ห้องสมุดคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การใช้ CARD TMS320C31 ในการประมวลผลภาพ

(IMAGE PROCESSING BY CARD TMS320C31)

โดย

1. นายวัชระ	ลักษณ์อนันต์กูร	39012102	ห้อง 3I
2. นายวีระพล	คงนุ่น	39012106	ห้อง 3I
3. นายสมบัติ	อิสระวาณิชย์	39012107	ห้อง 3I

อาจารย์ที่ปรึกษา

รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์

หมวดหมู่ 35 ส1
เลขทะเบียน 040582
วัน เดือน ปี 18 ตค 42

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2541

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

040582

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง **การใช้ CARD TMS320C31 ในการประมวลผลภาพ**

(Image Processing By CARD TMS320C31)

ผู้จัดทำ

1. นาย วัชระ ลักษณ์อนันต์กูร 39012102 3I
2. นาย วีระพล คงนุ่น 39012106 3I
3. นาย สมบัติ อิสระวาณิชย์ 39012107 3I

รศ. ดร. ฟูศักดิ์ ชีวสุวิทย์ อาจารย์ที่ปรึกษา

(.....)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	การประมวลผลภาพโดยใช้ CARD TMS 230C31		
นักศึกษา	นาย วัชระ	ลักษณ์อนันต์กูร	39012102
	นาย วีระพล	คงนุ่น	39012106
	นาย สมบัติ	อิสระวาณิชย์	39012107
อาจารย์ที่ปรึกษา	รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์		
ระดับการศึกษา	วิศวกรรมศาสตร์บัณฑิตเทคโนโลยีการวัดคุมทางอุตสาหกรรม		
ปีการศึกษา	พ.ศ. 2541		

บทคัดย่อ

ดิจิตอลซิกแนลโปรเซสซิงถูกนำมาใช้งานได้อย่างกว้างขวาง ในการประยุกต์ใช้งานในด้านต่างๆ เช่น การประมวลผลภาพ , การประมวลผลความเร็วสูง , การสื่อสารข้อมูล , ในด้านเครื่องมือวัดอุตสาหกรรมและการวัด เป็นต้น

สำหรับปริญญานิพนธ์ฉบับนี้จะใช้ Digital Signal Processor TMS 320C31 ในการประมวลผลส่วนที่เป็นข้อมูลภาพ ซึ่งการ์ด TMS 320C31 จะถูกกำหนดการทำงานด้วยโปรแกรมประมวลผลภาพ เช่น โปรแกรม Gradient Image หรือ โปรแกรม Smooth Image เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title : IMAGE PROCESSING BY TMS 320C31

Name : Mr.Watchara Luxanankul 39012102

: Mr.Weerapon Kongnon 39012106

: Mr.Sombat Issarawanich 39012107

Thesis Advisor : Assoc. Prof. Dr. Fusak Cheevasuvit

Level of study : Engineer of Industrial Technology in Industrial Instrument

Academic Year : 1998

Abstract

Digital signal processing are widely used in many application such as image processing , speed processing , data communication , instrumentation and measurement etc.

In this thesis the digital signal processor TMS 320C31 will be used as a part of image data acquisition. The TMS 320C31 card will be assigned as a image solfware processor for some application such as to produce gradient image , smooth image etc.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทที่ 1 บทนำ	
แนะนำไอซี TMS 320C31	1
บทที่ 2 สถาปัตยกรรมของ TMS 320C31	
2.1 หน่วยประมวลผลกลาง (CPU)	6
2.1.1 Multiplier	7
2.1.2 Arithmetic Logic Unit	7
2.1.3 Auxiliary Register Arithmetic Unit	7
2.1.4 CPU Register File	7
2.2 หน่วยความจำ (Memory)	10
2.2.1 RAM,ROM	10
2.2.2 Memory Maps	10
2.2.3 Memory Address Modes	11
2.3 Internal Bus Operation	12
2.4 External Bus Operation	12
2.5 ระบบติดต่อภายนอก (Peripharals)	15
2.6 ระบบถ่ายโอนข้อมูล (DMA)	17
2.7 คำสั่งของไอซี TMS 320C31	17
บทที่ 3 การปรับปรุงภาพ (IMAGE ENHANCEMENT)	
3.1 กระบวนการในสปาเชียลโดเมน	25
3.2 กระบวนการในโดเมนความถี่	28
3.3 การปรับปรุงภาพโดยวิธีการประมวลผลจุด	30
3.4 การหาขอบภาพโดยหลักการของ SOBEL	33
3.5 หลักการฟิลเตอร์ภาพโดยการกรองความถี่ต่ำ (Low-Pass Filter)	37
3.6 หลักการฟิลเตอร์ภาพโดยการกรองความถี่สูง (High-Pass Filter)	40
3.7 การหาขอบภาพโดยใช้หลักการลาปาเชียล (Laplacian)	42
3.8 การหาขอบภาพโดยใช้หลักการ Edge Enhancement	44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
บทที่ 4 IBM PC กับการอินเตอร์เฟส	
4.1 AT SLOT และ XT SLOT ของ IBM PC	45
4.2 รายละเอียดของสัญญาณต่างๆ บนสล็อต	46
4.2.1 เพาเวอร์ซัพพลาย	46
4.2.2 แอดเดรสบัส และสัญญาณต่างๆ ที่เกี่ยวข้อง	46
4.2.3 สัญญาณอินเตอร์รัพต์	47
4.2.4 สัญญาณที่ใช้ในกระบวนการ DMA	48
4.2.5 สัญญาณควบคุมต่างๆ	48
4.2.6 สัญญาณที่ใช้ในการสร้าง Wait State	49
4.2.7 สัญญาณนาฬิกา	50
4.3 การจัดการ Address ของ Memory และ Port	50
4.4 อินเตอร์รัพต์ (Interrupt)	52
บทที่ 5 การออกแบบวงจร	
5.1 การแปลงสัญญาณภาพ	55
5.2 การกำเนิดสัญญาณแอดเดรส	57
5.3 การแปลงสัญญาณอนาล็อกเป็นดิจิตอล	58
5.4 การอ้างตำแหน่งหน่วยความจำ	58
5.5 ชุดควบคุม	60
บทที่ 6 การทดลอง	
6.1 ทดลองกับภาพที่ได้จาก File Image	63
6.2 ทดลองกับภาพที่ได้จากกล้อง CCD	66
6.3 การเปรียบเทียบการประมวลผลระหว่าง PC กับ DSP	69
บทที่ 7 บทวิจารณ์และสรุป	
7.1 สรุป	70
7.2 ข้อจำกัดของระบบ	70
7.3 ปัญหาและอุปสรรคที่พบ	70

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

หน้า

รูปที่ 1.1	แสดงการประมวลเฉพาะอย่าง	1
รูปที 1.2	แสงการใช้ไมโครโปรเซสเซอร์ประมวลผล	1
รูปที 1.3	แสดงส่วนประกอบของ DSP	2
รูปที 1.4	แสดงวิวัฒนาการของไอซีตระกูล TMS 320xx	4
รูปที 1.5	แสดงการประยุกต์ใช้งาน TMS 320C31	5
รูปที่ 2.1	แสดงสถาปัตยกรรมของไอซี TMS 320C31	6
รูปที่ 2.2	แสดงหน่วยความจำของไอซีเบอร์ TMS 320C31	10
รูปที่ 2.3	แสดงการจัดหน่วยความจำของไอซีเบอร์ TMS 320C31	11
รูปที่ 2.4	แสดง Memory-Mapped External Interface Control Resister	13
รูปที่ 2.5	แสดง Primary Bus Control Register	14
รูปที่ 2.6	แสดง Expansion bus Control Register	14
รูปที่ 2.7	แสดง Peripheral Moduls	16
รูปที่ 2.8	แสดง DMA Controller	17
รูปที่ 3.1	แสดงเนเบอร์ฮูดขนาด 3x3 ของจุด (x,y) ในภาพ	26
รูปที่ 3.2	แสดฟังก์ชันระดับสีเทาสำหรับการปรับปรุงคอนทราส	26
รูปที่ 3.3	แสดงมาสก์ที่ใช้สำหรับการทำภาพให้คมชัดขึ้น	27
รูปที่ 3.4	แสดงมาสก์ขนาด 3x3 โดยทั่วไป	27
รูปที่ 3.5	แสดงการจัดการของระบบเชิงเส้น	29
รูปที่ 3.6	แสดงการทำการสลับสีภาพ	30
รูปที่ 3.7	แสดงการทำเทรสโฮล	31
รูปที่ 3.8	แสดงลักษณะของภาพกับฮีสโตแกรม	32
รูปที่ 3.9	แสดงการหาผลรวมของข้อมูลภาพตำแหน่งที่ (x=1,y=1)	35
รูปที่ 3.10	แสดงผลรวมของข้อมูลภาพตำแหน่งที่ (x=1,y=1)	35
รูปที่ 3.11	แสดงผลลัพธ์การรวมค่าของข้อมูลภาพครบทุกจุดภาพ	35
รูปที่ 3.12	แสดงผลลัพธ์การปรับค่าของข้อมูลภาพ	36
รูปที่ 3.13	แสดงการหาขอบภาพด้วยวิธีการของ Sobel	36
รูปที่ 3.14	แสดงภาพต้นแบบก่อนการกรองแบบ Low-Pass	38
รูปที่ 3.15	แสดงภาพต้นแบบก่อนการกรองแบบ High-Pass	38

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

หน้า

รูปที่ 3.16 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีต่ำที่ถูกมาส์กด้วยการกรองแบบ Low-pass 38

รูปที่ 3.17 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีต่ำที่ผ่าการกรองแบบ Low-pass 39

รูปที่ 3.18 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีสูงที่ถูกมาส์กด้วยการกรองแบบ Low-pass 39

รูปที่ 3.19 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีสูงที่ผ่าการกรองแบบ Low-pass 39

รูปที่ 3.20 แสดงภาพต้นแบบก่อนการกรองแบบ High-Pass 41

รูปที่ 3.21 แสดงภาพต้นแบบหลังการกรองแบบ High-Pass 41

รูปที่ 3.22 แสดงภาพต้นแบบก่อนการหาขอบภาพด้วยวิธี Laplacian 42

รูปที่ 3.23 แสดงภาพต้นแบบหลังการหาขอบภาพด้วยวิธี Laplacian 42

รูปที่ 4.1 แสดงตำแหน่งของสัญญาณบนสล็อต IBM 45

รูปที่ 5.1 แสดงบล็อกไดอะแกรมของทั้ง 3 ส่วนที่ต่อเข้าด้วนกัน 54

รูปที่ 5.2 แสดงบล็อกไดอะแกรมของส่วนการแปลงสัญญาณภาพ 55

รูปที่ 5.3 แสดงเวลาที่ใช้ในการสแกนภาพของสัญญาณทางแนวนอน 56

รูปที่ 5.4 แสดงส่วนประกอบของสัญญาณภาพระบบ PAL 56

รูปที่ 5.5 แสดงการสุ่มสัญญาณภาพและการคำนวณหาค่าเวลาหน่วง 57

รูปที่ 5.6 แสดงสัญญาณภาพกับแรงดันอ้างอิงและการต่อวงจร A/D Converter 58

รูปที่ 5.7 แสดงบล็อกไดอะแกรมของวงจร Multipler 59

รูปที่ 5.8 แสดงการอ้างหน่วยความจำของจรเก็บข้อมูลภาพ 59

รูปที่ 5.9 แสดงการอ้างหน่วยความจำของหน่วยประมวลผลภาพ 60

รูปที่ 5.10 แสดงการอ้างหน่วยความจำของคอมพิวเตอร์ (PC) 60

รูปที่ 6.1 แสดงภาพ Lena ที่ได้จากไฟล์ข้อมูลภาพ 63

รูปที่ 6.2 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Sobel Enhancement 64

รูปที่ 6.3 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Laplacian Enhancement 64

รูปที่ 6.4 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Low-Pass Filter 65

รูปที่ 6.5 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี High-Pass Filter 65

รูปที่ 6.6 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Edge Enhancement 66

รูปที่ 6.7 แสดงภาพที่ได้จากกล้อง CCD 66

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

หน้า

รูปที่ 6.8 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Sobel Enhancement 67

รูปที่ 6.9 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Laplacian Enhancement 67

รูปที่ 6.10 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Low-Pass Filter 68

รูปที่ 6.11 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี High-Pass Filter 68

รูปที่ 6.12 แสดงผลของภาพที่ผ่านการประมวลผลโดยวิธี Edge Enhancement 69

รูปที่ 6.13 กราฟแสดงการเปรียบเทียบการประมวลผลภาพระหว่าง PC กับ DSP 69

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้า

ตารางที่ 2.1	แสดงชื่อและหน้าที่ต่าง ๆ ของรีจีสเตอร์ใน CPU	9
ตารางที่ 2.2	แสดงหน้าที่ต่าง ๆ ของ primary bus control register	14
ตารางที่ 2.3	แสดงคำสั่งของไอซี TMS 320C31	18
ตารางที่ 4.1	แสดงการจัดแอดเดรสของหน่วยความจำบน IBM	51
ตารางที่ 4.2	แสดงการใช้งานของพอร์ตบน IBM	52
ตารางที่ 4.3	แสดงการจัดฮาร์ดแวร์อินเตอร์รัพต์	53
ตารางที่ 5.1	แสดงค่าพารามิเตอร์ของระบบต่างๆที่มีใช้กันอยู่	55
ตารางที่ 5.2	แสดงสัญญาณในการควบคุมต่างๆ	61

