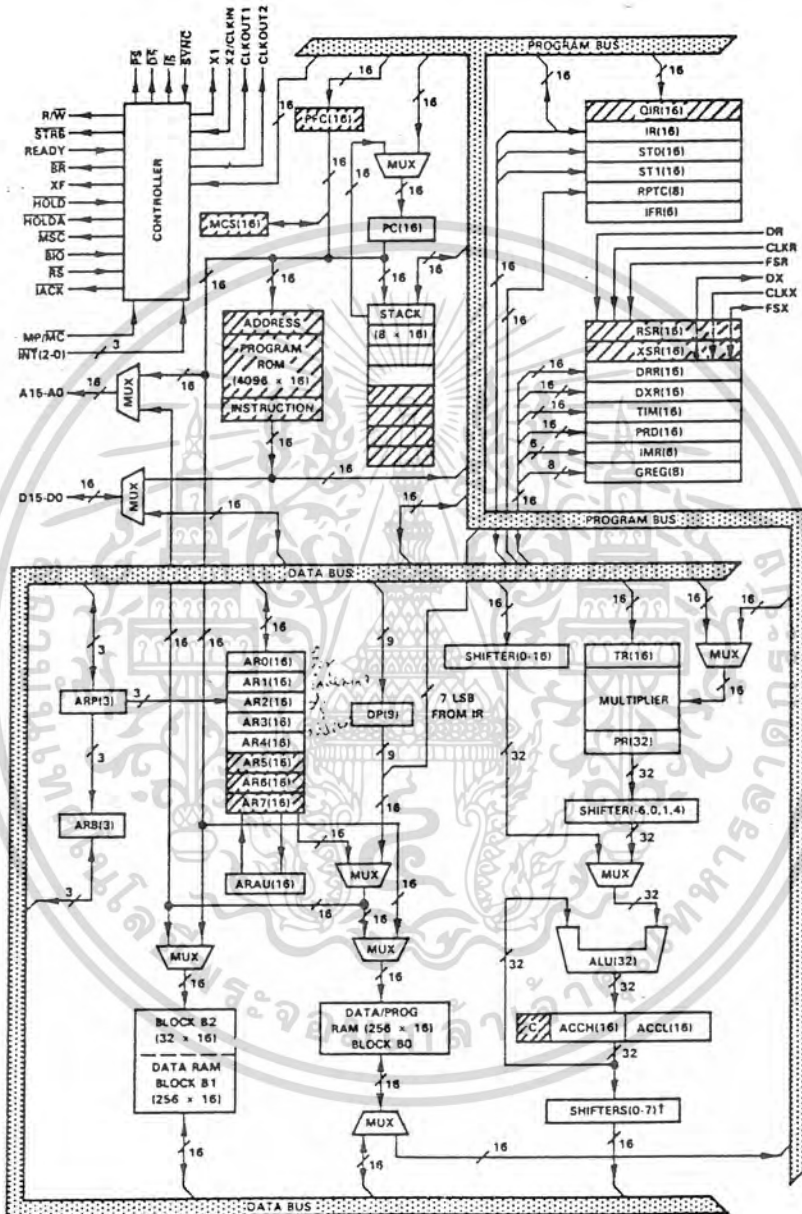
- ได้เพิ่มจำนวนหน่วยความจำขึ้นอีก โดยมี RAM ภายในถึง 544 เวิร์ด (1 เวิร์ด มี 16 บิต) หน่วยความจำภายนอกที่ใช้สามารถต่อได้ถึง 64K เวิร์ด ซึ่งมีจำนวนสูงมาก จึงเป็น DSP ที่มีการต่อหน่วยความจำภายนอกได้สูง

- มี ROM ให้ใช้ตามผู้สั่งทำอยู่บนชิพอีก 4K เวิร์ด การทำงานของชิพนี้ให้ข้อเด่นอีกข้อคือ สามารถโหลดโปรแกรมจากหน่วยความจำภายนอกเข้าไปเก็บใน RAM ภายใน เพื่อให้การทำงานทำได้เร็วยิ่งขึ้น

- ขีดความสามารถที่เพิ่มเติมมาอีกประการหนึ่งคือ การมีฮาร์ดแวร์ไทมเมอร์และขีดความสามารถในการโอนย้ายข้อมูลเป็นบล็อกๆ

ไดอะแกรมของ TMS32025 แสดงได้ดังรูปที่ 3.1

สถาปัตยกรรมของ TMS320C25 สร้างขึ้นมาเพื่อความเร็วในการทำงานและโครงสร้างที่ทำงานได้ด้วยขีดความสามารถที่สูงขึ้น และเพื่อให้การทำงานของบัสไม่ขึ้นต่อกัน จึงแยกบัสเป็น บัสของโปรแกรมและบัสของข้อมูลจากกัน โดยบัสของโปรแกรมจะเป็นทางเข้าออกที่คำสั่ง และ โอเพอร์เรนด์ของคำสั่ง ส่วนบัสข้อมูลจะเชื่อมต่อโดยตรงกับวงจรการทำงานการประมวลผล เช่น CALU (Central Arithmetic Logic Unit) และรีจิสเตอร์ที่เป็นไฟล์ข้อมูลย่อย ARO-AR7 และยังมี ARAU ซึ่งเป็น Auxiliary Register Arithmetic Unit การทำโครงสร้างการคำนวณทางคณิตศาสตร์นี้ ยึดหลักการให้ทำงานด้วยประสิทธิภาพ เช่น การเลื่อนบิต (shift) การคูณหารหรือคำสั่งทางลอจิก



¹ Shifters on TMS32020 (0, 1, 4)
 NOTE: Shaded areas are for TMS320C25 only.

รูปที่ 3.1 โครงสร้างของ TMS320C25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 การจัดหน่วยความจำ

TMS320C25 มีการแบ่งหน่วยความจำเป็นส่วน program และ data ตามรูปที่ 3.2

3.1.1 data memory

บล็อก B0 256 เวิร์ด (1 เวิร์ด มี 16 บิต) B1 256 เวิร์ด B2 32 เวิร์ด เป็นหน่วยความจำที่อยู่ในชิป (on-chip Ram) ซึ่งสามารถทำงานได้ด้วยความเร็วเต็มที่ โดยสามารถโปรแกรมให้เป็น Data Memory หรือ Program Memory ได้โดยคำสั่ง CNFD หรือ CNFP รูปที่ 3.2a แสดง Memory Map หลังจากใช้คำสั่ง CNFD ส่วนรูปที่ 3.2b แสดง Memory Map หลังจากใช้คำสั่ง CNFP ส่วนบล็อก B1 และ B2 จะเป็น Data Memory เสมอ

TMS320c25 สามารถอ้างอิงหน่วยความจำข้อมูลได้ทั้งหมด 64 Kwords หากเราใช้หน่วยความจำข้อมูลบนชิปด้วย ตำแหน่งของมันจะถูก Map ลงในตำแหน่งที่ต่ำกว่า 1 Kwords ของ Data memory space และหน่วยความจำข้อมูลสามารถขยายเพิ่มขึ้นโดยตรงจนมีขนาด 64 Kwords ขณะที่ยังคงทำงานด้วยความเร็วเต็มที่

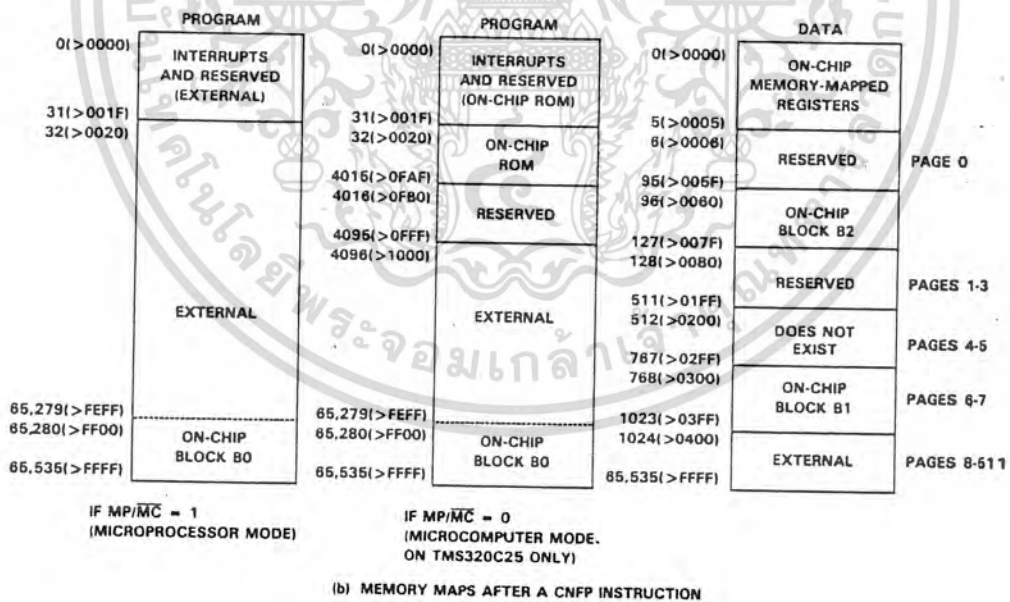
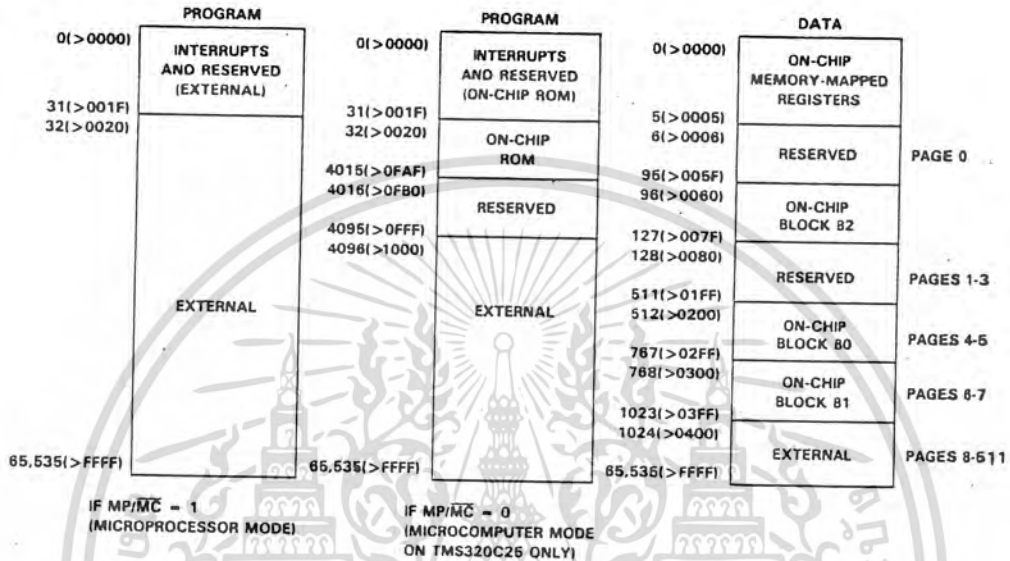
3.1.2 program memory

โปรแกรม RAM และ ROM บนชิปหรือหน่วยความจำภายนอกความเร็วสูงสามารถถูกใช้ด้วยความเร็วเต็มที่โดยไม่มีสภาวะรอ (wait state) หรือมีสถานะ READY ก็สามารถติดต่อกับ TMS320c25 ให้ข้างล่างหากใช้หน่วยความจำภายนอกก็ทำงานซ้ำ TMS320c25 สามารถมีหน่วยความจำโปรแกรมได้ทั้งหมด 64 Kwords โดย RAM ภายใน (block B0) สามารถกำหนดให้เป็นหน่วยความจำโปรแกรมได้ โดยใช้ software ซึ่งการเอ็ทซีคิวต์จาก block B0 สามารถเริ่มได้หลังจาก Memory space ถูกกำหนดใหม่

หาก TMS320C25 ถูกใช้ร่วมกับ on-chip program ROM ขนาด 4 Kwords ซึ่งสามารถโปรแกรมจากโรงงาน On-chip ROM ทำให้การเอ็ทซีคิวต์โปรแกรมได้ด้วยความเร็วเต็มที่ การกำหนดตำแหน่ง 4kwords แรกว่าจะ เป็นหน่วยความจำภายนอกชิปหรือบนชิปสามารถเลือกได้โดยกำหนดจากขา MP/~MC (Microprocessor/ Microcomputer) หากขาดังกล่าวเป็น High ตำแหน่ง 4kwords แรกจะเป็นหน่วยความจำนอกชิป ถ้ามีสถานะเป็น Low ตำแหน่ง 4kwords แรกเป็น On-chip ROM

มีการใช้แอสแตริสค์ระหว่างโปรแกรมข้อมูล และ I/O ทำให้เกิดความคล่องตัวในการทำงาน หน่วยความจำภายในของ DSP นี้แบ่งเป็นรีจิสเตอร์สำหรับเพริเฟอรัล (peripheral register) 6 ตัว ซึ่งประกอบด้วย DRR และ DXR (serial port register), TIM (timer register), PRD (period register), IMR (interrupt mask register) และ GREG (global memory allocation register)

ภายใน TMS320C25 ยังมีรีจิสเตอร์ไฟล์ 8 ตัว คือ AR0- AR7 ไว้สำหรับการทำการอ้างแอดเดรสโดยอ้อมหรือเป็นที่เก็บข้อมูลชั่วคราว รีจิสเตอร์ทั้ง 8 ตัวนี้สามารถเรียกใช้โดยตรงจากคำสั่งหรือจากการอ้างอิงเพียง 3 บิต จาก ARP(Auxiliary Register Pointer)ARP นี้รับข้อมูลได้โดยตรงจากหน่วยความจำข้อมูลหรือมาจากโอเปอร์แรนด์ของคำสั่ง



รูปที่ 3.2 การจัดหน่วยความจำของ TMS320C25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

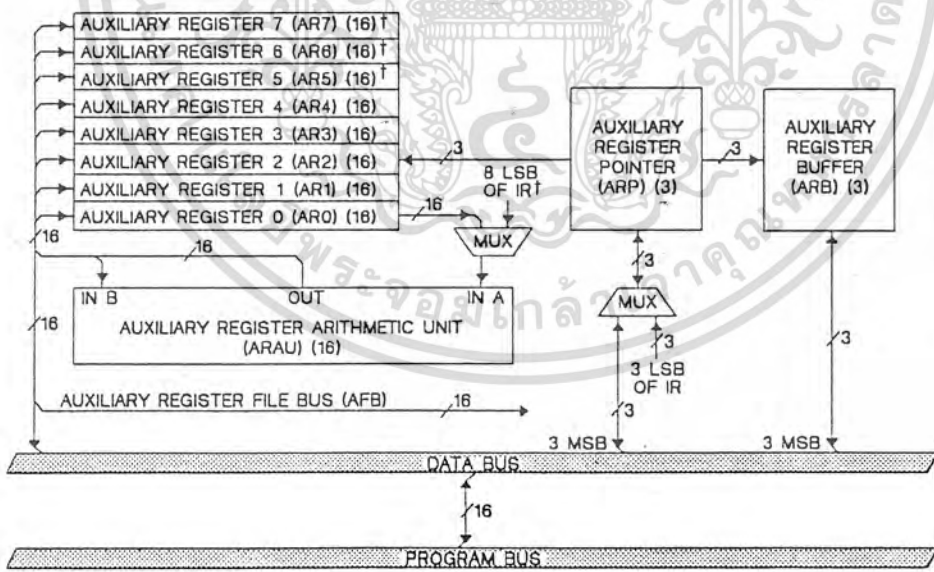
รีจิสเตอร์ไฟล์โดยตรงเข้ากับ ARAU ดังแสดงในรูปที่ 3.3 ARAU สามารถทำการเพิ่มค่าหรือลดค่าเพื่อทำการออดิโอสแตรส ซึ่งทำให้การออดิโอสแตรสเพื่อนำข้อมูลจากตารางทำได้ง่ายและรวดเร็ว ดังนั้น ARAU จึงมีหน้าที่สำคัญสำหรับการสนับสนุน การออดิโอสแตรสให้ได้ขีดความสามารถสูงขึ้น โครงสร้างของ ARAU แสดงได้ดังรูปที่ 3.3

นอกจากนี้ ARAU ยังสามารถเป็นหน่วยคำนวณทางคณิตศาสตร์ได้ด้วย โดยสามารถทำการคำนวณตัวเลขขนาด 16 บิต ในขณะที่ CALU จำนวนตัวเลขขนาด 32 บิต

3.2 หน่วยคำนวณทางคณิตศาสตร์และลอจิกกลาง: CALU

ประกอบด้วยวงจรรถเคลื่อนบิตขนาด 16 บิต, วงจรคูณเลขขนาด 16 x 16 บิต และ ALU ขนาด 32 บิต โดยมีแอดคิมูเลเตอร์รับข้อมูลขนาด 32 บิต วงจรเคลื่อนบิตจะทำการเคลื่อนบิตไปทางซ้ายได้ตั้งแต่ 0 - 16 บิต โดยโปรแกรมได้จากคำสั่ง เมื่อเคลื่อนบิตแล้วบิตทางขวามือจะได้รับการเติม 0 ให้

ALU ขนาด 32 บิต จะทำงานตามคำสั่งทางคณิตศาสตร์และลอจิก สิ่งเกตว่าการทำงานของ ALU นี้จะมีวงจรผ่านค่า เช่น ผ่านค่าการถ่วงบิตแล้วมาทำใน ALU (ดูไดอะแกรมรูปที่ 3.3) ซึ่งทำให้การทำงานแต่ละคำสั่งทำงานได้เร็วมาก โดยเฉพาะคำสั่งการคูณบวกสำหรับการประมวลผลสัญญาณดังที่กล่าวมาแล้ว



† TMS320C25 specific.

รูปที่ 3.3 โครงสร้างของ ARAU

3.3 วัจนกรรม

TMS320C25 มีการคูณตัวเลขขนาด 16 x 16 บิต ซึ่งให้เอาต์พุตตัวเลขเป็น 32 บิต การคูณจะนำข้อมูลจาก TR(Temporary Register) มาคูณ ผลลัพธ์จะได้ตัวเลข 32 บิตเก็บไว้ที่ PR(Product Register) เอาต์พุตจากทริกเกอร์นี้จะสามารถ ส่งต่อไปทำอย่างอื่นอีก เช่น การเลื่อนบิตหรือการปิดจุดทศนิยม นอกจากนี้ยังทำการเลื่อนบิตไปทางขวา 6 บิต หรือกระทำในรูปแบบทางคณิตศาสตร์อื่นได้

การคูณทำได้ทั้งแบบคิดเครื่องหมายและไม่คิดเครื่องหมาย คำสั่งที่น่าสนใจเช่น MAC-multiply/accumulate เป็นคำสั่งที่คูณแล้วบวกสะสม หรือคำสั่ง MACD-multiply/accumulate and Data move ซึ่งเป็นการใช้คำสั่งเดียวแต่ทำงานได้หลายอย่างและทำได้รวดเร็ว เพราะโครงสร้างเป็นแบบ pipelining ทั้งสองคำสั่งนี้จะใช้โอเปอร์เรนด์สองตัว โดยอาจจะนำมาจากโปรแกรมหรือจากข้อมูลก็ได้

3.4 การควบคุมการทำงาน

TMS320C25 มีส่วนการควบคุมการทำงานที่สำคัญภายใน ซึ่งได้แก่วงจรตั้งเวลาวงจรนับทำให้เกิดการทำงาน มาสเคเบิลอินเตอร์รัพจากภายนอก 3 ตัว และอินเตอร์รัพภายในซึ่งเกิดจากพอร์ตอนุกรมหรือวงจรตั้งเวลา

วงจรตั้งเวลาประกอบด้วยรีจิสเตอร์วงจรนับขนาด 16 บิต ซึ่งเป็นวงจรนับลง (down counter) การทำงานจะทำการนับสัญญาณนาฬิกาจาก CLKOUT1 และเมื่อวงจรนับเวลาลดค่ารีจิสเตอร์ลงมาเหลือ 0 ก็จะทำให้เกิดการอินเตอร์รัพ (TINT) และจากนั้นวงจรตั้งเวลาก็จะโหลดค่าจาก PRD เข้ายังรีจิสเตอร์วงจรนับใหม่เป็นไซเคิลต่อไป ดังนั้นช่วงเวลาที่เกิดอินเตอร์รัพแต่ละครั้งจึงเท่ากับ $(PRD+1)*CLKOUT$ ซึ่งทำให้เกิดประโยชน์ในการกำหนดการซิงโครไนซ์กับอุปกรณ์ภายนอก

ค่าของรีจิสเตอร์ RPTC (repeat counter) เป็นตัวกำหนดจำนวนวงรอบของการทำงานของคำสั่ง ซึ่งนำค่ามาจากหน่วยความจำข้อมูลหรือจากคำสั่งโดยตรง การกำหนด RPTC จะทำให้คำสั่งได้รับการทำซ้ำเป็นจำนวนครั้งเท่ากับค่าที่โหลดไว้ ซึ่งจะทำการซ้ำได้ถึง 256 ครั้ง การอินเตอร์รัพที่ขั้วจากภายนอกจะมีขาอินเตอร์รัพ 3 ขา คือ INTO ถึง INT2

3.5 พอร์ตอนุกรม

พอร์ตอนุกรมเป็นพอร์ตที่ใช้เชื่อมโยงกับโลกภายนอกโดยการใช้ข้อมูลแบบอนุกรม การเชื่อมโยงข้อมูลนี้อาจจะเชื่อมต่อกับตัวอุปกรณ์ A/D และ D/A โดยวงจรเชื่อมต่อกับภายนอกจะต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นฮาร์ดแวร์ที่รับสัญญาณนี้ได้ ซึ่งภายในชิพมีรีจิสเตอร์ 2 ตัวที่ทำหน้าที่เก็บข้อมูล โดยรีจิสเตอร์ตัวหนึ่งเป็นรีจิสเตอร์ส่งข้อมูล อีกตัวหนึ่งเป็นรีจิสเตอร์รับข้อมูล การรับข้อมูลจะรับข้อมูลแบบ 8 บิตหรือ 16 บิตก็ได้ โดยแยกโหมดการรับส่งเป็นแบบไบต์หรือเวิร์ด การส่งรับข้อมูลนี้ได้รับการสร้างสัญญาณซิงค์จากภายในซึ่งสามารถรับส่งสัญญาณข้อมูลได้สูงถึง 5 MHz

ข้อปรับปรุงพอร์ตคอนโทรลของ TMS20C25 ได้แก่การสร้างบัฟเฟอร์ของตัวรับและตัวส่งข้อมูล และ CLK $\bar{?}$ CLKX คือ สัญญาณนาฬิกาสำหรับการรับส่งข้อมูล สามารถใช้สัญญาณนาฬิกาได้ตั้งแต่ 0 Hz เป็นต้นไป นอกจากนี้ยังเพิ่ม FSM-frame sync mode เพื่อให้ใช้กับระบบโทรศัพทแบบ PCM ตามมาตรฐาน AT&T T-1 และ CCITT G.711/712 จึงทำให้การประยุกต์ TMS320C25 ในระบบโทรศัพททำได้ง่าย

3.6 การเชื่อมต่อกับอินพุต-เอาต์พุต

TMS320C25 มีอินพุต-เอาต์พุตพอร์ตต่ออย่างละ 16 พอร์ต แต่ละพอร์ตเป็นพอร์ตแบบขนาน 16 บิต โดยเชื่อมต่อกับบัสข้อมูล คำสั่งที่เกี่ยวข้องกับ IN กับ OUT จะใช้เวลา 2 ไชเคิล แต่ถ้าให้ทำงานแบบเข้าโดยกำหนดค่าเข้าไปในรีจิสเตอร์ทำเข้าแล้ว คำสั่ง IN หรือ OUT นี้จะใช้เวลาได้เพียงไชเคิลเดียว การเชื่อมโยงทาง I/O จะแยกออกจากระบบเพราะไมโครโปรเซสเซอร์ DSP นี้มีช่องทางสำหรับโปรแกรมข้อมูลและ I/O นอกจากนี้ยังสามารถมอง I/O เหมือนเป็นหน่วยความจำได้

บนตัวไอซีจะประกอบด้วย DO-D15 เป็นบัสข้อมูล A0-A15 เป็นบัสของแอดเดรส และยังประกอบด้วยขาไอซี 3 ขา เพื่อเลือกว่าเป็นโปรแกรม, ข้อมูลหรืออินพุตเอาต์พุต (DS, PS และ IS) การกำหนดทิศทางจะใช้ R/W และมีสัญญาณ STRB เป็นตัวควบคุมการรับส่งข้อมูล

TMS320C25 มีขา HOLD เพื่อใช้ทำ DMA โดยเมื่อนานแอดคัพจะทำให้สถานะของบัสแอดเดรส บัสข้อมูล และสายสัญญาณควบคุม อยู่ในสภาวะอิมพีแดนซ์สูง เพื่อให้อุปกรณ์ภายนอกติดต่อกับ ROM และ RAM ต่อไป

INTERRUPT NAME	MEMORY LOCATION	PRIORITY	FUNCTION
RS	0	1 (highest)	External reset signal
INT0	2	2	External user interrupt #0
INT1	4	3	External user interrupt #1
INT2	6	4	External user interrupt #2
	8-23		Reserved locations
TINT	24	5	Internal timer interrupt
RINT	26	6	Serial port receive interrupt
XINT	28	7 (lowest)	Serial port transmit interrupt
TRAP	30	N/A	TRAP instruction address

ตารางที่ 3.1 Interrupt Locations และ Priorities

3.7 INTERRUPT

การจัดการ interrupt ของ TMS320C25 แสดงได้ดังตารางที่ 3.1 ซึ่งเป็นการ interrupt ทาง hardware ทั้งหมดยกเว้น TRAP เป็นการ interrupt ทาง software แต่ละ interrupt address จะใช้เนื้อที่ 2 address

3.8 โหมดการอ้างอิงแอดเดรสของ TMS320C25

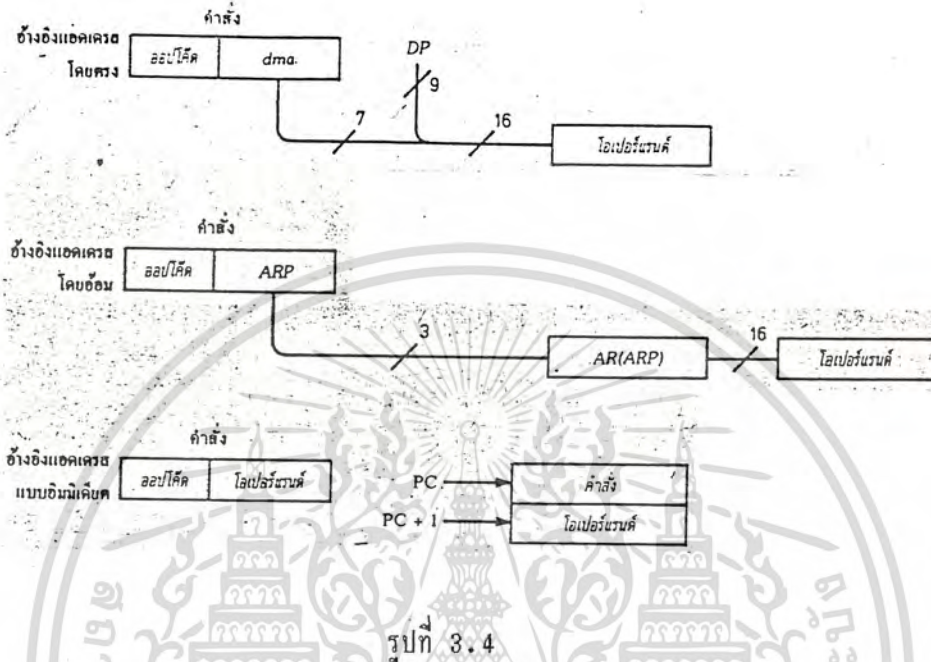
คำสั่งส่วนใหญ่ของ TMS320C25 เป็นคำสั่งที่ใช้รหัส 16 บิต เพื่อว่าการเพชท์และเอ็กซ์คิวจะกระทำได้เพียงใช้เคิลแอดเดรส 8 บิต การอ้างอิงแอดเดรสจะกระทำด้วยโหมดสำคัญ 3 โหมดดังรูปที่ 3.4 คือแบบอ้างอิงแอดเดรสโดยตรง (direct) แบบอ้างอิงแอดเดรสโดยอ้อม (indirect) และแบบอิมมิเตียต การอ้างอิงแอดเดรสโดยตรงและโดยอ้อมเป็นการต้องการติดต่อกับหน่วยความจำ ส่วนการอ้างอิงแอดเดรสแบบอิมมิเตียตเป็นการนำเอาโอเปอร์แรนด์มาใช้โดยตรง

การอ้างอิงแอดเดรสโดยตรงจะใช้คำสั่ง 16 บิต โดยแยกออกเป็นออปโค้ด 9 บิตและอีก 7 บิต เป็นโอเปอร์แรนด์ ในการทำงานจะใช้ 7 บิตร่วมกับ 9 บิตจากเพชพอยน์เตอร์ (DP) เพื่อรวมให้เป็น 16 บิต สำหรับอ้างอิงหน่วยความจำ 64k ดังนั้นการจัดหน่วยความจำจึงแบ่งเป็นเพจ ละ 128 เวิร์ดและมีจำนวน 512 เพจ

การอ้างอิงแอดเดรสแบบทางอ้อมจะอ้างผ่านรีจิสเตอร์ $AR_0 - AR_7$ รีจิสเตอร์เหล่านี้เป็นรีจิสเตอร์ตัวนับรูป หรือสำหรับเก็บข้อมูลชั่วคราวก็ได้ รูปที่ 3.5 เป็นการใช้อ้างอิงแอดเดรสไปยังหน่วยความจำ การกำหนดรีจิสเตอร์ทำได้โดยการใช้อ้างอิงแอดเดรส (Auxiliary Register Pointer) ARP ขนาด 3 บิต ซึ่งมีค่าได้จาก 0-7 เพื่อแทน $AR_0 - AR_7$ ค่าใน ARP นี้สามารถไหลเข้าโดยตรงจากข้อมูลในหน่วยความจำ หรือจากคำสั่งที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีการกำหนดค่าแบบอิมมิตีตโอเปอร์เรนด์ และเราสามารถข้อมูลจากรีจิสเตอร์ช่วยเหล่านี้อีก
กลับเข้าหน่วยความจำได้เช่นกัน



จากตารางที่ 3.2 จะเห็นว่ามีการอ้างแอดเดรสแบบทางอ้อม 7 แบบ คือ

- ∇ การอินเด็กซ์ด้วยการเพิ่มค่า
- ∇ การอินเด็กซ์ด้วยการลดค่า
- ∇ การอินเด็กซ์โดยการบวกข้อมูลของ AR₀
- ∇ การอินเด็กซ์โดยการลบข้อมูลของ AR₀
- ∇ การอินเด็กซ์โดยการบวกข้อมูลของ AR₀ สำหรับการทำให้ bit-reversing ใน

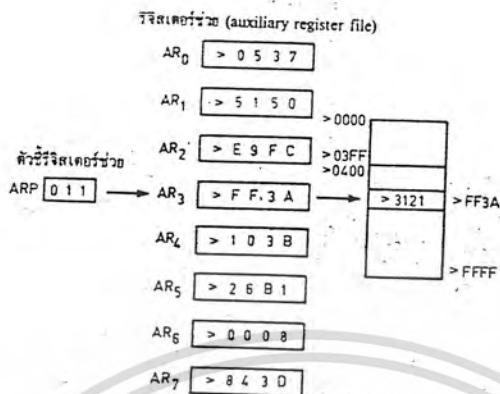
algorithm FFT

- ∇ การอินเด็กซ์โดยการลบข้อมูลของ AR₀ สำหรับการทำให้ bit-reversing ใน

Algorithm FFT

- ∇ ไม่มีการอินเด็กซ์

การอ้างอิงแอดเดรสแบบอิมมิตีตจะมีส่วนของโอเปอร์เรนด์ประกอบอยู่ในคำสั่ง โดยค่าตัวเลขนี้จะเป็นไปได้ 2 ขนาด คือ แบบ 8 บิตหรือแบบ 13 บิต และสำหรับคำสั่งแบบที่มีขนาด 2 เวิร์ดจะมีขนาดของตัวเลขโอเปอร์เรนด์ที่ใช้งานขนาด 16 บิต



รูปที่ 3.5

โหมดแอดเดรส	ลักษณะการช่วย	การทำงาน
OP A	แอดเดรสโดยตรง	
OP • (,NARP)	แอดเดรสโดยทางอ้อม	ไม่มีการเปลี่ยนค่าใน AR
OP • + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR เดิมเพิ่มขึ้น I
OP • - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR เดิมลดค่าลง I
OP • 0 + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR ₀ บวกกับค่าของ AR เดิม
OP • 0 - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR ₀ ลบกับค่าของ AR เดิม
OP • BRO + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR ₀ บวกกับค่าของ AR เดิม โดยมีทำให้บิตทศกระจายไว้
OP • BRO - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR ₀ ลบจากค่าของ AR เดิม โดยมีทำให้บิตทศกระจายไว้

ตารางที่ 3.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8 ชุดคำสั่งของ TMS320C25

สำหรับ TMS320C25 นี้ ได้ปรับปรุงโครงสร้างทางสถาปัตยกรรมเพิ่มขึ้นอีก ดังนั้นจึงมีชุดคำสั่งที่เพิ่มเติมขึ้นอีกหลายคำสั่งและบางคำสั่งก็มียุติความสามารถสูงขึ้น สัญลักษณ์ที่ใช้แทนคำสั่งมีดังนี้

- ACC - แอคคิวมูเลเตอร์
- ARB - รีจิสเตอร์ช่วยซีบิฟเฟอร์
- AR_n - รีจิสเตอร์ช่วย (auxiliary register) ตัวที่ n
- ARP - ตัวชี้รีจิสเตอร์ช่วย (auxiliary register pointer)
- BIO - Branch control input
- C - บิตตัวกต
- CM - 2 บิตที่ใช้สำหรับการกำหนดโหมดการเปรียบเทียบ (compare mode)
- CNF - บิตที่ใช้สำหรับการควบคุม RAM บนชิพ (configuration control bit)
- dma - แอดเดรสหน่วยความจำข้อมูล
- DP - พอยเตอร์บอกเพจข้อมูล (data page pointer)
- FO - บิตบอกสถานะฟอร์แมต (format status bit)
- FSM - บิตบอกโหมดเฟรมซิงโครไนซ์
- HM - บิตบอกโหมดการ hold
- INTM - บิตแฟล็กโหมดอินเตอร์รัพท์
- >nn - บอกว่า nn คือ เลขฐาน 16
- OV - แฟล็กโอเวอร์โฟลว์
- OVM - บิตโหมดโอเวอร์โฟลว์
- P - รีจิสเตอร์ผลคูณ
- PA - แอดเดรสพอร์ต โดยใช้ PAO หมายถึง พอร์ต 0
- PC - โปรแกรมเคาท์เนอร์
- PM - 2 บิตที่บอก P รีจิสเตอร์ให้เอาท์พุตเลื่อนบิต
- pma - แอดเดรสหน่วยความจำโปรแกรม
- Preg - รีจิสเตอร์ผลคูณ
- RPTC - เคาท์เนอร์นับทำซ้ำ
- ST_n - รีจิสเตอร์สถานะ (ST0 หรือ ST1)
- SXM - บิตบอกโหมด sign extension

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- T - รีจิสเตอร์ชั่วคราว
 TC - บิตทดสอบการควบคุม (test control register)
 TOS - จุดสูงสุดของสแตค (top of stack)
 Treg - รีจิสเตอร์ชั่วคราว
 TXM - บิตบอกโหมดส่งข้อมูลอนุกรม
 Usgn - ค่าที่ไม่คิดเครื่องหมาย
 XF - บิตบอกสถานะของขา XF

ACCUMULATOR MEMORY REFERENCE INSTRUCTIONS			
Mnemonic and Description		Words	16-Bit Opcode
			MSB LSB
ABS	Absolute value of accumulator	1	1100 1110 0001 1011
ADD	Add to accumulator with shift	1	0000 SSSS I DDD DDDD
ADDC†	Add to accumulator with carry	1	0100 0011 I DDD DDDD
ADDH	Add to high accumulator	1	0100 1000 I DDD DDDD
ADDK†	Add to accumulator short immediate	1	1100 1100 KKKK KKKK
ADDS	Add to low accumulator with sign-extension suppressed	1	0100 1001 I DDD DDDD
ADDT†	Add to accumulator with shift specified by T register	1	0100 1010 I DDD DDDD
ADLKT	Add to accumulator long immediate with shift	2	1101 SSSS 0000 0010
AND	AND with accumulator	1	0100 1110 I DDD DDDD
ANDKT	AND immediate with accumulator with shift	2	1101 SSSS 0000 0100
CPLT	Complement accumulator	1	1100 1110 0010 0111
LAC	Load accumulator with shift	1	0010 SSSS I DDD DDDD
LACK	Load accumulator short immediate	1	1100 1010 KKKK KKKK
LACT†	Load accumulator with shift specified by T register	1	0100 0010 I DDD DDDD
LALK†	Load accumulator long immediate with shift	2	1101 SSSS 0000 0001
NEG†	Negate accumulator	1	1100 1110 0010 0011
NORM†	Normalize contents of accumulator	1	1100 1110 1010 0010
OR	OR with accumulator	1	0100 1101 I DDD DDDD
ORKT	OR immediate with accumulator with shift	2	1101 SSSS 0000 0101
ROL†	Rotate accumulator left	1	1100 1110 0011 0100
ROR†	Rotate accumulator right	1	1100 1110 0011 0101
SACH	Store high accumulator with shift	1	0110 1XXX I DDD DDDD
SACL	Store low accumulator with shift	1	0110 0XXX I DDD DDDD
SBLKT	Subtract from accumulator long immediate with shift	2	1101 SSSS 0000 0011
SFLT	Shift accumulator left	1	1100 1110 0001 1000
SFRT	Shift accumulator right	1	1100 1110 0001 1001
SUB	Subtract from accumulator with shift	1	0001 SSSS I DDD DDDD
SUBB†	Ssubtract from accumulator with borrow	1	0100 1111 I DDD DDDD
SUBC	Conditional subtract	1	0100 0111 I DDD DDDD
SUBH	Subtract from high accumulator	1	0100 0100 I DDD DDDD
SUBK†	Ssubtract from accumulator short immediate	1	1100 1101 KKKK KKKK
SUBS	Subtract from low accumulator with sign extension suppressed	1	0100 0101 I DDD DDDD
SUBT†	Subtract from accumulator with shift specified by T register	1	0100 0110 I DDD DDDD
XOR	Exclusive-OR with accumulator	1	0100 1100 I DDD DDDD
XORK†	Exclusive-OR immediate with accumulator with shift	2	1101 SSSS 0000 0110
ZAC	Zero accumulator	1	1100 1010 0000 0000
ZALH	Zero low accumulator and load high accumulator	1	0100 0000 I DDD DDDD
ZALR†	Zero low accumulator and load high accumulator with rounding	1	0111 1011 I DDD DDDD
ZALS	Zero accumulator and load low accumulator with sign extension suppressed	1	0100 0001 I DDD DDDD

†These instructions are specific to the TMS320C2x instruction set.

‡These instructions are specific to the TMS320C25 instruction set.

ตารางที่ 3.3 ชุดคำสั่งของ TMS320C25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CONTROL INSTRUCTIONS				
Mnemonic and Description		Words	16-Bit Opcode	
			MSB	LSB
BIT†	Test bit	1	1001	BBBB I DDD DDDD
BITT	Test bit specified by T register	1	0101	0111 I DDD DDDD
CNFDT	Configure block as data memory	1	1100	1110 0000 0100
CNFPT	Configure block as program memory	1	1100	1110 0000 0101
DINT	Disable interrupt	1	1100	1110 0000 0001
EINT	Enable interrupt	1	1100	1110 0000 0000
IDLET	Idle until interrupt	1	1100	1110 0000 1111
LST	Load status register ST0	1	0101	0000 I DDD DDDD
LST1†	Load status register ST1	1	0101	0001 I DDD DDDD
NOP	No operation	1	0101	0001 I DDD DDDD
POP	Pop top of stack to low accumulator	1	1100	1110 0000 0000
POPDT	Pop top of stack to data memory	1	1100	1110 0001 1101
PSHDT	Push data memory value onto stack	1	0111	1010 I DDD DDDD
PUSH	Push low accumulator onto stack	1	0101	0100 I DDD DDDD
RC†	Reset carry bit	1	1100	1110 0001 1100
RHM†	Reset hold mode	1	1100	1110 0011 0000
ROVM	Reset overflow mode	1	1100	1110 0011 1000
RPT†	Repeat instruction as specified by data memory value	1	1100	1110 0000 0010
RPTKT	Repeat instruction as specified by immediate value	1	1100	1011 KKKK KKKK
RSXMT	Reset sign-extension mode	1	1100	1110 0000 0110
RTC†	Reset test/control flag	1	1100	1110 0011 0010
SC†	Set carry bit	1	1100	1110 0011 0001
SHM†	Set hold mode	1	1100	1110 0011 1001
SOVM	Set overflow mode	1	1100	1110 0000 0011
SST	Store status register ST0	1	0111	1000 I DDD DDDD
SST1†	Store status register ST1	1	0111	1001 I DDD DDDD
SSXMT	Set sign-extension mode	1	1100	1110 0000 0111
STC†	Set test/control flag	1	1100	1110 0011 0011

Mnemonic and Description		Words	16-Bit Opcode	
			MSB	LSB
BLKDT	Block move from data memory to data memory	2	1111	1101 I DDD DDDD
BLKPT	Block move from program memory to data memory	2	1111	1100 I DDD DDDD
DMOV	Data move in data memory	1	0101	0110 I DDD DDDD
FORT†	Format serial port registers	1	1100	1110 0000 111K
IN	Input data from port	1	1000	AAAA I DDD DDDD
OUT	Output data to port	1	1110	AAAA I DDD DDDD
RFSM†	Reset serial port frame synchronization mode	1	1100	1110 0011 0110
RTXM†	Reset serial port transmit mode	1	1100	1110 0010 0000
RXFT	Reset external flag	1	1100	1110 0000 1100
SFSM†	Set serial port frame synchronization mode	1	1100	1110 0011 0111
STXM†	Set serial port transmit mode	1	1100	1110 0010 0001
SXFT	Set external flag	1	1100	1110 0000 1101
TBLR	Table read	1	0101	1000 I DDD DDDD
TBLW	Table write	1	0101	1001 I DDD DDDD

†These instructions are specific to the TMS320C2x instruction set.
 ‡These instructions are specific to the TMS320C25 instruction set.

ตารางที่ 3.3 ชุดคำสั่งของ TMS320C25 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BRANCH/CALL INSTRUCTIONS			
Mnemonic and Description	Words	16-Bit Opcode	
		MSB	LSB
B	Branch unconditionally	2	1111 1111 1DDD DDDD
BACCT	Branch to address specified by accumulator	1	1100 1110 0010 0101
BANZ	Branch on auxiliary register not zero	2	1111 1011 1DDD DDDD
BBNZT	Branch if TC bit \neq 0	2	1111 1001 1DDD DDDD
BBZT	Branch if TC bit = 0	2	1111 1000 1DDD DDDD
BCCT	Branch on carry	2	0101 1110 1DDD DDDD
BGEZ	Branch if accumulator \geq 0	2	1111 0100 1DDD DDDD
BGZ	Branch if accumulator $>$ 0	2	1111 0001 1DDD DDDD
BIOZ	Branch on I/O status = 0	2	1111 1010 1DDD DDDD
BLEZ	Branch if accumulator \leq C	2	1111 0010 1DDD DDDD
BLZ	Branch if accumulator $<$ 0	2	1111 0011 1DDD DDDD
BNCCT	Branch on no carry	2	0101 1111 1DDD DDDD
BNVT	Branch if no overflow	2	1111 0111 1DDD DDDD
BNZ	Branch if accumulator \neq 0	2	1111 0101 1DDD DDDD
BV	Branch on overflow	2	1111 0000 1DDD DDDD
BZ	Branch if accumulator = 0	2	1111 0110 1DDD DDDD
CALA	Call subroutine indirect	1	1100 1110 0010 0100
CALL	Call subroutine	2	1111 1110 1DDD DDDD
RET	Return from subroutine	1	1100 1110 0010 0110
TRAPT	Software interrupt	1	1100 1110 0001 1110

AUXILIARY REGISTERS AND DATA PAGE POINTER INSTRUCTIONS			
Mnemonic and Description	Words	16-Bit Opcode	
		MSB	LSB
ADRKCT	Add to auxiliary register short immediate	1	0111 1110 KKKK KKKK
CMPT	Compare auxiliary register with auxiliary register ARO	1	1100 1110 0101 00KK
LAR	Load auxiliary register	1	0011 0RRR 1DDD DDDD
LARK	Load auxiliary register short immediate	1	1100 0RRR KKKK KKKK
LARP	Load auxiliary register pointer	1	0101 0101 1000 1RRR
LDP	Load data memory page pointer	1	0101 0010 1DDD DDDD
LDPK	Load data memory page pointer immediate	1	1100 100K KKKK KKKK
LRLKCT	Load auxiliary register long immediate	2	1101 0RRR 0000 0000
MAR	Modify auxiliary register	1	0101 0101 1DDD DDDD
SAR	Store auxiliary register	1	0111 0RRR 1DDD DDDD
SBRKCT	Subtract from auxiliary register short immediate	1	0111 1111 KKKK KKKK

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS			
Mnemonic and Description	Words	16-Bit Opcode	
		MSB	LSB
APAC	Add P register to accumulator	1	1100 1110 0001 0101
LPHT	Load high P register	1	0101 0011 1DDD DDDD
LT	Load T register	1	0011 1100 1DDD DDDD
LTA	Load T register and accumulate previous product	1	0011 1101 1DDD DDDD
LTD	Load T register, accumulate previous product, and move data	1	0011 1111 1DDD DDDD
LTPCT	Load T register and store P register in accumulator	1	0011 1110 1DDD DDDD
LTST	Load T register and subtract previous product	1	0101 1011 1DDD DDDD
MACT	Multiply and accumulate	2	0101 1101 1DDD DDDD
MACDCT	Multiply and accumulate with data move	2	0101 1100 1DDD DDDD
MPY	Multiply (with T register, store product in P register)	1	0011 1000 1DDD DDDD
MPYACT	Multiply and accumulate previous product	1	0011 1010 1DDD DDDD
MPYK	Multiply immediate	1	101K KKKK KKKK KKKK
MPYST	Multiply and subtract previous product	1	0011 1011 1DDD DDDD
MPYUCT	Multiply unsigned	1	1100 1111 1DDD DDDD
PAC	Load accumulator with P register	1	1100 1110 0001 0100
SPAC	Subtract P register from accumulator	1	1100 1110 0001 0110
SPHT	Store high P register	1	0111 1101 1DDD DDDD
SPLCT	Store low P register	1	0111 1100 1DDD DDDD
SPMCT	Set P register output shift mode	1	1100 1110 0000 10KK
SQRACT	Square and accumulate	1	0011 1001 1DDD DDDD
SQRST	Square and subtract previous product	1	0101 1010 1DDD DDDD

†These instructions are specific to the TMS320C2x instruction set.

‡These instructions are specific to the TMS320C25 instruction set.

ตารางที่ 3.3 ชุดคำสั่งของ TMS320C25 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

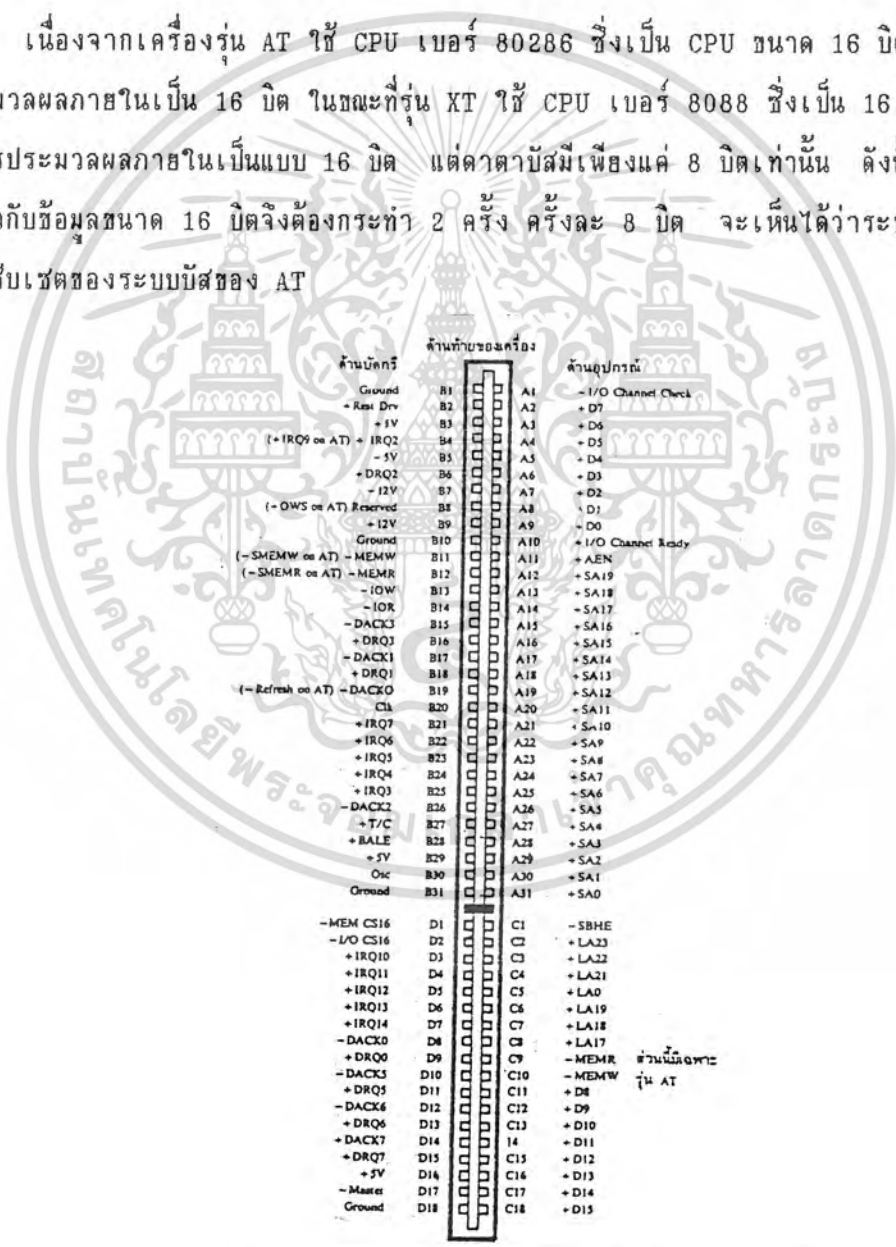
บทที่ 4

IBM PC กับการ์ดอินเตอร์เฟส

ในตอนนี้จะขอแนะนำส่วนต่างๆของ IBM PC ที่เกี่ยวข้องกับการ์ดอินเตอร์เฟส ซึ่งต้องต่อโดยตรงกับระบบบัสของ IBM PC ให้เข้าใจก่อน

4.1 AT SLOT และ XT SLOT ของ IBM PC

เนื่องจากเครื่องรุ่น AT ใช้ CPU เบอร์ 80286 ซึ่งเป็น CPU ขนาด 16 บิตแต่ก็มีการประมวลผลภายในเป็น 16 บิต ในขณะที่รุ่น XT ใช้ CPU เบอร์ 8088 ซึ่งเป็น 16 บิตเทียม คือมีการประมวลผลภายในเป็นแบบ 16 บิต แต่ดาตาบัสมีเพียงแค่ 8 บิตเท่านั้น ดังนั้นการจัดการเกี่ยวกับข้อมูลขนาด 16 บิตจึงต้องกระทำ 2 ครั้ง ครั้งละ 8 บิต จะเห็นได้ว่าระบบบัสของ XT เป็นซิปเซตของระบบบัสของ AT



รูปที่ 4.1 แสดงตำแหน่งของสัญญาณบนสล๊อต IBM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บนเมนบอร์ดของ AT จะมีสล๊อตอยู่ 2 ชนิด คือสล๊อตสั้นและสล๊อตยาว ซึ่งสล๊อตยาวจะมีจำนวนขาสัญญาณและตำแหน่งของขาสัญญาณบนสล๊อตเหมือนกับ XT ส่วนขาสัญญาณที่อยู่บนสล๊อตสั้นที่เพิ่มขึ้น จะเป็นสัญญาณที่มีแต่เฉพาะบน AT เท่านั้น ประกอบด้วยข้อมูลครึ่งบน 8 บิต แอดเดรสที่เพิ่มขึ้นมาอีก 4 บิต และขาสัญญาณควบคุมที่เพิ่มขึ้นจาก XT สำหรับรูปของสล๊อตและตำแหน่งของสัญญาณแสดงไว้ในรูปที่ 4.1

สำหรับเครื่องคอมพิวเตอร์ที่ใช้ CPU สูงกว่า 80286 นั้น เช่น 80386 ที่เป็น CPU ขนาด 32 บิต ก็ยังคงใช้สล๊อตแบบ AT อยู่ ซึ่งมีการทำงานเหมือนเดิมในรุ่นก่อนๆ เช่นกัน

4.2 รายละเอียดของสัญญาณต่าง ๆ บนสล๊อต

(I), (O) และ (I/O) หมายถึง ทิศทางของขาสัญญาณเมื่อเทียบกับเมนบอร์ด โดยที่

(I) หมายถึง ขาสัญญาณอินพุต

(O) หมายถึง ขาสัญญาณเอาต์พุต

(I/O) หมายถึง ขาสัญญาณที่เป็นได้ทั้งอินพุตและเอาต์พุต

(*I/O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณเอาต์พุต แต่จะเป็นอินพุตในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้าจะหมายถึงขาสัญญาณที่แอดคัพที่ลอจิก "0" และขาสัญญาณที่ไม่มี หรือมีเครื่องหมายบวกนำหน้าจะหมายถึง ขาสัญญาณที่แอดคัพที่ลอจิก "1" สัญญาณที่อยู่บนสล๊อตนั้นสามารถขับไอซีที่ทอลล์ชนิดโลว์เพาเวอร์ได้สองตัว โดยที่ไม่ทำให้เกิดการโหลดหรือการเพี้ยนของสัญญาณ ขาสัญญาณต่าง ๆ บนสล๊อตของ XT และ AT สามารถแบ่งออกเป็นกลุ่ม ๆ ได้ดังนี้

เพาเวอร์ซัพพลาย

Ground	ขาสัญญาณนี้ต่ออยู่กับกราวด์ของระบบเรกูเลเตอร์
+5 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ +5 V
-5 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ -5 V
+12 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ +12 V
-12 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC เรกูเลเตอร์ -12 V

แอดเดรสบัส และ สัญญาณต่าง ๆ ที่เกี่ยวข้อง

SA0-SA19	เป็นแอดเดรสบิตที่ 0 ถึง 19 โดยที่ SA0 มีนัยสำคัญต่ำสุด ขาสัญญาณนี้จะแอดคัพ
(*I/O)	เมื่อขาสัญญาณ BALE มีสถานะเป็น "1" และจะถูกแลตซ์ไว้ตอนขอบขาลงของขา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณ BALE แอดเดรสทั้ง 20 บิตนี้ สามารถอ้างหน่วยความจำได้ถึง 1 เมกะไบต์ XT สำหรับ AT เมื่อใช้ร่วมกับ LA17-LA23 จะอ้างได้ถึง 16 เมกะไบต์ LA17-LA23 (เฉพาะรุ่น AT) ขาสัญญาณนี้จะแอกตีฟเมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก (*I/O) "1" แต่จะไม่มีแลตช์ไว้ตอนขอบขาลงของสัญญาณ BALE ดังนั้นถ้าอุปกรณ์ I/O ไม่มีการอ้างแอดเดรสเกิน 1 เมกะไบต์ ขาสัญญาณนี้ก็ไม่ใช่บังคับใช้ แต่ถ้ามีการอ้างแอดเดรสเกิน อุปกรณ์ I/O จะต้องทำการแลตช์สัญญาณนี้ โดยใช้ขอบขาลงของสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR

AEN (Address Enable) ขาสัญญาณนี้จะแอกตีฟเมื่อตัวควบคุม DMA ได้ทำการควบคุม บัสต่าง ๆ ของระบบแล้ว ดังนั้นการอ้างพอร์ตของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ ในการดีโค๊ดด้วย เพื่อที่จะไม่ทำให้เกิดการติดต่อกันระหว่างระบบกับอุปกรณ์ I/O ตัว อื่นยกเว้นตัวที่กำลังทำขบวนการ DMA อยู่

BALE (Address Latch Enable) ขาสัญญาณนี้ใช้ในการแสดงการเริ่มต้นของขบวนการ ต่าง ๆ ที่มีการติดต่อกับหน่วยความจำ โดยจะแอกตีฟเมื่อค่าแอดเดรสที่ CPU ต้องการติดต่อดำเนินแอดเดรสบัสเรียบร้อยแล้ว ตามปกติขอบขาลงของสัญญาณนี้จะ ทำให้เกิดการแลตช์สัญญาณ SA0-SA19 และถ้ามีการอ้างแอดเดรสเกิน 1 เมกะไบต์ใน AT จะใช้ขอบขาลงของสัญญาณนี้ในการแลตช์สัญญาณ LA17-LA23 ด้วยเช่นกัน แต่สำหรับในขบวนการ DMA สัญญาณนี้จะมีสถานะเป็น "1" ตลอด

SBHE (เฉพาะรุ่น AT) (Bus High Enable) เป็นขาสัญญาณที่ใช้แสดงว่ามีการรับส่ง ข้อมูลในบิตที่ SD8-SD15

คาตาบัส

SD0-SD7 สำหรับรุ่น AT จะมี SD8-SD15 เพิ่มขึ้นมาด้วย คือ คาตาบิต 0 ถึง 7 สำหรับรุ่น (I/O) XT และสำหรับรุ่น AT คือ คาตาบิต 0 ถึง 15 โดยที่ SDO มีนัยสำคัญต่ำสุด สำหรับ AT ถ้ามีการติดต่อกับบิตที่ SD8-SD15 สามารถตรวจสอบได้จากขาสัญญาณ SBHE

สัญญาณอินเทอร์รัพต์

IRQ2-IRQ7 (Interrupt Request) (สำหรับรุ่น AT จะเป็น IRQ3-7, 9-12, 14, 15) เป็น ขาสัญญาณอินเทอร์รัพต์ CPU สำหรับ AT ลำดับความสำคัญของสัญญาณอินเทอร์ IRQ เป็นดังนี้ คือ 9, 10, 11, 12, 14, 15, 3, 4, 5, 6 และ 7 โดย IRQ9 มีลำดับความ สำคัญน้อยที่สุด สำหรับ XT IRQ2 จะมีลำดับความสำคัญมากที่สุด รองๆ ลงไป คือ IRQ3, 4, 5, 6, 7 โดยปกติสัญญาณนี้จะมีสถานะเป็น "0" เสมอ ถ้าต้องการอินเทอร์

รีเฟรช CPU ให้ส่งพัลส์ที่เป็นลอจิก "1" ให้กับมัน โดยไม่จำเป็นต้องคำนึงถึงคาบเวลาของพัลส์ ทั้งนี้เพราะระบบของ IBM ตัวอินเทอร์รัพต์คอนโทรลเลอร์ (8259 Interrupt controller) จะถูกโปรแกรมให้ตรวจสอบสัญญาณอินเทอร์รัพต์โดยใช้ขอบขาลงของสัญญาณนี้

-I/O CH CK (I/O Channel Check) เป็นขาสัญญาณที่บอกถึงความผิดพลาดในการรับส่งข้อมูล ซึ่งตรวจสอบจากพาริตีบิต ถ้าพาริตีบิตที่อ่านจากหน่วยความจำกับพาริตีบิตที่สร้างขึ้นจากขบวนการรับส่งข้อมูล สัญญาณนี้จะทำให้เกิดการอินเทอร์รัพต์ CPU แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิด Parity Error ขึ้น CPU จะแสดงข้อความบอกความผิดพลาดขึ้นและจะหยุดการทำงาน (Halt) เพื่อให้ผู้ใช้ตรวจสอบหาสาเหตุของการผิดพลาด

สัญญาณที่ใช้ในขบวนการ DMA

DRQ1-DRQ3 (DMA Request) (สำหรับรุ่น AT จะเป็น DRQ0-3, 5-7) เป็นขาสัญญาณใช้ในการขอทำขบวนการ DMA โดยที่ DRQ0 มีลำดับความสำคัญมากที่สุด และ DRQ3 มีลำดับความสำคัญน้อยที่สุดสำหรับรุ่น XT และสำหรับรุ่น AT ขา DRQ7 จะมีลำดับความสำคัญน้อยที่สุด

-DACK03 (DMA Acknowledge) (สำหรับรุ่น AT จะเป็น -DACK0-3, 5-7) เป็นสัญญาณตอบสนองการขอทำ DMA ของอุปกรณ์ I/O เพื่อให้อุปกรณ์ I/O ทราบว่าการขอทำขบวนการ DMA นั้นได้รับการตอบสนองแล้ว เช่น ถ้ามีการขอทำ DMA ผ่านทาง DRQ2 และเมื่อ CPU รับรู้แล้ว จะทำให้สัญญาณ DACK2 แอคติฟ

Refresh (เฉพาะรุ่น AT) (Memory Refresh) มีหน้าที่เหมือนกับขาสัญญาณ DACK0 ในรุ่น XT คือ ใช้แสดงขบวนการรีเฟรชหน่วยความจำ เพราะว่าในรุ่น AT จะมีวงจรที่ใช้ในการรีเฟรชหน่วยความจำโดยตรงอยู่แล้ว ดังนั้นจึงไม่จำเป็นต้องใช้ขาสัญญาณ DRQ0 และ DACK0

-Master (เฉพาะรุ่น AT) (Master) ขาสัญญาณนี้จะใช้ร่วมกับ DMA Request ในการเข้าควบคุมระบบบัสในขบวนการ DMA โดยที่ตัว DMA คอนโทรลเลอร์จะส่งสัญญาณ DMA Request แล้วรอจนกระทั่งได้รับการตอบสนองโดยสัญญาณ DACK เกิดการแอคติฟขึ้น แล้วจึงจะส่งสัญญาณนี้ให้กับ CPU จะทำให้แอดเดรสบัส คาต้าบัส และคอนโทรลบัส เข้าสู่สถานะไตรสแตต หรือ ไฮอิมพีแดนซ์ หลังจากนั้นตัว DMA คอนโทรลเลอร์จะต้องรออีกหนึ่งคาบสัญญาณคล็อก ก่อนที่จะเข้าควบคุมบัสต่างๆ และจะต้องรออีก 2 ไชเคิล ก่อนที่จะทำการอ่านหรือเขียนข้อมูล ช่วงเวลาที่สัญญาณนี้

แอดค็พไม่ควรถูกเกิน 15 ไมโครวินาที มิฉะนั้นข้อมูลภายในหน่วยความจำจะสูญหายไป เนื่องจากขาดสัญญาณรีเฟรชหน่วยความจำ

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

สัญญาณควบคุมต่างๆ

-MEMR (Memory Read) [สำหรับรุ่น AT คือ ขาสัญญาณ -SMEMR (System Memory Read)] ขาสัญญาณนี้เป็นตัวบอกให้หน่วยความจำส่งข้อมูลออกมาที่ดาต้าบัส แต่สำหรับ AT สัญญาณ -SMEMR จะแอดค็พ เมื่อการอ่านข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 เมกกะไบต์แรกเท่านั้น

MEMR (เฉพาะรุ่น AT) (Memory Read) ขาสัญญาณนี้ไม่ใช่สัญญาณเดียวกันกับสัญญาณ -MEMR ใน XT มันจะแอดค็พในทุกๆขบวนการอ่านข้อมูลที่เกิดขึ้น ไม่ว่าจะอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์แรกหรือไม่

MEMW (Memory Write) [สำหรับรุ่น AT คือ ขาสัญญาณ -SMEW (System Memory Write)] ขาสัญญาณนี้เป็นตัวบอกให้หน่วยความจำเก็บข้อมูลจากดาต้าบัส แต่สำหรับ AT สัญญาณ -SMEW จะแอดค็พ เมื่อเกิดการเก็บข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 เมกกะไบต์แรกเท่านั้น

MEMW (เฉพาะรุ่น AT) (Memory Write) ขาสัญญาณนี้ไม่ใช่สัญญาณเดียวกันกับสัญญาณ -MEMW ใน XT มันจะแอดค็พในทุกๆขบวนการเก็บข้อมูลที่เกิดขึ้น ไม่ว่าจะอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์แรกหรือไม่

-IOR (I/O Read) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการส่งข้อมูลลงมาที่ดาต้าบัส

-IOW (I/O Write) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการเก็บข้อมูลจากดาต้าบัสเข้าไป