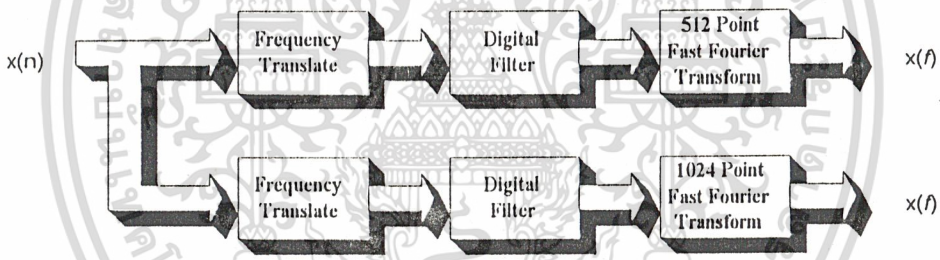
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ดิจิตอลซิกแนลโปรเซสชิง (DSP) คือการเปลี่ยนรูปแบบของสัญญาณ โดยใช้การคำนวณทางคณิตศาสตร์ด้วยดิจิตอลซิกแนลโปรเซสเซอร์ ซึ่งจะเห็นได้ว่าเราสามารถจะนำสัญญาณนั้นใส่ลงไปในสมการคณิตศาสตร์ แล้วได้ผลลัพธ์ออกมาโดยตรงเลยทีเดียว

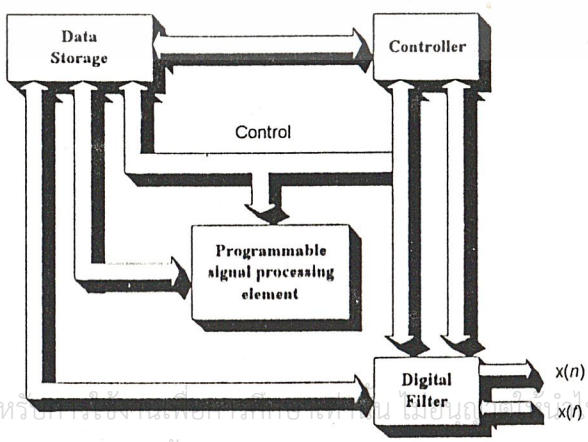
ในช่วง 10 กว่าปีที่ผ่านมา ความก้าวหน้าทางด้านดิจิตอลซิกแนลโปรเซสชิงได้เกิดขึ้นเป็นอย่างมากมีการค้นพบอัลกอริทึม (Algorithms) ใหม่ ๆ เพิ่มขึ้นเรื่อย ๆ และการพัฒนาเครื่องมือใหม่ ๆ เพื่อช่วยในการออกแบบตามอัลกอริทึมนั้นอีกด้วย ระบบดิจิตอลซิกแนลโปรเซสชิงนั้นจะมีอยู่ 2 แบบใหญ่ ๆ คือ

1. ระบบที่ออกแบบมาเพื่อให้ทำงานอย่างใดอย่างหนึ่งโดยเฉพาะ



รูปที่ 1.1 แสดงการประมวลผลเฉพาะอย่าง

2. ระบบที่ใช้ไมโครโปรเซสเซอร์ ซึ่งจะสามารถทำงานเป็นอะไรก็ได้ ขึ้นอยู่กับโปรแกรมที่ใส่



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 1.2 แสดงการใช้ไมโครโปรเซสเซอร์ประมวลผล

การใช้ CARD TMS320C31 ในการประมวลผลภาพ

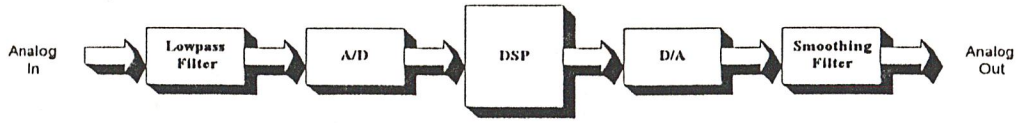
ความก้าวหน้าทางอิเล็กทรอนิกส์ครั้งใหญ่ เกิดขึ้นเมื่อมีการพัฒนาสร้างดิจิตอลซิกแนลโปรเซสเซอร์แบบความเร็วสูงขึ้น (High-Speed Signal Processor) ซึ่งในปัจจุบันนี้ได้มีใช้กันอย่างแพร่หลายในราคาไม่แพงนัก

ดิจิตอลซิกแนลโปรเซสเซอร์จำเป็นจะต้องมีความเร็วสูงและถูกออกแบบมาโดยเฉพาะให้สามารถทำงานได้อย่างมีประสิทธิภาพ โดยใช้ข้อดีของการประมวลผลแบบขนาน และการใช้ชุดคำสั่งที่สร้างขึ้นมาโดยเฉพาะ ทั้งนี้ก็เพื่อการประมวลผลแบบเวลาจริง (Real time) นั่นเองการประมวลผลแบบเวลาจริงคือเมื่อเราใส่สัญญาณอินพุทเข้าไปก็จะได้สัญญาณเอาท์พุทออกมาเลยโดยจะมีเวลาหน่วง (Delay) น้อยมาก ซึ่งจะมีประโยชน์ในการใช้งานมากกว่าการประมวลผลแบบไม่ใช่เวลาจริง

ด้วยความสามารถของดิจิตอลซิกแนลโปรเซสซิง จึงได้นำไปใช้งานในรูปแบบต่างๆ ซึ่งให้ผลลัพธ์ที่ดีกว่าใช้วงจรทางอนาลอก เช่น ความน่าเชื่อถือ ความยืดหยุ่น การทดแทนได้ง่าย ขนาดที่เล็กกว่า และเสถียรภาพในระยะยาวและความสามารถของดิจิตอลซิกแนลโปรเซสเซอร์ที่สามารถเปลี่ยนโปรแกรมการทำงานได้ง่าย ทำให้การใช้งานดิจิตอลซิกแนลโปรเซสเซอร์นั้นน่าสนใจ เช่น สามารถเปลี่ยนให้ดีขึ้นเมื่ออัลกอริทึมใหม่ ๆ ถูกค้นพบหรือความสามารถทำงานได้หลาย ๆ อย่างโดยเพียงแค่เปลี่ยนโปรแกรมให้กับมันเท่านั้น

ในทางอิเล็กทรอนิกส์ โดยส่วนใหญ่มักจะเกี่ยวข้องกับสัญญาณทางไฟฟ้า เช่น การกรองสิ่งที่ไม่ต้องการออกจากสัญญาณอินพุท การแยกเอาข้อมูลออกมาหรือไม่ ก็เป็นการสร้างรูปคลื่นที่ต้องการขึ้นมาหรือการเปลี่ยนลักษณะทางแอมปลิจูดของสัญญาณ ซึ่งในอดีตการกระทำเหล่านี้จะทำโดยใช้วงจร อนาลอกที่ออกแบบมาเพื่องานนั้น ๆ นอกจากนั้นวงจรทางอนาลอกยังไม่เหมาะที่จะสร้างขึ้นเป็นวงจรขนาดใหญ่ (VLSI) อีกด้วย คุณสมบัติของวงจรอนาลอกนั้นจะขึ้นอยู่กับค่าของอุปกรณ์ที่ใช้ เช่น R , C ที่มีค่าไม่แน่นอน ที่สำคัญจะมีกระแสรั่วไหลที่แปรตามเวลา และยังขึ้นอยู่กับ โวลท์เตจและอุณหภูมิด้วย การออกแบบวงจรทางอนาลอกนั้นก็ยังน่าเบื่อเพราะวงจรหนึ่ง ๆ จะทำงานได้อย่างเดียวเท่านั้นแต่ถ้าต้องการให้ทำงานอื่น ๆ ก็ต้องเปลี่ยนอุปกรณ์และก็ต้องเดินสายกันใหม่ นอกจากนี้แล้วการทดสอบวงจร อนาลอกอาจเป็นปัญหาได้เนื่องจากอิมพิแดนซ์ของเครื่องมือวัดที่ต่อเข้าไปในวงจรปัญหาพื้นฐานของวงจรอนาลอกเหล่านี้ สามารถแก้ได้โดยใช้ DSP ข้อได้เปรียบอีกประการของการใช้ DSP แทนวงจรอนาลอกคือ การเปลี่ยนคุณสมบัติของการประมวลผลสามารถทำได้โดยง่าย ซึ่งจะทำให้เราสามารถทดลอง ในทุกทางที่เป็นไปได้ เพื่อเลือกเอาผลดีที่สุดที่เราพอใจ

โดยทั่วไประบบ DSP จะประกอบด้วยส่วนต่างๆดังรูป 1.3



รูปที่ 1.3 แสดงส่วนประกอบของ DSP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

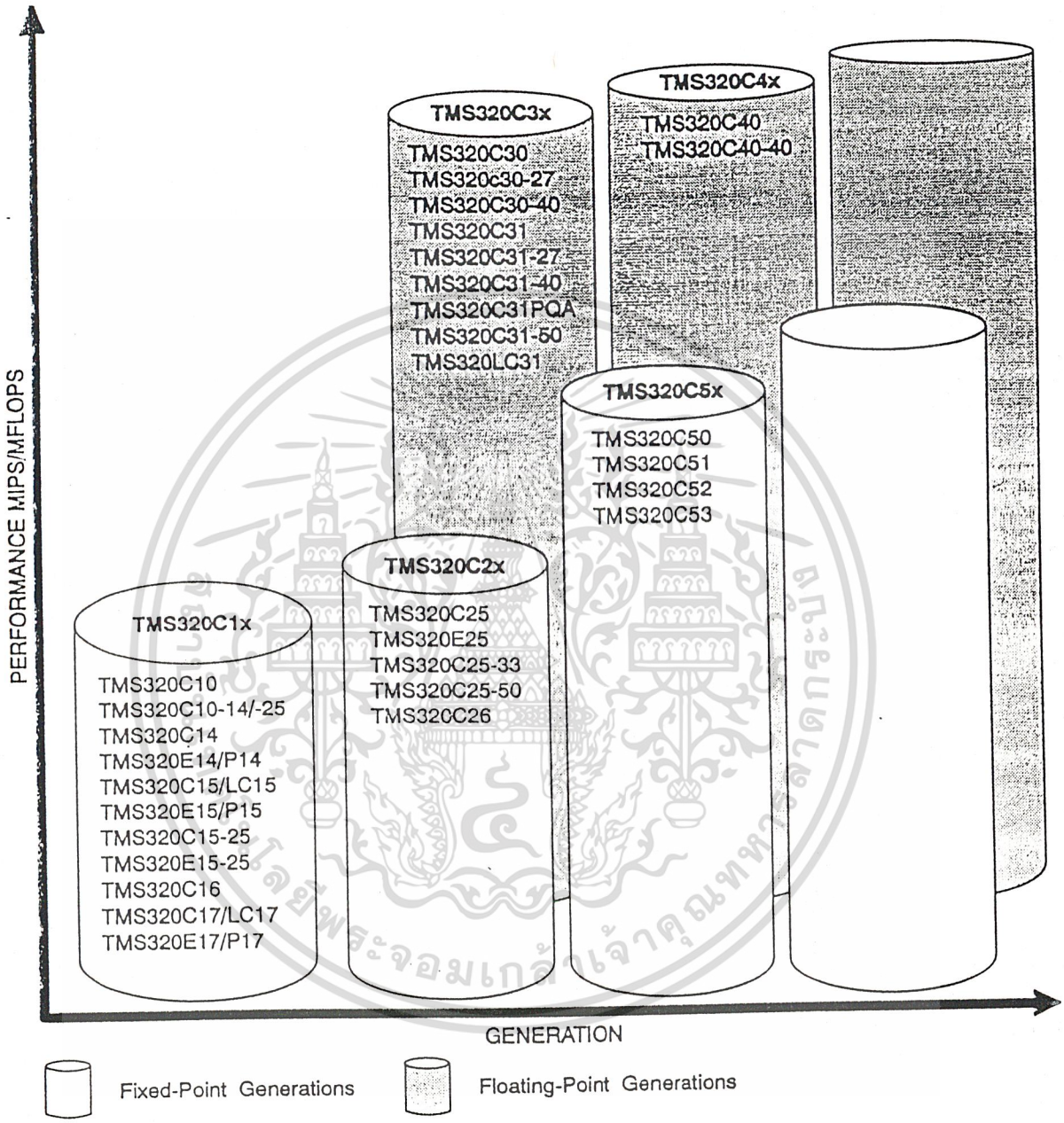
สัญญาณอนาลอกอินพุท จะถูกป้อนเข้าสู่ Low pass filter เพื่อที่จะจำกัดช่วงความถี่ของสัญญาณอินพุท จากนั้นก็จะส่งต่อไปยัง อนาลอก/ดิจิตอล คอนเวอร์เตอร์ (Analog to Digital Converter ; A/D) แล้วส่งให้ระบบประมวลสัญญาณดิจิตอลต่อไป ระบบประมวลผลสัญญาณดิจิตอลก็จะนำเอาอัลกอริทึมที่มีอยู่มากระทำกับข้อมูล จากนั้นก็จะส่งสัญญาณข้อมูลที่ถูกแก้ไขแล้ว ไปให้ส่วนดิจิตอล / อนาลอก คอนเวอร์เตอร์ (Digital to Analog Converter :D/A) เพื่อเปลี่ยนสัญญาณข้อมูลกลับเป็นสัญญาณอนาลอกตามที่ต้องการ แต่จากเอาท์พุทของ D/A ซึ่งจะมีลักษณะเป็นขั้น ๆ แยกกัน ดังนั้นเราจำเป็นต้องใช้วงจรกรองสัญญาณให้เรียบ (Low Pass Filter) เพื่อที่จะขจัดสัญญาณความถี่สูงที่ไม่ต้องการออกไป

หากเราจะมองข้อเสียของ DSP ก็มีแต่ข้อเสียหรือปัญหาที่เกิดขึ้น จะแตกต่างไปจากที่พบในวงจรอนาลอก เช่น การเกิดควอนไตซ์เซชัน(Quantization) ,การเกิดน๊อยส์ (Noise) , Idle channal noise ,ความไม่ราบเรียบของสัญญาณ , ความละเอียดของสัญญาณ เป็นต้น ตัวแปรที่สำคัญอีกตัวหนึ่งที่จะแยกระบบ DSP ต่าง ๆ ออกตามการใช้งานก็คือ ค่าอัตราการสุ่มข้อมูล (Sampling Rate) ยิ่งอัตราการสุ่มข้อมูลมากยิ่งต้องใช้ระบบประมวลผลที่มีความเร็วสูงยิ่งขึ้นและซับซ้อนยิ่งขึ้นด้วย อัตราการสุ่มข้อมูลนี้จะมีผลต่อการนำไปใช้งาน ถ้ามีอัตราการสุ่มข้อมูลที่สูง ก็จะใช้งานในย่านความถี่ที่สูงขึ้นได้ โดยคุณภาพของสัญญาณข้อมูลยังคงยอมรับได้

บริษัทเทกซัสอินสตูเมนต์ ได้ผลิตไอซีที่เป็นตัวประมวลผลออกมาสำหรับใช้งานทางด้านของดิจิตอลซิกแนลโปรเซสซิงโดยเฉพาะ ซึ่งเป็นตระกูลที่มีชื่อว่า TMS 320 ซึ่งมีวิวัฒนาการดังรูปที่ 1.4 ไอซีตระกูล TMS 320 นี้ จะเป็นลักษณะไมโครโปรเซสเซอร์แบบชิพเดี่ยว (Single Chip) สามารถทำงานได้กว้างขวาง และมีความเร็วในการประมวลผลสูง

ไอซีเบอร์ TMS 320C31 ที่ใช้ในโครงงานนี้เป็นไอซีที่ได้ถูกพัฒนาขึ้นมาทำให้มีความเร็วในการประมวลผลสูงและสามารถนำไปใช้งานได้อย่างกว้างขวาง โดยเฉพาะการประมวลผลแบบเวลาจริง (Real Time) การประยุกต์ใช้งานไอซีเบอร์ TMS 320C31 ได้แสดงในรูปที่ 1.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 1.4 แสดงวิวัฒนาการของไอซีตระกูล TMS 320xx

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

General-Purpose DSP	Graphics/Imaging	Instrumentation
Digital Filtering	3-D Transformations Rendering	Spectrum Analysis
Convolution	Robot Vision	Function Generation
Correlation	Image Transmission/Compression	Pattern Matching
Hilbert Transforms	Pattern Recognition	Seismic Processing
Fast Fourier Transforms	Image Enhancement	Transient Analysis
Adaptive Filtering	Homomorphic Processing	Digital Filtering
Windowing	Workstations	Phase-Locked Loops
Waveform Generation	Animation/Digital Map	
Voice/Speech	**Control**	**Military**
Voice Mail	Disk Control	Secure Communications
Speech Vocoding	Servo Control	Radar Processing
Speech Recognition	Robot Control	Sonar Processing
Speaker Verification	Laser Printer Control	Image Processing
Speech Enhancement	Engine Control	Navigation
Speech Synthesis	Motor Control	Missile Guidance
Text-to-Speech	Kalman Filtering	Radio Frequency Modems
Neural Networks		Sensor Fusion
Telecommunications		**Automotive**
Echo Cancellation	FAX	Engine Control
ADPCM Transcoders	Cellular Telephones	Vibration Analysis
Digital PBXs	Speaker Phones	Antiskid Brakes
Line Repeaters	Digital Speech	Adaptive Ride Control
Channel Multiplexing	Interpolation (DSI)	Global Positioning
1200- to 19200-bps Modems	X.25 Packet Switching	Navigation
Adaptive Equalizers	Video Conferencing	Voice Commands
DTMF Encoding/Decoding	Spread Spectrum	Digital Radio
Data Encryption	Communications	Cellular Telephones
Consumer	**Industrial**	**Medical**
Radar Detectors	Robotics	Hearing Aids
Power Tools	Numeric Control	Patient Monitoring
Digital Audio/TV	Security Access	Ultrasound Equipment
Music Synthesizer	Power Line Monitors	Diagnostic Tools
Toys and Games	Visual Inspection	Prosthetics
Solid-State Answering Machines	Lathe Control	Fetal Monitors
	CAM	MR Imaging

รูปที่ 1.5 แสดงการประยุกต์ใช้งาน TMS 320C31

ในปริญญานิพนธ์นี้เรานำ TMS 320C31 มาสร้างเป็นระบบ DSP ซึ่งเรากำหนดคุณสมบัติ เบื้องต้นดังนี้

- สามารถนำไปประยุกต์ใช้งานในด้านการประมวลผลภาพดิจิตอลที่มีความแตกต่างระดับสีเทา (Gray Level) 256 ระดับ 512 x 512 Pixel
- สามารถเปลี่ยนแปลงแก้ไขอัลกอริทึมในการประมวลได้โดยง่าย
- แหล่งข้อมูลภาพสามารถนำมาจากกล้อง CCD ได้

ทั้งนี้ก็เพื่อความยืดหยุ่นของการประมวลผลภาพที่จะนำไปใช้งานได้หลาย ๆ แบบ และจะได้นำไปใช้เป็นชุดทดลองและพัฒนา DSP ต่อไป

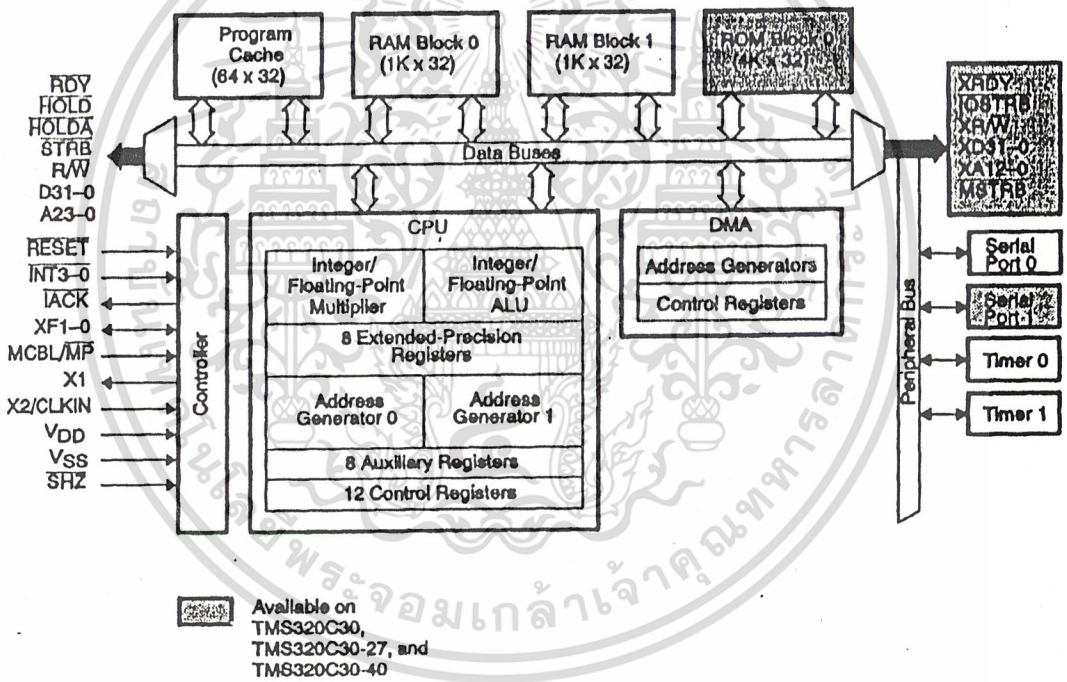
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารนี้ทุกครั้งที่มีการนำไปใช้

บทที่ 2

สถาปัตยกรรมของ TMS320C31

ไอซีเบอร์ TMS320C31 ถูกสร้างและออกแบบมาเพื่อตอบสนองความต้องการทางด้านการประมวลผลบนพื้นฐานของคณิตศาสตร์ชั้นสูงและการแก้ปัญหาทาง hard ware และ software

ประสิทธิภาพอันสูงของ TMS320C31 เกิดจากความแน่นอนและความกว้างทางไดนามิกของส่วนคำนวณทางทศนิยม มีหน่วยความจำขนาดใหญ่ การทำงานแบบขนานและมี DMA Controller เพื่อการทำงานที่มีประสิทธิภาพมากขึ้น



รูปที่ 2.1 แสดงสถาปัตยกรรมของไอซีเบอร์ TMS320C3X

สถาปัตยกรรม TMS320C31 ประกอบด้วยส่วนต่าง ๆ สำคัญดังนี้

2.1 หน่วยประมวลผลกลาง (CPU)

หน่วยประมวลผลกลางจะประกอบไปด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.1 Multiplier

มีหน้าที่ในการคูณ และสามารถคูณจำนวนเต็ม 24 bit และ จำนวนทศนิยม 32 bit โดยใช้เวลาเพียง 1 รอบ ซึ่งการคำนวณจำนวนทศนิยมจะใช้เวลา 50 ns ต่อรอบ และสามารถประมวลผลแบบขนานได้เพื่อที่จะเพิ่มความสามารถในการประมวลผลข้อมูล โดยใช้คำสั่งแบบขนานให้ทำการคูณและการทำงานของ ALU ภายในหนึ่งรอบ

เมื่อ multiplier ทำการคำนวณทศนิยม อินพุทที่ใช้จะมีขนาด 32 บิต และผลลัพธ์ที่ได้จะมีขนาด 40 บิต แต่ถ้า multiplier ทำการคูณจำนวนเต็มอินพุทจะมีขนาดเพียง 24 บิต และจะได้ผลลัพธ์ขนาด 32 บิต

2.1.2 หน่วยคำนวณทางคณิตศาสตร์ (Arithmetic Logic Unit (ALU))

ใน 1 รอบการทำงานของ ALU จะประมวลผลจำนวนเต็มได้ 32 บิตและทศนิยมได้ 40 บิต โดยผลลัพธ์จาก ALU ยังคงมีขนาด 32 บิตถ้าเป็นจำนวนเต็ม และ 40 บิตถ้าเป็นทศนิยม barrel shifter จะมีหน้าที่ในการเลื่อนข้อมูล 32 บิต ทั้ง ซ้ายและขวาใน 1 รอบของการทำงาน

บัสภายใน (CPU1/CPU2 และ REG1/REG2) จะเป็นตัวพาข้อมูลที่จะนำมาประมวลผล 2 ตัวจากหน่วยความจำ และอีก 2 ตัวจากรีจีสเตอร์ ดังนั้นจึงสามารถทำการคูณ บวก หรือลบแบบขนานได้ ใน 1 รอบการทำงาน

2.1.3 หน่วยช่วยคำนวณทางคณิตศาสตร์ (Auxiliary Register Arithmetic Unit (ARAUs))

ARAUS เป็นรีจีสเตอร์ที่ช่วยในการทำงานแบบขนานกับตัวคูณและ ALU โดย ARAUs จะมี 2 ตัว คือ ARAU0 และ ARAU1 ARAU จะสามารถเก็บแอดเดรสได้ 2 แอดเดรสใน 1 รอบการทำงานเพื่อใช้ในการอ้างตำแหน่งแบบต่าง ๆ

2.1.4 CPU Register File

ใน TMS320C31 จะมีรีจีสเตอร์อยู่ 28 ตัว ซึ่งตรงกับ CPU โดย รีจีสเตอร์ทั้งหลายเหล่านี้จะทำงานกับ multiplier และ ALU และยังสามารถใช้เพื่อวัตถุประสงค์อื่นอีกด้วย อย่างไรก็ตามรีจีสเตอร์เหล่านี้ก็ยังมีหน้าที่พิเศษบางอย่างด้วย เช่น ใช้สำหรับการทำงานเกี่ยวกับจำนวนเลขทศนิยม และรีจีสเตอร์ช่วย (Auxiliary register) ทั้ง 8 ตัวยังใช้ในการอ้างแอดเดรสแบบ indirect addressing register ด้วย รีจีสเตอร์ที่เหลือจะทำหน้าที่เกี่ยวกับระบบต่างๆ เช่น การอ้างแอดเดรส การจัดการเกี่ยวกับแสตก (stack) การบอกสถานะของโปรเซสเซอร์ การอินเตอร์รัพต์ และการเคลื่อนย้ายข้อมูลเป็นบล็อก

ชื่อต่างๆของรีจีสเตอร์และหน้าที่ของมันแสดงดังตาราง 2.1

-exten-precision register (R7-R0) :สามารถใช้เก็บค่าจำนวนเต็ม 32 บิต และทศนิยม 40 บิต โดยจะเก็บค่าลงในบิตที่ 0-39 แต่ถ้าจำนวนเต็มแบบมีเครื่องหมายก็จะทำการเก็บที่ บิต 0-31 ส่วนบิตที่ 32-39 จะไม่ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-auxilairy register (AR7-AR0) : จะถูกใช้โดย CPU และถูกควบคุมโดย ARAU ทั้ง 2 ตัว หน้าที่แรกของ AR ยังใช้ให้ส่วนที่เก็บตัวนับในการวนลูปหรือจะใช้เพื่องานอื่นๆ ที่เกี่ยวกับตัวมัลติเพล็กเซอร์และ ALU