RESET DRV (Reset Driver) เป็นขาสัญญาณที่แอดค็พตอนที่เรารับจ่ายไฟให้กับระบบเพื่อใช้ในการรีเซ็ต CPU และ อุปกรณ์ต่างๆในระบบคอมพิวเตอร์ รวมทั้งอุปกรณ์ I/O ที่ต่ออยู่ด้วย

-MEM CS16 (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่า ต้องการรับส่งข้อมูลกับหน่วยความจำที่ละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือ ทำการรับส่งข้อมูลที่ละ 8 บิต สองครั้งเพื่อให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ในเชิงพาณิชย์ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้ข้อมูลขนาด 16 บิต

- I/O CS16 (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่าจะต้องการรับส่งข้อมูลกับอุปกรณ์ I/O ที่ละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือ ทำการรับส่งข้อมูลทีละ 8 บิต สองครั้งเพื่อให้ได้ข้อมูลขนาด 16 บิต

สัญญาณที่ใช้สร้าง Wait States

- I/O CH RDY (I/O Channel Ready) ขาสัญญาณนี้จะถูกทำให้แอกตีฟโดยอุปกรณ์ I/O หรือหน่วยความจำที่ไม่สามารถทำงานได้ทันกับระบบ ดังนั้น จะต้องทำการหน่วงระบบให้ทำงานช้าลง ด้วยการเพิ่ม Wait States โดยการทำให้สัญญาณนี้แอกตีฟในช่วงเวลาที่ I/O ได้รับสัญญาณจากการตีโค้ดแอกเดอเรส, สัญญาณ -MEMR, สัญญาณ -MRMW, สัญญาณ -IOR, สัญญาณ -IOW
- (เฉพาะรุ่น AT) (Zero Wait State) การแอกตีฟของขาสัญญาณนี้จะบังคับไม่ให้เกิดการสร้าง Wait States โดยอัตโนมัติ นั่นคือ การที่จะเกิด Wait States ขึ้นได้ จะต้องขึ้นอยู่กับสัญญาณนี้ เช่น การทำงานในขบวนการอ่านเขียนข้อมูลขนาด 16 บิต โดยไม่ใช้ Wait States ทำได้โดยการสร้างสัญญาณ OWS จากสัญญาณการตีโค้ดแอกเดอเรส และสัญญาณที่ใช้ในการอ่านเขียน หรือการลด Wait States ในขบวนการอ่านเขียนข้อมูลขนาด 8 บิต ให้เหลือเพียง 2 Wait States ทำได้โดยให้สัญญาณ OWS แอกตีฟหลังจากการอ่านหรือเขียนไปแล้ว 1 คล็อก โดยปกติการขับสัญญาณนี้ควรใช้เกตที่มีเอาท์พุทเป็นแบบ Open Collector ที่ทนกระแสได้ 20 mA (Sinking Current)

สัญญาณนาฬิกา

- CLK (System Clock) สำหรับ XT ขาสัญญาณนี้จะมีค่าประมาณ 4.77 MHz หรืออาจจะสูงกว่านี้ก็ได้สำหรับรุ่นใหม่ๆ และสำหรับ AT จะมีค่าประมาณ 6 MHz หรือในรุ่นใหม่ๆ อาจจะมีค่าสูงถึง 15 MHz
- โดยปกติ ขาสัญญาณนี้มีควิตซ์ไชเคิล 50% สำหรับ CPU เบอร์ 80286 ตัวกำเนิดสัญญาณนาฬิกาที่ป้อนให้จะมีความถี่เป็น 2 เท่าของความถี่ที่ CPU ทำงาน แต่ขาสัญญาณนี้จะยังคงมีความถี่เท่ากับความถี่ที่ CPU ทำงานอยู่เสมอ
- OSC (Oscillator) เป็นขาสัญญาณที่มีความถี่สูง คือ 14.31818 MHz ความถี่ของสัญญาณนี้จะคงที่เสมอ และจะไม่ซิงโครนัสกับสัญญาณอื่นๆ ในระบบ ดังนั้นจึงไม่ควรนำสัญญาณนี้ไปใช้ เป็นสัญญาณคล็อกของอุปกรณ์ I/O ที่ต่ออยู่กับระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การจัด Address ของ Memory และ Port

การควบคุมอุปกรณ์ I/O ที่ต่ออยู่กับ IBM PC จะกระทำผ่านพอร์ต โดยการอ้างถึงแอดเดรสของพอร์ตที่อุปกรณ์นั้นต่ออยู่โดยตรง ดังนั้นการที่จะใช้งานหรือควบคุมอุปกรณ์เหล่านี้จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ต ใน IBM PC พอร์ตและหน่วยความจำจะแยกจากกันโดยเด็ดขาด ถึงแม้ว่าการอ้างถึงจะใช้สัญญาณจากแอดเดรสบัสเหมือนกันก็ตาม แต่สัญญาณที่ใช้ในการอินพุตในการอ่านและเขียนข้อมูลจะต่างกัน ดังนั้นการติดต่อกับพอร์ตจึงมีคำสั่งแยกต่างหากออกจากคำสั่งที่ใช้ติดต่อกับหน่วยความจำ คือ IN และ OUT ด้วยเหตุนี้ การจัดแอดเดรสของหน่วยความจำ และแอดเดรสของพอร์ต I/O จึงแยกออกจากกัน ใน IBM PC การจัดแอดเดรสของหน่วยความจำ แสดงได้ในตารางที่ 4.1 และในตารางที่ 4.2 แสดงการจัดแอดเดรสของพอร์ต I/O

Block 0	00000-0FFFF	RAM to 64K
Block 1	10000-1FFFF	RAM to 128K
Block 2	20000-2FFFF	RAM to 192K
Block 3	30000-3FFFF	RAM to 256K
Block 4	40000-4FFFF	RAM to 320K
Block 5	50000-5FFFF	RAM to 384K
Block 6	60000-6FFFF	RAM to 448K
Block 7	70000-7FFFF	RAM to 512K
Block 8	80000-8FFFF	RAM to 576K
Block 9	90000-9FFFF	RAM to 640K
Block A	A0000-AFFFF	Extended video memory
Block B	B0000-BFFFF	Standard video memory
Block C	C0000-CFFFF	BIOS extension (eg EGA)
Block D	D0000-DFFFF	Other use
Block E	E0000-EFFFF	Other use
Block F	F0000-FFFFF	BIOS EPROM

ตารางที่ 4.1 แสดงการจัดแอดเดรสของหน่วยความจำบน IBM

การอ้างแอดเดรสของหน่วยความจำ จะใช้แอดเดรสทั้งหมด 20 เส้น คือ A0-A19 ในรุ่น XT (8088) แต่ในรุ่น AT (80286) ใช้แอดเดรส 24 เส้น คือ A0-A23 การอ้างแอดเดรสของพอร์ตสำหรับ CPU เบอร์ 8088 และ 80286 สามารถอ้างได้ถึง 64 k พอร์ต แต่ใน IBM PC ทั้งในรุ่น AT และ XT ออกแบบให้ใช้แอดเดรสเพียง 10 เส้น เท่านั้น คือ A0-A9 ดังนั้นจำนวนพอร์ตสูงสุดที่สามารถอ้างได้คือ 1024 พอร์ต ในจำนวนทั้งหมดนี้ ยังแบ่งออกเป็น 2 กลุ่มคือกลุ่มพอร์ตที่มีแอดเดรสอยู่ในช่วง 000H-0FFH จะใช้บนเมนบอร์ดสำหรับชิพพอร์ทัลเท่านั้น เช่น 8259 (Interrupt Controller) , 8257 (DMA Controller) , 8253 (Timer & Counter) และกลุ่มที่มีแอดเดรสอยู่ในช่วง 100H-3FFH จะใช้งานกับการ์ดขยายต่างๆที่เสียบในสล롯

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Address	Description	Note
1F0-1F8	Fixed disk	1
200-20F	Games adapter	
210-217	Expansion unit	2
278-27F	2nd parallel printer port	1
2B0-2DF	Alternate EGA	
2F8-2FF	2nd serial port	
2E1	GPIB (0)	4
2E2-2E3	Data Acquisition (0)	4
300-31F	Prototype card	
320-32F	Fixed disk	2
360-36F	PC Network	
378-37F	1st parallel printer port	
380-38F	SDLC/2nd Bisynchronous	3
390-393	Cluster (0)	4
3A0-3AF	1st Bisynchronous	1
3B0-3BF	Monochrome display/printer	
3C0-3CF	EGA	
3D0-3DF	CGA	
3F0-3F7	Floppy disk	
3F8-3FF	1st serial port	
Notes:		
	Devices on main board not included	
1	AT only	
2	PC only	
3	2nd Bisynchronous on AT only	
4	These devices decode the full 16 address bits thereby allowing further devices in the same category above 3FF (eg GPIB (1) = 22E1 etc)	

ตารางที่ 4.2 แสดงการใช้งานของพอร์ตบน IBM

จากตารางที่ 4.2 จะเห็นได้ว่าแอดเดรสของพอร์ตถูกแบ่งออกเป็นช่วงย่อยๆ ซึ่งจะกำหนดไว้ให้ใช้กับอุปกรณ์ I/O เฉพาะอย่าง ถ้าในระบบของเราไม่ได้ใช้งานอุปกรณ์นั้น เราสามารถนำแอดเดรสของพอร์ตในช่วงนั้นมาใช้งานได้ เช่น ถ้าระบบเราไม่ได้ใช้จอยสติค (Joystick) เราสามารถใช้งานพอร์ตในช่วง (200H-20FH) ได้ แต่อย่างไรก็ตาม การเลือกใช้งานพอร์ตที่ไม่ได้ถูกกำหนดให้ใช้กับอุปกรณ์อื่นจะดีกว่า ทำให้อุปกรณ์ที่ใช้งานผ่านพอร์ตนี้สามารถใช้ได้กว้างขวางยิ่งขึ้น

4.4 อินเตอรรัพต์ (Interrupts)

การอินเตอรรัพต์ใน CPU เบอร์ 8088 และ 80286 แบ่งออกเป็น 2 ชนิด คือ NMI (Non-Maskable Interrupts) และ INT (Maskable Interrupts) แต่สำหรับ IBM PC NMI ถูกใช้ในการเช็คความผิดพลาดของการรับส่งข้อมูล

การอินเตอรรัพต์แบบ Maskable มีจำนวน 256 เวกเตอร์ (เวกเตอร์ในที่นี้คือแอดเดรสเริ่มต้นของโปรแกรมย่อย (Interrupt Service Routine) ที่ถูกกระทำเมื่อได้รับการเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อินเทอร์รัพต์) ในจำนวนทั้งหมด 256 เวกเตอร์นี้ จะรวมถึงซอฟต์แวร์อินเทอร์รัพต์ (B DOS Call) , สัญญาณอินเทอร์รัพต์ที่เกิดจากตัว CPU เองด้วย เช่น การหารด้วยศูนย์ (Divide By Zero) , การทำที่ละคำสั่งที่ใช้ในโปรแกรมดีบัก (Single Step) และฮาร์ดแวร์อินเทอร์รัพต์สำหรับฮาร์ดแวร์อินเทอร์รัพต์แบ่งออกได้ตามลำดับความสำคัญดังนี้ สำหรับรุ่น XT แบ่งได้ 8 ระดับ และ 15 ระดับในรุ่น AT ตารางที่ 4.3 แสดงฮาร์ดแวร์อินเทอร์รัพต์ใน IBM PC อินเทอร์รัพต์บางระดับจะถูกใช้ในเมนบอร์ด เช่น อินเทอร์รัพต์ระดับ 0 เป็นต้น ส่วนที่เหลือจะถูกใช้ในการ์ดขยายต่างๆ เช่น ตัวควบคุมดิสก์ (Floppy & Hard Disk Controller)

ใน IBM PC การขอทำแบบฮาร์ดแวร์อินเทอร์รัพต์แบบ Maskable เราไม่จำเป็นต้องให้ค่าเวกเตอร์ เพราะว่า 8259 (Interrupt Controller) จะเป็นตัวจัดการเองหมด เราเพียงแค่ให้สัญญาณขอทำอินเทอร์รัพต์ผ่านทางสลอตก็พอ

Bussed Signal	Hardware Interrupt Level		Priority interrupt number (Hex)	Function
	Int. Ctrl 1	Int. Ctrl 2 (AT only)		
PC	AT			
		IRQ0	08	Timer output 0
		IRQ1	09	Keyboard
		IRQ2	0A	Keyboard (Int. Ctrl 2 on AT)
		IRQ8	70	Realtime clock
		IRQ9	71	S/W Redirection to IRQ2
		IRQ10	72	Reserved
		IRQ11	73	Reserved
		IRQ12	74	Reserved
		IRQ13	75	Co-processor
		IRQ14	76	Hard disk controller
		IRQ15	77	Reserved
		IRQ3	0B	Serial port 2
		IRQ4	0C	Serial port 1
		IRQ5	0D	Hard disk (Printer 2 on AT)
		IRQ6	0E	Floppy disk controller
		IRQ7	0F	Printer port 1

* The PC has IRQ2 as a bussed signal. On the AT IRQ2 connects to the second interrupt controller to fan this one interrupt out into another 8. In order to provide compatibility within the PC, the software on the AT redirects IRQ9 to the IRQ2 handler. Accordingly, the same pin on the expansion slot which is IRQ2 on the PC is IRQ9 on the AT.

The function of IRQ2 (or IRQ9) is officially reserved and would not be a good one to consider using as it is used by EGA cards.

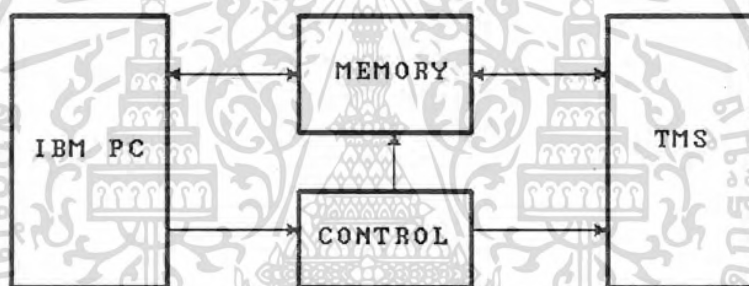
ตารางที่ 4.3 แสดงการจัดฮาร์ดแวร์อินเทอร์รัพต์

บทที่ 5

รายละเอียดโครงการ

5.1 หลักการเบื้องต้น

จะขอกล่าวถึงจุดประสงค์ของโครงการนี้อีกครั้ง คือเพื่อทำการเพิ่มประสิทธิภาพในการประมวลผลสัญญาณเชิงเลขให้แก่คอมพิวเตอร์ โดยจะเน้นในงานด้าน Image Processing โดยหลักการพื้นฐานก็คือใช้ CPU DSP มาช่วยในการคำนวณให้กับคอมพิวเตอร์ โดย CPU ที่นำมาใช้นี้จะถูกออกแบบมาเฉพาะ มีคุณสมบัติที่สามารถคำนวณตัวเลขจำนวนมากๆได้อย่างมีประสิทธิภาพ เมื่อนำมาช่วยคำนวณแทน CPU ของเครื่องคอมพิวเตอร์ ก็สามารถที่จะช่วยลดเวลาในการคำนวณลงไปได้มาก โดยมีโครงสร้างพื้นฐานเป็นดังรูปด้านล่าง



รูปที่ 5.1 แสดง Block Diagram พื้นฐาน

จากรูปแสดง Block Diagram พื้นฐานของระบบ ซึ่งจะประกอบด้วย ส่วนของคอมพิวเตอร์ (IBM PC) ซึ่งโครงการนี้เลือกใช้คอมพิวเตอร์ในตระกูล 80x86 กับส่วนของวงจรภายนอกที่จะต่อเพิ่มเข้าไป ได้แก่ วงจร CPU พร้อมกับอุปกรณ์ประกอบ, หน่วยความจำ Memory ซึ่งจะมีอยู่ 2 ชุดด้วยกัน คือ Program Memory และ Data Memory และวงจรควบคุมการทำงานของ TMS โดยได้รับคำสั่งจากตัวคอมพิวเตอร์อีกที

หลักการทำงานของระบบก็คือ ในส่วนของ TMS จะมี Program สำหรับการประมวลผลอยู่ โดยเก็บอยู่ใน RAM ส่วน Program memory เมื่อคอมพิวเตอร์ต้องการที่จะประมวลผล คอมพิวเตอร์ก็จะทำการย้ายข้อมูลภาพเข้าไปไว้ใน RAM ส่วน Data Memory จากนั้นก็ให้ TMS เริ่มทำการประมวลผล และเมื่อ TMS ประมวลผลเสร็จคอมพิวเตอร์ก็จะอ่านข้อมูลที่ได้จากการประมวลผลใน Data Memory ไปใช้ต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 แนวทางในการออกแบบ

การทำโครงการชุดนี้ ได้เลือกคอมพิวเตอร์ที่ใช้ CPU ในตระกูล 80x86 ของ IBM PC หรือคอมแพคทีเบิ้ลมาทำการศึกษาคลวง เนื่องจากเป็นเครื่องที่นิยมใช้กันอย่างกว้างขวาง ดังนั้นในการออกแบบเพื่อทำการ Interface กับคอมพิวเตอร์ จึงต้องพิจารณาระบบการทำงานของคอมพิวเตอร์ก่อนเป็นอันดับแรก

สิ่งที่ต้องพิจารณาก่อนก็คือ การติดต่อระหว่างคอมพิวเตอร์ TMS และหน่วยความจำบนการ์ด โดยทั่วไปคอมพิวเตอร์สามารถติดต่อกับอุปกรณ์ภายนอกได้ก็คือทาง I/O port เป็นสำคัญ ซึ่งอาจจะเป็นการติดต่อทาง พอร์ตขนาน หรือ พอร์ตอนุกรม หรือจะเป็นการต่อจากสล๊อตของเครื่องโดยตรงก็ได้ เมื่อพิจารณาถึงการย้ายข้อมูลจากคอมพิวเตอร์มายัง RAM ภายนอก ถ้าผ่านทาง I/O port ในการประมวลผลแต่ละครั้ง จะต้องใช้เวลาในการย้ายข้อมูลเป็นจำนวนมาก เนื่องจาก การติดต่อกับ I/O port จะใช้เวลามากกว่าการติดต่อกับหน่วยความจำ จะเห็นว่าไม่มีประสิทธิภาพพอ ดังนั้นในการย้ายข้อมูลภาพระหว่าง PC กับการ์ด TMS จึงเลือกใช้การติดต่อผ่าน Memory แทนการย้ายผ่าน I/O port

หลักการทำงานก็คือหน่วยความจำที่อยู่บนการ์ด TMS กับ บล็อกของหน่วยความจำบน PC จะเป็นตัวเดียวกัน เมื่อใดที่ PC ต้องการโหลดข้อมูลภาพไปยังหน่วยความจำหรือดูข้อมูลจากหน่วยความจำ ก็สามารถกระทำบนตัวหน่วยความจำได้เลย และเมื่อการ์ด TMS ต้องการนำข้อมูลภาพไปประมวลผล ก็จะนำเอาข้อมูลที่อยู่ใน RAM ชุดเดียวกับที่ PC ใช้ในการโหลดหรือดึงข้อมูลนั่นเอง ซึ่งในลักษณะนี้จะช่วยลดเวลาในการเคลื่อนย้ายข้อมูลลงได้มาก

เมื่อพิจารณาการจัดหน่วยความจำภายในเครื่องคอมพิวเตอร์ IBM PC- จะเห็นได้ว่าในพื้นที่ของหน่วยความจำในคอมพิวเตอร์ PC ขนาด 1 เมกกะไบต์ จะมีบล็อกว่างที่ไม่ได้ใช้งานอยู่ 2 บล็อก ตามการจัดการหน่วยความจำของ DOS คือ บล็อก D (D000:0000-D000:FFFF) และ บล็อก E (E000:0000-E000:FFFF) ว่างอยู่สามารถนำมาใช้งานได้ โดยที่แต่ละบล็อกมีขนาดเท่ากับ 64 กิโลไบต์ ดังนั้นจึงได้เลือกใช้วิธีการ map หน่วยความจำบนการ์ดลงไปในระบบของคอมพิวเตอร์ โดยเลือกใช้แค่เพียง 1 บล็อก ดังนั้นคอมพิวเตอร์จะสามารถติดต่อกับหน่วยความจำภายนอกได้เหมือนกับเป็นหน่วยความจำในระบบ ในขณะที่ TMS ก็ยังสามารถติดต่อได้เช่นเดิมนั่นคือวิธีที่ใช้เพิ่มประสิทธิภาพให้กับการโอนข้อมูลระหว่าง PC กับ TMS

ในการใช้งาน เราจะทำการโหลดโปรแกรมจากคอมพิวเตอร์ไปเก็บใน RAM ส่วนของ Program Memory ก่อน ซึ่งโปรแกรมที่โหลดให้ TMS คือโปรแกรมการคำนวณต่าง ๆ ที่ต้องการให้ TMS ประมวลผล จากนั้นเมื่อต้องการให้เริ่มประมวลผล ก็จะโหลดข้อมูลต่างๆไปยัง RAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วน Data Memory (โดยการไหลค่าต่างๆไปยัง RAM บนการ์ดจะผ่านทาง การ map Memory บน PC ทั้งสิ้น) จากนั้นก็ทำการส่งคำสั่งจากพอร์ตของ PC เพื่อไปควบคุมให้ TMS ติดต่อกับ RAM ได้ และเริ่มทำงาน (ให้สัญญาณ reset ของ TMS ไม่ active , เป็น "1") เมื่อ TMS ประมวลผลเสร็จ PC ก็นำข้อมูลที่ได้ไปใช้จาก RAM ในส่วนของ Data Memory โดยการที่ PC จะรู้ได้อย่างไรว่า TMS ประมวลผลเสร็จแล้วนั้น จะใช้วิธีการ interrupt PC โดยให้ TMS out ข้อมูลออกมาทางพอร์ตของ TMS เพื่อเป็นสัญญาณไป interrupt PC เมื่อ PC ได้รับ interrupt ก็จะมีตัวเองว่า TMS ทำงานเสร็จแล้ว จากนั้นก็ทำการส่งคำสั่งควบคุมให้ RAM ติดต่อกับ PC เพื่อนำข้อมูลมาใช้งานต่อ พร้อมทั้งหยุดการทำงานของ TMS (ป้อน reset ค้าง ให้กับ TMS)

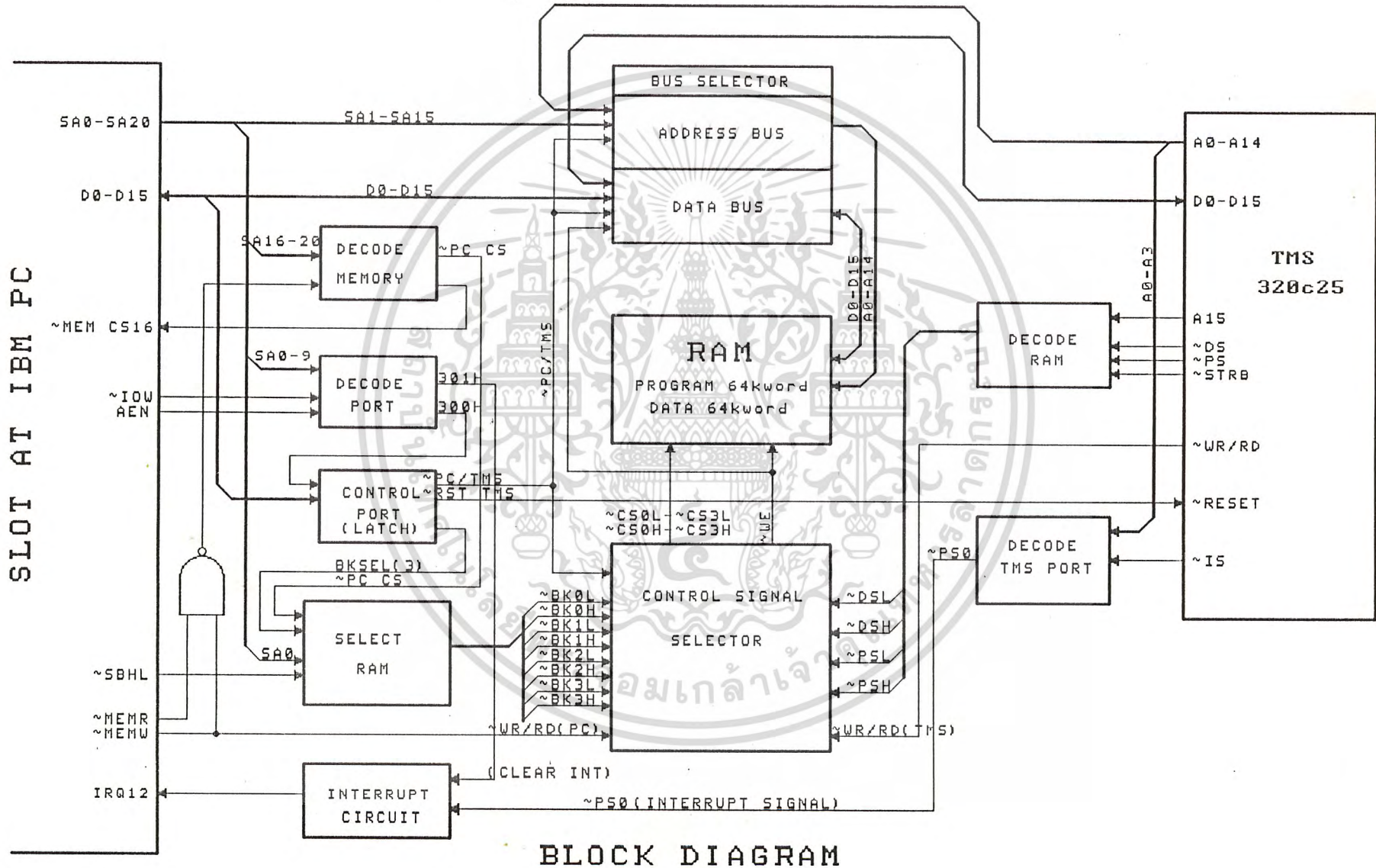
5.3 รายละเอียดของวงจร

โครงการนี้เป็น การสร้างวงจรขึ้นมาชุดหนึ่ง โดยจะทำการติดต่อกับ PC ผ่านทางสล็อตของเมนบอร์ด ดังนั้นจึงมีลักษณะเป็นการ์ดสำหรับเสียบเข้ากับสล็อต นอกจากนี้ การที่จะให้การ์ดสามารถทำงานได้ยังต้องประกอบด้วยโปรแกรมการประมวลผลสำหรับ TMS และโปรแกรมควบคุมการทำงานของการ์ดด้วย เพื่อให้วงจรสามารถทำงานประมวลผลได้ตามต้องการ ซึ่งจากหลักการและแนวทางการออกแบบที่ได้กล่าวไปแล้ว สามารถออกแบบวงจรได้ดังในรูปวงจรมุมที่ 1-6 โดยมี Block Diagram ที่สมบูรณ์ในรูปที่ 5.2

หลักการการทำงานของวงจร

จาก Block diagram ของวงจร วงจรนี้จะประกอบด้วย RAM จำนวน 128 k word โดยจะแบ่งเป็นส่วนของ Program Memory 64 k word และ Data Memory 64 k word ซึ่ง RAM ชุดนี้สามารถติดต่อได้ 2 ทาง คือ จาก PC และ TMS ดังนั้นจึงต้องมีวงจรสำหรับทำการสวิตช์เลือกให้ RAM ติดต่อกับทางใด ซึ่งได้แก่ วงจรใน Block Bus Selector และ Control Signal Selector สำหรับส่วนสำคัญในการประมวลผลก็คือ วงจรของ CPU TMS 320C25 ซึ่งจะประกอบด้วยวงจรในการตีโค้ดเลือก RAM สำหรับติดต่อกับ RAM และวงจรตีโค้ดพอร์ตเพื่อใช้สร้างสัญญาณอินเตอร์รัพต์ให้กับ PC นอกจากนี้ก็เป็นส่วนของวงจรทาง PC ได้แก่ ส่วนของการตีโค้ด address ติดต่อกับ RAM และวงจรควบคุมการทำงานต่างๆบนการ์ดผ่านทางพอร์ต I/O ของ PC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



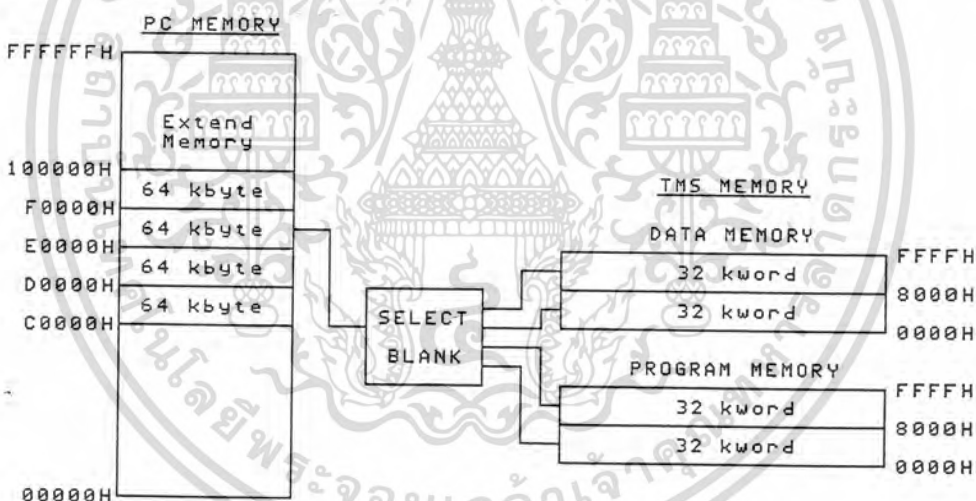
BLOCK DIAGRAM

รูปที่ 5.2 Block Diagram ของการ์ด

การจัดการ map Memory บน Card

ขนาดของ RAM ที่ใช้นั้น ถูกเลือกใช้เท่ากับขนาดของหน่วยความจำที่ TMS ต้องการ ซึ่งก็คือแบ่งเป็นส่วนของ Program memory 65,536 ตำแหน่ง และ Data memory อีก 65,536 ตำแหน่ง (อย่างละ 64 k ตำแหน่ง) โดยที่ TMS มีขา Data Bus ทั้งสิ้น 16 เส้น นั่นคือ TMS ประมวลผลครั้งละ 16 bit (2 byte หรือ 1 word) ดังนั้นจึงต้องการหน่วยความจำทั้งหมดเป็นจำนวน 128 k word หรือ 256 k byte

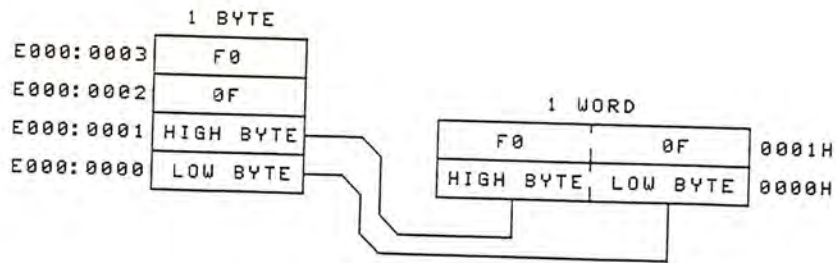
เมื่อมาพิจารณาถึงหน่วยความจำบน PC จะเห็นว่าจำนวนหน่วยความจำบนการ์ด ถ้าจะให้ map ลงบน PC ทั้งหมด จะใช้เนื้อที่มาก ซึ่งบน PC มีที่ว่างไม่เพียงพอ โดยเนื้อที่หน่วยความจำบน PC ที่ว่าง ได้แก่ส่วนของหน่วยความจำบล็อก D และ E เป็นจำนวน 128 k byte เมื่อไม่สามารถ map ได้ทั้งหมด ดังนั้นจึงใช้วิธีแบ่งหน่วยความจำบนการ์ดออกเป็นส่วนๆ (แบงค์) แล้วทำการ map ทีละส่วน โดยมีวงจรเลือกอีกชุดหนึ่งสำหรับเลือกแบงค์ที่จะ map ดังนั้นจึงสามารถ map หน่วยความจำทั้งหมดได้