-data page pointer (DP) : ส่วนรีจีสเตอร์ขนาด 32 บิต โดยที่ 8 บิตแรก จะถูกใช้ในการอ้างแอดเดรสแบบ direct addressing mode ซึ่งใช้เป็นตัวชี้ (pointer) ไปยังหน้า (page) ต่าง ๆ ของข้อมูล

-index register (IR0,IR1) : จะถูกใช้โดย ARAU ในการเก็บค่าของดัชนีในการชี้แอดเดรส (indexed address)

-block size register (BK) : มีขนาด 32 บิตซึ่ง ARAU จะใช้ในการอ้างแอดเดรสแบบเก็บ circular addressing เพื่อเป็นตัวเก็บค่าของขนาดของบล็อกข้อมูล

-system stack pointer (SP) : เป็นรีจีสเตอร์ขนาด 32 บิต ซึ่งเป็นค่าของแอดเดรสของยอดของแสตค โดยปกติ SP จะชี้ไปยังค่าสุดท้ายซึ่งถูกเก็บอยู่บนแสตค การ PUSH จะกระทำ ก่อนที่จะเพิ่มค่าของ SP และการ POP หลังการลดค่าของ SP คำสั่งที่จะทำงานกับ SP ได้คือ interrupts, traps, returns, PUSH และ POP

-status register (ST) : เป็นรีจีสเตอร์ที่เก็บข้อมูลต่างๆ ที่แสดงสถานะของ CPU เช่นค่าของ flag ซึ่งจะแสดงสถานะของผลลัพธ์ที่ได้จากการคำนวณทางคณิตศาสตร์ ว่า เป็น ศูนย์ ลบ ฯลฯ รวมถึงการคำนวณทางลอจิกด้วย

-CPU/DMA interrupt enable register (IE) : เป็นรีจีสเตอร์ขนาด 32 บิต ใช้สำหรับการกำหนดการเอ็นนาเบิ้ลอินเตอร์รัพต์ของ CPU โดยการเอ็นนาเบิ้ลของ CPU จะใช้บิตที่ 0-10 และการเอ็นนาเบิ้ลของ DMA จะใช้บิตที่ 16-26 "1" จะเป็นการเอ็นนาเบิ้ล และ "0" จะเป็นการดีสเอเบิ้ล

-I/O Flage register (IOF) : เป็นรีจีสเตอร์ในการควบคุมขา XF0 และ XF1

- repeat counter (RC) : เป็นรีจีสเตอร์ขนาด 32 บิตใช้เก็บค่าของจำนวนเท่าหรือจำนวนครั้งของบล็อกข้อมูลซึ่งถูกกระทำซ้ำ เมื่อโปรเซสเซอร์ทำงานในโหมดของการกระทำในข้อมูลซ้ำ (repeat mode) repeat star address register (RS) รีจีสเตอร์ขนาด 32 บิต จะถูกใช้ในการเก็บแอดเดรสเริ่มต้นของบล็อกข้อมูลที่จะถูกกระทำซ้ำ และ repeat end address register (RE) : :ซึ่งเป็นรีจีสเตอร์ขนาด 32 บิต จะถูกเก็บแอดเดรสสุดท้ายของบล็อกข้อมูลซึ่งจะถูกกระทำซ้ำ

-program counter (PC) : เป็นรีจีสเตอร์ขนาด 32 บิต ซึ่งเก็บแอดเดรสของคำสั่งที่จะถูกนำมาอ่านคำสั่งถัดไป

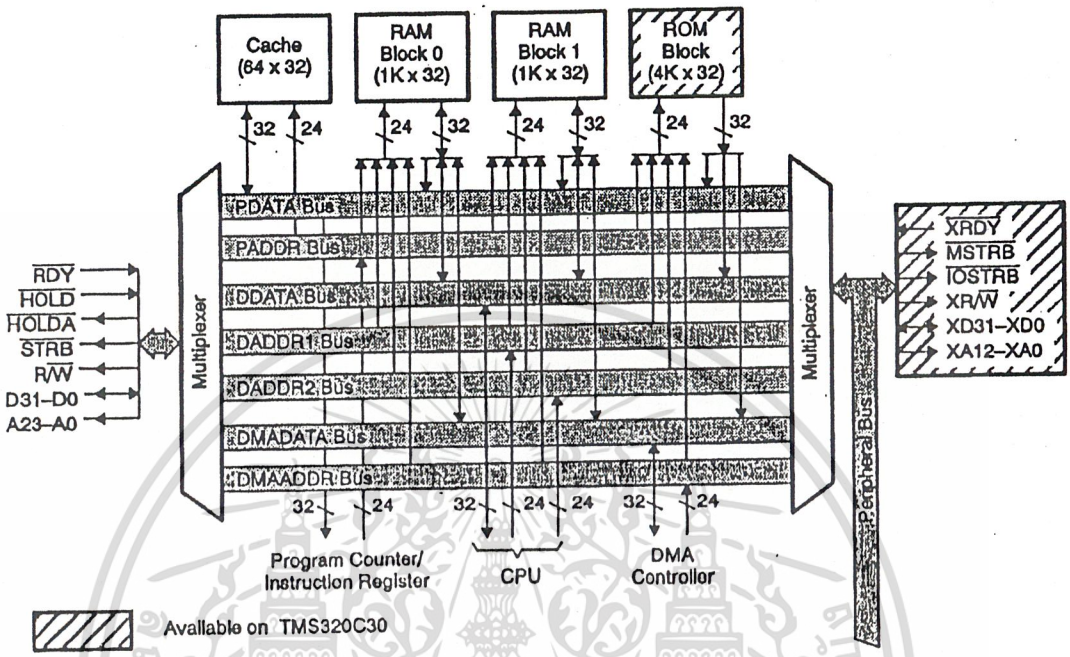
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Register Name	Assigned Function
R0	Extended-precision register 0
R1	Extended-precision register 1
R2	Extended-precision register 2
R3	Extended-precision register 3
R4	Extended-precision register 4
R5	Extended-precision register 5
R6	Extended-precision register 6
R7	Extended-precision register 7
AR0	Auxiliary register 0
AR1	Auxiliary register 1
AR2	Auxiliary register 2
AR3	Auxiliary register 3
AR4	Auxiliary register 4
AR5	Auxiliary register 5
AR6	Auxiliary register 6
AR7	Auxiliary register 7
DP	Data-page pointer
IR0	Index register 0
IR1	Index register 1
BK	Block-size register
SP	System-stack pointer
ST	Status register
IE	CPU/DMA interrupt-enable
IF	CPU interrupt flag
IOF	I/O flag
RS	Repeat start-address
RE	Repeat end-address
RC	Repeat counter
PC	Program counter

ตารางที่ 2.1 ชื่อและหน้าที่ต่างๆของรีจีสเตอร์ใน CPU

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 หน่วยความจำ (Memory Organization)



รูปที่ 2.2 แสดงหน่วยความจำของไอซีเบอร์ TMS 320C31

หน่วยความจำของ TMS320C31 จะมีขนาด 16M และมีขนาดของ 1 word เท่ากับ 32บิต โดยในหน่วยความจำนี้จะใช้กับโปรแกรมข้อมูล และการอินพุท เอาท์พุท ดังนั้นค่าของสัมประสิทธิ์โปรแกรมหรือข้อมูลจะถูกเก็บได้ทั้ง RAM และ ROM

2.2.1 RAM, ROM

รูปที่ 2.2 จะแสดงการจัดหน่วยความจำภายใน TMS320C31 โดยแรมจะแบ่งออกเป็น บล็อก 0 และ 1 ซึ่งจะมีขนาดบล็อกละ 1K x 32 และ ROM จะมีขนาด 4K x 32 ทั้ง RAM และ ROM สามารถถูกเข้าใช้ได้โดย CPU 2 ครั้งใน 1 รอบของการทำงานได้ดังนั้นการที่มีบัสของโปรมแกรมแยกกันและบัสของ DMAแยกกันทำให้การอ่านโปรแกรม การอ่านและการเขียนข้อมูล หรือ การทำงานของ DMA ที่สามารถทำงานแบบขนานได้

2.2.2 Memory Maps

ตารางหน่วยความจำจะขึ้นอยู่กับว่าจะให้โปรเซสเซอร์ประมวลผลในโหมดของไมโครโปรเซสเซอร์ หรือไมโครคอมพิวเตอร์ แต่ในโครงงานนี้จะให้โปรเซสเซอร์ประมวลผลในโหมดของไมโครโปรเซสเซอร์ แสดงดังรูปที่ 2.3 ในแอดเดรสที่ 800000h จนถึง 801FFFh จะถูกใช้สำหรับ expansion bus ซึ่งจะถูกเข้าถึงได้โดยให้สัญญาณ / MSTRB แอคตีฟ แอดเดรสที่ 80200h จนถึง 803FFFh จะไม่ใช้แอดเดรสที่ 80400h ถึง 805FFFh จะถูกใช้สำหรับ expansion bus ซึ่งจะถูกทำให้เข้าถึงโดยให้สัญญาณ /IOSTRB แอคตีฟ แอดเดรส 80600h ถึง 807FFFh จะไม่ใช้ทุกๆ รีจีสเตอร์จะอยู่ที่แอดเดรส 80800h ถึง 8097FFh RAM บล็อก 0 จะอยู่

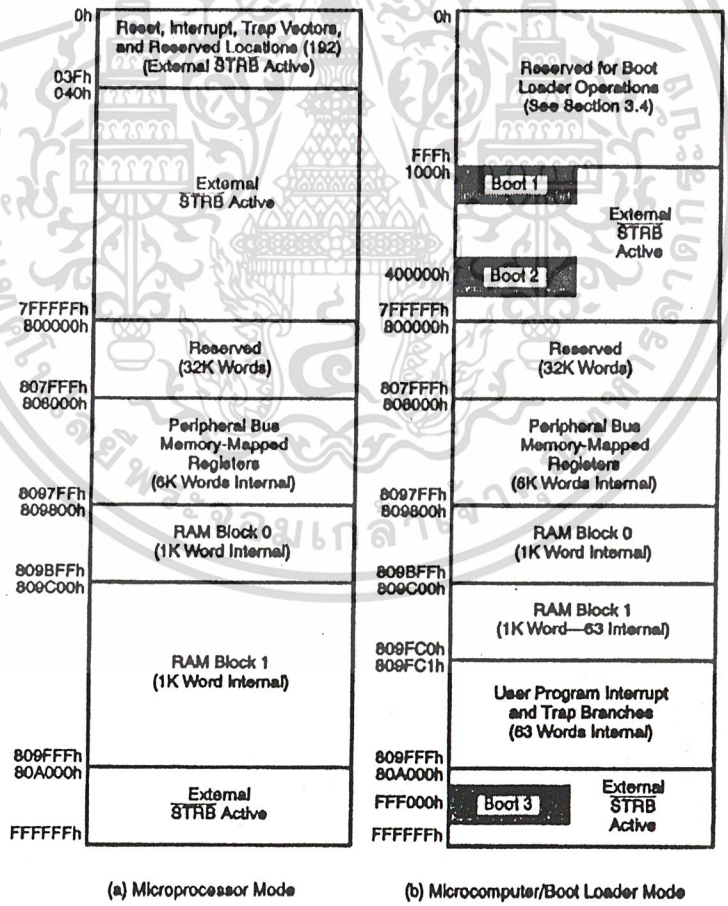
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่ 809800h ถึง 809BFFh และ บล็อก 1 จะอยู่ที่ 809C00h ถึง 809FFh แอดเดรส 80A00h จนถึง 0FFFFFFh จะถูกใช้โดยอุปกรณ์ภายนอก (เมื่อสัญญาณ /STRB แอคตีฟ)

2.2.3 Memory Address Modes

TMS 320C31 จะแบ่งโหมดของการอ้างแอดเดรสออกเป็น 5 กลุ่ม โดยมีโหมดการอ้างแอดเดรสทั้งหมด 6 แบบ ดังนี้

1. General addressing mode
   - Register : ค่าที่นำมาคำนวณจะเก็บที่ CPU รีจิสเตอร์
   - Short immediate : ค่าที่นำมาคำนวณจะเป็นค่าที่โหลดโดยตรงขนาด 16 บิต จากคำสั่ง
   - Direct : ค่าที่นำมาคำนวณจะถูกเก็บอยู่ในตำแหน่ง ซึ่งมีแอดเดรสขนาด 24 บิตแสดงอยู่ในคำสั่ง
   - Indirect : รีจิสเตอร์ช่วยจะเก็บแอดเดรสของข้อมูล



รูปที่ 2.3 แสดงการจัดหน่วยความจำของไอซี TMS320C31

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. Three-operand addressing mode :
   - Register
   - Indirect
3. Parallel addressing mode :
   - Register : ข้อมูลจะอยู่ใน extended-precision register
   - Indirect
4. Long-immediate addressing mode :
   - Long-immediate : ข้อมูลจะเป็นค่าที่โหลดโดยตรงขนาด 24 บิต จากคำสั่ง
5. Conditional branch addressing mode :
   - Register
   - PC-relative : ค่าที่มีเครื่องหมายขนาด 16 บิต จะถูกบวกกับค่าใน PC