รูปที่ 5.3 แสดงการ map RAM บนการ์ดกับ memory ของ PC

สำหรับวงจรที่ออกแบบ จะใช้หน่วยความจำบน PC เพียง 1 บล็อกเท่านั้น (ปกติจะใช้ บล็อก E แต่ก็สามารถใช้ บล็อก D ได้เช่นกัน โดยจะต้องแน่ใจว่าหน่วยความจำส่วนนั้นไม่มีการใช้งาน และต้องว่างไม่มี RAM ในส่วนนั้น) ดังนั้น RAM บนการ์ดจะถูกแบ่งเป็น 4 ส่วน ได้แก่ Program Memory ส่วนล่างและส่วนบน กับ Data Memory ส่วนล่างและส่วนบน จะได้ส่วนละ 64 k byte (32 k word) เท่ากับ 1 บล็อก ใน PC พอดี จากการแบ่งจะได้เป็น Blank0 (BK0) DATA LOW MEMORY , Blank1 (BK1) DATA HIGH MEMORY , Blank2 (BK2) PROGRAM LOW MEMORY , Blank3 (BK3) PROGRAM HIGH MEMORY

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.4 แสดงการ map memory ระหว่าง 1 word กับ 1 byte

ในการ map Memory 64 k byte ด้วย 32 k word นั้น 1 word จะเท่ากับ 2 byte ซึ่งใช้เนื้อที่ Memory 2 ตำแหน่งบน PC การจัดการจะใช้วิธี map 1 word byte ล่าง เข้าที่ตำแหน่งแรก byte บนลงตำแหน่งที่ติดกันถัดไป หรือกล่าวได้ว่า map byte ล่างด้วย address เลขคู่ และ map byte บนด้วย address เลขคี่ ซึ่งเป็นวิธีเดียวกับที่ใช้ใน PC เพื่อให้สามารถประมวลผลแบบ 16 bit ได้ด้วย

การทำงานของวงจร

การใช้งานของการ์ดคือ การโหลดโปรแกรมและข้อมูลต่างๆ ลงบน RAM จากนั้นก็ให้ TMS ทำการประมวลผล เมื่อเสร็จสิ้นการประมวลผล TMS ก็จะส่งสัญญาณกลับมายังอินเตอร์เฟซ PC เพื่อเป็นการบอกให้รู้ จาก Block diagram ส่วนที่เป็นตัวเชื่อมระหว่าง RAM กับ PC และ TMS ได้แก่ วงจรในบล็อค Bus Selector และ Control Signal Selector ซึ่งเป็นส่วนที่เลือกให้สัญญาณจากด้านใดติดต่อกับ RAM โดย Bus Selector เป็นตัวสวิตช์เลือก Address bus และ Data bus ส่วนบล็อค Control Signal Selector จะสวิตช์เลือกสัญญาณ $\sim WR$ และ $\sim CS$ ของ RAM โดยที่ทั้งสองบล็อคจะถูกควบคุมโดยส่วนบล็อค Control port ซึ่งสัญญาณจากบล็อค Control port ก็ถูกส่งมาจาก PC อีกต่อหนึ่ง

ในการติดต่อระหว่าง TMS กับ RAM จะมีวงจร decode address (ในบล็อค decode RAM) เพื่อสร้างสัญญาณ enable RAM (Chip select -CS) ได้สัญญาณออกมา 4 เส้น คือ $\sim DSL$, $\sim DSH$, $\sim PSL$ และ $\sim PSH$ โดย $\sim DSL$ และ $\sim DSH$ เป็นสัญญาณติดต่อกับ Data Memory ส่วนล่างและบน ส่วน $\sim PSL$ และ $\sim PSH$ ใช้ติดต่อกับ Program Memory ส่วนล่างและบนเช่นกัน นอกจากนี้ยังมีวงจร decode port ซึ่งใช้สำหรับสร้างสัญญาณอินเตอร์เฟซให้ PC หลังจากประมวลผลเสร็จ ซึ่งจะทำงานร่วมกับบล็อค Interrupt circuit อีกต่อหนึ่ง

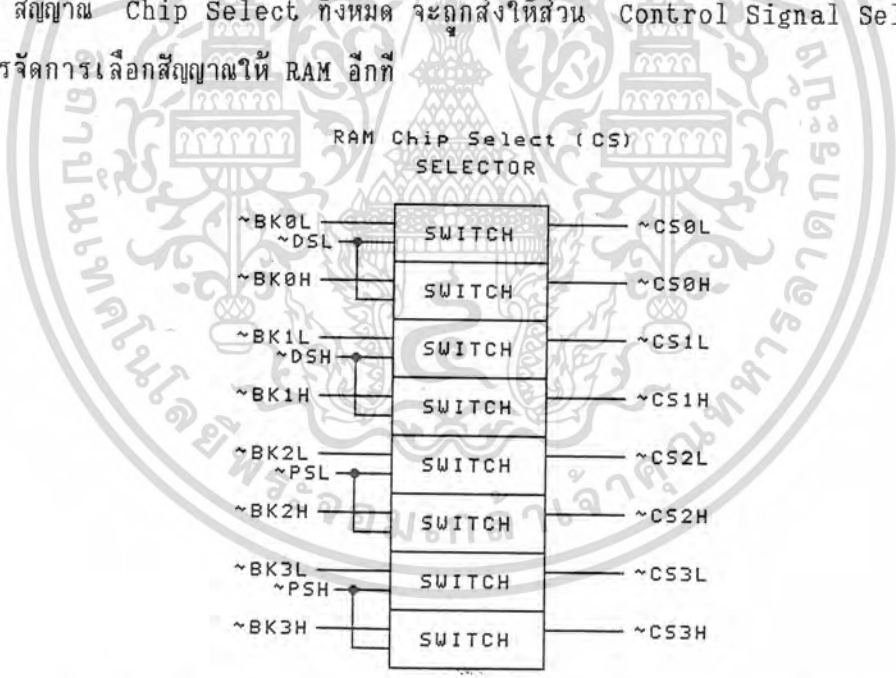
ด้านของ PC ก็ประกอบด้วย วงจร decode memory สำหรับ Decode หน่วยความจำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บล็อกที่จะ map ซึ่งจะใช้เฉพาะสัญญาณ Address bus A16-A19 เท่านั้น ส่วนอื่นได้แก่บล็อก select RAM เป็นตัวสร้างสัญญาณ Chip Select สำหรับ enable RAM โดยใช้กลุ่มสัญญาณ BK SEL จาก Control port เป็นสัญญาณเลือก RAM ว่าให้ map แบนด์ใด นอกจากนี้ยังใช้สัญญาณ A0 กับ ~SBHL ในการเลือก RAM byte ล่างและ byte บน ซึ่งจะได้เอาท์พุทออกไป 8 เส้น ได้แก่ สัญญาณ ~BK0L-BK3L และ ~BK0H-BK3H

เมื่อเปรียบเทียบสัญญาณ Chip Select ระหว่างทางด้าน TMS กับ PC จะเห็นว่าต่างกัน 1 เท้า เนื่องจาก TMS เป็น CPU 16 bit แท้ 1 แอดเดรสมี ค่า 16 bit ในขณะที่ทาง PC ไม่ใช่ CPU 16 bit โดยตรง คือสามารถทำงานได้ทั้ง 8 bit และ 16 bit แต่หน่วยความจำ 1 แอดเดรสจะมีค่าที่อยู่เพียง 8 bit ในกรณีที่ต้องการให้ทำงานแบบ 16 bit จะแบ่งเป็นไบท์คู่และไบท์คี่ หรือไบท์ล่างและไบท์บน ดังเช่นในโครงงานนี้จะให้ทำงานแบบ 16 bit ซึ่งจะมีสัญญาณควบคุมแยกกัน คือ A0 สำหรับไบท์ล่าง (คี่) และ SBHL สำหรับไบท์บน (คู่) ทำให้ PC ต้องใช้สัญญาณมากกว่า 1 เท้า

สัญญาณ Chip Select ทั้งหมด จะถูกส่งให้ส่วน Control Signal Select ซึ่งจะทำการจัดการเลือกสัญญาณให้ RAM อีกที



รูปที่ 5.5 แสดงช่องทำงานภายในของส่วน Control Signal Select

จากรูป จะเห็นว่าสัญญาณจาก TMS 1 สัญญาณ ใช้คู่กับสัญญาณจาก PC 2 เส้น เพราะ TMS ไม่มีสัญญาณแยก byte บน , ล่าง (ที่สามารถแบ่งได้เป็นเช่นนี้ก็เนื่องจากใช้ RAM ทั้งหมด 8 ตัว ขนาดตัวละ 32k * 8bit 1 แบนด์จะใช้ RAM 2 ตัว โดยแยกตัวหนึ่งเป็น data byte บน อีกตัวเป็น byte ล่าง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รายละเอียดของวงจรแต่ละส่วน

วงจร CPU

แสดงได้ในรูปวงจรแผ่นที่ 1 มี CPU คือ TMS 320C25 ซึ่งมี address และ data bus อย่างละ 16 เส้น ส่วนประกอบที่ทำให้ CPU ทำงานได้ คือ Crystal ซึ่งต่ออยู่ระหว่างขา X2 และ X7 ของ TMS ใช้สร้างสัญญาณ Clock ภายในตัว TMS โดยเลือกใช้ใช้งานที่ความถี่เพียง 20 MHz เท่านั้น จากความสามารถเต็มที่ซึ่งทำงานได้ที่ 40 MHz เนื่องจากโครงการนี้เป็นภาระเริ่มต้นศึกษาทดลองใช้งาน CPU เบอร์นี้เป็นครั้งแรก ดังนั้นถ้าหากเลือกใช้ความถี่ 40 MHz ที่ความถี่สูง การเดินสายจะมีปัญหามากขึ้น เพื่อลดปัญหาจึงใช้ความถี่ต่ำลง ซึ่งการเดินสายไม่มีผลมากนัก

จากรูปวงจรแผ่นที่ 1 สำหรับตัว CPU นั้นขา Ready จะต่อกับไฟ Vcc เป็นการให้ TMS ทำงานโดยไม่มี Wait State เพราะ RAM ที่ใช้ในโครงการนี้เป็นสแตติก RAM ที่มี Access Time น้อยมาก ทำให้ไม่ต้องการ Wait State ในการอ่านเขียน RAM และเป็นการเพิ่มความเร็วในการประมวลผลด้วยในตัว ส่วนขา MP/~MC จะต่อกับ Vcc ด้วย เพื่อ set ให้ TMS อยู่ใน mode "Microprocessor" คือ จะใช้ Program Memory จาก RAM ภายนอกทั้งหมด สัญญาณอีกเส้นที่ใช้ควบคุม TMS คือ ~RST หรือ Reset โดยขาจะถูกควบคุมจาก Control port อีกต่อ เพื่อเป็นการกำหนดให้ TMS ทำงานหรือไม่

ในรูปวงจรแผ่นที่ 1 ยังมี IC อีกตัว คือ 74LS138 (decoder 3-8) ทำหน้าที่ decode พอร์ตของ TMS ซึ่งจะใช้สาย address เพียง 4 เส้น เพราะ TMS มีพอร์ต I/O เพียง 16 พอร์ตเท่านั้น ร่วมกับสัญญาณ ~IS ในการ decode สำหรับสัญญาณที่สร้างขึ้นนี้ มีไว้เพื่อไปป้อนให้กับวงจรสร้างสัญญาณอินเตอร์รัพต์ในรูปวงจรแผ่นที่ 6 เพื่อสร้างสัญญาณไปอินเตอร์รัพต์ PC อีกต่อหนึ่ง (ใช้ในกรณีที่ TMS ประมวลผลเสร็จสิ้น ซึ่งในตอนท้ายของโปรแกรมจะมีการสั่งให้ส่งค่าออกที่พอร์ตนี้) โดยสัญญาณที่ใช้นี้ต้องการแค่เพียงสัญญาณการ decode port เท่านั้น เพื่อไปทริกวงจรต่อไป จึงไม่มีการต่อ data bus สำหรับ I/O port

วงจร RAM

แสดงได้ในรูปวงจรแผ่นที่ 2 ซึ่งประกอบด้วย RAM จำนวน 8 ตัว เบอร์ MT5C2568 เป็น สแตติก RAM ขนาด $32 \text{ k} * 8 \text{ bit}$ ซึ่งเป็น RAM ชนิดพิเศษที่มีเวลาในการเข้าถึงต่ำ เพื่อลด ความยุ่งยากในส่วนของวงจร CPU ที่จะต้องมีวงจรสร้าง Wait State ถ้าใช้ RAM ที่มีเวลา เข้าถึงมาก

ลักษณะการต่อ RAM ขา address และขา $\sim\text{WE}/\text{RD}$ จะต่อร่วมกันหมด ส่วน data bus จะแบ่งเป็น 2 ส่วน แถวบน U3-U6 จะต่อกับ data bus D8-D15 ใช้เก็บ data byte บน ส่วนแถวล่าง U7-U10 จะต่อกับ data bus D0-D7 ใช้เก็บ data byte ล่าง สำหรับขา $\sim\text{CS}$ ของแต่ละตัวจะแยกกัน ซึ่งเป็นผลจากวิธีการติดต่อระหว่าง RAM กับ PC ซึ่งแบ่งเป็นแบบ 4 แบนด์ โดยแต่ละแบนด์ยังแบ่งเป็นไบท์คู่และไบท์คี่ ทำให้ RAM แต่ละตัวต้องมี $\sim\text{CS}$ แยกกัน

วงจรจัดการสัญญาณ $\sim\text{CS}$ จะอยู่ในรูปวงจรแผ่นที่ 4 มี IC 74LS157 2 ตัว ทำหน้าที่ สวิตช์เลือกสัญญาณจาก PC และ TMS จากรูปวงจรจะเห็นว่าสัญญาณจาก PC จะมีด้วยกัน 8 เส้น ในขณะที่จาก TMS จะมีเพียง 4 เส้น โดยที่อินพุทของ IC 74LS157 ทางด้าน TMS จะ มีการต่อขาอินพุทเป็นคู่ ทำให้กรณี TMS ติดต่อ RAM จะแอดคัฟ RAM 2 ตัวพร้อมกัน คือ ไบท์บน และไบท์ล่างทำงานพร้อมกัน ส่วนทาง PC การแอดคัฟ ไบท์บนกับไบท์ล่างจะแยก

สัญญาณ select RAM ของ TMS มาจาก IC 74LS139 2 ชุด (IC 1 ตัวมี decoder 2-4 อยู่ 2 ชุด) ทำการ decode สัญญาณจาก TMS โดยชุดแรกสำหรับ decode เลือก Data Memory ซึ่งใช้สัญญาณ $\sim\text{DS}$, A15 และ $\sim\text{STRB}$ ร่วมกัน อีกชุดหนึ่งสำหรับ decode เลือก Program Memory ใช้สัญญาณ $\sim\text{PS}$, A15 $\sim\text{STRB}$ โดย A15 ใช้สำหรับ decode address ส่วนบนกับส่วนล่าง $\sim\text{STRB}$ เป็นสัญญาณที่แอดคัฟเมื่อเป็นการติดต่อกับหน่วยความจำ สำหรับ $\sim\text{DS}$ และ $\sim\text{PS}$ เป็นสัญญาณที่แอดคัฟเมื่อ CPU ติดต่อกับหน่วยความจำส่วน Data และ Program ตามลำดับ จากการ decode จะได้เป็นสัญญาณ $\sim\text{DSL}$, $\sim\text{DSH}$, $\sim\text{PSL}$ และ $\sim\text{PSH}$ คือ สัญญาณเลือก RAM Data Memory ส่วนล่าง, Data Memory ส่วนบน, Program Memory ส่วนล่างและ Program Memory ส่วนบน ตามลำดับ

ส่วนทางด้าน PC สัญญาณ select ทั้ง 8 เส้น ได้จากวงจร gate ในรูปวงจรแผ่นที่ 4 ซึ่งประกอบด้วย OR gate 8 ตัว แบ่งเป็น 2 ชุด ชุดหนึ่งจะมีอินพุทหนึ่งต่อกับขา A0 ของ PC สำหรับสร้างสัญญาณสำหรับติดต่อ RAM ส่วนไบท์ล่าง (ไบท์คู่) อีกชุดอินพุทจะต่อกับสัญญาณ $\sim\text{SBHL}$ จาก PC เพื่อเลือกติดต่อกับไบท์บน (ไบท์คี่) (สัญญาณ $\sim\text{SBHL}$ นี้จะมีเฉพาะในสล็อตแบบ AT เป็นส่วนที่ขยายจากแบบ XT จะ active กรณีที่ต้องการใช้งานแบบ 16 bit เท่านั้นคือจะต้องมี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณลอจิก "0" ป้อนให้กับขา \sim MEMCS16 ก่อนในช่วงไซเคิลแรกๆ ของการอ่าน/เขียนหน่วยความจำหรือพอร์ต โดยสัญญาณนี้จะแอคติฟเมื่อมีการอ่านหรือเขียนข้อมูลทาง data bus ไปที่บน D8-D15) ส่วนอินพุตอีกขาหนึ่งของ gate แต่ละตัว จะต่อกับ IC 74LS138 ซึ่งทำหน้าที่ decode เลือกแบริ่ง โดยสัญญาณเลือกแบริ่ง BKEN, BKSELO และ 1 มาจาก Control port เอาท์พุทของ 74LS138 จะใช้เพียง 4 เส้นเท่านั้นคือมี 4 แบริ่ง แต่ละขาจะต่อกับ gate 2 ตัว คือ ตัวที่ decode ไปที่กลางตัว และ ไปที่บนอีกตัว เพราะฉะนั้น OR gate แต่ละตัวจะเลือกแอคติฟ (Low) ต่างๆกัน คือ \sim BK0L- \sim BK3L สำหรับ Blank0-3 Low Byte และ \sim BK0H- \sim BK3H สำหรับ Blank0-3 High Byte

วงจร PC Decode

ส่วนนี้จะเป็นส่วนที่ติดต่อกับ PC โดยตรง คือ ทำการ decode memory และ port ของ PC ซึ่งวงจรแสดงในรูปวงจรแผ่นที่ 5 จากรูปวงจร decode port ได้แก่ IC 74LS688 และ 74LS138 โดย 74LS688 จะทำการเปรียบเทียบ address A3-A9 กับค่าตั้งไว้ที่ Dip Switch ตามเบอร์พอร์ตที่ใช้ จากนั้นเอาท์พุทของ 74LS688 จะไปเข้า 74LS138 เพื่อ decode ร่วมกับ A0-A2 อีกครั้งได้เป็นสัญญาณ Select port ในปริภูมิต่างกันเลือกใช้ที่พอร์ต 300H กับ 301H โดยพอร์ต 300H จะเป็น Control port สัญญาณ Select port จาก 74LS138 ขาแรกจะต่อเข้าวงจร Latch IC 74LS273 เพื่อ Latch ข้อมูลจาก D0-D8 ให้เป็นสัญญาณควบคุมวงจรส่วนอื่นต่อไป ที่ขา CLR ของ 74LS273 นั้นจะต่อกับสัญญาณ Rest DRV เพื่อให้ขณะที่ Reset PC ก็จะทำให้ Reset ตัว 74LS273 ด้วย ทำให้สัญญาณ \sim BKEN เป็น 0 นั่นคือเป็นสภาวะที่ไม่มีส่วนใดติดต่อกับ RAM บนการ์ดได้ และสัญญาณ Reset TMS เป็น 0 ด้วย หรือกล่าวได้ว่า เป็นการ Reset การ์ดไปด้วยพร้อมกับ PC

การ decode หน่วยความจำจะใช้ขาแอคติฟเพียง 4 เส้น คือ A16-A19 เพราะว่าเป็นการ decode บิตของหน่วยความจำที่จะใช้เท่านั้น โดยใช้ IC Comparator 4 bit 74LS85 เป็นตัวเปรียบเทียบกับ Dip switch ที่ set ค่าของบิตที่ต้องการไว้ สัญญาณเอาท์พุทจาก 74LS85 จะต่อเข้าวงจร gate ร่วมกับสัญญาณอ่าน/เขียนหน่วยความจำ (\sim MEMR & \sim MEMW) สร้างเป็นสัญญาณ \sim PC CS ป้อนให้กับวงจรอื่นต่อไป สำหรับแอคติฟ RAM

จากวงจรแผ่นที่ 5 ยังมีสัญญาณจาก PC ที่สำคัญอีก 2 เส้น คือ AEN และ \sim MEM CS16 สัญญาณ AEN เป็นสัญญาณที่บอกถึงกรณีการทำ DMA ซึ่งจะต้อง disable พอร์ตอื่นๆ ที่ไม่เกี่ยวข้องทั้งหมด ดังนั้นในวงจร decode port จึงต้องมีการ decode สัญญาณ AEN ไว้ด้วย ส่วน

สัญญาณ \sim MEM CS16 เป็นอินพุทของ PC ใช้เพื่อบอกให้ PC ทำการอ่าน/เขียนหน่วยความจำแบบ 16 bit ซึ่งถ้าไม่มีสัญญาณนี้ป้อนให้ ก็จะไม่มีการส่ง data ออกมาบน data bus D8-D15 สัญญาณ \sim SBHL ก็ไม่แอกต์ฟด้วย ก็จะเป็นการทำงานแบบ 8 bit ตามธรรมดา จึงต้องมีการป้อนสัญญาณนี้ให้ด้วย โดยสัญญาณนี้จะต้องป้อนให้ PC ในช่วงเวลาแรกของไซเคิลการอ่าน/เขียนข้อมูล คือหลังจากที่ค่า address ถูกปล่อยลง bus ต่อจากนั้นจะต้องมีการป้อนอินพุทให้กับขา \sim MEM CS16 ทันที จึงจะสามารถทำงานแบบ 16 bit ได้ ดังนั้นในวงจรจึงใช้สัญญาณจาก 74LS85 มาทำการ enable บัฟเฟอร์ เพื่อป้อนลอจิก "0" ให้กับขา \sim MEM CS16 สาเหตุที่ต้องมีบัฟเฟอร์เนื่องจากขา \sim MEM CS16 หรือขาอินพุททั่วไปของสล๊อคจะต้องการกระแส sink ประมาณ 20 mA จึงต้องต่อบัฟเฟอร์ และต้องเป็นบัฟเฟอร์ที่มีเอาต์พุทเป็น Tri State หรือ Open Collector ด้วย

Bus selector

วงจร Bus Selector เป็นส่วนที่สวิตช์เลือก address bus และ data bus ระหว่าง PC กับ TMS ให้ติดต่อกับ RAM วงจรจริงแสดงในรูปวงจรแผ่นที่ 3 วงจรนี้จะใช้บัฟเฟอร์ 2 ทาง 74LS245 จำนวน 8 ตัว เป็นตัวจัดการ แบ่งเป็นสำหรับ data bus และ address bus อย่างละ 4 ตัว โดยโครงสร้างการต่อของ data bus และ address bus จะคล้ายกัน คือทางด้านขา B ของ 74LS245 จะต่อกับ bus ของ PC และของ TMS แต่ที่ขั้วขา A จะมีการต่อถึงกันเป็นคู่ระหว่างขั้วของ PC กับของ TMS แล้วไปต่อเข้ากับ RAM ต่อไป ลักษณะการต่อของขา A นั้น สำหรับ data bus จะต่อแบบให้ D0 ของ PC ต่อกับ D0 ของ TMS ไล่ไปจนถึง D15 ของ PC กับ D15 ของ TMS แต่สำหรับ address bus จะต่างจาก data bus เล็กน้อย คือขา A0 ของ TMS จะต่อกับขา A1 ของ PC ไล่ไปจนถึง A14 ของ TMS ซึ่งจะต่อกับ A14 ของ PC สำหรับการที่ PC ไม่ใช้ A0 เนื่องจาก A0 ถูกใช้สำหรับเลือก address คู่ร่วมกับ \sim SBHL สำหรับเลือก address คู่ ซึ่งจะพิจารณาอีกแบบก็คือ 1 address ของ TMS จะเท่ากับ 2 address ของ PC นั่นคือ สัญญาณ A1 ของ PC จึงตรงกับ A0 ของ TMS

สำหรับการควบคุมการสวิตช์เลือกนั้นใช้สัญญาณ \sim PC/TMS จาก Control port ซึ่งจะป้อนเข้ากับขา \sim E ของ 74LS245 เพื่อ Enable บัฟเฟอร์ โดยที่ขั้วบัฟเฟอร์ของ TMS นั้น สัญญาณ \sim PC/TMS จะผ่าน Inverter ก่อน เพื่อให้ทำงานที่สัญญาณ \sim PC/TMS เป็นลอจิก "1" ส่วนบัฟเฟอร์ด้าน PC นั้น สัญญาณ \sim PC/TMS จะถูกต่อเข้า OR gate ก่อน ร่วมกับ สัญญาณ decode หน่วยความจำ \sim PC CS เพื่อทำให้บัฟเฟอร์ทำงานเฉพาะกรณีที่ตั้ง Address ตรงกับคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นุญญาติให้นำไปใช้ประโยชน์ด้านนการ

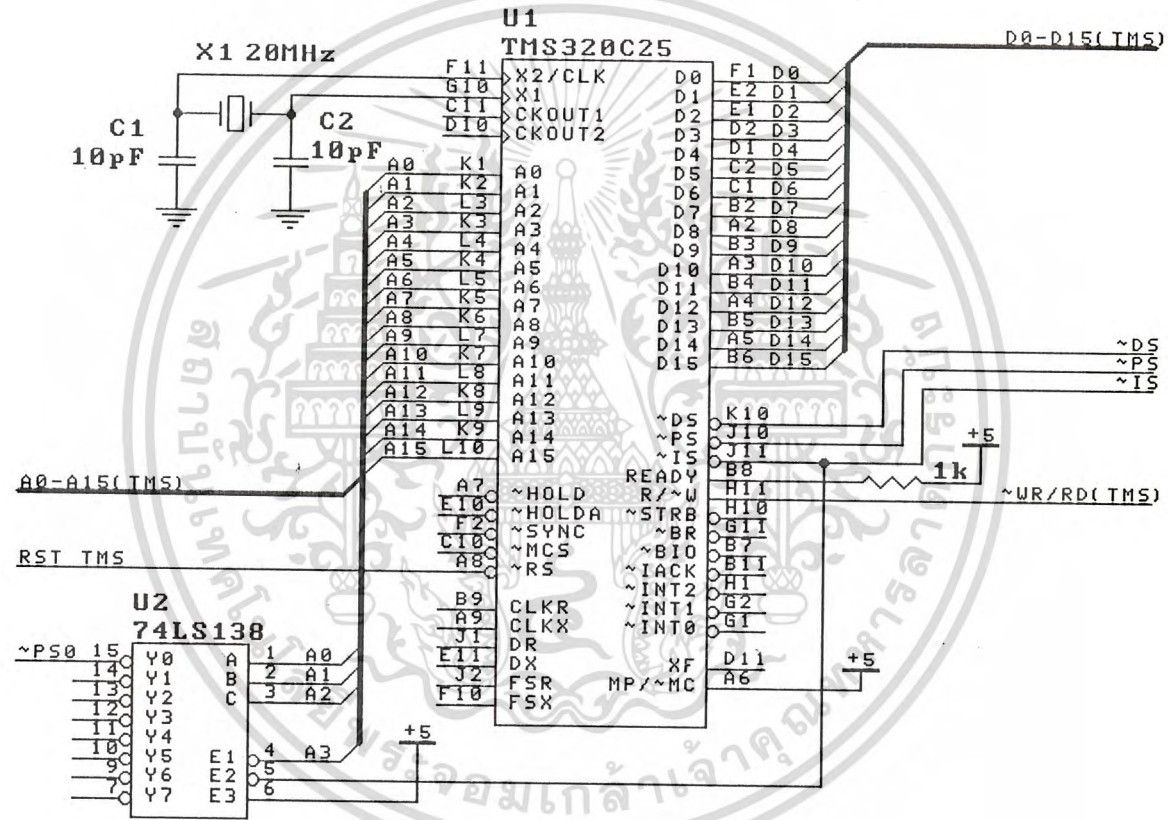
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เท่านั้น

นอกจากนี้ ในส่วน data selector นั้น ถ้า DIR ของ 74LS245 จึงจะต้องถูกควบคุมด้วยเช่นกัน ซึ่งต่างจากส่วน address ที่ถ้า DIR จะต่อลงกราวด์ เพราะเป็นการติดต่อทางเดียว แต่สำหรับ data bus ต้องติดต่อสองทาง โดยสัญญาณที่ใช้ควบคุมได้แก่ สัญญาณ $\sim WE$ ของ RAM นั้นเอง ซึ่งสัญญาณนี้ผ่านการ select มาแล้วอีกต่อเช่นกัน จากรูปในวงจรแผ่นที่ 3 จะเห็นว่าสัญญาณ $\sim WE$ จะถูก select ร่วมมากับ address bus เนื่องจากบัฟเฟอร์ในตัว 74LS245 เหลืออีก 1 คู่ ซึ่งสัญญาณ $\sim WE$ ที่ได้ จะถูกนำไปต่อเข้ากับ RAM ทุกตัว เพื่อบอกสภาวะการเขียนหรืออ่าน RAM และยังต่อเข้าบัฟเฟอร์ของ data Selector เพื่อควบคุมทิศทางของบัฟเฟอร์ 2 ทาง ตามการเขียนหรืออ่าน RAM

วงจรสร้างสัญญาณอินเทอร์รัพต์ PC

วงจรนี้ได้แสดงในรูปวงจรแผ่นที่ 6 อุปกรณ์ที่ทำหน้าที่หลักคือ D-FLIP FLOP 74LS74 โดยจะรับสัญญาณจากทาง TMS ได้แก่สัญญาณ $\sim WE$ Or กับ สัญญาณ $\sim PS0$ (สัญญาณจากการ decode port 0) ซึ่งแอสคิต์ฟที่ขอบขาขึ้นของสัญญาณ เมื่อมีสัญญาณจาก TMS ที่ขา Q ของ D-FLIP FLOP ซึ่งปกติจะเป็น 0 อยู่ ก็จะเปลี่ยนเป็น 1 ส่งผ่านบัฟเฟอร์ เพื่อไปอินเทอร์รัพต์ PC สำหรับที่ขา CLR ของ D-FF นี้จะเป็นวงจร reset เพื่อ CLR สัญญาณอินเทอร์รัพต์ โดยใช้สัญญาณ reset DRV จากสล็อตสำหรับ Clear ตอนเริ่มต้นทุกครั้ง และสัญญาณ $\sim CLR INT$ เอาท์จากการ decode พอร์ตเบอร์ 301H เพื่อให้ Clear อินเทอร์รัพต์ หลังจาก PC รับรู้การอินเทอร์รัพต์

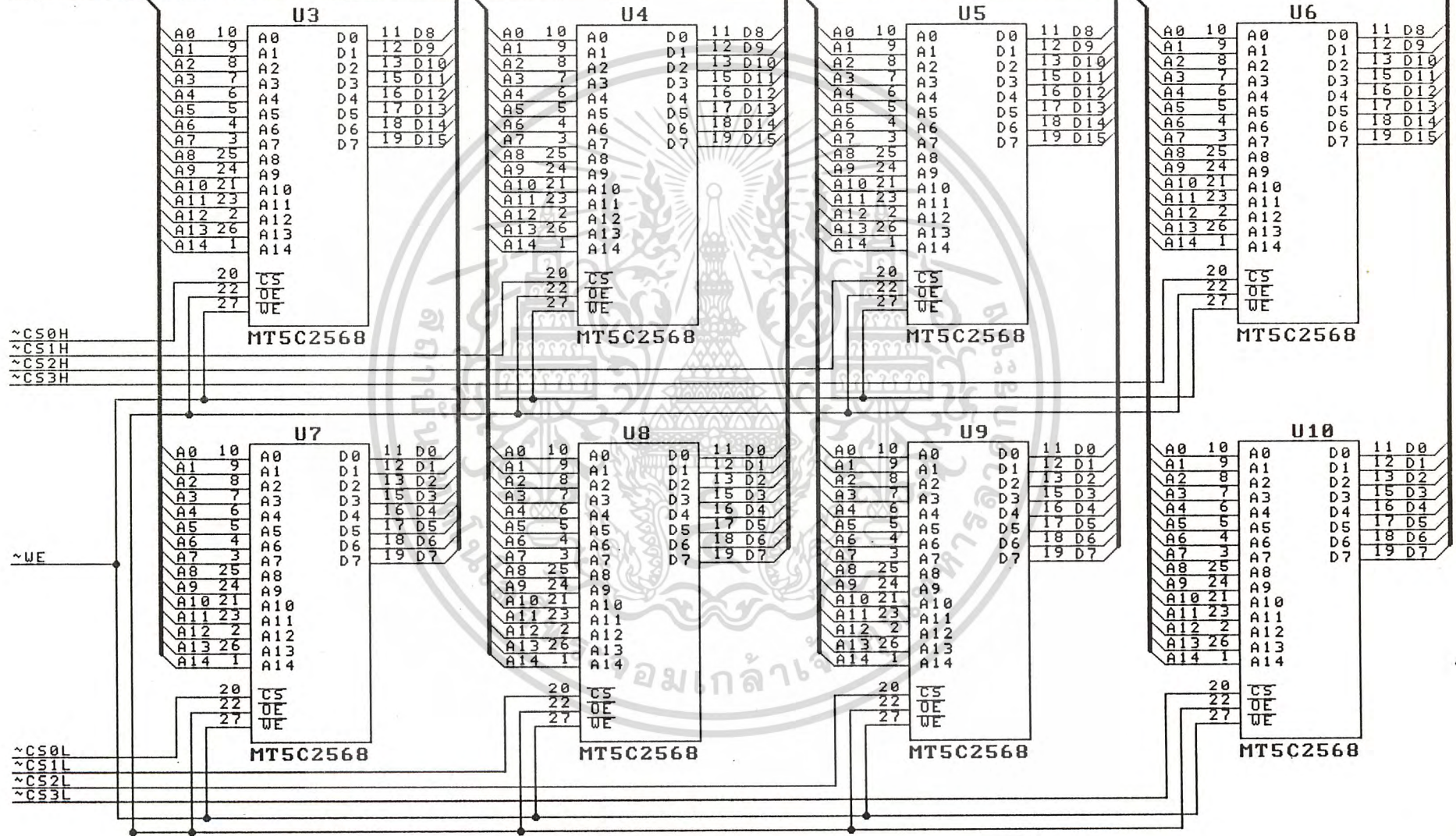


CPU CIRCUIT

รูปวงจรแผ่นที่ 1 วงจร CPU

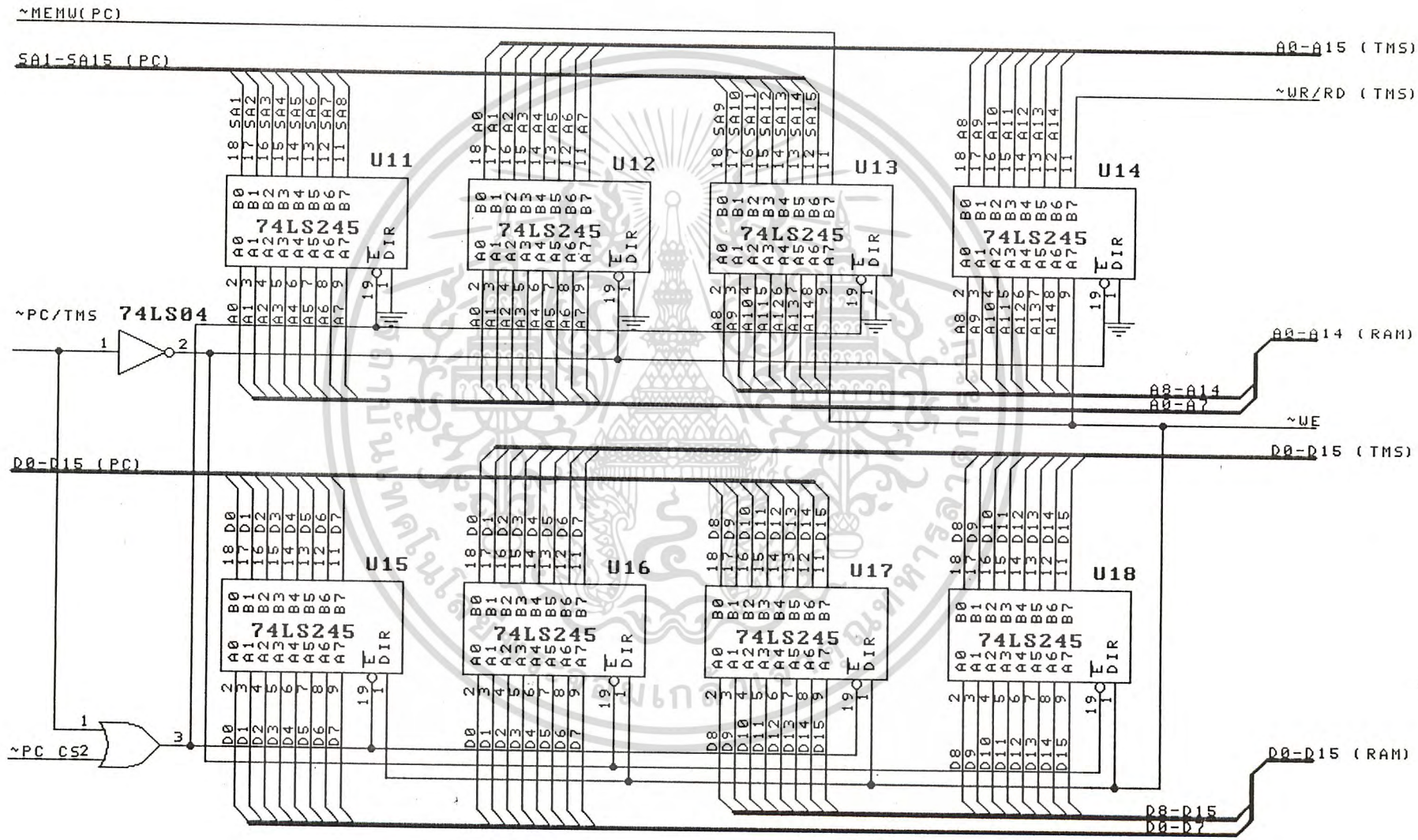
D0-D15 (RAM)

A0-A14 (RAM)



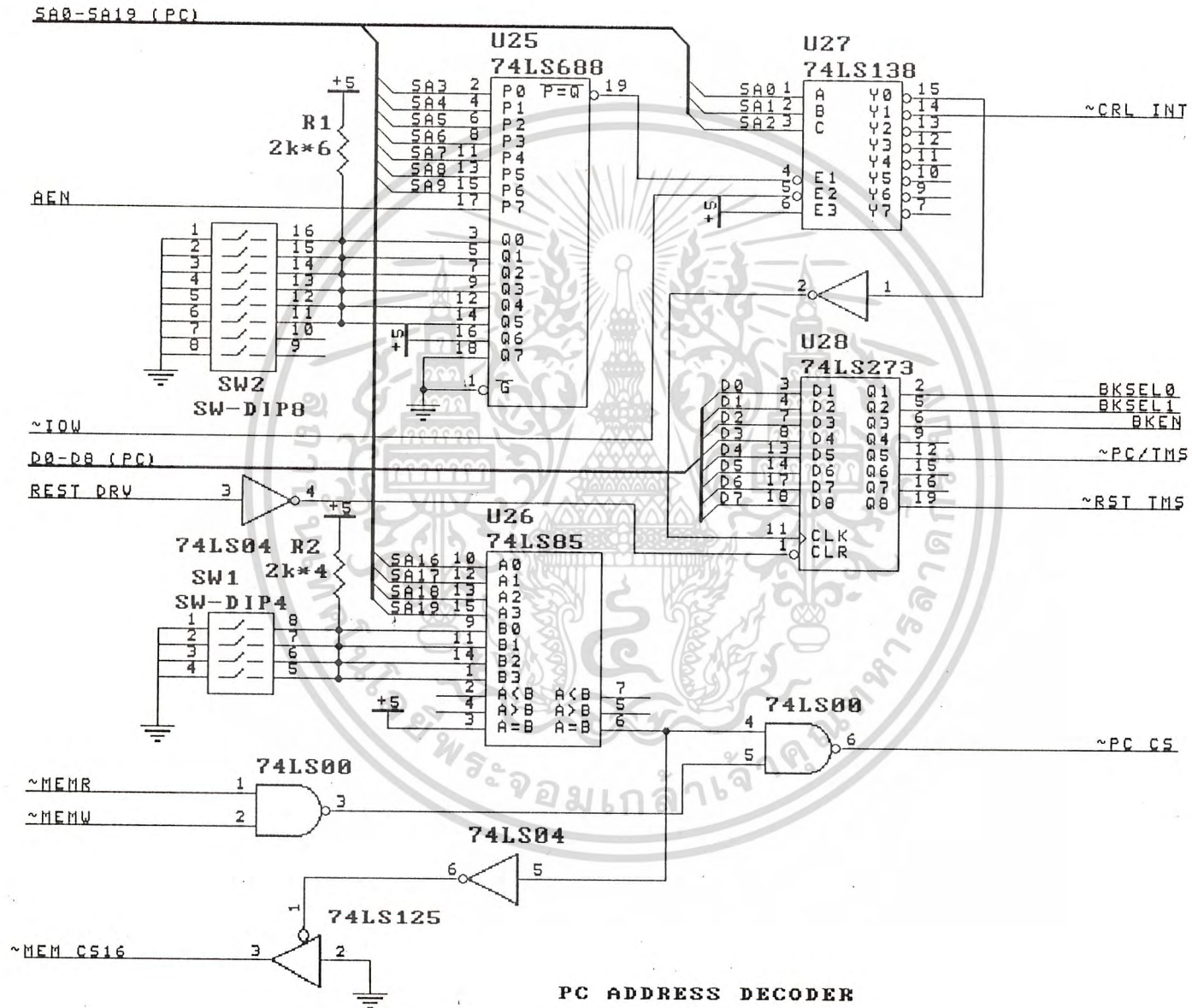
MEMORY (RAM) ON CARD

รูปวงจรมemory 2 วงจร RAM



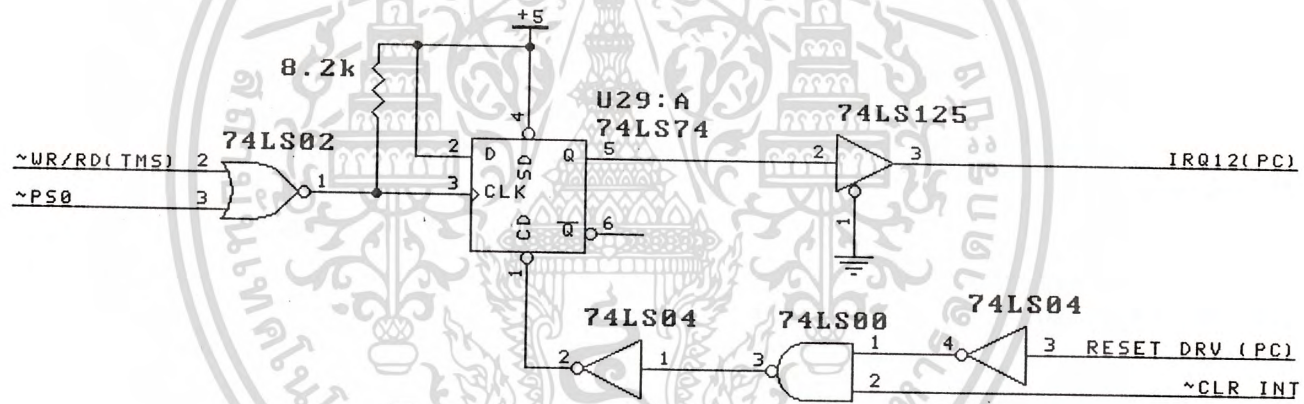
ADDRESS AND DAT BUS SELECTOR

รูปวงจรแผ่นที่ 3 วงจรสวิตซ์เลือก Address และ Data Bus



PC ADDRESS DECODER

รูปวงจรแผ่นที่ 5 วงจร decode หน่วยความจำและพอร์ตของ PC



INTERRUPT PC CIRCUIT

รูปวงจรแผ่นที่ 6 และวงจรสร้างสัญญาณอินเตอร์รัปต์

การใช้งานการ์ด

การ์ดที่สร้างขึ้นนี้จะไม่สามารถทำงานได้ถ้าหากขาดโปรแกรมควบคุมการทำงาน ซึ่งโปรแกรมที่ใช้สำหรับการ์ดนี้จะแบ่งได้เป็น 2 ส่วน ส่วนแรกคือโปรแกรมการประมวลผลของตัว CPU TMS บนการ์ด และอีกส่วนคือ โปรแกรมควบคุมการทำงานของการ์ดซึ่งเป็นโปรแกรมบน PC โดยจะกล่าวในรายละเอียดต่อไป

โปรแกรมการประมวลผลของ TMS

เนื่องจากบนการ์ดนี้จะมี CPU ซึ่งทำหน้าที่ในการประมวลผลอยู่ การที่จะให้ CPU ใดๆ ก็ตามสามารถทำงานได้ หัวใจสำคัญก็คือโปรแกรมการทำงานของ CPU ดังนั้นการที่จะให้ CPU ทำงานประมวลผลอะไรก็ต้องเขียนโปรแกรมขึ้นมาสำหรับงานนั้นๆ สำหรับปัญหานี้พบที่ใช้ CPU เบอร์ TMS 320C25 การเขียนโปรแกรมต้องเขียนด้วยภาษาแอสเซมบลี และด้วยลักษณะพิเศษเฉพาะด้านของ CPU เบอร์นี้ ซึ่งมีการทำงานของแต่ละคำสั่งแปลกไปจาก CPU ทั่วไปการออกแบบโปรแกรมจึงค่อนข้างยาก เพื่อความสะดวกในการพัฒนาโปรแกรมต่อไปจึงได้ออกแบบให้หน่วยความจำบนการ์ดเป็น RAM ทั้งหมด คือทั้ง Data และ Program Memory แทนที่จะเป็นเพียงเฉพาะ Data Memory และยังติดต่อกับ PC ได้เหมือนกันด้วย ดังนั้นโปรแกรมของ TMS จึงสามารถโหลดผ่านทาง PC ได้ นอกจากนี้ยังสามารถพัฒนาโปรแกรมบนเครื่อง PC ได้ด้วย ซึ่งเป็นการเพิ่มประสิทธิภาพอย่างสูงสำหรับการพัฒนาโปรแกรมของ TMS

ในการพัฒนาโปรแกรม จะใช้วิธีเขียนโปรแกรมในรูปของ Text ไฟล์ โดยใช้โปรแกรมอิดีเตอร์ใดๆ จากนั้นก็ใช้โปรแกรมคอมไพเลอร์ทำการคอมไพล์ให้เป็น Binary ไฟล์ ซึ่งเป็นไฟล์ที่เก็บข้อมูลในรูปของภาษาเครื่องของ TMS ในการใช้งานก็จะโหลด Binary ไฟล์เหล่านี้ลงในหน่วยความจำ Program Memory เมื่อ TMS ประมวลผลก็จะอ่านคำสั่งจาก Program Memory ไปทำงาน จะเห็นได้ว่าใช้ PC เป็นตัวจัดการทั้งหมด ซึ่งสะดวกมากและสามารถแก้ไขโปรแกรมได้ง่าย สำหรับคอมไพเลอร์ที่ใช้ได้แก่ C32 ซึ่งจะแปลได้เป็นไฟล์ .HEX จากนั้นก็ใช้โปรแกรมอื่นเช่น HEXBIN2.EXE แปลงไฟล์ .HEX เป็น Binary ไฟล์ .BIN

สำหรับตัวโปรแกรมของ TMS นั้น ก็เป็นโปรแกรมที่เขียนเพื่อให้ TMS ทำการประมวลผลแบบต่างๆ แล้วแต่ผู้ออกแบบโปรแกรม แต่สิ่งหนึ่งที่ต้องมีคือในตอนท้ายของโปรแกรมหลังประมวลผลเสร็จ จะต้องทำคำสั่ง OUT พอร์ต 0 ด้วยค่าอะไรก็ได้ เพื่อบอกแก่ PC ว่าเสร็จแล้ว

โปรแกรมควบคุมการทำงานบนการ์ด

การทำงานของการ์ดจะทำงานอะไรนั้น ทุกอย่างจะอยู่ในการควบคุมของ PC หมด ดังนั้นการใช้งานการ์ดก็คือการควบคุมผ่านทาง PC ซึ่งจะต้องทราบถึงขั้นตอนต่างๆ จึงจะสามารถควบคุมให้ทำงานได้ถูกต้อง ในการควบคุมการทำงานสามารถแบ่งได้เป็น 2 ชนิด คือ

- การโหลดข้อมูลหรือโปรแกรมระหว่างหน่วยความจำบนการ์ดกับ PC
- การ RUN โปรแกรมบนการ์ด

1. การโหลดข้อมูลและโปรแกรมระหว่างหน่วยความจำบนการ์ดกับ PC

เป็นการโหลดข้อมูลและโปรแกรมลง RAM ซึ่งมีขั้นตอนดังนี้

7	6	5	4	3	2	1	0
RST	-	-	PC/TMS	-	BKEN	BKSELO	BKSEL1

บิต 0,1 Blank Select 0,1 เป็นบิตเลือกแ่งค์ของ RAM บนการ์ด

บิต 1 บิต 0

0 0 Data Memory ส่วนล่าง 0000-7FFFH

0 1 Data Memory ส่วนบน 8000-FFFFH

1 0 Program Memory ส่วนล่าง 0000-7FFFH

1 1 Program Memory ส่วนบน 8000-FFFFH

บิต 2 Blank Enable กำหนดให้ PC ติดต่อกับ RAM ได้หรือไม่ เช็ทเป็น "1" คือให้ติดต่อกได้ เป็น "0" ไม่ให้ติดต่อก

บิต 4 ~PC/TMS เป็นบิตที่กำหนดสวิตซ์เลือกระหว่าง PC กับ TMS ให้ติดต่อกับ RAM เช็ทเป็น "1" คือติดต่อกับ TMS เป็น "0" คือติดต่อกับ PC

บิต 7 Reset TMS เป็นบิตควบคุม TMS ให้ทำงานหรือไม่ โดยให้ ถ้า Reset เป็นสัญญาณควบคุม ถ้าต้องการให้ทำงานก็เช็ทเป็น "1" ถ้าไม่ต้องการให้ทำงานก็เช็ทเป็น "0"

ตารางที่ 5.1 แสดงหน้าที่ของแต่ละบิตใน Control port (port 300H)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.1 สวิตช์ให้ RAM ติดต่อกับ PC และเลือกแบริ่งค้ที่ต้องการติดต่อด้วย โดยส่งค่าสั่งควบคุมไปที่ Control port ซึ่งแต่ละบิตมีหน้าที่อย่างไร แสดงในตารางที่ 5.1 สำหรับขั้นตอนนี้จะเซ็ตให้ บิต 7 เป็น "0" คือ reset TMS อยู่ , บิต 4 เป็น "0" เพื่อเลือกให้ RAM ติดต่อกับ PC บิต 2 เป็น "1" เป็นการ enable ให้ PC ติดต่อกับ RAM ได้ และ บิต 0 และ 1 จะเซ็ตตามแบริ่งค้ที่ต้องการจะติดต่อด้วย คือ ถ้าต้องการติดต่อกับหน่วยความจำส่วนใดก็เซ็ตแบริ่งค้ให้ตรงกัน สำหรับ bit อื่นไม่ได้ใช้จะเซ็ตเป็นอะไรก็ได้ แต่สำหรับกรณีนี้จะเซ็ตเป็น 0 หมด ดังนั้น ค่าที่จะส่งไปก็คือ 000001xxb โดยค่า xx เป็นอะไรก็ได้ แล้วแต่จะเซ็ตเป็นแบริ่งค้ใด

1.2 หลังจากผ่านขั้นตอนแรกแล้ว ก็จะสามารถติดต่อกับ RAM บนการ์ดได้ โดยผ่านทางหน่วยความจำของ PC ในบล็อกที่ map ไว้ เช่น map ที่บล็อก E ก็ติดต่อด้วยทาง address E000:0000H-E000:FFFFH ในการจะโหลดข้อมูลลง RAM ก็โหลดข้อมูลไปยังหน่วยความจำช่วงบล็อก E หรือถ้าจะอ่านข้อมูลก็โหลดข้อมูลจากหน่วยความจำบล็อก E

ในการโหลดข้อมูลสิ่งที่จะต้องระวัง คือตำแหน่งของข้อมูลใน PC กับตำแหน่งของข้อมูลใน TMS คือการย้ายข้อมูลต่างๆจะต้องสัมพันธ์กับตำแหน่งที่ TMS จะทำงาน เพื่อที่จะได้ทำงานได้ถูกต้อง จึงต้องเข้าใจความสัมพันธ์ระหว่างตำแหน่งข้อมูลใน PC กับ TMS ก่อน

จากที่ทราบแล้วว่า ข้อมูลใน PC 1 ตำแหน่ง เป็นแบบ 8 บิต ในขณะที่ TMS เป็น 16 บิต ดังนั้นในการอ้างอิงข้อมูลใน TMS 1 ตำแหน่ง PC จะต้องใช้ 2 ตำแหน่ง สามารถเขียนความสัมพันธ์ได้ดังนี้

กรณีเป็นหน่วยความจำส่วนล่าง

$$\text{Address บน PC เก็บไปที่ต่ำ} = \text{E0000H} + \text{Address บน TMS} * 2$$

$$\text{Address บน PC เก็บไปที่สูง} = \text{E0001H} + \text{Address บน TMS} * 2$$

กรณีเป็นหน่วยความจำส่วนบน

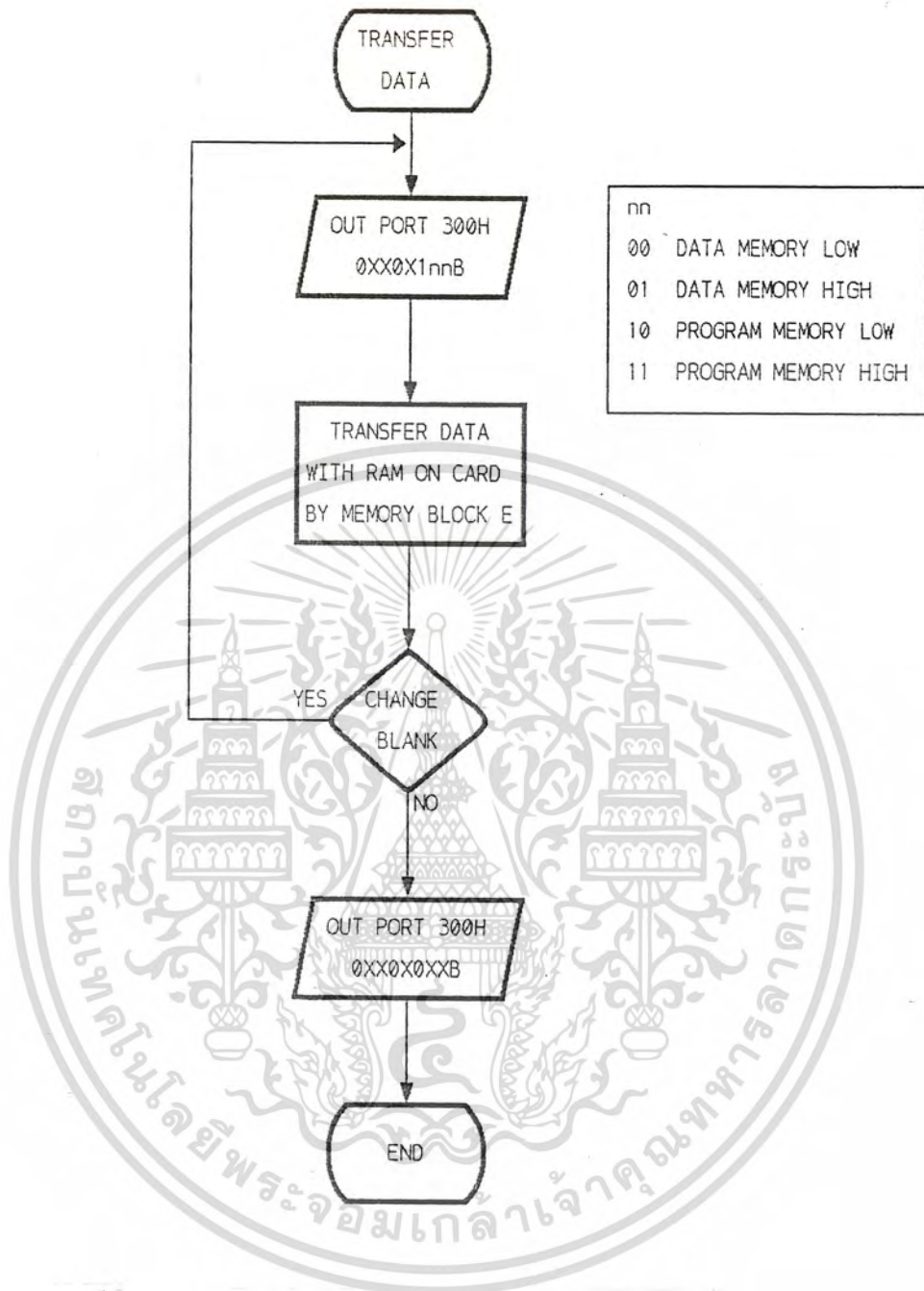
$$\text{Address บน PC เก็บไปที่ต่ำ} = \text{E0000H} + (\text{Address บน TMS} - 8000\text{H}) * 2$$

$$\text{Address บน PC เก็บไปที่สูง} = \text{E0001H} + (\text{Address บน TMS} - 8000\text{H}) * 2$$

จากสมการ คือ Address 0000-FFFFH จะซ้ำกับ 8000H-FFFFH แต่อยู่ต่างแบริ่งค้กัน

1.3 หลังจากทำการโหลดข้อมูลเสร็จ ถ้าต้องการเปลี่ยนแบริ่งค้ ก็ส่งค่าสั่งไปที่ Control port ด้วยค่าสั่งเดียวกับในข้อ 1.1 เพียงแต่เปลี่ยนค่า บิต 0 และ 1 ตามแบริ่งค้ที่ต้องการติดต่อกใหม่

1.4 หลังจากสิ้นสุดการโหลดข้อมูลบน RAM แล้ว ให้ส่งค่า 00000000B ไปยัง Control port คือ เซ็ตบิต 3 ให้เป็น "0" เพื่อขอกเลกมิให้ PC ติดต่อกับ RAM บนการ์ด บิต 0 และ 1 เป็นอะไรก็ได้ ส่วนบิตอื่นๆเหมือนเดิม



รูปที่ 5.6 Flow Chart ของการโหลดข้อมูลและโปรแกรมลง RAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การ RUN โปรแกรมบนการ์ด

ในการ RUN โปรแกรมจะกระทำหลังจากที่มีการโหลดตัวโปรแกรมและข้อมูลต่างๆลง RAM เสร็จสิ้นแล้ว ซึ่งการ RUN โปรแกรม นอกจากจะมีการควบคุมตัวการ์ดแล้ว ยังต้องเซ็ต อินเทอร์เน็ตของ PC ด้วย เพื่อให้รับรู้การอินเทอร์เน็ต มีขั้นตอนดังนี้

2.1 จัดการเกี่ยวกับอินเทอร์เน็ตบน PC คือ

- อ่านาเบิลฮาร์ดแวร์ อินเทอร์เน็ตเบอร์ 12 ที่ตัวอินเทอร์เน็ตคอนโทรลเลอร์ โดยการเซ็ตค่าบิตที่ 4 ของพอร์ต A1H ให้เป็น "0" (พอร์ตของอินเทอร์เน็ตคอนโทรลเลอร์ตัวที่ 2)

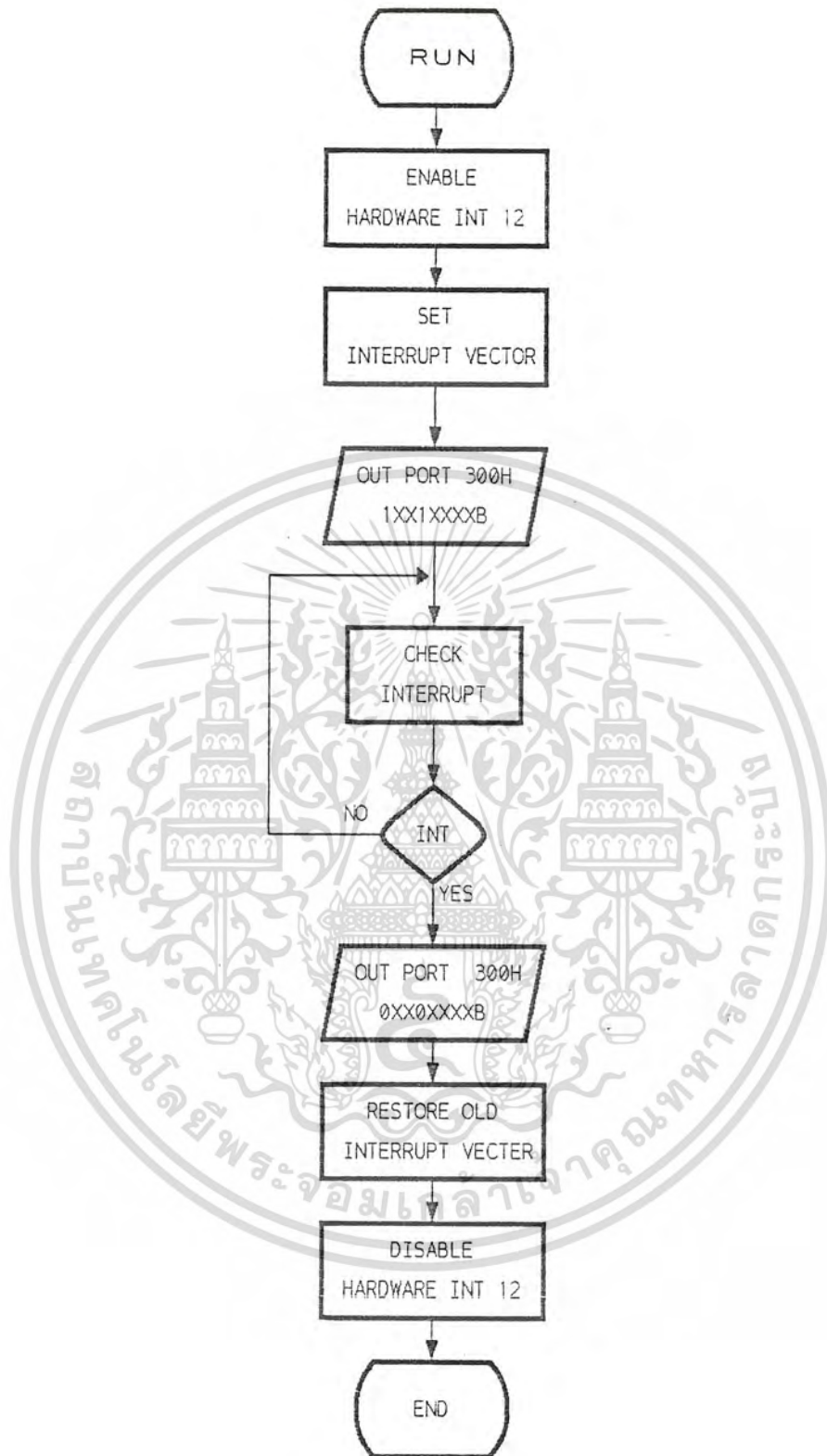
- เซ็ตอินเทอร์เน็ตเวคเตอร์ ของฮาร์ดแวร์ อินเทอร์เน็ตเบอร์ 12 ซึ่งตรงกับอินเทอร์เน็ตที่ 74 บน PC ซึ่งมีเวคเตอร์อยู่ที่ Address 1D0-1D3H โดยเซ็ตไปที่โปรแกรมจัดการอินเทอร์เน็ตที่เขียนขึ้นใหม่

2.2 ส่งคำสั่งไปที่ Control port เพื่อให้ CPU เริ่มประมวลผล ซึ่งคำสั่งที่ส่งไป คือ 1001000B มีความหมายดังนี้ ให้บิต 7 ซึ่งเป็นสัญญาณ Reset TMS เป็น "1" คือ เริ่มให้ TMS ประมวลผล ขณะเดียวกัน บิต 4 ซึ่งเป็นตัวสวิตช์ RAM ระหว่าง PC กับ TMS ก็จะต้องเป็น "1" เช่นกัน คือให้สวิตช์ไปที่ TMS สำหรับบิตอื่นๆจะไม่มีผลสำหรับกรณีนี้ ซึ่งปกติจะเซ็ตเป็น "0" ทั้งหมด

2.3 หลังจากเริ่มให้ TMS ประมวลผล ทางด้าน PC ก็จะคอยสัญญาณตอบกลับจาก TMS ซึ่งจะอยู่ในรูปของอินเทอร์เน็ต เมื่อใด PC ได้รับอินเทอร์เน็ต ก็หมายถึง TMS ประมวลผลเสร็จแล้ว ก็จะทำการหยุดการทำงานของ TMS โดยส่งคำสั่งควบคุมการไปที่ Control port set ให้บิตที่ 7 และ 4 เป็น "0"

2.4 จัดการอินเทอร์เน็ตของ PC ให้เหมือนเดิม ได้แก่ ค่าที่อินเทอร์เน็ตคอนโทรลเลอร์ซึ่งปกติจะ Disable อยู่ และคืนค่าอินเทอร์เน็ตเวคเตอร์เก่าด้วย