2.3 Internal Bus Operation

ข้อดีที่สำคัญประการหนึ่งของ TMS 320C31 คือการที่มีบัสภายในและสามารถประมวลผลแบบขนานได้ บัสโปรแกรมที่แยกกัน (PADDRและPDATA) บัสข้อมูลที่แยกกัน (DADDR1,DADDR2และDDATA) และบัส DMA ที่แยกกัน (DMAADDR และ DMADATA) ทำให้สามารถทำการประมวลผลแบบขนานได้ บัสเหล่านี้สามารถต่อได้กับหน่วยความจำภายใน หน่วยความจำภายนอก หรือ อุปกรณ์ภายนอกอื่นๆ รูปที่ 2.2 จะแสดงการต่อบัสต่างๆ เหล่านี้

PC จะถูกต่ออยู่กับ PADDR (ขนาด 24 บิต) IR จะต่ออยู่กับ PDATA (ขนาด 32 บิต) ซึ่งบัสเหล่านี้สามารถอ่านคำสั่ง ทุกๆรอบของการทำงาน

DADDR1 และ DADDR2 (ขนาด 24 บิต) และ DDATA (ขนาด 32 บิต) จะใช้ในการติดต่อกับหน่วยความจำในทุกๆรอบของการทำงาน DDATA จะใช้ในการส่งข้อมูลไปยัง CPU โดยผ่านบัส CPU1 และ CPU2 บัส CPU1 และ CPU2 สามารถส่งข้อมูลจากหน่วยความจำ 2 ข้อมูลไปยัง Multiplier ALU และรีจิสเตอร์ในทุกๆ รอบของการทำงาน นอกจากนี้ภายใน CPU มีบัสรีจิสเตอร์ REG1 และ REG2 ซึ่งใช้ในการหาข้อมูล 2 ค่า จากรีจิสเตอร์ไปยัง Multiplier และ ALU ในทุกๆ รอบการทำงาน รูปที่ 2.2 แสดงบัสภายในต่างๆ DMA Controller จะใช้ DMAADDR (ขนาด24 บิต)และ DMADATA (ขนาด 32 บิต) โดยบัสเหล่านี้ทำให้ DMA สามารถติดต่อกับหน่วยความจำแบบขนานได้

2.4 External Bus Operation

หน่วยความจำและอุปกรณ์ภายนอกสามารถอินเตอร์เฟสกับ TMS 320C31 ได้ 2 ทางคือ ทาง Primary bus expansion bus การอินเตอร์เฟสกับหน่วยความจำหรืออุปกรณ์ภายนอกที่มีความเร็วช้ากว่า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TMS 320C31 จะต้องเพิ่ม Wait state เข้าไป โดยอาศัยการควบคุมของ memory mapped control register และสัญญาณจากภายนอก

primary bus มีขนาดของ data bus 32 บิต, address bus 24 บิต และcontrol signal อีก 1 ชุด expansion bus มีขนาด data bus 32 บิต, address bus 13 บิต และcontrol signal อีก 1 ชุด โดยบัสทั้งสอง สามารถถูกควบคุมได้โดยการใช้โปรแกรม(Software) สั่งงานและสัญญาณควบคุมจากภายนอก

Register	Peripheral Address
Expansion-Bus Control (see subsection 7.1.2)†	808060h
Reserved	808061h
Reserved	808062h
Reserved	808063h
Primary-Bus Control (see subsection 7.1.1)	808064h
Reserved	808065h
Reserved	808066h
Reserved	808067h
Reserved	808068h
Reserved	808069h
Reserved	80806Ah
Reserved	80806Bh
Reserved	80806Ch
Reserved	80806Dh
Reserved	80806Eh
Reserved	80806Fh

† Reserved on the TMS320C31

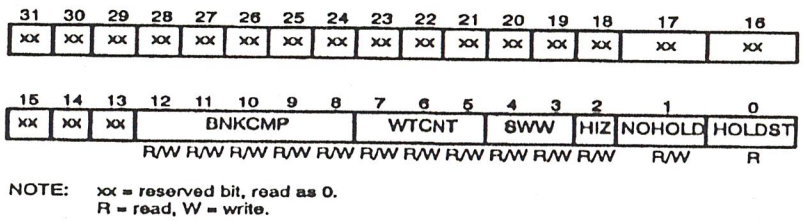
รูปที่ 2.4 memory-mapped external Interface control register

อุปกรณ์ภายนอกสามารถเข้าถึงได้โดยการใช้สัญญาณ /STRB,/MSTRB และ /IOSTRB โดยสัญญาณ /STRB จะใช้เมื่อต้องการ primary bus ส่วน expansion bus จะสามารถใช้งานได้โดย 2 แบบ คือ

- การเข้าใช้ memory สามารถทำได้โดยให้ /MSTRB เป็น 0 โดย /MSTRB จะมีวงรอบการทำงานเหมือนกับ /STRB

- อุปกรณ์ภายนอกสามารถเข้าใช้ได้โดยให้สัญญาณ /IOSTRB เป็น 0

primary bus และ expansion bus จะมี control register ดังรูปที่ 2.5



รูปที่ 2.5 primary bus control register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

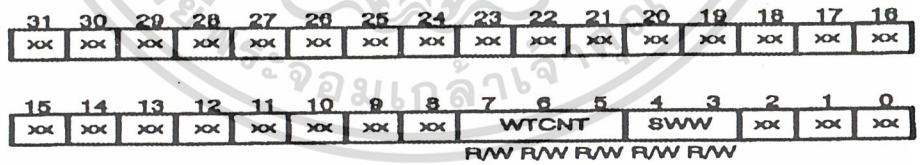
Bit	Name	Reset Value	Function
0	HOLDST	x †	Hold status bit. This bit signals whether the port is being held (HOLDST = 1) or is not being held (HOLDST = 0). This status bit is valid whether the port has been held via hardware or software.
1	NOHOLD	0	Port hold signal. NOHOLD allows or disallows the port to be held by an external $\overline{\text{HOLD}}$ signal. When NOHOLD = 1, the TMS320C3x takes over the external bus and controls it, regardless of serviced or pending requests by external devices. No hold acknowledge ($\overline{\text{HOLDA}}$) is asserted when a $\overline{\text{HOLD}}$ is received. However, it is asserted if an internal hold is generated (HIZ = 1). NOHOLD is set to 0 at reset.
2	HIZ	0	Internal hold. When set (HIZ = 1), the port is put in hold mode. This is equivalent to the external $\overline{\text{HOLD}}$ signal. By forcing a high-impedance condition, the TMS320C3x can relinquish the external memory port through software. $\overline{\text{HOLDA}}$ goes low when the port is placed in the high-impedance state. HIZ is set to 0 at reset.
4–3	SWW	11	Software wait mode. In conjunction with WTCNT, this two-bit field defines the mode of wait-state generation. It is set to 1 1 at reset.
7–5	WTCNT	111	Software wait mode. This three-bit field specifies the number of cycles to use when in software wait mode for the generation of internal wait states. The range is 0 (WTCNT = 0 0 0) to 7 (WTCNT = 1 1 1) H1/H3 cycles. It is set to 1 1 1 at reset.
12–8	BNKCMP	10000	Bank compare. This five-bit field specifies the number of MSBs of the address to be used to define the bank size. It is set to 1 0 0 0 0 at reset.
31–13	Reserved	0–0	Read as 0.

† x = 0 or 1

ตารางที่ 2.2 แสดงหน้าที่ต่าง ๆ ของ primary bus control register

control register ของ primary bus จะมีขนาด 32 บิต ดังรูปที่ 2.5 และตาราง 2.2

control register ของ expansion bus จะมีขนาด 32 บิต ดังรูปที่ 2.6 และตาราง 2.3



NOTE: xx = reserved bit, read as 0.
R = read, W = write.

รูปที่ 2.6 expansion bus control register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Bit	Name	Reset Value	Function
2–0	Reserved	000	Read as 0.
4–3	SWW	11	Software wait-state generation. In conjunction with the WTCNT, this two-bit field defines the mode of wait-state generation. It is set to 1 1 at reset.
7–5	WTCNT	111	Software wait mode. This three-bit field specifies the number of cycles to use when in software wait mode for the generation of internal wait states. The range is 0 (WTCNT = 0 0 0) to 7 (WTCNT = 1 1 1) H1/H3 clock cycles. It is set to 1 1 1 at reset.
31–8	Reserved	0–0	Read as 0.

ทุกๆ วงรอบของสัญญาณบนสายบัสจะขึ้นอยู่กับสัญญาณนาฬิกา H1 โดย 1 รอบของ H1 เริ่มนับจากขอบขาลงจนถึงขอบขาลงของลูกถัดไป สำหรับการทำงานแบบความเร็วเต็มที่(คือไม่ใช้ wait state) ในการเขียนข้อมูลจะใช้เวลา 2 รอบของ H1 และการอ่านข้อมูลจะใช้ 1 รอบของ H1 แต่ถ้าการอ่านข้อมูลกระทำต่อจากคำสั่งการเขียนข้อมูล การอ่านข้อมูลจะใช้เวลา 2 รอบของ H1 ซึ่งหลักการนี้จะใช้ได้ทั้งกับ primary bus และ expansion bus (เมื่อใช้ /MSTRB)

ถ้าการเขียนข้อมูลเป็นการกระทำภายใน โดยไม่มีการอินเตอร์เฟสกับอุปกรณ์ภายนอก(คือการใช้จาก CPU และ DMA) วงรอบของการเขียนข้อมูลจะใช้เพียง 1 รอบของ H1 เท่านั้น

สัญญาณ /MSTRB และ /STRB จะแอคตีฟที่สถานะ 0 สำหรับการอ่านและการเขียนข้อมูล ในขณะที่สภาวะก่อนหลังของการแอคตีฟของ /MSTRB และ /STRB ในการเขียนข้อมูลจะมี transion cycle ของ H1 ในระหว่าง transion cycle จะเกิด

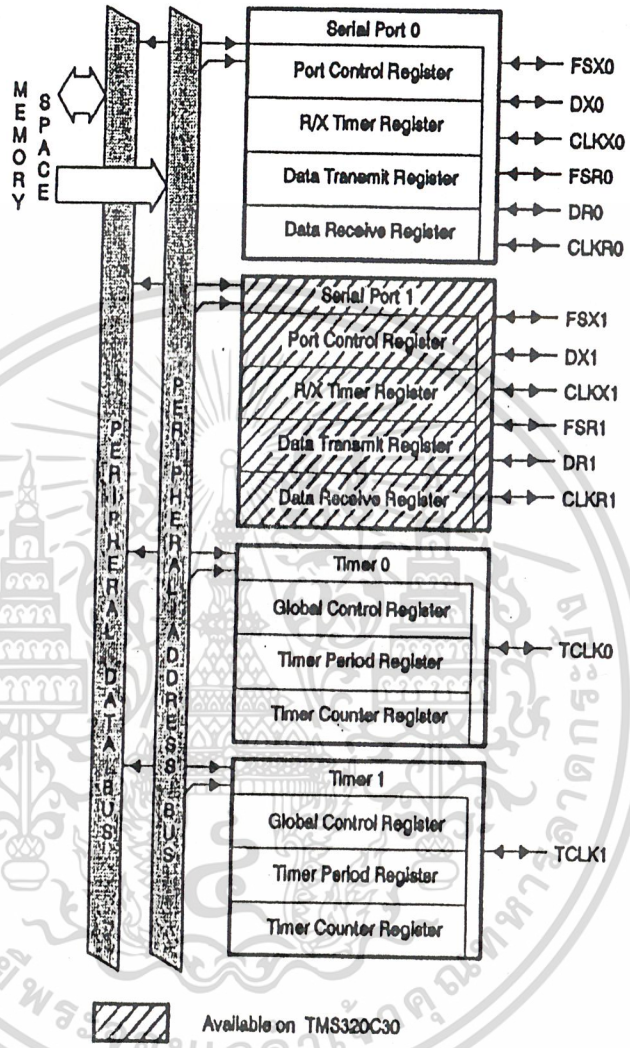
1. /MSTRB และ /STRB จะเป็นสถานะ high
2. (X)R/W จะเปลี่ยนสถานะที่ขอบขาขึ้นของ H1
3. ถ้าเป็นสถานะของการเขียนข้อมูล address จะเปลี่ยนสถานะที่ขอบขาขึ้นของ H1 และถ้าเป็นสถานะของการอ่านข้อมูล address จะเปลี่ยนที่ขอบขาลงของ H1

2.5 ระบบติดต่อภายนอก (Peripherals)

อุปกรณ์ภายนอกที่ติดต่อกับ TMS 320C31 จะควบคุมโดยผ่านหน่วยความจำในตำแหน่ง 8097FFh ถึง 808000h บัสที่ติดต่อกับอุปกรณ์ภายนอกหรือ peripheral bus จะมีบัสข้อมูลขนาด 32 บิต และบัส แอดเดรสขนาด 24 บิต

การติดต่อกับอุปกรณ์ภายนอกจะผ่าน timer 2 ตัว และพอร์ตอนุกรม 2 ตัว รูปที่ 2.7 จะแสดงบัสและขาสัญญาณต่างๆ ที่ใช้ในการติดต่อกับอุปกรณ์ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