2.5 หลังจากขั้นตอนที่ 2.4 ก็เป็นการจบขั้นตอนที่จะให้ TMS ประมวลผล จากนั้นการนำข้อมูลที่ได้ไปใช้ ก็จะอยู่ในหัวข้อแรก คือ การโหลดข้อมูล



รูปที่ 5.7 Flow Chart ของการสั่งให้การ์ดประมวลผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

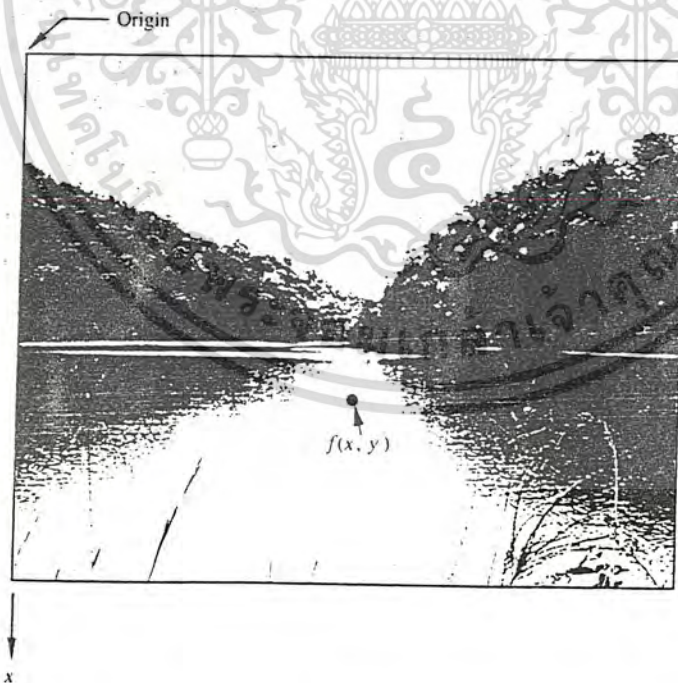
บทที่ 6

การประยุกต์ใช้งานและการทดลอง

การประยุกต์ใช้งานในปริศยานิพจน์นี้ จะทำการศึกษาทดสอบโดยการเขียนโปรแกรมการทำ FFT (FAST FOURIER TRANSFORM) 2 มิติ ซึ่งการทำ FFT นั้นสามารถที่จะประยุกต์ใช้กับการประมวลผลอื่นได้มากมาย โดยจะใช้ INPUT เป็นสัญญาณภาพต่างๆ และทำการเปรียบเทียบผลลัพธ์ที่ได้จากการประมวลผลระหว่างการใช้ TMS320C25 ประมวลผล กับ การใช้ COMPUTER ประมวลผล ซึ่งเป็นการเปรียบเทียบทั้งทางด้านความเร็วและความผิดพลาดข้อมูล

6.1 ลักษณะของสัญญาณภาพ

สัญญาณภาพประกอบด้วยจุดขนาดเล็กๆจำนวนมาก มาเรียงติดต่อกันตามแนวตั้งและแนวนอน โดยในแต่ละจุดจะแสดงสีหนึ่งสี แต่ละจุดเรียกว่าจุดภาพ (PIXEL) เพื่อให้ง่ายแก่การเปรียบเทียบ จึงใช้ลักษณะของภาพเป็นภาพขาวดำ ซึ่งสามารถเปรียบเทียบก่อนและหลังประมวลผลได้ง่าย ในแต่ละจุดภาพที่เป็นขาวดำจะแบ่งระดับความเข้มเป็น 64 ระดับ (0-63) โดยค่า 0 จะแทนจุดมืด และค่า 63 จะแทนจุดสว่างที่สุด



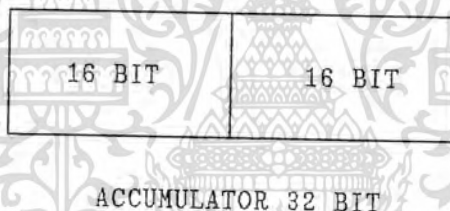
รูปที่ 6.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 ระบบตัวเลข

ระบบตัวเลขที่ใช้ใน ระบบ DIGITAL เป็นแบบเลขฐานสอง ซึ่งโดยทั่วไปนิยมใช้กันอยู่ 2 แบบ คือ FIXED POINT FORMAT และ FLOATING POINT FORMAT โดยแบบ FLOATING POINT นั้นสามารถแสดงค่าได้ละเอียดและแม่นยำกว่า เหมาะสำหรับการคำนวณที่ต้องใช้ความละเอียดหรือมีจุดทศนิยมมาก แต่อย่างไรก็ตามการคำนวณจะยุ่งยาก ช้าขึ้น และสิ้นเปลืองหน่วยความจำมากกว่า

การคำนวณ FFT นั้นค่าส่วนใหญ่เป็นจุดทศนิยม เพื่อให้ได้ผลลัพธ์ของการคำนวณที่ละเอียด จึงต้องคำนวณส่วนที่เป็นจุดทศนิยมด้วย ในส่วนของการประมวลผลโดยใช้คอมพิวเตอร์ เพื่อที่จะเป็นต้นแบบในการเปรียบเทียบการคำนวณจะใช้แบบ FLOATING POINT ทั้งหมด ในส่วนการประมวลผลด้วย TMS 320C25 เพื่อไม่ให้ยุ่งยากมากนักและประหยัดหน่วยความจำ จึงใช้การคำนวณโดยมี 32 บิต ประกอบด้วย ส่วนจำนวนเต็ม 16 บิต และส่วนหลังจุดทศนิยมอีก 16 บิต



และเนื่องจาก TMS320C25 มี ACCUMULATOR ขนาด 32 บิต จึงเพียงพอในการคำนวณ แต่ขนาดของ DATA แต่ละ ADDRESS เป็นแบบ 16 บิต ดังนั้นการเก็บค่าของผลลัพธ์จึงต้องใช้ 2 ADDRESS ซึ่งสิ้นเปลือง MEMORY มากขึ้นเป็น 2 เท่า การเก็บผลลัพธ์จึงเก็บเพียงส่วนที่เป็นจำนวนเต็มเท่านั้น ส่วนที่เป็นจุดทศนิยมจะตัดทิ้งไป ในส่วนของจำนวนลบนั้น ทั้งคอมพิวเตอร์และ TMS320C25 จะใช้แบบ 2'S COMPLEMENT

ตัวอย่างการคำนวณที่ใช้ใน TMS 320C25

$\sin 45 = 0.7071$ เมื่อแปลงเป็นจุดทศนิยมแบบเลขฐาน 16 ได้ B504

ต้องการหาค่า $5 + (0.7071 * 2)$

DECIMAL	HEXADECIMAL
$5 + (0.7071 * 2)$	$5 + (.B504 * 2)$
$5 + (1.4142)$	$5 + (1.6A08)$
6.4142	6.6aA8

ซึ่ง 0.4142 เมื่อแปลงเป็นจุดทศนิยมฐาน 16 ได้ 6A08

6.3 ค่าการคำนวณ FFT

ในการแปลง DFT สมมติให้สัญญาณที่ส่งมาทำการครบคาบ จากการสุ่ม N ครั้งพอดี สมการของ DFT N จุดจะได้เป็น

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

เมื่อ W_N คือ PHASE หรือ TWIDDLE FACTOR

k คือ ลำดับของช่วงความถี่

จากสมการของ butterfly unit

$$\text{คือ } P_{m+1} = P_m + W_N^k \cdot Q_m$$

$$Q_{m+1} = P_m - W_N^k \cdot Q_m$$

$$\text{ซึ่ง } W_N^k = e^{-jk(2\pi/N)}$$

$$= \cos(X) - j\sin(X) \quad \text{เมื่อ } X = (2\pi/N)$$

โดย P_m และ Q_m สามารถแบ่งเป็นส่วนจริงและส่วนจินตภาพ

$$P_m = PR + PI$$

$$Q_m = QR + QI$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะได้

$$\begin{aligned}
 P_{m+1} &= PR + jPI + [QR\cos(X)+QI\sin(X)] + j[QI\cos(X)-QR\sin(X)] \\
 &= [PR+QR\cos(X)+QI\sin(X)] + j[PI+QI\cos(X)-QR\sin(X)] \\
 Q_{m+1} &= PR + jPI - [QR\cos(X)+QI\sin(X)] - j[QI\cos(X)-QR\sin(X)] \\
 &= [PR-QR\cos(X)-QI\sin(X)] + j[PI-QI\cos(X)+QR\sin(X)]
 \end{aligned}$$

พิจารณาค่าสูงสุดที่เป็นไปได้ของ OUTPUT จาก $PR+QR\cos(X)+QI\sin(X)$

$$1 + 1\sin 45 + 1\cos 45 = 2.414213562$$

ซึ่งเกิน 1 ทำให้เกิดการ OVERFLOW ได้ ดังนั้นการทำให้ BUTTERFLY ในแต่ละ stage จึงควรมีการ scale ค่าลงโดยการหาร 2 หรือควรจะมีการ scale ค่าลงบ้าง ถ้าการทำ FFT มี M stage OUTPUT ที่ได้จะถูก scale ลงเท่ากับ $2^M = N$ ในกรณีที่ทำ FFT N จุด

การทำ BIT-REVERSE

หากพิจารณาการทำ FFT 8 จุด จะเห็นว่า OUTPUT ที่ได้จะสลับตำแหน่งไป แต่ถ้าเราทำ BIT-REVERSE ที่ INPUT ก่อน ก็จะได้ OUTPUT เรียงตามลำดับตั้งทฤษฎีที่ได้กล่าวไว้ในบทที่ 2 การประมวลผลด้วย TMS 320C25 นั้นจะมีคำสั่งพิเศษในการทำ BIT REVERSE โดยเฉพาะซึ่งทำให้สามารถการประมวลผลได้รวดเร็วยิ่งขึ้น

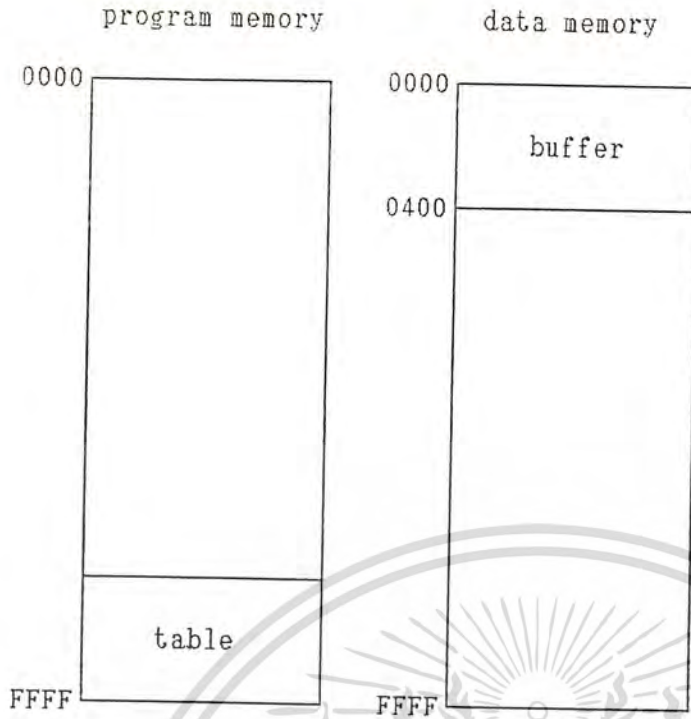
6.4 การทำ FFT โดยใช้ TMS320C25

โดยขอแบ่งเป็น 3 ส่วน คือ

- INPUT
- PROCESS
- OUTPUT

-INPUT เมื่อ computer ทำการ load data ภาพ (ซึ่งจะมีเฉพาะส่วน real ส่วน image เป็น 0) ลงใน data memory และ load program ลงในส่วน program memory จะเห็นได้ว่าค่า P_n ซึ่งประกอบด้วย $\cos(X)$ และ $\sin(X)$ ซึ่งเป็นค่าจุดทศนิยม เพื่อให้ง่ายต่อการเขียนโปรแกรม จึงคำนวณค่า $\cos(X)$ และ $\sin(X)$ ที่ต้องใช้งานไว้ก่อน เมื่อทำการ load program ก็จะเก็บค่าเหล่านี้เป็น table เอาไว้ เมื่อต้องการใช้ตัวใดก็ load ค่ามาใช้ได้เลย ในส่วนของ TMS320C25 จะจัดให้ data memory address 0-400h เป็น ram ภายใน ที่ทำงานได้รวดเร็ว ซึ่งมีอยู่ในตัว TMS320C25 อยู่แล้ว ดังนั้นจึงใช้ส่วนนี้เป็น buffer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



-PROCESS การ process จะเริ่มเมื่อขา reset ของ TMS320C25 เป็น high เมื่อทำการ load data ภาพมาแล้ว data ภาพ จะเริ่มต้นที่ address 400h เป็นต้นไป ถ้า data ภาพมีขนาด 64x64 จะใช้ memory ส่วน real 4kword และส่วน image อีก 4kword รวมเป็น 8kword ถ้าเป็น 128x128 จะใช้ memory ทั้งหมด 32kword การทำ FFT 2 มิติ สามารถทำได้โดยการคำนวณ FFT ในแนวหนึ่งแนวใดก่อนแล้วทำการคำนวณ FFT อีกแนวหนึ่งต่อไป ดังรูปที่ 6.2 เป็นการคำนวณ FFT 2 มิติ ขนาด X x Y จุด โดยในขั้นแรกจะคำนวณ FFT X จุดตามแนวนอนก่อน จำนวน Y ครั้ง จากนั้นจะคำนวณ FFT Y จุดตามแนวตั้ง จำนวน X ครั้ง

หลังจากเราทำ FFT เรียบร้อยแล้ว data ที่ได้จะมีทั้งส่วน real และ image ซึ่งจะนำไปหาขนาดของจำนวนเชิงซ้อนและนำไปแสดงผลต่อไป การ PROCESS ทั้งหมดสามารถเขียนเป็น flow chart ซึ่งใช้ทั้งใน TMS 320C25 และ ในคอมพิวเตอร์ ดังรูปที่ 6.3

-OUTPUT ค่าที่ได้จากการหาขนาดของจำนวนเชิงซ้อน จะมีค่าที่เป็นไปได้มากมาย แต่ในการ display จะแสดงเป็นภาพขาวดำ 64 ระดับ ซึ่งทำให้ค่าบางค่าไม่สามารถแสดงได้ จึงต้องมีการเทียบค่าโดยให้ค่าผลลัพธ์สูงสุด แสดงเป็นจุดสว่างที่สุด และผลลัพธ์ที่น้อยกว่า แสดงด้วยจุดสว่างน้อยลงตามลำดับ โดยผลลัพธ์ที่น้อยที่สุดแสดงด้วยจุดมืด การ display ก็จะมีประสิทธิภาพมากขึ้นและง่ายแก่การเปรียบเทียบจากที่กล่าวมา 3 ส่วนนั้น ส่วนที่ใช้ TMS320C25 ประมวลผล คือส่วน PROCESS เพียงส่วนเดียว นอกนั้นจะใช้ computer ควบคุมทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.5 การคำนวณขนาดของจำนวนเชิงซ้อน

จากสูตรการหาขนาดของจำนวนเชิงซ้อน

$$|Z| = \sqrt{a^2 + b^2}$$

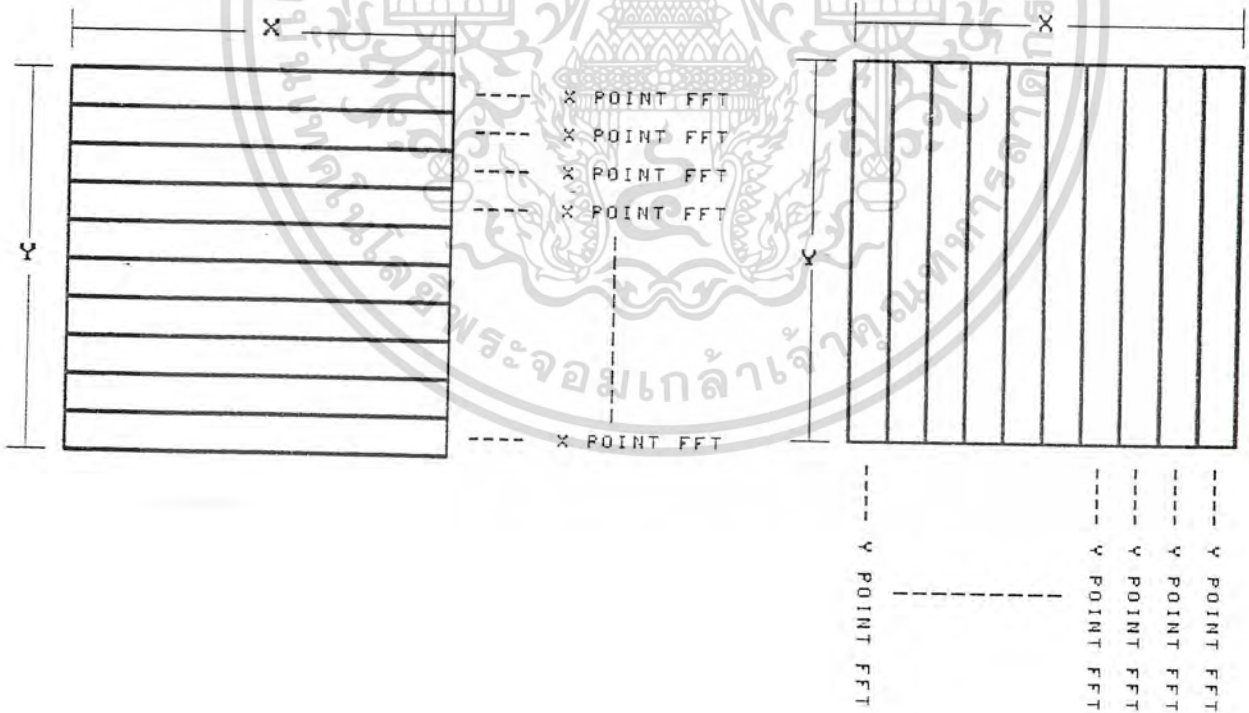
ซึ่ง $Z = a + bi$

จะเห็นได้ว่าการหาขนาดจำนวนเชิงซ้อนมีด้วยกัน 3 ขั้นตอน

-การหาค่ากำลังสอง ใน TMS 320C25 จะมีคำสั่งสำหรับการหาค่ากำลังสองอยู่แล้ว สามารถเรียกใช้ได้เลย

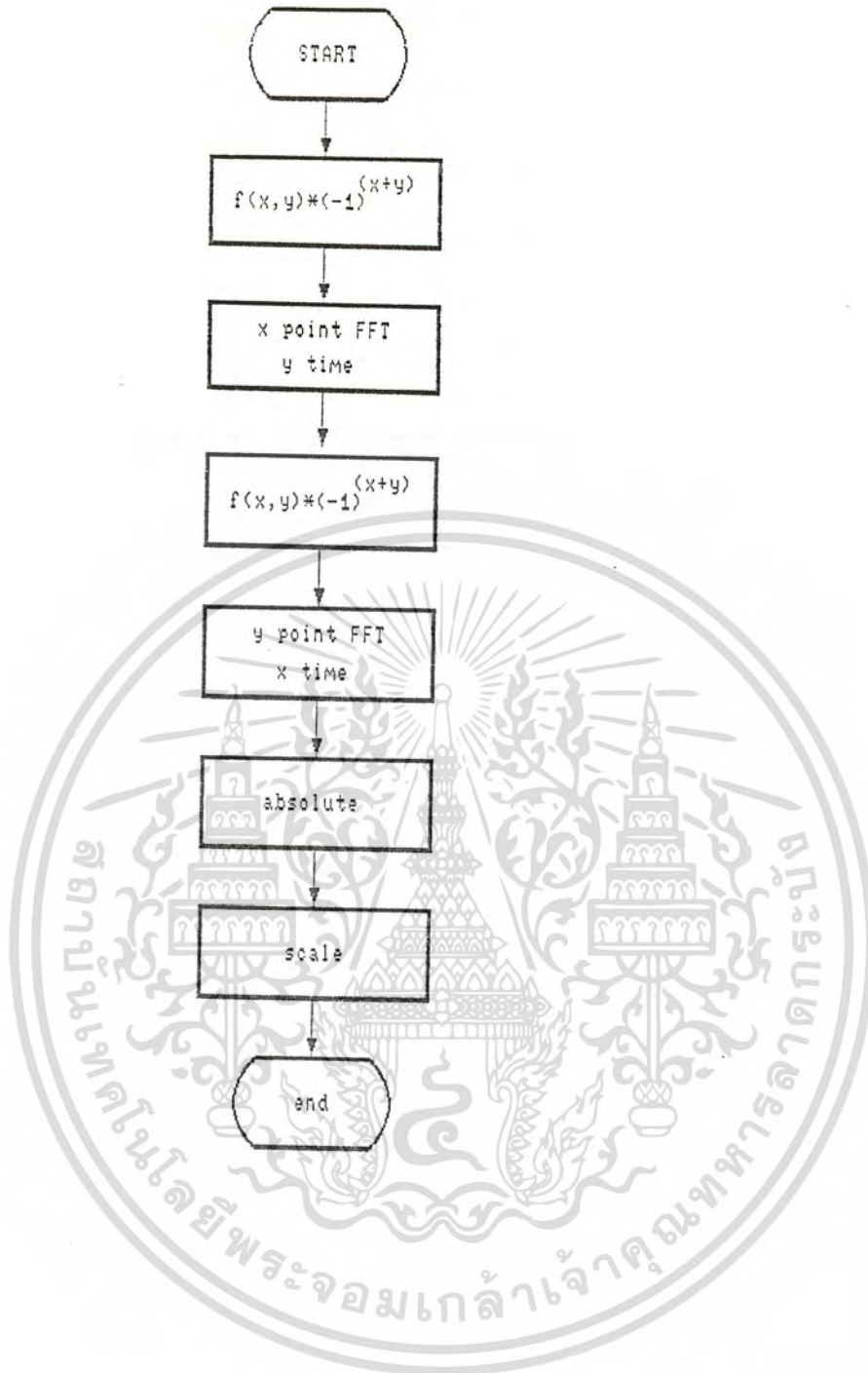
-การบวก ใน TMS 320C25 จะมีคำสั่งสำหรับการบวกอยู่แล้ว สามารถเรียกใช้ได้เลย

-การหารากที่สอง จากสมการทางคณิตศาสตร์ ถ้า $a * a = z$ โดยที่ $a, z > 0$ จะได้ว่า a จะเป็นค่ารากที่สองของจำนวนเต็มบวกของ z ดังนั้นในการหาจะใช้หลักการทาง Numerical ในการประมาณค่า a และทำการคำนวณเปรียบเทียบ $a*a$ กับ z จะเปลี่ยนค่า a ให้ $a*a$ เข้าใกล้ z จนกว่าจะได้ผลลัพธ์ตามต้องการ สามารถเขียน flow chart ได้ดังรูป 6.4



รูปที่ 6.2 การทำ FFT สองมิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.3 FLOW CHART ที่ใช้ในการทำ FFT 2 มิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

ผลการทดลอง และ สรุป

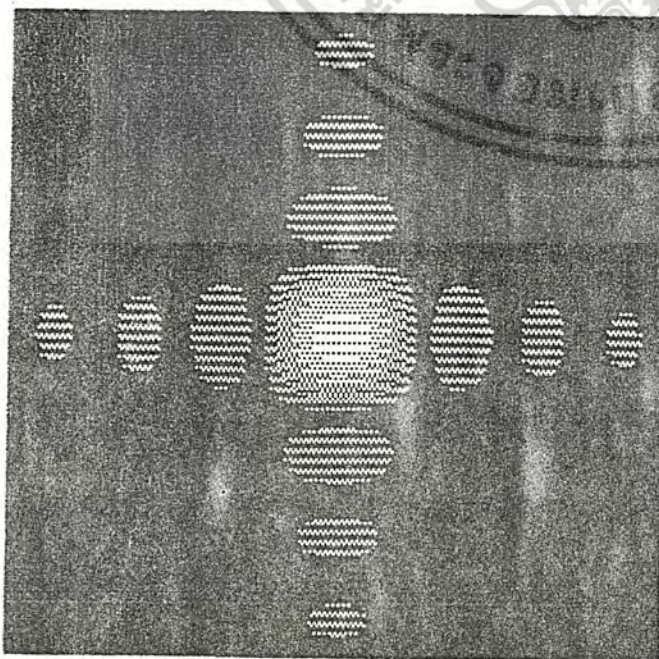
ผลการทดลอง

จากการแปลง FFT ของภาพต่าง ๆ ขนาด 128*128 จุด โดยใช้โปรแกรมภาษา assembly ของ TMS320C25 และโปรแกรมภาษาซี จะได้ผลการแปลงซึ่งสามารถแสดงได้ดัง

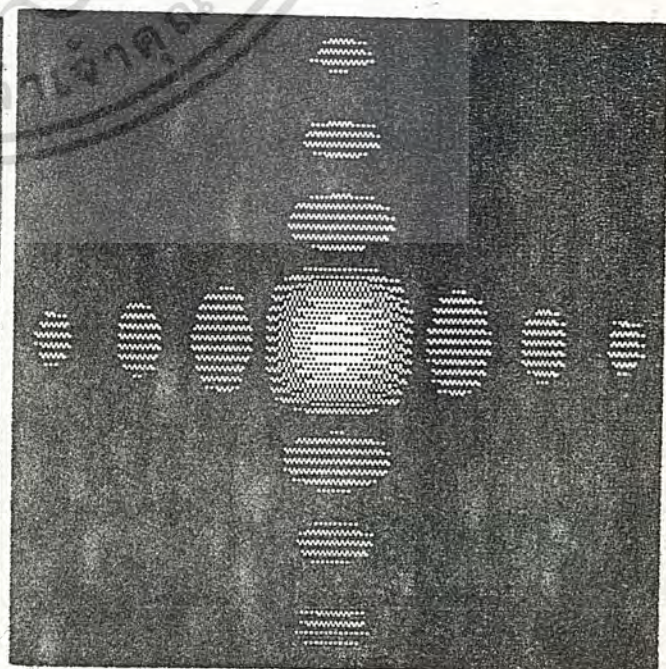
รูป



ภาพก่อนการแปลง FFT

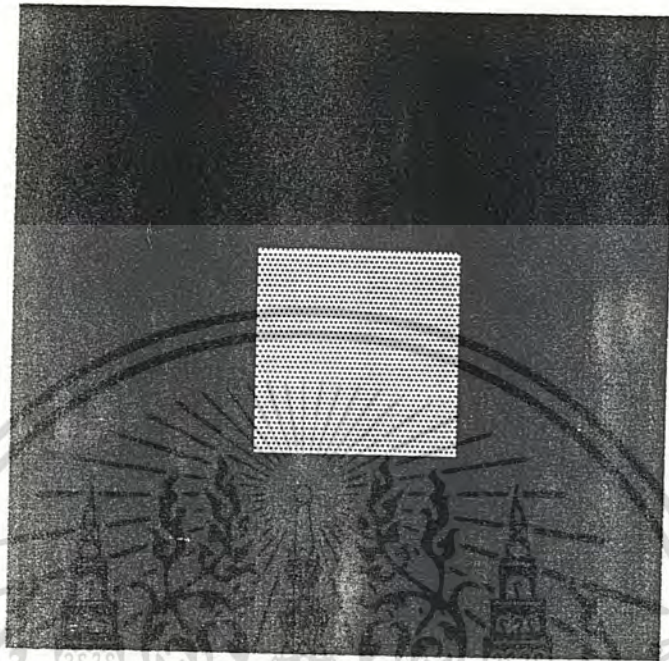


ภาพที่แปลง FFT โดยใช้โปรแกรมภาษาซี

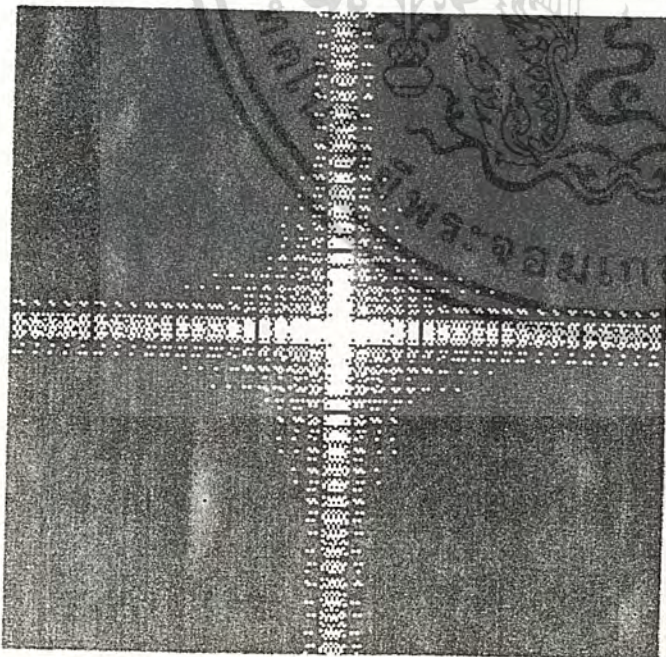


ภาพที่แปลง FFT โดยใช้ TMS320C25

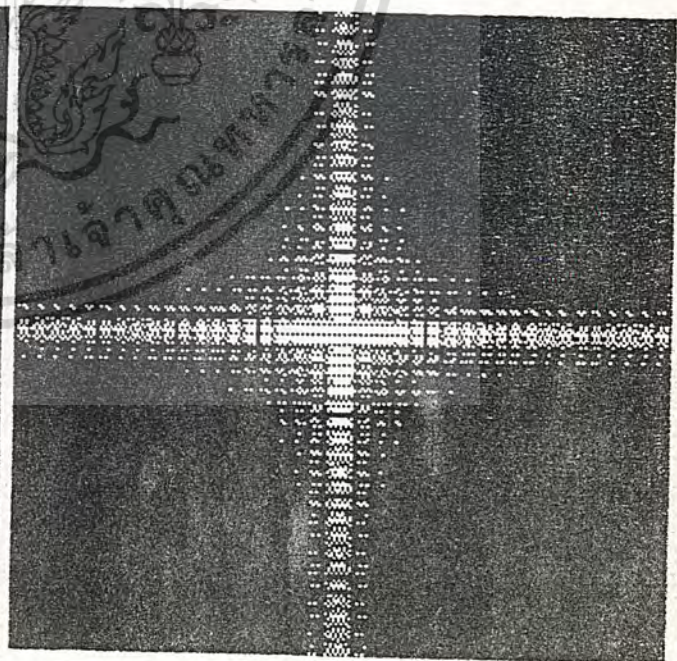
ไม่ว่ากรณีใดก็ตาม ผู้ใช้ต้องปฏิบัติตามเงื่อนไขการใช้งานและต้องอ้างอิงถึงเจ้าของลิขสิทธิ์ที่ปรากฏบนเว็บไซต์