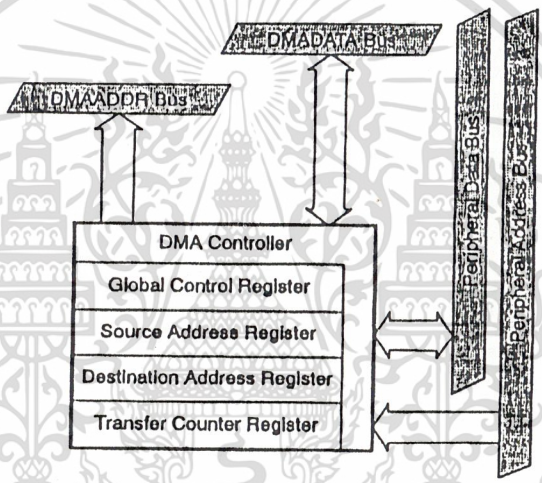
รูปที่ 2.7 peripheral module

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 ระบบถ่ายโอนข้อมูล (Direct Memory Access (DMA)

TMS 320C31 จะมี DMA controller ภายใน ซึ่งสามารถอ่านหรือเขียนลงไปในตำแหน่งใดๆ ก็ได้ บนหน่วยความจำโดยไม่ไปขัดการทำงานของ CPU ดังนั้น TMS 320C31 จึงสามารถติดต่อกับหน่วยความจำภายนอกที่มีความเร็วช้าได้ และสามารถติดต่อกับอุปกรณ์ภายนอก โดยไม่ไปรบกวนการทำงานของ CPU

DMA controller จะมี address generator , source register , destination register และ transfer counter การที่ DMA มีบัสข้อมูลและบัสแอดเดรสเป็นของตัวเอง จะทำให้การทำงานจะไม่ไปรบกวนการทำงานของ CPU การทำงานของ DMA จะทำการย้ายข้อมูลเป็นบล็อกหรือเพียง 1 word ไปลงหน่วยความจำหรือนำออกมาจากหน่วยความจำ รูป 2.7 แสดง DMA controller



รูปที่ 2.8 DMA controller

2.7 คำสั่งของไอซีเบอร์ TMS 320C31

คำสั่งของไอซีเบอร์ TMS 320C31 เป็นคำสั่งภาษาแอสแซมบลี้ ซึ่งเป็นคำสั่งเฉพาะของ ไอซีตระกูล TMS 320C3x ซึ่งประกอบไปด้วยคำสั่งแบบธรรมดากับคำสั่งแบบขนาน ที่จะทำให้มีการประมวลผลที่เร็วขึ้น ซึ่งคำสั่งต่างๆจะแสดงดังตารางที่ 2.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mnemonic	Description	Operation
ABSF	Absolute value of a floating-point number	|*src*| → Rn
ABSI	Absolute value of an integer	|*src*| → Dreg
ADDC	Add integers with carry	*src* + Dreg + C → Dreg
ADDC3	Add integers with carry (3 operand)	*src*1 + *src*2 + C → Dreg
ADDF	Add floating-point values	*src* + Rn → Rn
ADDF3	Add floating-point values (3 operand)	*src*1 + *src*2 → Rn
ADDI	Add integers	*src* + Dreg → Dreg
ADDI3	Add integers (3 operand)	*src*1 + *src*2 + → Dreg
AND	Bitwise logical AND	Dreg AND *src* → Dreg
AND3	Bitwise logical AND (3 operand)	*src*1 AND *src*2 → Dreg
ANDN	Bitwise logical AND with complement	Dreg AND $\overline{src}$ → Dreg
ANDN3	Bitwise logical ANDN (3 operand)	*src*1 AND $\overline{src2}$ → Dreg
ASH	Arithmetic shift	If count ≥ 0: (Shifted Dreg left by count) → Dreg Else: (Shifted Dreg right by |count|) → Dreg
ASH3	Arithmetic shift (3 operand)	If count ≥ 0: (Shifted *src* left by count) → Dreg Else: (Shifted *src* right by |count|) → Dreg
B*cond*	Branch conditionally (standard)	If *cond* = true: If C*src* is a register, C*src* → PC If C*src* is a value, C*src* + PC → PC Else, PC + 1 → PC
B*condD*	Branch conditionally (delayed)	If *cond* = true: If C*src* is a register, C*src* → PC If C*src* is a value, C*src* + PC + 3 → PC Else, PC + 1 → PC
BR	Branch unconditionally (standard)	Value → PC
BRD	Branch unconditionally (delayed)	Value → PC
CALL	Call subroutine	PC + 1 → TOS Value → PC

Legend:					
	C	carry bit	C*src*	conditional-branch addressing modes
	cond	condition code	count	shift value (general addressing modes)
	Dreg	register address (any register)	PC	program counter
	Rn	register address (R7–R0)	*src*	general addressing modes
	*src*1	three-operand addressing modes	*src*2	three-operand addressing modes

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ห้องสมุดคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Mnemonic	Description	Operation
CALL*cond*	Call subroutine conditionally	If *cond* = true: PC + 1 → TOS If C*src* is a register, C*src* → PC If C*src* is a value, C*src* + PC → PC Else, PC + 1 → PC
CMPF	Compare floating-point values	Set flags on Rn – *src*
CMPF3	Compare floating-point values (3 operand)	Set flags on *src*1 – *src*2
CMPI	Compare integers	Set flags on Dreg – *src*
CMPI3	Compare integers (3 operand)	Set flags on *src*1 – *src*2
DB*cond*	Decrement and branch conditionally (standard)	ARn – 1 → ARn If *cond* = true and ARn ≥ 0: If C*src* is a register, C*src* → PC If C*src* is a value, C*src* + PC + 1 → PC Else, PC + 1 → PC
DB*condD*	Decrement and branch conditionally (delayed)	ARn – 1 → ARn If *cond* = true and ARn ≥ 0: If C*src* is a register, C*src* → PC If C*src* is a value, C*src* + PC + 3 → PC Else, PC + 1 → PC
FIX	Convert floating-point value to integer	Fix (*src*) → Dreg
FLOAT	Convert integer to floating-point value	Float(*src*) → Rn
IACK	Interrupt acknowledge	Dummy read of *src* IACK toggled low, then high
IDLE	Idle until interrupt	PC + 1 → PC Idle until next interrupt
LDE	Load floating-point exponent	*src*(exponent) → Rn(exponent)
LDF	Load floating-point value	*src* → Rn
LDF*cond*	Load floating-point value conditionally	If *cond* = true, *src* → Rn Else, Rn is not changed
LDFI	Load floating-point value, interlocked	Signal interlocked operation *src* → Rn
LDI	Load integer	*src* → Dreg
LDI*cond*	Load integer conditionally	If *cond* = true, *src* → Dreg Else, Dreg is not changed

Legend:

ARn	auxiliary register n (AR7–AR0	Rn	register address (R7 — R0)
C*src*	conditional-branch addressing modes	*src*	general addressing modes
cond	condition code	*src*1	three-operand addressing modes
Dreg	register address (any register)	*src*2	three-operand addressing modes
PC	program counter	TOS	top of stack

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปพ
ก3882ก
2541

C10582

Mnemonic	Description	Operation
LDII	Load integer, interlocked	Signal interlocked operation *src* → Dreg
LDM	Load floating-point mantissa	*src* (mantissa) → Rn (mantissa)
LSH	Logical shift	If count ≥ 0: (Dreg left-shifted by count) → Dreg Else: (Dreg right-shifted by |count|) → Dreg
LSH3	Logical shift (3-operand)	If count ≥ 0: (*src* left-shifted by count) → Dreg Else: (*src* right-shifted by |count|) → Dreg
MPYF	Multiply floating-point values	*src* × Rn → Rn
MPYF3	Multiply floating-point value (3 operand)	*src*1 × *src*2 → Rn
MPYI	Multiply integers	*src* × Dreg → Dreg
MPYI3	Multiply integers (3 operand)	*src*1 × *src*2 → Dreg
NEGB	Negate integer with borrow	0 – *src* – C → Dreg
NEGF	Negate floating-point value	0 – *src* → Rn
NEGI	Negate integer	0 – *src* → Dreg
NOP	No operation	Modify ARn if specified
NORM	Normalize floating-point value	Normalize (*src*) → Rn
NOT	Bitwise logical complement	$\overline{src}$ → Dreg
OR	Bitwise logical OR	Dreg OR *src* → Dreg
OR3	Bitwise logical OR (3 operand)	*src*1 OR *src*2 → Dreg
POP	Pop integer from stack	*SP–– → Dreg
POPF	Pop floating-point value from stack	*SP–– → Rn
PUSH	Push integer on stack	Sreg → *++ SP
PUSHF	Push floating-point value on stack	Rn → *++ SP

Legend:				
	ARn	auxiliary register n (AR7–AR0)	SP	stack pointer
	C	carry bit	Sreg	register address (any register)
	Dreg	register address (any register)	*src*	general addressing modes
	PC	program counter	*src*1	3-operand addressing modes
	Rn	register address (R7–R0)	*src*2	3-operand addressing modes

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mnemonic	Description	Operation
RETI*cond*	Return from interrupt conditionally	If *cond* = true or missing: *SP–– → PC 1 → ST (GIE) Else, continue
RETS*cond*	Return from subroutine conditionally	If *cond* = true or missing: *SP–– → PC Else, continue
RND	Round floating-point value	Round (*src*) → Rn
ROL	Rotate left	Dreg rotated left 1 bit → Dreg
ROLC	Rotate left through carry	Dreg rotated left 1 bit through carry → Dreg
ROR	Rotate right	Dreg rotated right 1 bit → Dreg
RORC	Rotate right through carry	Dreg rotated right 1 bit through carry → Dreg
RPTB	Repeat block of instructions	*src* → RE 1 → ST (RM) Next PC → RS
RPTS	Repeat single instruction	*src* → RC 1 → ST (RM) Next PC → RS Next PC → RE
SIGI	Signal, interlocked	Signal interlocked operation Wait for interlock acknowledge Clear interlock
STF	Store floating-point value	Rn → Daddr
STFI	Store floating-point value, interlocked	Rn → Daddr Signal end of interlocked operation
STI	Store integer	Sreg → Daddr
STII	Store integer, interlocked	Sreg → Daddr Signal end of interlocked operation
SUBB	Subtract integers with borrow	Dreg – *src* – C → Dreg

Legend:					
	C	carry bit	RM	repeat mode bit
	cond	condition code	RS	repeat start register
	Daddr	destination memory address	Rn	register address (R7–R0)
	Dreg	register address (any register)	SP	stack pointer
	GIE	global interrupt enable register	ST	status register
	PC	program counter	Sreg	register address (any register)
	RC	repeat counter register	*src*	general addressing modes
	RE	repeat interrupt register		

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mnemonic	Description	Operation
SUBB3	Subtract integers with borrow (3 operand)	*src*1 – *src*2 – C → Dreg
SUBC	Subtract integers conditionally	If Dreg – *src* ≥ 0: [(Dreg – *src*) << 1] OR 1 → Dreg Else, Dreg << 1 → Dreg
SUBF	Subtract floating-point values	Rn – *src* → Rn
SUBF3	Subtract floating-point values (3 operand)	*src*1 – *src*2 → Rn
SUBI	Subtract integers	Dreg – *src* → Dreg
SUBI3	Subtract integers (3 operand)	*src*1 – *src*2 → Dreg
SUBRB	Subtract reverse integer with borrow	*src* – Dreg – C → Dreg
SUBRF	Subtract reverse floating-point value	*src* – Rn → Rn
SUBRI	Subtract reverse integer	*src* – Dreg → Dreg
SWI	Software interrupt	Perform emulator interrupt sequence
TRAP*cond*	Trap conditionally	If *cond* = true or missing: Next PC → *++ SP Trap vector N → PC 0 → ST (GIE) Else, continue
TSTB	Test bit fields	Dreg AND *src*
TSTB3	Test bit fields (3 operand)	*src*1 AND *src*2
XOR	Bitwise exclusive OR	Dreg XOR *src* → Dreg
XOR3	Bitwise exclusive OR (3 operand)	*src*1 XOR *src*2 → Dreg

Legend:				
	C	carry bit	Rn	register address (R7–R0)
	cond	condition code	SP	stack pointer
	Dreg	register address (any register)	*src*	general addressing modes
	GIE	global interrupt enable register	*src*1	3-operand addressing modes
	N	any trap vector 0–27	*src*2	3-operand addressing modes
	PC	program counter	ST	status register

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mnemonic	Description	Operation
	Parallel Arithmetic With Store Instructions	
ABSF || STF	Absolute value of a floating point	|*src2*| → *dst1* || *src3* → *dst2*
ABSI || STI	Absolute value of an integer	|*src2*| → *dst1* || *src3* → *dst2*
ADDF3 || STF	Add floating point	*src1* + *src2* → *dst1* || *src3* → *dst2*
ADDI3 || STI	Add integer	*src1* + *src2* → *dst1* || *src3* → *dst2*
AND3 || STI	Bitwise logical AND	*src1* AND *src2* → *dst1* || *src3* → *dst2*
ASH3 || STI	Arithmetic shift	If count ≥ 0: *src2* << count → *dst1* || *src3* → *dst2* Else: *src2* >> |count| → *dst1* || *src3* → *dst2*
FIX || STI	Convert floating point to integer	Fix(*src2*) → *dst1* || *src3* → *dst2*
FLOAT || STF	Convert integer to floating point	Float(*src2*) → *dst1* || *src3* → *dst2*
LDF || STF	Load floating point	*src2* → *dst1* || *src3* → *dst2*
LDI || STI	Load integer	*src2* → *dst1* || *src3* → *dst2*
LSH3 || STI	Logical shift	If count ≥ 0: *src2* << count → *dst1* || *src3* → *dst2* Else: *src2* >> |count| → *dst1* || *src3* → *dst2*
MPYF3 || STF	Multiply floating point	*src1* x *src2* → *dst1* || *src3* → *dst2*
MPYI3 || STI	Multiply integer	*src1* x *src2* → *dst1* || *src3* → *dst2*