ภาพก่อนการแปลง FFT

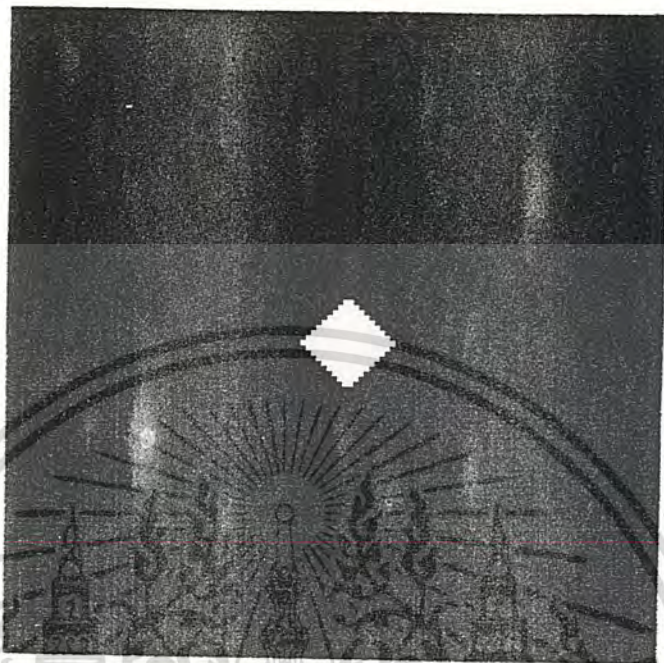


ภาพที่แปลง FFT โดยใช้โปรแกรมภาษาซี

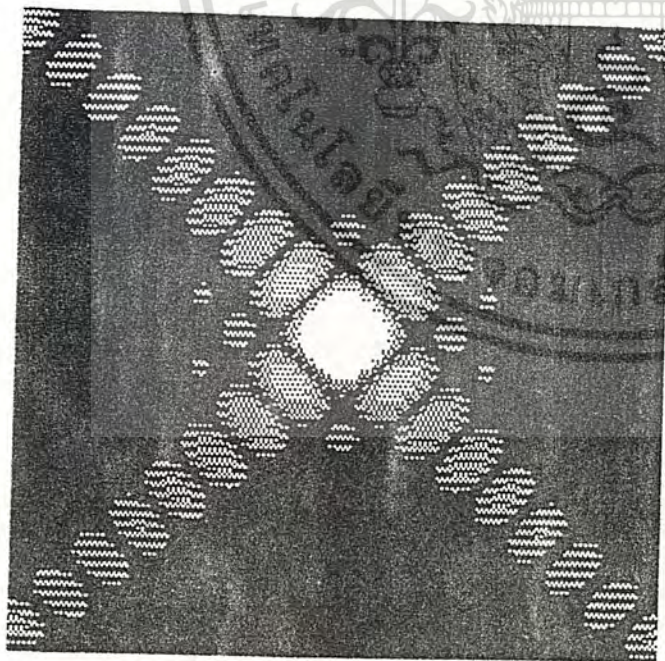


ภาพที่แปลง FFT โดยใช้ TMS320C25

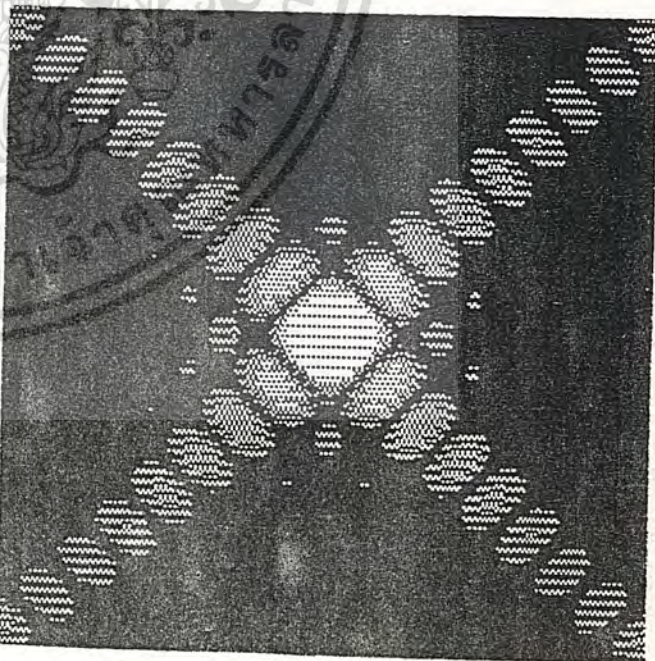
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพก่อนการแปลง FFT

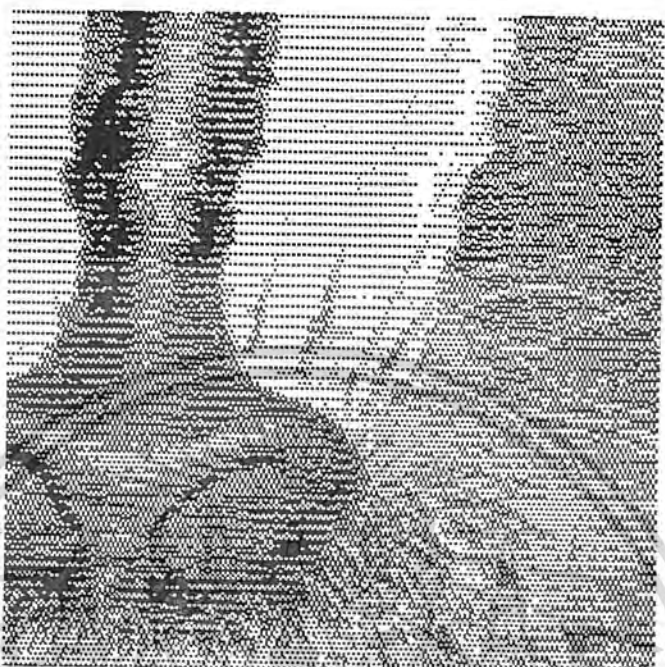


ภาพที่แปลง FFT โดยใช้โปรแกรมภาษาซี

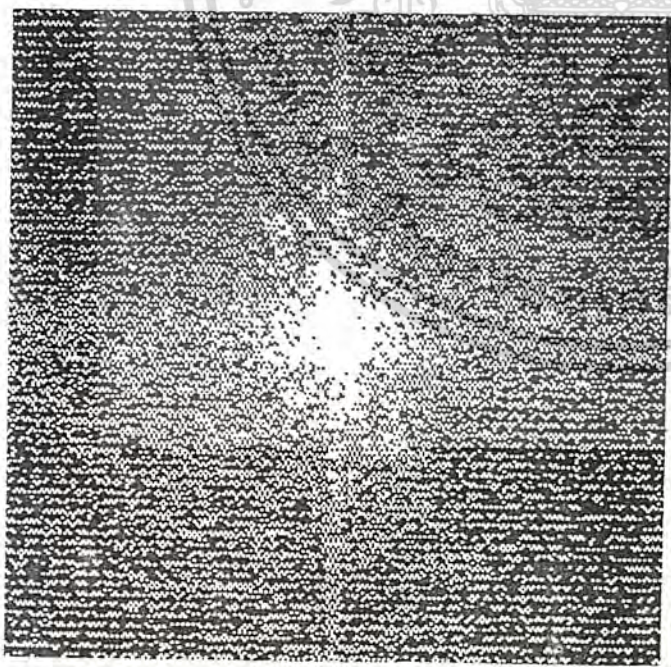


ภาพที่แปลง FFT โดยใช้ TMS320C25

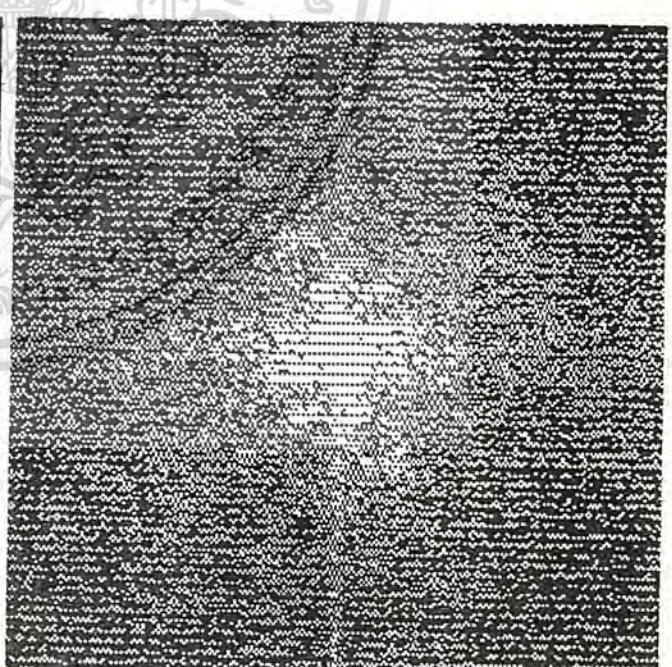
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพก่อนการแปลง FFT



ภาพที่แปลง FFT โดยใช้โปรแกรมภาษาซี



ภาพที่แปลง FFT โดยใช้ TMS320C25

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทางด้านความเร็วของการแปลง FFT ภาพขนาด 128*128 จุด การใช้การ์ด TMS ซึ่งเป็นโปรแกรมภาษา assembly ของ TMS320C25 จะใช้เวลาประมาณ 3 วินาที โดยไม่ขึ้นกับว่าคอมพิวเตอร์จะเป็นรุ่นใด และมี clock เท่าใด ส่วนการใช้โปรแกรมภาษาซี ความเร็วของการแปลง FFT จะขึ้นอยู่กับรุ่น และ clock ของคอมพิวเตอร์ โดยทั่วไปจะใช้เวลาประมาณ 3 - 10 วินาที

จากการทดลองกับเครื่องคอมพิวเตอร์รุ่นต่าง ๆ ในด้านความเร็วของการแปลง FFT สามารถแสดงได้ดังตาราง

รุ่นของคอมพิวเตอร์	เวลา (วินาที)	
	คอมพิวเตอร์	TMS320C25
486DX2-50	3.02	3.03
486DX-33	4.61	3.03
386DX-40	8.62	3.03
386SX	9.86	3.03

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สรุป และวิจารณ์

จากผลการทดลองจะเห็นได้ว่า การใช้การ์ด TMS ซึ่งเป็นการ์ดที่ใช้ทางด้าน DSP โดยเฉพาะ จะสามารถแปลง FFT ได้เร็วกว่าคอมพิวเตอร์ที่ใช้ภาษาซี ซึ่งความเร็วขึ้นอยู่กับรุ่น และ clock ของคอมพิวเตอร์ และถึงแม้ว่า คอมพิวเตอร์รุ่นใหม่ ๆ (เช่น 486DX2-50) จะสามารถทำงานได้เร็วเท่ากับการ์ด TMS แต่เมื่อเปรียบเทียบกับ clock แล้ว จะเห็นว่า การ์ด TMS ใช้ clock ต่ำกว่ามาก ซึ่งการ์ดในปริมาณนี้ใช้ในการ wire lab ทำให้จำกัด clock ที่ใช้ได้เพียง 20MHz และถ้าพัฒนาต่อไป โดยทำแผงวงจรพิมพ์จะทำให้สามารถใช้ clock สูงสุดของ TMS320C25 คือ 40MHz ก็จะทำให้ความเร็วในการแปลง FFT เพิ่มมากขึ้น



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. Texas Instrument "Second-Generation TMS 320 User's Guide", 1987
2. Rafael C. Gonzales, Richard E. Wood "Digital-Image Processing" Addison-Wesley Publishing Company, 1992.
3. E. Oran Brigham "The Fast Fourier Transform and Its Application" Prentice-Hall International, Inc., 1988.
4. Paul M. Embree, Bruce Kimble "C Language Algorithms for Digital Signal Processing " Prentice-Hall International, Inc., 1991.
5. สันภูวราวรรณ "การประมวลผลสัญญาณดิจิทัล" วารสารเซมิคอนดักเตอร์ อิเล็กทรอนิกส์ เล่ม 76-79 , ซีเอ็ดดูเคชั่น , 2530.
6. ไพศาล เตชะรัตนประเสริฐ "มม พีซีชาร์ตแวร์ ตอน การ์ดอินเตอร์เฟสอเนกประสงค์" วารสารเซมิคอนดักเตอร์ อิเล็กทรอนิกส์ เล่ม 104 , ซีเอ็ดดูเคชั่น , 2534
7. กนกวรรณ แซ่แข็ง , ชิตาพร พิทักษ์พรพันธ์ , เอี่ยมพร จารุกิตติยนต์ "การประยุกต์ใช้งาน การประมวลผลสัญญาณเชิงเลข" วิทยานิพนธ์ ปีการศึกษา 2534 ภาควิชา อิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สจล.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

;      MAIN PROGRAM FOR 128x128 POINT FFT

CPU      "320C2X.TBL"      ; CPU TABLE
HOF      "INT8"            ; HEX OUTPUT FORMAT
WDLN     2                  ; 2 BYTE WORD LENGTH
INCL     "128_7ST.MAC"
INCL     "FFT.MAC"

```

```

AR0:     EQU 0
AR1:     EQU 1
AR2:     EQU 2
AR3:     EQU 3
AR4:     EQU 4
AR5:     EQU 5
AR6:     EQU 6
AR7:     EQU 7

```

```

X0R:     EQU 00
X0I:     EQU 01
X1R:     EQU 02
X1I:     EQU 03
X2R:     EQU 04
X2I:     EQU 05
X3R:     EQU 06
X3I:     EQU 07
X4R:     EQU 08
X4I:     EQU 09
X5R:     EQU 10
X5I:     EQU 11
X6R:     EQU 12
X6I:     EQU 13
X7R:     EQU 14
X7I:     EQU 15
X8R:     EQU 16
X8I:     EQU 17
X9R:     EQU 18
X9I:     EQU 19
X10R:    EQU 20
X10I:    EQU 21
X11R:    EQU 22
X11I:    EQU 23
X12R:    EQU 24
X12I:    EQU 25
X13R:    EQU 26
X13I:    EQU 27
X14R:    EQU 28
X14I:    EQU 29
X15R:    EQU 30
X15I:    EQU 31
X16R:    EQU 32
X16I:    EQU 33
X17R:    EQU 34
X17I:    EQU 35
X18R:    EQU 36
X18I:    EQU 37
X19R:    EQU 38
X19I:    EQU 39
X20R:    EQU 40
X20I:    EQU 41

```



X21R:	EQU	42
X21I:	EQU	43
X22R:	EQU	44
X22I:	EQU	45
X23R:	EQU	46
X23I:	EQU	47
X24R:	EQU	48
X24I:	EQU	49
X25R:	EQU	50
X25I:	EQU	51
X26R:	EQU	52
X26I:	EQU	53
X27R:	EQU	54
X27I:	EQU	55
X28R:	EQU	56
X28I:	EQU	57
X29R:	EQU	58
X29I:	EQU	59
X30R:	EQU	60
X30I:	EQU	61
X31R:	EQU	62
X31I:	EQU	63
X32R:	EQU	64
X32I:	EQU	65
X33R:	EQU	66
X33I:	EQU	67
X34R:	EQU	68
X34I:	EQU	69
X35R:	EQU	70
X35I:	EQU	71
X36R:	EQU	72
X36I:	EQU	73
X37R:	EQU	74
X37I:	EQU	75
X38R:	EQU	76
X38I:	EQU	77
X39R:	EQU	78
X39I:	EQU	79
X40R:	EQU	80
X40I:	EQU	81
X41R:	EQU	82
X41I:	EQU	83
X42R:	EQU	84
X42I:	EQU	85
X43R:	EQU	86
X43I:	EQU	87
X44R:	EQU	88
X44I:	EQU	89
X45R:	EQU	90
X45I:	EQU	91
X46R:	EQU	92
X46I:	EQU	93
X47R:	EQU	94
X47I:	EQU	95
X48R:	EQU	96
X48I:	EQU	97
X49R:	EQU	98
X49I:	EQU	99
X50R:	EQU	100
X50I:	EQU	101



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X51R:	EQU	102
X51I:	EQU	103
X52R:	EQU	104
X52I:	EQU	105
X53R:	EQU	106
X53I:	EQU	107
X54R:	EQU	108
X54I:	EQU	109
X55R:	EQU	110
X55I:	EQU	111
X56R:	EQU	112
X56I:	EQU	113
X57R:	EQU	114
X57I:	EQU	115
X58R:	EQU	116
X58I:	EQU	117
X59R:	EQU	118
X59I:	EQU	119
X60R:	EQU	120
X60I:	EQU	121
X61R:	EQU	122
X61I:	EQU	123
X62R:	EQU	124
X62I:	EQU	125
X63R:	EQU	126
X63I:	EQU	127
W:	EQU	0b50h
WVALUE:	EQU	5A82H ;VALUE FOR SIN(45) OR COS(45)

COS1:	EQU	0febh ;COS(1/32)
COS2:	EQU	0fb0h ;COS(2/32)
COS3:	EQU	0f4fh ;COS(3/32)
COS4:	EQU	0ec7h ;COS(4/32)
COS5:	EQU	0e1ch ;COS(5/32)
COS6:	EQU	0d4dh ;COS(6/32)
COS7:	EQU	0c5eh ;COS(7/32)
COS8:	EQU	0b50h
COS9:	EQU	0a26h ;COS(9/32)
COS10:	EQU	08e3h ;COS(10/32)
COS11:	EQU	078ah
COS12:	EQU	061fh
COS13:	EQU	04a4h
COS14:	EQU	031eh
COS15:	EQU	0191h
COS16:	EQU	0000h
COS17:	EQU	-0191h
COS18:	EQU	-031eh
COS19:	EQU	-04a4h
COS20:	EQU	-061fh
COS21:	EQU	-078ah
COS22:	EQU	-08e3h
COS23:	EQU	-0a26h
COS24:	EQU	-0b50h
COS25:	EQU	-0c5eh
COS26:	EQU	-0d4dh
COS27:	EQU	-0e1ch
COS28:	EQU	-0ec7h
COS29:	EQU	-0f4fh
COS30:	EQU	-0fb0h
COS31:	EQU	-0febh

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SIN1: EQU 0191h
 SIN2: EQU 031eh
 SIN3: EQU 04a4h
 SIN4: EQU 061fh
 SIN5: EQU 078ah
 SIN6: EQU 08e3h
 SIN7: EQU 0a26h
 SIN8: EQU 0b50h
 SIN9: EQU 0c5eh
 SIN10: EQU 0d4dh
 SIN11: EQU 0e1ch
 SIN12: EQU 0ec7h
 SIN13: EQU 0f4fh
 SIN14: EQU 0fb0h
 SIN15: EQU 0febh
 SIN16: EQU 0ffffh
 SIN17: EQU 0febh
 SIN18: EQU 0fb0h
 SIN19: EQU 0f4fh
 SIN20: EQU 0ec7h
 SIN21: EQU 0e1ch
 SIN22: EQU 0d4dh
 SIN23: EQU 0c5eh
 SIN24: EQU 0b50h
 SIN25: EQU 0a26h
 SIN26: EQU 08e3h
 SIN27: EQU 078ah
 SIN28: EQU 061fh
 SIN29: EQU 04a4h
 SIN30: EQU 031eh
 SIN31: EQU 0191h

COSI1: EQU 0ffah ;COS(1/64)
 COSI2: EQU 0febh ;COS(2/64)
 COSI3: EQU 0fd3h ;COS(3/64)
 COSI4: EQU 0fb0h ;COS(4/64)
 COSI5: EQU 0f85h ;COS(5/64)
 COSI6: EQU 0f4fh ;COS(6/64)
 COSI7: EQU 0f10h ;COS(7/64)
 COSI8: EQU 0ec7h ;COS(8/64)
 COSI9: EQU 0e76h ;COS(9/64)
 COSI10: EQU 0e1ch ;COS(10/64)
 COSI11: EQU 0db9h
 COSI12: EQU 0d4dh
 COSI13: EQU 0cd9h
 COSI14: EQU 0c5eh
 COSI15: EQU 0bdah
 COSI16: EQU 0b50h
 COSI17: EQU 0abeh
 COSI18: EQU 0a26h
 COSI19: EQU 0987h
 COSI20: EQU 08e3h
 COSI21: EQU 0839h
 COSI22: EQU 078ah
 COSI23: EQU 06d7h
 COSI24: EQU 061fh
 COSI25: EQU 0563h
 COSI26: EQU 04a4h
 COSI27: EQU 03e2h

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

COSI28:	EQU	031eh
COSI29:	EQU	0258h
COSI30:	EQU	0191h
COSI31:	EQU	00c8h
COSI32:	EQU	0000h
COSI33:	EQU	-00c8h
COSI34:	EQU	-0191h
COSI35:	EQU	-0258h
COSI36:	EQU	-031eh
COSI37:	EQU	-03e2h
COSI38:	EQU	-04a4h
COSI39:	EQU	-0563h
COSI40:	EQU	-061fh
COSI41:	EQU	-06d7h
COSI42:	EQU	-078ah
COSI43:	EQU	-0839h
COSI44:	EQU	-08e3h
COSI45:	EQU	-0987h
COSI46:	EQU	-0a26h
COSI47:	EQU	-0abeh
COSI48:	EQU	-0b50h
COSI49:	EQU	-0bdah
COSI50:	EQU	-0c5eh
COSI51:	EQU	-0cd9h
COSI52:	EQU	-0d4dh
COSI53:	EQU	-0db9h
COSI54:	EQU	-0e1ch
COSI55:	EQU	-0e76h
COSI56:	EQU	-0ec7h
COSI57:	EQU	-0f10h
COSI58:	EQU	-0f4fh
COSI59:	EQU	-0f85h
COSI60:	EQU	-0fb0h
COSI61:	EQU	-0fd3h
COSI62:	EQU	-0febh
COSI63:	EQU	-0ffah

SINI1:	EQU	00c8h
SINI2:	EQU	0191h
SINI3:	EQU	0258h
SINI4:	EQU	031eh
SINI5:	EQU	03e2h
SINI6:	EQU	04a4h
SINI7:	EQU	0563h
SINI8:	EQU	061fh
SINI9:	EQU	06d7h
SINI10:	EQU	078ah
SINI11:	EQU	0839h
SINI12:	EQU	08e3h
SINI13:	EQU	0987h
SINI14:	EQU	0a26h
SINI15:	EQU	0abeh
SINI16:	EQU	0b50h
SINI17:	EQU	0bdah
SINI18:	EQU	0c5eh
SINI19:	EQU	0cd9h
SINI20:	EQU	0d4dh
SINI21:	EQU	0db9h
SINI22:	EQU	0e1ch
SINI23:	EQU	0e76h

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SINI24: EQU 0ec7h
SINI25: EQU 0f10h
SINI26: EQU 0f4fh
SINI27: EQU 0f85h
SINI28: EQU 0fb0h
SINI29: EQU 0fd3h
SINI30: EQU 0febh
SINI31: EQU 0ffah
SINI32: EQU 0FFFh
SINI33: EQU 0ffah
SINI34: EQU 0febh
SINI35: EQU 0fd3h
SINI36: EQU 0fb0h
SINI37: EQU 0f85h
SINI38: EQU 0f4fh
SINI39: EQU 0f10h
SINI40: EQU 0ec7h
SINI41: EQU 0e76h
SINI42: EQU 0e1ch
SINI43: EQU 0db9h
SINI44: EQU 0d4dh
SINI45: EQU 0cd9h
SINI46: EQU 0c5eh
SINI47: EQU 0bdah
SINI48: EQU 0b50h
SINI49: EQU 0abeh
SINI50: EQU 0a26h
SINI51: EQU 0987h
SINI52: EQU 08e3h
SINI53: EQU 0839h
SINI54: EQU 078ah
SINI55: EQU 06d7h
SINI56: EQU 061fh
SINI57: EQU 0563h
SINI58: EQU 04a4h
SINI59: EQU 03e2h
SINI60: EQU 031eh
SINI61: EQU 0258h
SINI62: EQU 0191h
SINI63: EQU 00c8h

```

```

SH_DP: EQU 68H
SH_DP1: EQU 69H

```

```

;-----
; SART PROGRAM
;-----

```

```

ORG 0

```

```

; INITIALIZE FFT PROCESSING.

```

```

FFT: SPM 0
SSXM
ROVM

```

```

LRLK AR2,401H ;SOURCE
LRLK AR4,8191 ;COUNT Y
SHIFT_1: LARP AR2
LT *
MPYK -1

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SPL    *+
ADRK   1                ;ACC = (AR2)
LARP   ar4
BANZ   SHIFT_1,*-
```

```
;-----
;BIT-REVERSE 1
```

```
LARK   ARO,128          ;SIZE BITREV
LRLK   AR1,8000H        ;DESTINATE
LRLK   AR2,400H         ;SOURCE
LARK   AR4,127          ;COUNT Y
BITY1: LARK   AR3,127    ;COUNT X
BITX1: LARP   AR2
LAC    *+,0,AR1         ;ACC = (AR2)
SACL   *BR0+,0,AR3
BANZ   BITX1,*-
LARP   AR1
ADRK   255              ;SIZE OF X * 2
LARP   AR4
BANZ   BITY1,*-
```

```
;-----
; END BITREVERSE 1
```

```
;START 128 POINT FFT
```

```
LOOP64_0: LARK   AR3,255    ;LOOP 128x2 - 1          STAGE 1-6
LRLK   AR2,256          ;FIRST DP = 256
LARK   AR1,68H          ;ADDR 68H STORE DP
LARP   AR1
SAR    AR2,*
LDP    *                ;DATA PAGE POINTER
```

```
;-----
INCL   "64FFT.MAC"      ;64FFT 1 TIME
```

```
;-----
LARK   AR1,68H
LARP   AR1
LAR    AR2,* ,AR2
ADRK   1
LARP   AR3
BANZ   LOOP64_0,*-
```

```
;-----
;END STAGE 1-6 64FFT 256 TIMES _1
```

```
LRLK   AR2,256
LARK   AR1,SH_DP        ;68H
LARP   1
SAR    AR2,*+
LRLK   AR2,257
SAR    AR2,*
```

```
LOOP1281: LARK   AR3,127    ;LOOP 128 - 1          START STAGE 7
LRLK   AR2,256          ;FIRST DP = 256
LARK   AR1,68H          ;ADDR 68H STORE DP
LARP   AR1
SAR    AR2,*
LDP    *+                ;DATA PAGE POINTER
LARP   AR2
ADRK   1
LARP   ARr1
SAR    AR2,*
```

```
;-----
INCL   "128_MAC.MAC"    ;STAGE 7TH 128FFT
```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นโดยไม่เอาใจใส่ในประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
LARK AR1,68H
LARP AR1
LAR AR2,*,AR2
ADRK 2
LARP AR3
BANZ LOOP1281,*-
```

```
;END STAGE 1-7 128FFT 128 TIMES ___1
```

```
INCL "128TP.MAC" ;TRANSPPOST 128X128
```

```
LRLK AR2,8002H ;SOURCE
LRLK AR4,8191 ;COUNT Y
SHIFT_21: LARP AR2
LT *
MPYK -1
SPL *+
ADRK 3 ;ACC = (AR2)
larp ar4
BANZ SHIFT_21,*-
```

```
LRLK AR2,8003H ;SOURCE
LRLK AR4,8191 ;COUNT Y
SHIFT_22: LARP AR2
LT *
MPYK -1
SPL *+
ADRK 3 ;ACC = (AR2)
larp ar4
BANZ SHIFT_22,*-
```

```
; BIT-REVEES2
```

```
LRLK AR1,400H ;DESTINATE
LRLK AR2,8000H ;SOURCE
LRLK AR4,16383 ;COUNT Y
rey1_1: LARP AR2
ZALH *+
ADRK 1 ;ACC = (AR2)
larp ar1
SACH *+,0,ar4
BANZ rey1_1,*-
```

```
LARK ARO,128 ;SIZE BITREV
LRLK AR1,8000H ;DESTINATE
LRLK AR2,400H ;SOURCE
LARK AR4,127 ;COUNT Y
BITY2_1: LARK AR3,127 ;COUNT X
BITX2_1: LARP AR2
LAC *+,0,AR1 ;ACC = (AR2)
SACL *BR0+,0,AR3
BANZ BITX2_1,*-
LARP AR1
ADRK 255 ;SIZE OF X * 2
LARP AR4
BANZ BITY2_1,*-
```

```
LRLK AR1,400H ;DESTINATE
LRLK AR2,8001H ;SOURCE
```

เอกสารนี้เป็นเอกสารราชการสำหรับการใช้งานภายในเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

;-----
INCL "128_MAC.MAC"                STAGE 7TH 128FFT
;-----
LARK    AR1,68H
LARP    AR1
LAR     AR2,*,AR2
ADRK    2
LARP    AR3
BANZ    LOOP1282,*-
;END STAGE 1-7 128FFT 128 TIMES __2
;-----
INCL "128ABS.MAC"
;-----
stop:   out *,0          ; SEND INTERRUPT TO PC
        nop
        b stop
        END

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

; 64FFT.MAC

;1ST & 2ND STAGES COMBINED WITH DIVIDE-BY-4 INTERSTAGE SCALING

COMBO X0R,X0I,X1R,X1I,X2R,X2I,X3R,X3I,
COMBO X4R,X4I,X5R,X5I,X6R,X6I,X7R,X7I
COMBO X8R,X8I,X9R,X9I,X10R,X10I,X11R,X11I
COMBO X12R,X12I,X13R,X13I,X14R,X14I,X15R,X15I
COMBO X16R,X16I,X17R,X17I,X18R,X18I,X19R,X19I
COMBO X20R,X20I,X21R,X21I,X22R,X22I,X23R,X23I
COMBO X24R,X24I,X25R,X25I,X26R,X26I,X27R,X27I
COMBO X28R,X28I,X29R,X29I,X30R,X30I,X31R,X31I
COMBO X32R,X32I,X33R,X33I,X34R,X34I,X35R,X35I
COMBO X36R,X36I,X37R,X37I,X38R,X38I,X39R,X39I
COMBO X40R,X40I,X41R,X41I,X42R,X42I,X43R,X43I
COMBO X44R,X44I,X45R,X45I,X46R,X46I,X47R,X47I
COMBO X48R,X48I,X49R,X49I,X50R,X50I,X51R,X51I
COMBO X52R,X52I,X53R,X53I,X54R,X54I,X55R,X55I
COMBO X56R,X56I,X57R,X57I,X58R,X58I,X59R,X59I
COMBO X60R,X60I,X61R,X61I,X62R,X62I,X63R,X63I

;3RD STAGE WITH DIVIDE-BY-2 INTERSTAGE SCALING.

ZERO X0R,X0I,X4R,X4I
PIBY4 X1R,X1I,X5R,X5I,W
PIBY2 X2R,X2I,X6R,X6I
PI3BY4 X3R,X3I,X7R,X7I,W
ZERO X8R,X8I,X12R,X12I
PIBY4 X9R,X9I,X13R,X13I,W
PIBY2 X10R,X10I,X14R,X14I
PI3BY4 X11R,X11I,X15R,X15I,W
ZERO X16R,X16I,X20R,X20I
PIBY4 X17R,X17I,X21R,X21I,W
PIBY2 X18R,X18I,X22R,X22I
PI3BY4 X19R,X19I,X23R,X23I,W
ZERO X24R,X24I,X28R,X28I
PIBY4 X25R,X25I,X29R,X29I,W
PIBY2 X26R,X26I,X30R,X30I
PI3BY4 X27R,X27I,X31R,X31I,W
ZERO X32R,X32I,X36R,X36I
PIBY4 X33R,X33I,X37R,X37I,W
PIBY2 X34R,X34I,X38R,X38I
PI3BY4 X35R,X35I,X39R,X39I,W
ZERO X40R,X40I,X44R,X44I
PIBY4 X41R,X41I,X45R,X45I,W
PIBY2 X42R,X42I,X46R,X46I
PI3BY4 X43R,X43I,X47R,X47I,W
ZERO X48R,X48I,X52R,X52I
PIBY4 X49R,X49I,X53R,X53I,W
PIBY2 X50R,X50I,X54R,X54I
PI3BY4 X51R,X51I,X55R,X55I,W
ZERO X56R,X56I,X60R,X60I
PIBY4 X57R,X57I,X61R,X61I,W
PIBY2 X58R,X58I,X62R,X62I
PI3BY4 X59R,X59I,X63R,X63I,W

;4TH STAGE WITH INTERSTAGE SCALING.

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ZERO X0R,X0I,X8R ,X8I
 BTRFL X1R,X1I,X9R ,X9I ,COS4 ,SIN4
 PIBY4 X2R,X2I,X10R,X10I,W
 BTRFL X3R,X3I,X11R,X11I,COS12,SIN12
 PIBY2 X4R,X4I,X12R,X12I
 BTRFL X5R,X5I,X13R,X13I,COS20,SIN20
 PI3BY4 X6R,X6I,X14R,X14I,W
 BTRFL X7R,X7I,X15R,X15I,COS28,SIN28

ZERO X16R,X16I,X24R,X24I
 BTRFL X17R,X17I,X25R,X25I,COS4 ,SIN4
 PIBY4 X18R,X18I,X26R,X26I,W
 BTRFL X19R,X19I,X27R,X27I,COS12,SIN12
 PIBY2 X20R,X20I,X28R,X28I
 BTRFL X21R,X21I,X29R,X29I,COS20,SIN20
 PI3BY4 X22R,X22I,X30R,X30I,W
 BTRFL X23R,X23I,X31R,X31I,COS28,SIN28

ZERO X32R,X32I,X40R,X40I
 BTRFL X33R,X33I,X41R,X41I,COS4 ,SIN4
 PIBY4 X34R,X34I,X42R,X42I,W
 BTRFL X35R,X35I,X43R,X43I,COS12,SIN12
 PIBY2 X36R,X36I,X44R,X44I
 BTRFL X37R,X37I,X45R,X45I,COS20,SIN20
 PI3BY4 X38R,X38I,X46R,X46I,W
 BTRFL X39R,X39I,X47R,X47I,COS28,SIN28

ZERO X48R,X48I,X56R,X56I
 BTRFL X49R,X49I,X57R,X57I,COS4 ,SIN4
 PIBY4 X50R,X50I,X58R,X58I,W
 BTRFL X51R,X51I,X59R,X59I,COS12,SIN12
 PIBY2 X52R,X52I,X60R,X60I
 BTRFL X53R,X53I,X61R,X61I,COS20,SIN20
 PI3BY4 X54R,X54I,X62R,X62I,W
 BTRFL X55R,X55I,X63R,X63I,COS28,SIN28

; 5TH STAGE CALCULATE 64 POINTS

ZERO X0R ,X0I ,X16R,X16I
 BTRFL X1R ,X1I ,X17R,X17I,COS2 ,SIN2
 BTRFL X2R ,X2I ,X18R,X18I,COS4 ,SIN4
 BTRFL X3R ,X3I ,X19R,X19I,COS6 ,SIN6
 PIBY4 X4R ,X4I ,X20R,X20I,W
 BTRFL X5R ,X5I ,X21R,X21I,COS10 ,SIN10
 BTRFL X6R ,X6I ,X22R,X22I,COS12 ,SIN12
 BTRFL X7R ,X7I ,X23R,X23I,COS14 ,SIN14
 PIBY2 X8R ,X8I ,X24R,X24I
 BTRFL X9R ,X9I ,X25R,X25I,COS18 ,SIN18
 BTRFL X10R,X10I,X26R,X26I,COS20 ,SIN20
 BTRFL X11R,X11I,X27R,X27I,COS22 ,SIN22
 PI3BY4 X12R,X12I,X28R,X28I,W
 BTRFL X13R,X13I,X29R,X29I,COS25 ,SIN25
 BTRFL X14R,X14I,X30R,X30I,COS28 ,SIN28
 BTRFL X15R,X15I,X31R,X31I,COS30 ,SIN30

ZERO X32R,X32I,X48R,X48I
 BTRFL X33R,X33I,X49R,X49I,COS2 ,SIN2
 BTRFL X34R,X34I,X50R,X50I,COS4 ,SIN4
 BTRFL X35R,X35I,X51R,X51I,COS6 ,SIN6
 PIBY4 X36R,X36I,X52R,X52I,W

เอกสารนี้เป็นเอกสารตัวอย่าง ไม่สามารถนำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BTRFL X37R,X37I,X53R,X53I,COS10,SIN10
 BTRFL X38R,X38I,X54R,X54I,COS12,SIN12
 BTRFL X39R,X39I,X55R,X55I,COS14,SIN14
 PIBY2 X40R,X40I,X56R,X56I
 BTRFL X41R,X41I,X57R,X57I,COS18,SIN18
 BTRFL X42R,X42I,X58R,X58I,COS20,SIN20
 BTRFL X43R,X43I,X59R,X59I,COS22,SIN22
 PI3BY4 X44R,X44I,X60R,X60I,W
 BTRFL X45R,X45I,X61R,X61I,COS25,SIN25
 BTRFL X46R,X46I,X62R,X62I,COS28,SIN28
 BTRFL X47R,X47I,X63R,X63I,COS30,SIN30

; 6TH STAGE CALCULATE 64 POINT

ZERO X0R ,X0I ,X32R,X32I
 BTRFL X1R ,X1I ,X33R,X33I,COS1 ,SIN1
 BTRFL X2R ,X2I ,X34R,X34I,COS2 ,SIN2
 BTRFL X3R ,X3I ,X35R,X35I,COS3 ,SIN3
 BTRFL X4R ,X4I ,X36R,X36I,COS4 ,SIN4
 BTRFL X5R ,X5I ,X37R,X37I,COS5 ,SIN5
 BTRFL X6R ,X6I ,X38R,X38I,COS6 ,SIN6
 BTRFL X7R ,X7I ,X39R,X39I,COS7 ,SIN7
 PIBY4 X8R ,X8I ,X40R,X40I,W
 BTRFL X9R ,X9I ,X41R,X41I,COS9 ,SIN9
 BTRFL X10R,X10I,X42R,X42I,COS10,SIN10
 BTRFL X11R,X11I,X43R,X43I,COS11,SIN11
 BTRFL X12R,X12I,X44R,X44I,COS12,SIN12
 BTRFL X13R,X13I,X45R,X45I,COS13,SIN13
 BTRFL X14R,X14I,X46R,X46I,COS14,SIN14
 BTRFL X15R,X15I,X47R,X47I,COS15,SIN15
 PIBY2 X16R,X16I,X48R,X48I
 BTRFL X17R,X17I,X49R,X49I,COS17,SIN17
 BTRFL X18R,X18I,X50R,X50I,COS18,SIN18
 BTRFL X19R,X19I,X51R,X51I,COS19,SIN19
 BTRFL X20R,X20I,X52R,X52I,COS20,SIN20
 BTRFL X21R,X21I,X53R,X53I,COS21,SIN21
 BTRFL X22R,X22I,X54R,X54I,COS22,SIN22
 BTRFL X23R,X23I,X55R,X55I,COS23,SIN23
 PI3BY4 X24R,X24I,X56R,X56I,W
 BTRFL X25R,X25I,X57R,X57I,COS25,SIN25
 BTRFL X26R,X26I,X58R,X58I,COS26,SIN26
 BTRFL X27R,X27I,X59R,X59I,COS27,SIN27
 BTRFL X28R,X28I,X60R,X60I,COS28,SIN28
 BTRFL X29R,X29I,X61R,X61I,COS29,SIN29
 BTRFL X30R,X30I,X62R,X62I,COS30,SIN30
 BTRFL X31R,X31I,X63R,X63I,COS31,SIN31

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

; 128_MAC.MAC

; 7TH STAGE CALCULATE 64 POINT'

```
FZERO      X0R ,X0I ,X0R ,X0I
FBTRFL     X1R ,X1I ,X1R ,X1I ,COSI1 ,SINI1
FBTRFL     X2R ,X2I ,X2R ,X2I ,COSI2 ,SINI2
FBTRFL     X3R ,X3I ,X3R ,X3I ,COSI3 ,SINI3
FBTRFL     X4R ,X4I ,X4R ,X4I ,COSI4 ,SINI4
FBTRFL     X5R ,X5I ,X5R ,X5I ,COSI5 ,SINI5
FBTRFL     X6R ,X6I ,X6R ,X6I ,COSI6 ,SINI6
FBTRFL     X7R ,X7I ,X7R ,X7I ,COSI7 ,SINI7
FBTRFL     X8R ,X8I ,X8R ,X8I ,COSI8 ,SINI8
FBTRFL     X9R ,X9I ,X9R ,X9I ,COSI9 ,SINI9
FBTRFL     X10R,X10I,X10R,X10I,COSI10,SINI10
FBTRFL     X11R,X11I,X11R,X11I,COSI11,SINI11
FBTRFL     X12R,X12I,X12R,X12I,COSI12,SINI12
FBTRFL     X13R,X13I,X13R,X13I,COSI13,SINI13
FBTRFL     X14R,X14I,X14R,X14I,COSI14,SINI14
FBTRFL     X15R,X15I,X15R,X15I,COSI15,SINI15
FPIBY4     X16R,X16I,X16R,X16I,W
FBTRFL     X17R,X17I,X17R,X17I,COSI17,SINI17
FBTRFL     X18R,X18I,X18R,X18I,COSI18,SINI18
FBTRFL     X19R,X19I,X19R,X19I,COSI19,SINI19
FBTRFL     X20R,X20I,X20R,X20I,COSI20,SINI20
FBTRFL     X21R,X21I,X21R,X21I,COSI21,SINI21
FBTRFL     X22R,X22I,X22R,X22I,COSI22,SINI22
FBTRFL     X23R,X23I,X23R,X23I,COSI23,SINI23
FBTRFL     X24R,X24I,X24R,X24I,COSI24,SINI24
FBTRFL     X25R,X25I,X25R,X25I,COSI25,SINI25
FBTRFL     X26R,X26I,X26R,X26I,COSI26,SINI26
FBTRFL     X27R,X27I,X27R,X27I,COSI27,SINI27
FBTRFL     X28R,X28I,X28R,X28I,COSI28,SINI28
FBTRFL     X29R,X29I,X29R,X29I,COSI29,SINI29
FBTRFL     X30R,X30I,X30R,X30I,COSI30,SINI30
FBTRFL     X31R,X31I,X31R,X31I,COSI31,SINI31
FPIBY2     X32R,X32I,X32R,X32I
FBTRFL     X33R,X33I,X33R,X33I,COSI33,SINI33
FBTRFL     X34R,X34I,X34R,X34I,COSI34,SINI34
FBTRFL     X35R,X35I,X35R,X35I,COSI35,SINI35
FBTRFL     X36R,X36I,X36R,X36I,COSI36,SINI36
FBTRFL     X37R,X37I,X37R,X37I,COSI37,SINI37
FBTRFL     X38R,X38I,X38R,X38I,COSI38,SINI38
FBTRFL     X39R,X39I,X39R,X39I,COSI39,SINI39
FBTRFL     X40R,X40I,X40R,X40I,COSI40,SINI40
FBTRFL     X41R,X41I,X41R,X41I,COSI41,SINI41
FBTRFL     X42R,X42I,X42R,X42I,COSI42,SINI42
FBTRFL     X43R,X43I,X43R,X43I,COSI43,SINI43
FBTRFL     X44R,X44I,X44R,X44I,COSI44,SINI44
FBTRFL     X45R,X45I,X45R,X45I,COSI45,SINI45
FBTRFL     X46R,X46I,X46R,X46I,COSI46,SINI46
FBTRFL     X47R,X47I,X47R,X47I,COSI47,SINI47
FPI3BY4    X48R,X48I,X48R,X48I,W
FBTRFL     X49R,X49I,X49R,X49I,COSI49,SINI49
FBTRFL     X50R,X50I,X50R,X50I,COSI50,SINI50
FBTRFL     X51R,X51I,X51R,X51I,COSI51,SINI51
FBTRFL     X52R,X52I,X52R,X52I,COSI52,SINI52
FBTRFL     X53R,X53I,X53R,X53I,COSI53,SINI53
FBTRFL     X54R,X54I,X54R,X54I,COSI54,SINI54
```

เอกสารนี้เป็นเอกสารที่สงวนไว้ใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FBTRFL	X55R,X55I,X55R,X55I,COSI55,SINI55
FBTRFL	X56R,X56I,X56R,X56I,COSI56,SINI56
FBTRFL	X57R,X57I,X57R,X57I,COSI57,SINI57
FBTRFL	X58R,X58I,X58R,X58I,COSI58,SINI58
FBTRFL	X59R,X59I,X59R,X59I,COSI59,SINI59
FBTRFL	X60R,X60I,X60R,X60I,COSI60,SINI60
FBTRFL	X61R,X61I,X61R,X61I,COSI61,SINI61
FBTRFL	X62R,X62I,X62R,X62I,COSI62,SINI62
FBTRFL	X63R,X63I,X63R,X63I,COSI63,SINI63



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

;FFT.MAC

COMBO: MACRO R1,I1,R2,I2,R3,I3,R4,I4
;CALCULATE PARTIAL TERMS FOR R3, R4, I3, AND I4.

```
LAC      R3,14      ;ACC = 1/4(R3)
ADD      R4,14      ;ACC = 1/4(R3+R4)
SACH     R3,1       ;R3  = 1/2(R3+R4)
SUB      R4,15      ;ACC = 1/4(R3+R4) - 1/2(R4)
SACH     R4,1       ;R4  = 1/2(R3-R4)
LAC      I3,14      ;ACC = 1/4(I3)
ADD      I4,14      ;ACC = 1/4(I3+I4)
SACH     I3,1       ;I3  = 1/2(I3+I4)
SUB      I4,15      ;ACC = 1/4(I3+I4)-1/2(I4)
SACH     I4,1       ;I4  = 1/2(I3-I4)
```

;CALCULATE PARTIAL TERMS FOR R2 R4 I2 I4

```
LAC      R1,14      ;ACC = 1/4(R1)
ADD      R2,14      ;ACC = 1/4(R1+R2)
SACH     R1,1       ;R3  = 1/2(R1+R2)
SUB      R2,15      ;ACC = 1/4(R1+R2) - 1/2(R2)
ADD      I4,15      ;ACC = 1/4[(R1-R2)+(I3-I4)]
SACH     R2         ;R2  = 1/4[(R1-R2)+(I3-I4)]
SUBH     I4         ;ACC = 1/4[(R1-R2)-(I3-I4)]
```

```
LT       R4         ;T = R4
MPYK     1          ;P = R4
SPL      I4         ;I4  = R4 = 1/2(R3-R4)
SACH     R4         ;R4  = 1/4[(R1-R2)-(I3-I4)]
LAC      I1,14      ;ACC = 1/4(I1)
ADD      I2,14      ;ACC = 1/4(I1+I2)
SACH     I1,1       ;I1  = 1/2(I1+I2)
SUB      I2,15      ;ACC = 1/4[(I1+I2)-1/2(I2)]
SUB      I4,15      ;ACC = 1/4[(I1-I2)-(R3-R4)]
SACH     I2         ;I2  = 1/4[(I1-I2)-(R3-R4)]
ADDH     I4         ;ACC = 1/4[(I1-I2)+(R3-R4)]
SACH     I4         ;I4  = 1/4[(I1-I2)+(R3-R4)]
```

;CALCULATE PARTIAL TERMS FOR R1 R3 I1 I3

```
LAC      R1,15      ;ACC = 1/4(R1+R2)
ADD      R3,15      ;ACC = 1/4[(R1+R2)+(R3+R4)]
SACH     R1         ;R1  = 1/4[(R1+R2)+(R3+R4)]
SUBH     R3         ;ACC = 1/4[(R1+R2)-(R3+R4)]
SACH     R3         ;R3  = 1/4[(R1+R2)-(R3+R4)]
LAC      I1,15      ;ACC = 1/4(I1+I2)
ADD      I3,15      ;ACC = 1/4[(I1+I2)+(I3+I4)]
SACH     I1         ;I1  = 1/4[(I1+I2)+(I3+I4)]
SUBH     I3         ;ACC = 1/4[(I1+I2)-(I3+I4)]
SACH     I3         ;I3  = 1/4[(I1+I2)-(I3+I4)]
ENDM
```

ZERO: MACRO PR,PI,QR,QI

;CALCULATE Re[P+Q] AND Re[P-Q]

```
LAC      PR,15      ;ACC = 1/2(PR)
ADD      QR,15      ;ACC = 1/2(PR+QR)
SACH     PR         ;PR  = 1/2(PR+QR)
SUBH     QR         ;ACC = 1/2(PR+QR)-(QR)
SACH     QR         ;QR  = 1/2(PR-QR)
```

;CALCULATE Im[P+Q] AND Im[P-Q]

```
LAC      PI,15      ;ACC = 1/2(PI)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น มิอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

ADD      QI,15      ;ACC = 1/2(PI+QI)
SACH     PI         ;PI = 1/2(PI+QI)
SUBH     QI         ;ACC = 1/2(PI+QI)-(QI)
SACH     QI         ;QI = 1/2(PI-QI)
ENDM

```

PIBY4: MACRO PR,PI,QR,QI,W

```

LAC      QI         ;ACC = QI
SUB      QR         ;ACC = QI-QR
SACL     QI         ;QI = QI-QR
LAC      QI,14     ;ACC = 1/4(QI-QR)
SACH     QI,1      ;QI = 1/2(QI-QR)
ADD      QR,15     ;ACC = 1/4(QI-QR)+1/2(QR)
SACH     QR,1      ;QR = 1/2(QI+QR)
LAC      PR,11     ;ACC = 1/32(PR)
LT       QR         ;T = 1/2(QI+QR)
MPYK     W          ;P = 1/32(QI+QR)*W
APAC     ;ACC = 1/32(QI+QR)*W
SACH     PR,4      ;PR = 1/2[PR+(QI+QR)*W]
SPAC     ;ACC = 1/32(PR)
SPAC     ;ACC = 1/32[PR-(QI+QR)*W]
SACH     QR,4      ;QR = 1/2[PR-(QI+QR)*W]
LAC      PI,11     ;ACC = 1/32(PI)
LT       QI         ;T = 1/2(QI-QR)
MPYK     W          ;P = 1/32(QI-QR)*W
APAC     ;ACC = 1/32(QI-QR)*W
SACH     PI,4      ;PR = 1/2[PI+(QI-QR)*W]
SPAC     ;ACC = 1/32(PI)
SPAC     ;ACC = 1/32[PI-(QI-QR)*W]
SACH     QI,4      ;QR = 1/2[PI-(QI-QR)*W]
ENDM

```

PIBY2: MACRO PR,PI,QR,QI

;CALCULATE Re[P+jQ] AND Re[P-jQ]

```

LAC      PI,15     ;ACC = 1/2(PI)
SUB      QR,15     ;ACC = 1/2(PI-QR)
SACH     PI         ;PI = 1/2(PI-QR)
ADDH     QR         ;ACC = 1/2(PI-QR)+(QR)
SACH     QR         ;QR = 1/2(PR+QR)

```

;CALCULATE Im[P+jQ] AND Im[P-jQ]

```

LAC      PR,15     ;ACC = 1/2(PR)
ADD      QI,15     ;ACC = 1/2(PR+QI)
SACH     PR         ;PR = 1/2(PR+QI)
SUBH     QI         ;ACC = 1/2(PR+QI)-(QI)
LT       QR         ; T REGISTER = QR
MPYK     1          ; P REGISTER = QR
SPL      QI         ; QR -> QI
SACH     QR         ;QR = 1/2(PR-QI)
ENDM

```

PI3BY4: MACRO PR,PI,QR,QI,W

```

LAC      QI,14     ;ACC = 1/4(QI)
SUB      QR,14     ;ACC = 1/4(QI-QR)
SACH     QI,1      ;QI = 1/2(QI-QR)
ADD      QR,15     ;ACC = 1/4(QI+QR)
SACH     QR,1      ;QR = 1/2(QI+QR)
LAC      PR,11     ;ACC = 1/32(PR)
LT       QI         ;T = 1/2(QI-QR)
MPYK     W          ;P = 1/32(QI-QR)*W
APAC     ;ACC = 1/32(QI-QR)*W

```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการวิจัยและพัฒนาเทคโนโลยีสารสนเทศและการสื่อสาร

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SACH	PR, 4	;PR = 1/2[PR+(QI-QR)*W]
SPAC		;ACC = 1/32(PR)
SPAC		;ACC = 1/32[PR-(QI-QR)*W]
LT	QR	;T = 1/2(QI+QR)
MPYK	W	;P = 1/32(QI+QR)*W
SACH	QR, 4	;QR = 1/2[PR-(QI-QR)*W]
LAC	PI, 11	;ACC = 1/32(PI)
SPAC		;ACC = 1/32(QI+QR)*W
SACH	PI, 4	;PR = 1/2[PI-(QI+QR)*W]
APAC		;ACC = 1/32(PI)
APAC		;ACC = 1/32[PI+(QI+QR)*W]
SACH	QI, 4	;QR = 1/2[PI+(QI+QR)*W]
ENDM		

BTRFL: MACRO PR,PI,QR,QI,WR,WI

; CALCULATE QR*WR+QI*WI AND STORE INTO QI

LT	QI	;T = QI
MPYK	WI	;P = 1/16(QI*WI)
ZAC		;ACC = 0
LTA	QR	;ACC = 1/16(QI*WI), T = QR
MPYK	WR	;P = 1/16(QR*WR)
APAC		;ACC = 1/16(QR*WR+QI*WI)
SACH	QR, 4	;QR = (QR*WR+QI*WI)
LAC	QR, 15	;ACC = 1/2(QR*WR+QI*WI)
SACH	QR	;QR = 1/2(QR*WR+QI*WI)

;CALCULATE QI*WR-QR*WI AND STORE RESULT INT QR

MPYK	WI	;P = 1/16(QR*WI)
ZAC		;ACC = 0
LTS	QI	;ACC = 1/16(QR*WI), T = QI
MPYK	WR	;P = 1/16(QI*WR)
APAC		;ACC = 1/16(QI*WR-QR*WI)
SACH	QI, 4	;QI = (QI*WR-QR*WI)
LAC	QI, 15	;ACC = 1/2(QI*WR-QR*WI)
SACH	QI	;QI = 1/2(QI*WR-QR*WI)

;CALCULATE RE(P+JQ) & RE(P-JQ) STORE RESULT IN PR & QR

LAC	PR, 14	;ACC = 1/4(PR)
ADD	QR, 15	;ACC = 1/4[PR+(QR*WR+QI*WI)]
SACH	PR, 1	;PR = 1/2[PR+(QR*WR+QI*WI)]
SUBH	QR	;ACC = 1/4[PR-(QR*WR+QI*WI)]
SACH	QR, 1	;PR = 1/2[PR-(QR*WR+QI*WI)]

;CALCULATE IM(P+JQ) & IM(P-JQ) STORE RESULT IN PI & QI

LAC	PI, 14	;ACC = 1/4(PI)
ADD	QI, 15	;ACC = 1/4[PI+(QI*WR-QR*WI)]
SACH	PI, 1	;PR = 1/2[PI+(QI*WR-QR*WI)]
SUBH	QI	;ACC = 1/4[PI-(QI*WR-QR*WI)]
SACH	QI, 1	;PR = 1/2[PI-(QI*WR-QR*WI)]
ENDM		

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

; ABSOLUTE.MAC AND MOVE DATA TO ADDRESS 400H

; START AT ADDR 400H
; INPUT: BUFFER ADDR 70H
; A " ADDR 71H
; B " ADDR 72H
; N " ADDR 73H
; SRT ADDR 74H
; CNT ADDR 75H
; 2400H POINTER ADDR 76H
; INPUT REAL AT 00
; INPUT IMAGE AT 01
; OUTPUT AT 00

; AR1 POINTER ON-CHIP RAM
; AR2 POINTER START 400H
; AR3 COUNT DATA
; MACRO ABSOLUTE COUNT
; COUNT: EQU 4

```
INIT_AB:  LRLK  AR2,76H
          LALK  400H ;data move to when finish program
          LARP  AR2
          SACL  *
          LRLK  AR2,8000H ;START ADDRESS
          LRLK  AR1,74H
          LARP  AR1
          SAR   AR2,*+
          LRLK  AR3,16383 ;128x128 - 1 ;CNT
          SAR   AR3,*
START_AB:  LARP  AR1
          SAR   AR3,*
          LARP  AR2
          SQRA  *+ ;P REGISTER = A**2
          ZAC   ;ACC = 0
          SQRA  *- ;P REGISTER = B**2 ACC= A**2
          APAC  ;ACC = A**2 + B**2
          LARP  AR1
          LRLK  AR1,70H ;AR1 = 70H
          SFR   ;ACC = 1/2(A**2 + B**2)
          SFR   ;ACC = 1/4(A**2 + B**2)
          SFR   ;ACC = 1/8(A**2 + B**2)
          SFR   ;ACC = 1/16(A**2 + B**2)
          SFR   ;ACC = 1/32(A**2 + B**2)
          SFR   ;ACC = 1/64(A**2 + B**2)
          SFR   ;ACC = 1/128(A**2 + B**2)
          SFR   ;ACC = 1/256(A**2 + B**2)
          SACL  *
          ;ADDR 70H == ACC
CHK_7400H: LRLK  AR7,7400H
          SBLK  7400H ; IF DATA MORE THAN 7400H
          BLZ   SQROOT ; SET DATA => 7400H
          LARP  AR1
          SAR   AR7,*
SQROOT:   LRLK  AR1,71H
          LARP  AR1
          LALK  80H ; INITIAL PARAMETER AND
          SACL  *+ ; STORE 80H AT (71H)
          SACL  *+
          LACK  1
          SACL  *
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้ทำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

EQUOO:  LRLK  AR1,70H ;
        LAC   *       ;
        BNZ  EQUAA  ; CHECK DATA = 0
        LRLK  AR2,71H ;
        LARP  AR2    ;
        SACH  *       ;
        B    OPUT   ;
EQUAA:  LRLK  AR1,71H ;
        SQRA  *-     ; CHECK DATA = A**2
        LAC   *       ;
        SPAC  *       ;
        BZ   OPUT   ;
MOREAA: LRLK  AR1,70H ; CHECK DATA MORE THAN
        LAC   *       ; A**2
        SPAC  *       ;
        BGZ  ADBN   ;
SBBN:  LRLK  AR1,72H ;
        LAC   *,15  ;
        SACH  *-     ;
        LAC   *+     ;
        SUB  *-     ; A=A+(B/2**N)
        SACL  *       ;
        LRLK  AR1,73H ;
        LAC   *       ; N=N+1
        ADLK  1      ;
        SACL  *       ;
        B    NNN    ;
ADBN:  LRLK  AR1,72H ;
        LAC   *,15  ;
        SACH  *-     ;
        LAC   *+     ;
        ADD  *-     ;
        SACL  *       ; A=A-(B/2**N)
        LRLK  AR1,73H ;
        LAC   *       ; N=N+1
        ADLK  1      ;
        SACL  *       ;
NNN:   LRLK  AR1,73H ;
        LAC   *       ;
        SUBK  10H   ; CHECK LOOP N = 10H
        BNZ  EQUOO  ;
OPUT:  LARP  AR1    ;
        LRLK  AR1,71H ;
        LAC   *       ;
        LRLK  AR1,76H ;
        LARP  AR1    ; OUTPUT = A
        LAR  AR2,*   ;
        LARP  AR2    ;
        SACL  *+     ; STORE RESULT
        LARP  AR1    ;
        SAR  AR2,*+  ;
        LRLK  AR1,74H ;
        LARP  AR1    ;
        LAR  AR2,*   ;
        LARP  AR2    ;
        ADRK  2      ;
        LARP  AR1    ;
        SAR  AR2,*+  ;
CHKCOUNT: LARK  AR1,75H ;
        LARP  AR1    ; CHECK LOOP

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น มิอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

LAR AR3,* ;
LARP AR3 ;
BANZ START_AB,*- ; _____



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

;128X128TP.MAC      TRANSPOST 128x128
;TRANSPOST X*Y MATRIX
;X      AT      ADDRESS 60H
;BUFFER AT      ADDRESS 61H,62H
;Y      AT      ADDRESS 63H
;BUFFER AT      ADDRESS 64H,65H
;

```

```

LRLK AR1,0H      ;-----
LRLK AR2,60H    ;
LARP AR2        ; X=0 , Y=0
SAR  AR1,*      ;
LRLK AR1,0H    ;
LRLK AR2,63H   ;
SAR  AR1,*      ;
TRANSPOST: LRLK AR2,60H ;
LARP AR2        ;
LAC  *          ; CHECK X = Y
LRLK AR2,63H   ;
SUB  *          ;
BZ   X7F        ;
BUFF:  LRLK AR2,61H ;
LARP  2         ;
LRLK AR1,100H  ;
SAR  AR1,*     ;
LRLK AR2,60H  ;
LT   *+        ;
MPY  *+        ; 100H*X+2*Y+8000H ==> ADDR 61H
PAC  *          ;
ADLK 8000H     ;
LRLK AR2,63H  ;
ADD  *         ;
ADD  *         ;
LRLK AR2,61H  ;
SACL *+        ;
LRLK AR2,64H  ;
LARP  2         ;
LRLK AR1,100H ;
SAR  AR1,*     ;
LRLK AR2,63H  ;
LT   *+        ;
MPY  *+        ; 100H*Y+2*X+8000H ==> ADDR 64H
PAC  *          ;
ADLK 8000H     ;
LRLK AR2,60H  ;
ADD  *         ;
ADD  *         ;
LRLK AR2,64H  ;
SACL *         ;
LRLK AR3,61H  ;
LARP AR3       ;
LAR  AR1,*     ;
LARP AR1       ;
LAR  AR0,*     ;
LRLK AR2,66H  ;
LARP AR2       ; MOVE DATA AT (61H) TO BUFFER
SAR  AR0,*+    ;
LARP AR1       ;

```

เอกสารนี้เป็นเอกสารงานวิจัยที่ได้รับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามเผยแพร่ต่อสาธารณะ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

LARP AR2 ;
SAR AR0,*- ;
LRLK AR3,64H ;
LARP AR3 ;
LAR AR1,* ;
LARP AR1 ;
LAR AR0,* ;
LRLK AR3,61H ;
LARP AR3 ; MOVE DATA AT (63H) TO (61H)
LAR AR2,* ;
LARP AR2 ;
SAR AR0,*+ ;
LARP AR1 ;
ADRK 1 ;
LAR AR0,* ;
LARP AR2 ;
SAR AR0,*- ;
LRLK AR3,66H ;
LARP AR3 ;
LAR AR1,* ;
LRLK AR3,64H ;
LARP AR3 ;
LAR AR2,* ;
LARP AR2 ; MOVE DATA FROM BUFFER TO (63H)
SAR AR1,*+ ;
LRLK AR3,67H ;
LARP AR3 ;
LAR AR1,* ;
LARP AR2 ;
SAR AR1,* ;
X7F: LRLK AR2,60H ;
LARP AR2 ;
LACK 7FH ; CHECK X=7FH
SUB * ;
BZ Y7F ;
X1: LRLK AR2,60H ;
LARP AR2 ;
LAR AR1,* ;
LARP AR1 ;
ADRK 1 ; X=X+1
LARP AR2 ; GOTO CHECK Y=X
LRLK AR2,60H ;
SAR AR1,* ;
B TRANSPOST ;
Y7F: LRLK AR2,63H ;
LARP AR2 ;
LACK 7FH ; CHECK Y=7FH
SUB * ; IF Y=7FH PROGRAM FINNISH
BZ FINISH ;
Y1X0: LRLK AR2,63H ;
LARP AR2 ;
LAR AR1,* ; Y=Y+1
LARP AR1 ; X=Y
ADRK 1 ;
LARP AR2 ;
LRLK AR2,63H ;
SAR AR1,* ;
LRLK AR2,60H ;
SAR AR1,* ;
B TRANSPOST ;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

/* PROGRAM LOAD PROGRAMS TO TMS PROGRAMS MEMORY*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#define DATA_LOW 0x00
#define DATA_HIGH 0x01
#define PROG_LOW 0x02
#define PROG_HIGH 0x03
#define CARD_PORT 0x300

unsigned far *cardmem = (unsigned far *)MK_FP(0xe000,0);

void main(int argc,char *argv[])
{
    FILE *fp;
    unsigned char tex[50];
    long file_size,status,i,j,x;

    if(argc<2)
    {
        printf("Enter TMS program file (BINARY FILE) : ");
        gets(tex);
    }
    else strcpy(tex,argv[1]);
    if ((fp= fopen(tex,"rb")) == NULL)
    {
        fprintf(stderr, "Cannot open input file \n");
        exit(0);
    }
    fseek(fp,0L,SEEK_END);
    file_size=ftell(fp);
    fseek(fp,0L,SEEK_SET);
    printf("File size = %u \n",file_size);
    if(file_size>0x20000)
    {
        printf("File too big , can't load ");
        fclose(fp);
        exit(1);
    }
    if(file_size < 0x10000)
    {
        outportb(CARD_PORT,0x06); /* first program memory block
        status=fread(cardmem,sizeof(unsigned),file_size/2,fp);
        if(status != file_size/2)
        {
            printf("Read file error \n");
            fclose(fp);
            exit(0);
        }
        swab((char far *)cardmem,(char far *)cardmem,file_size);
    }
    else{
        outportb(CARD_PORT,0x06);
        status = fread(cardmem,sizeof(unsigned),0x8000,fp);
        if(status != file_size/2)
        {
            printf("Read file error \n");
            fclose(fp);
        }
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้เพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        exit(0);
    }
    swab((char far *)cardmem,(char far *)cardmem,0x10000);
    outportb(CARD_PORT,0x07);
    status=fread(cardmem,sizeof(unsigned),(file_size/2)-0x8000,fp);
    if(status != file_size/2-0x8000)
    {
        printf("Read file error \n");
        fclose(fp);
        exit(0);
    }
    swab((char far *)cardmem,(char far *)cardmem,file_size-0x10000);
    }
    fclose(fp);
    outportb(CARD_PORT,0x00);
    printf("Program are already load \n");
}
/* disable card */

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    gettime(&stop);
    printf("The current time is: %2d:%02d:%02d.%02d\n",
        stop.ti_hour, stop.ti_min, stop.ti_sec, stop.ti_hund);
    second = time(NULL);
    printf("Number of seconds is %ld", second);
    printf("\n\nThe difference is:
    %f seconds\n", difftime(second, first));
}

        /* program load O.K. return 1 */
else printf("\n Program error..%d\n", b);
end_usecard();
}
void interrupt newisr12(void)
{
    outportb(0x301, 0);
    intcode=0;
    outportb(INT_CTRL1, 0x64);
    oldvec();
}
void init_card(void)
{
    unsigned char intmask;

    outportb(CARD_PORT, 0x00); /* 0 - - 0 - 0 0 0 */
    oldvec = getvect(INT_NO);
    setvect(INT_NO, newisr12);
}
void end_usecard(void)
{
    outportb(CARD_PORT, 0x00); /* 0 - - 0 : - 0 0 0 */
    setvect(INT_NO, oldvec);
}

```