Legend: count register addr (R7–R0); *dst1* register addr (R7–R0); *dst2* indirect addr (disp = 0, 1, IR0, IR1); *src1* register addr (R7–R0); *src2* indirect addr (disp = 0, 1, IR0, IR1); *src3* register addr (R7–R0)

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Mnemonic	Description	Operation
	Parallel Arithmetic With Store Instructions (Concluded)	
NEGF || STF	Negate floating point	0– *src2* → *dst1* || *src3* → *dst2*
NEGI || STI	Negate integer	0 – *src2* → *dst1* || *src3* → *dst2*
NOT || STI	Complement	$\overline{src1}$ → *dst1* || *src3* → *dst2*
OR3 || STI	Bitwise logical OR	*src1* OR *src2* → *dst1* || *src3* → *dst2*
STF || STF	Store floating point	*src1* → *dst1* || *src3* → *dst2*
STI || STI	Store integer	*src1* → *dst1* || *src3* → *dst2*
SUBF3 || STF	Subtract floating point	*src1* – *src2* → *dst1* || *src3* → *dst2*
SUBI3 || STI	Subtract integer	*src1* – *src2* → *dst1* || *src3* → *dst2*
XOR3 || STI	Bitwise exclusive OR	*src1* XOR *src2* → *dst1* || *src3* → *dst2*
	Parallel Load Instructions	
LDF || LDF	Load floating point	*src2* → *dst1* || *src4* → *dst2*
LDI || LDI	Load integer	*src2* → *dst1* || *src4* → *dst2*
	Parallel Multiply And Add/Subtract Instructions	
MPYF3 || ADDF3	Multiply and add floating point	op1 x op2 → op3 || op4 + op5 → op6
MPYF3 || SUBF3	Multiply and subtract floating point	op1 x op2 → op3 || op4 – op5 → op6
MPYI3 || ADDI3	Multiply and add integer	op1 x op2 → op3 || op4 + op5 → op6
MPYI3 || SUBI3	Multiply and subtract integer	op1 x op2 → op3 || op4 – op5 → op6

Legend:
dst1 register addr (R7–R0)
dst2 indirect addr (disp = 0, 1, IR0, IR1)
op1, op2, op4, and op5 Any two of these operands must be specified using register addr; the remaining two must be specified using indirect.
op3 register addr (R0 or R1)
op6 register addr (R2 or R3)
src1 register addr (R7–R0)
src2 indirect addr (disp = 0, 1, IR0, IR1)
src3 register addr (R7–R0)

ตารางที่ 2.3 แสดงคำสั่งของไอซี TMS320C31 (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การปรับปรุงภาพ (Image Enhancement)

จุดประสงค์หลักของเทคนิคการปรับปรุงภาพคือ กระบวนการที่ทำให้ภาพมีความเหมาะสมกับ "งานเฉพาะ" ที่ใช้คำว่า "เฉพาะ" นี้สำคัญมากเพราะว่าวิธีการที่ดีสำหรับงานประเภทหนึ่งอาจจะกลับกลายเป็นวิธีการที่ไม่ได้ใช้เลยกับงานอีกประเภทหนึ่ง ยกตัวอย่างเช่น วิธีการที่ดีและมีประโยชน์อย่างมากในการปรับปรุงภาพถ่ายรังสีเอ็กซ์ (X-RAY) กลับไม่เป็นวิธีการที่ดีในการปรับปรุงภาพถ่ายดาวอังคารที่ได้จากยานสำรวจอวกาศ

กระบวนการที่จะกล่าวถึงในที่นี้แบ่งได้เป็น 2 แบบใหญ่ ๆ คือ กระบวนการในสปาเชียลโดเมน (Spatial Domain) กับกระบวนการในโดเมนความถี่ (Frequency Domain) สปาเชียลโดเมน หมายถึงตัวระนาบภาพ และกระบวนการในสปาเชียลโดเมนเป็นกระบวนการที่ทำบนจุดภาพโดยตรง เทคนิคการประมวลผลในโดเมนความถี่ ใช้ความรู้พื้นฐานในการแปลงฟูริเยอร์มาประยุกต์ใช้กับภาพ โดยการปรับปรุงผลการแปลงฟูริเยอร์ของภาพ

3.1 กระบวนการในสปาเชียลโดเมน

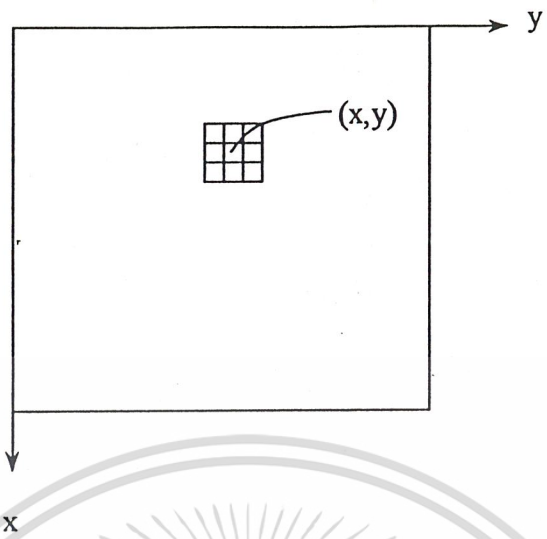
เทอมในสปาเชียลโดเมน หมายถึงการเรียงของจุดภาพ (pixel) รวมกันเป็นภาพและกระบวนการในสปาเชียลโดเมนเป็นกระบวนการที่ทำบนจุดภาพโดยตรงฟังก์ชันการประมวลผลภาพในสปาเชียล โดเมนสามารถแสดงได้ในสมการ 3.1.1

$$g(x,y) = T[f(x,y)] \qquad (3.1.1)$$

เมื่อฟังก์ชัน f(x,y) เป็นฟังก์ชันของภาพอินพุท g(x,y) เป็นฟังก์ชันภาพที่ผ่านการประมวลผล และ T เป็นตัวจัดการ (operator) บน f ที่กำหนดเหนือเนเบอร์ฮูด (neighborhood) ของจุด (x,y) และสามารถใช้ T จัดการบนเซ็ต(set) ของภาพอินพุท อย่างเช่นการรวมจุดภาพต่อจุดภาพ จะใช้สำหรับการลดนอยส์

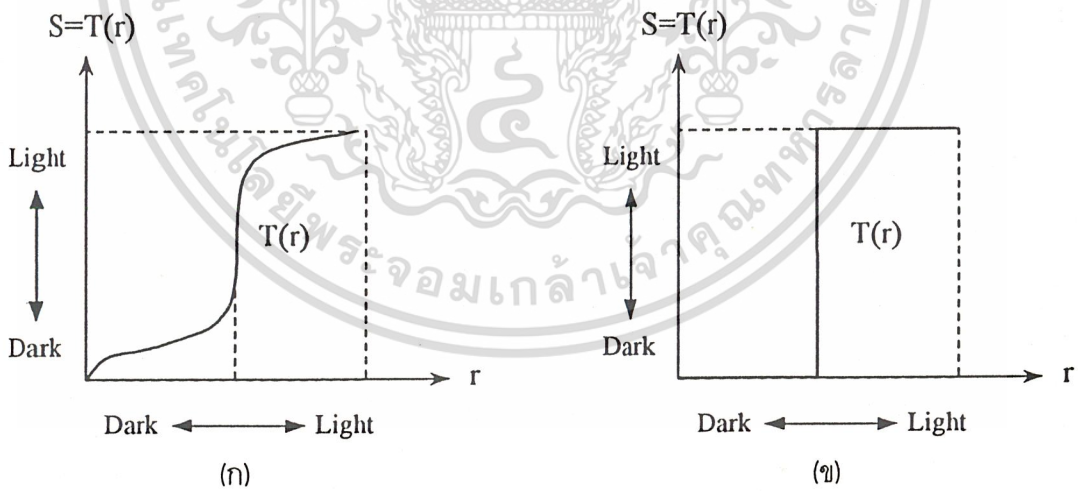
หลักการของการประมวลผล คือ การกำหนดเนเบอร์ฮูด รอบจุด (x,y) ที่มักจะเป็นซับอิมเมจ (subimage) ทรงสี่เหลี่ยมที่มีจุดศูนย์กลางที่ (x,y) ดังแสดงไว้ในรูปที่ 4.1 จุดศูนย์ของซับอิมเมจจะเคลื่อนที่ไปทีละจุด โดยเริ่มที่มุมบนด้านซ้ายของภาพ และกระทำตัวดำเนินการในแต่ละตำแหน่ง (x,y) เพื่อที่จะได้ค่า g ที่ตำแหน่งนั้นๆ แม้ว่ารูปร่างของเนเบอร์ฮูดจะเป็นรูปทรงอื่น ๆ เช่น วงกลม สามารถใช้ได้ในภาพสี่เหลี่ยม แต่อะเรย์ (array) สี่เหลี่ยม และสี่เหลี่ยมจัตุรัสนั้นเป็นตัวที่สำคัญเพราะว่าสามารถสร้างได้โดยง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.1 แสดงเนเบอร์ฮูดขนาด 3x3 ของจุด (x,y)ในภาพ

รูปแบบพื้นฐานของ 'T' คือเมื่อเนเบอร์ฮูดมีขนาด 1 x 1 ในกรณีนี้ g จะขึ้นกับค่า f ที่จุด (x,y) และ'T' จะกลายเป็นการแปลงระดับสีเทาหรือเรียกอีกอย่างหนึ่งว่าการส่ง (mapping) มีฟังก์ชันดังนี้



รูปที่ 3.2 แสดงฟังก์ชันระดับสีเทาสำหรับการปรับปรุงคอนทราส (contrast enhancement)

เมื่อเราให้ s และ r เป็นตัวแปลที่แสดงระดับสีเทาของฟังก์ชัน f (x,y) และ g(x,y) ที่ตำแหน่งต่าง ๆ ตามลำดับ ถ้า T(r) มีรูปแบบดังแสดงในรูปที่ 3.2 (ก) ผลของการแปลงที่เป็นการทำให้ contrast มีค่าสูงขึ้นโดยการทำให้ระดับสีเทาที่มีค่าต่ำกว่า m มีค่าเข้มขึ้น และทำให้ระดับสีเทาที่ค่าสูงกว่า m มีความสว่างขึ้น เทคนิคนี้เรียกว่า การยืดคอนทราส (contrast stretching) ระดับของที่มีค่าต่ำกว่า m จะถูกบีบอัดโดยฟังก์ชันการแปลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในช่วงแคบ ๆ ของ s จากจุดมืดผลตรงกันข้ามจะเกิดกับค่าของ r ที่มากกว่า m ในกรณีของภาพที่ 3.2 (ข) T(r) ทำให้มีสองระดับเป็นภาพสองระดับ (binary image) วิธีการนี้ดูธรรมดาแต่มีประโยชน์ในวิธีการประมวลผล สามารถเขียนเป็นสูตรการแปลงระดับสีเทา เพราะว่าการปรับปรุงที่จุดภาพใด ๆ ในภาพขึ้นอยู่กับระดับสีเทาที่จุดนั้นเท่านั้น วิธีการนี้มักจะถูกเรียกว่า การประมวลผลจุด (point processing)

-1	-1	-1
-1	8	-1
-1	-1	-1

รูปที่ 3.3 แสดงมาสก์ที่ใช้สำหรับการทำภาพให้คมชัดขึ้น

เนเบอร์ฮูดที่ใหญ่ขึ้นจะให้ฟังก์ชันที่การประมวณผลที่หลากหลายขึ้นถ้าไม่คำนึงถึงงานเฉพาะแล้วการที่จะให้ค่าของ f ผ่านในเนเบอร์ฮูดที่กำหนดไว้ของจุด (x,y) จะได้ค่าของ g ที่ (x,y) ในหลักการหนึ่งตามลูกศรนี้มีพื้นฐานการมาสก์(mask) (หรือเรียกอีกอย่างหนึ่งว่า Templates,Window หรือ Filters) โดยพื้นฐานแล้วมาสก์เป็นอะเรย์เล็กๆ อย่างเช่น 3x3 หรืออย่างที่แสดงในรูปที่ 3.1 ซึ่งสัมประสิทธิ์จะถูกเลือกให้ตรงกับความต้องการ อย่างเช่นเราต้องการทำภาพให้คมชัดขึ้น (sharpening) จะสามารถทำได้โดยการใช้การมาสก์ ดังแสดงในรูปที่ 3.3 โดยมีกระบวนการดังต่อไปนี้คือ จุดกึ่งกลางของมาสก์จะเคลื่อนไปทั่วทั้งภาพดังแสดงในรูปที่ 3.1 ในแต่ละจุดภาพของภาพที่อยู่ในพื้นที่ของมาสก์จะถูกคูณด้วยค่าสัมประสิทธิ์ที่ตำแหน่งนั้น ๆ คือจุดภาพที่ตำแหน่งจุด (x,y) จะถูกคูณด้วยค่า 8 ในขณะที่จุดรอบ ๆ 8 จุดจะถูกคูณด้วย -1 ผลลัพธ์จะได้มาจากการรวมผลการคูณทั้ง 9 ครั้งเข้าด้วยกัน ถ้าทุกจุดมีค่าเท่ากัน (เช่นเป็นฉากหลังที่มีค่าคงที่) ผลที่ได้จะเท่ากับศูนย์ในทางกลับกันถ้าจุดศูยน์กลางของมาสก์อยู่ที่จุดที่แตกต่างอยู่ภายในแต่ไม่ใช่ที่จุดศูนย์กลางผลรวมก็จะไม่เท่ากับศูนย์ แต่จะมีค่าต่ำลง ซึ่งจะเป็นการทำให้ขอบภาพมีความชัดเจนขึ้นคมชัดขึ้น

w_1 (x-1,y-1)	w_2 (x-1,y)	w_3 (x-1,y+1)
w_4 (x,y-1)	w_5 (x , y)	w_6 (x,y+1)
w_7 (x+1,y-1)	w_8 (x+1,y)	w_9 (x+1,y+1)

รูปที่ 3.4 แสดงมาสก์ขนาด 3x3 โดยทั่วไป

ดังที่แสดงในรูปที่ 3.4 ถ้าเราให้ w1 ,w2, w3 ....., w9 แสดงค่าสัมประสิทธิ์ของมาสก์ เมื่อพิจารณาจุดรอบ 8 จุดของจุด(x,y) เราจะได้สมการที่แสดงกระบวนการที่ได้กล่าวถึงไปแล้วตามสมการที่ 3.1.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$T[f(x,y)] = w_1 f(x-1,y-1) + w_2 f(x-1,y) + w_3 f(x-1,y+1) + w_4 f(x,y-1) + w_5 f(x,y) + w_6 f(x,y+1) + w_7 f(x+1,y-1) + w_8 f(x+1,y) + w_9 f(x+1,y+1) \quad (3.1.3)$$

ตัวมาสก์ที่ใหญ่กว่านี้ก็ทำได้ในรูปแบบเดียวกัน หมายเหตุในสมการที่ 3.1.3 นั้นถ้าเราเปลี่ยนค่าสัมประสิทธิ์ของมาสก์เป็น wi = 1/9 (I=1,2,3...9) และจะให้ g(x,y) = T[f(x,y)] ค่าของ g (x,y) ที่ได้จะเป็นค่าเฉลี่ยของ ระดับสีเทาของจุด (x,y) กับ 8 จุดรอบมันซึ่งก็คือการทำสภาพให้เบลอ

3.2 กระบวนการในโดเมนความถี่

พื้นฐานของการกระทำการทางโดเมนความถี่คือ ทฤษฎีการประสาน (convolution) ให้ g(x,y) เป็นภาพที่ได้จากการประสานภาพ f(x,y) กับตัวจัดการ ตำแหน่งยืนยงเชิงเส้น (Linear,Position invariant Operator) h(x,y) ที่มีความสัมพันธ์ดังสมการที่ 3.2.1

$$g(x,y) = h(x,y)f(x,y) \quad (3.2.1)$$

จากทฤษฎีการประสานในบทที่แล้วเราจะได้ความสัมพันธ์ในโดแมนความถี่คือ

$$G(u,v) = H(u,v)F(u,v) \quad (3.2.2)$$

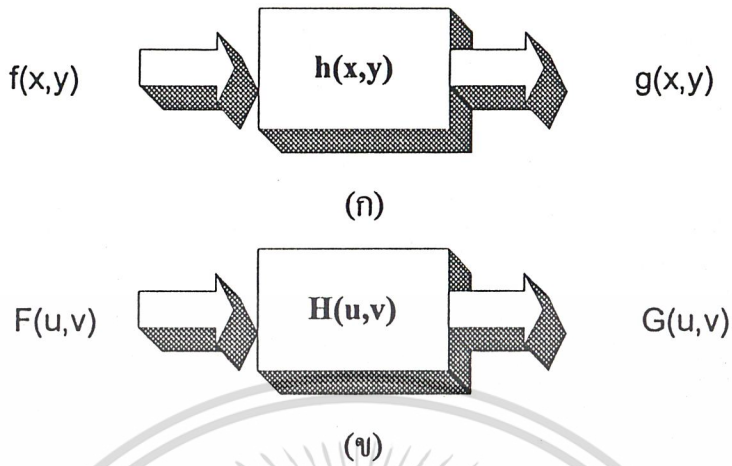
เมื่อ G(u,v) , H(u,v) และ F(u,v) เป็นผลการแปลงฟูริเยอร์ของ g(x,y) , h(x,y) และ f(x,y) ตามลำดับ H(u,v) บางทีก็เรียกว่า ฟังก์ชันถ่ายโอน (Transfer Eunction) ของกระบวนการการปรับปรุงภาพสามารถกระทำได้ในรูปแบบของสมการที่ 3.2.2 โดยทั่วไปแล้วการปรับปรุงภาพ ภาพ f(x,y) จะให้มาแล้วเราจะรู้ที่ต้องการ จะทำโดยการแปลงฟูริเยอร์หา F(u,v) จากการนั้นนำไปคูณกับ H(u,v) ที่เลือกให้เหมาะกับภาพที่ต้องการปรับปรุง แล้วทำการแปลงฟูริเยอร์ผกผันกับผลการคูณ ดังนั้นภาพที่ปรับปรุงจะได้ดังสมการที่ 3.2.3

$$g(x,y) = IFFT[H(u,v)F(u,v)] \quad (3.2.3)$$

การแสดงจุดเด่นบางประการของภาพ f(x,y) อย่างเช่นขอบภาพใน f(x,y) สามารถเน้นได้โดยการเลือกฟังก์ชัน H(u,v) ที่ขยายส่วนความถี่สูงของ F(u,v) ที่จะกล่าวถึงในหัวข้อที่ 3.4

ในรูปที่ 3.5 (ก) h(x,y) เป็นตัวกำหนดคุณสมบัติของระบบ ซึ่งมีฟังก์ชั่นที่จะผลิต g(x,y) จากภาพอินพุท f(x,y) ระบบจะทำการประสาน h(x,y) กับภาพ อินพุท f(x,y) แล้วได้ผลลัพธ์ที่ เอาท์พุทเป็น g(x,y) จากทฤษฎีการประสานเราสามารถ ใช้วิธีอื่นในการที่จะได้มาซึ่งผลอันเดียวกัน โดยการทำในโดเมนความถี่ โดยการคูณ F(u,v) ด้วย H(u,v) จะได้ G(u,v) แล้วทำการแปลงฟูริเยอร์ผกผันที่เอาท์พุทจะได้ภาพที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.5 แสดงการจัดการของระบบเชิงเส้น (ก) เอาท์พุทของระบบเป็นการประสาน h(x,y) กับอินพุท (ข) เอาท์พุทเป็นผลคูณของ H(u,v) กับอินพุท

ถ้าเราไม่รู้ h(x,y) แล้วเราป้อนฟังก์ชันอิมพัลส์ หนึ่งหน่วย (unit impulses function) ให้กับระบบที่เราทราบกันดีว่าผลการแปลงฟูริเยอร์ของอิมพัลส์เท่ากับ 1 ดังนั้นจากสมการที่ 3.2.2 จะได้ G(u,v) = H(u,v) ผลการแปลงฟูริเยอร์ผกผันจะได้ h(x,y) ผลนี้เป็นที่รู้จักกันดีในทฤษฎีระบบเชิงเส้น (liner , position invariant system) ซึ่งแสดงคุณสมบัติของการตอบสนองพัลส์ ถ้าผลการแปลงฟูริเยอร์ของอิมพัลส์ถูกป้อนให้ระบบเอาท์พุทของระบบจะเป็นฟังก์ชันถ่ายโอนของระบบ H(u,v) หรือกล่าวอีกนัยหนึ่งว่าป้อนอิมพัลส์โดยตรงจะได้ h(x,y) ที่เอาท์พุทซึ่ง (x,y) นี้จะเรียกว่า ผลการตอบสนองอิมพัลส์ (impulses response)

ในสมการที่ 3.2.1 อธิบายการทำงานทาง สปาเชียลซึ่งใช้การมาสก์ ที่กล่าวถึงในส่วนก่อนหน้านี้ ในความเป็นจริงแล้วเป็นการประสานเชิงตัวเลข (discrete convolution) จากเหตุผลนี้การสปาเชียลมาสก์ ที่กล่าวถึงในส่วนก่อนหน้านี้มักจะเรียกว่าการประสานมาสก์ แต่อย่างไรก็ตามจุดประสงค์หลักที่จะสนับสนุนนี้ จากทฤษฎีการประสาน ต้องการว่า H(u,v) กับ h(x,y) ต้องมีขนาดเดียวกันยกตัวอย่างเช่น ถ้า H(u,v) มีขนาด 512 x 512 h(x,y) ต้องมีขนาด 512 x 512 เช่นกัน จากสมการ 3.2.1 จะเป็นผลในสปาเชียลที่เท่ากันก็ต่อเมื่อ h(x,y) มีขนาดเท่ากัน การประสานเชิงเลขกับอะเรย์ขนาดใหญ่จะมีประสิทธิภาพมากกว่าถ้าทำในโดเมนความถี่โดยใช้การแปลงฟาสท์ฟูริเยอร์ (Fast Fourier Transform หรือ FFT) เข้าช่วยในตอนหลังเราสามารถพัฒนาวิธีการที่จะใช้การประสานมาสก์เล็กๆ ในสปาเชียลโดเมน ที่ได้จากการประมาณค่าของ H(u,v) แบบกำลังสองผิดพลาดน้อยที่สุด (least-square error) สามารถใช้กับการประมวลผลทางสปาเชียลได้ดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

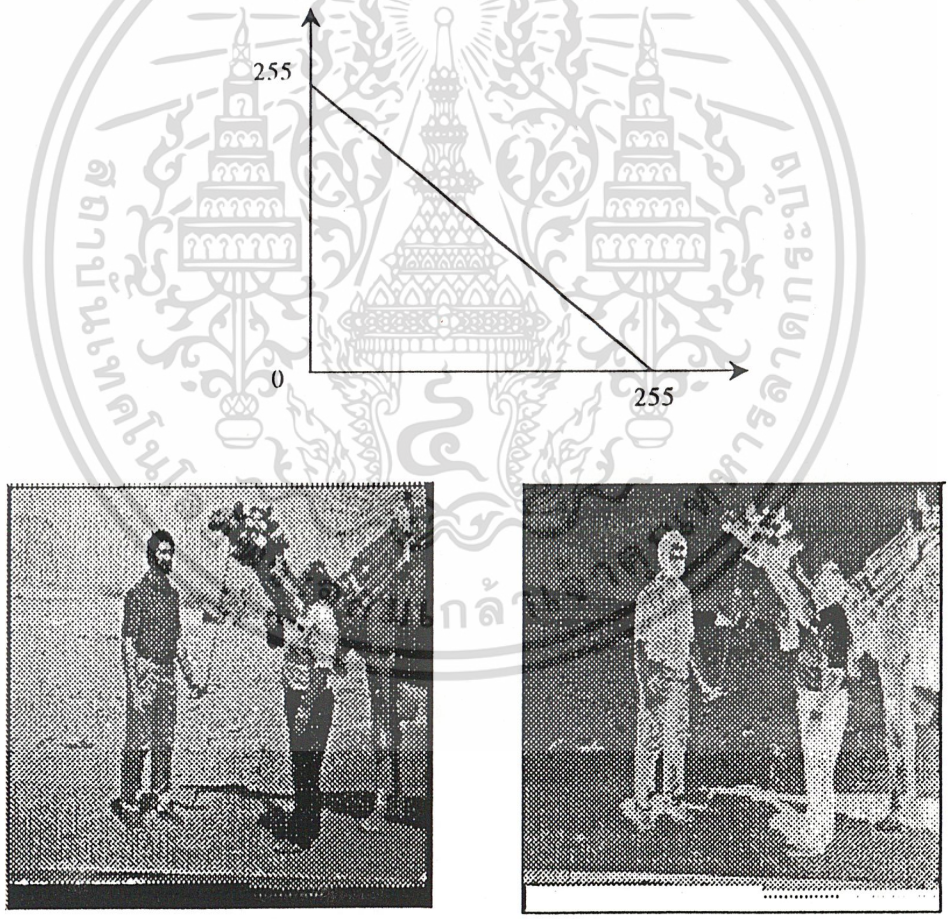
3.3 การปรับปรุงภาพโดยวิธีการประมวลผลจุด

เราจะมาเริ่มศึกษาเทคนิคการปรับปรุงภาพ ที่ใช้ความเข้มของจุดภาพเพียงอย่างเดียวดังที่กล่าวในหัวข้อ 3.1 วิธีการประมวลผลจุดเป็นวิธีการที่พื้นฐานที่สุดในการปรับปรุงภาพเราจะให้ r และ s แทนค่าความเข้มของต้นฉบับ กับภาพภายหลังการประมวลผลตามลำดับ

<u>ตัวอย่างการปรับปรุงความเข้มอย่างง่าย</u>

- ภาพกลับสี (image negative)

การกลับสีมีประโยชน์มากในการแสดงภาพทางการแพทย์ และทางด้านการถ่ายภาพ ภาพกลับสีได้มาจากการใช้ฟังก์ชันการแปลง $s = T(r)$ ดังแสดงในรูปที่ 3.6 ก เมื่อ L เป็นจำนวนระดับสีเทาทำโดยการใช้การกลับลำดับจากสีดำเป็นสีขาวผลการแปลงแสดงในรูปที่ 3.6

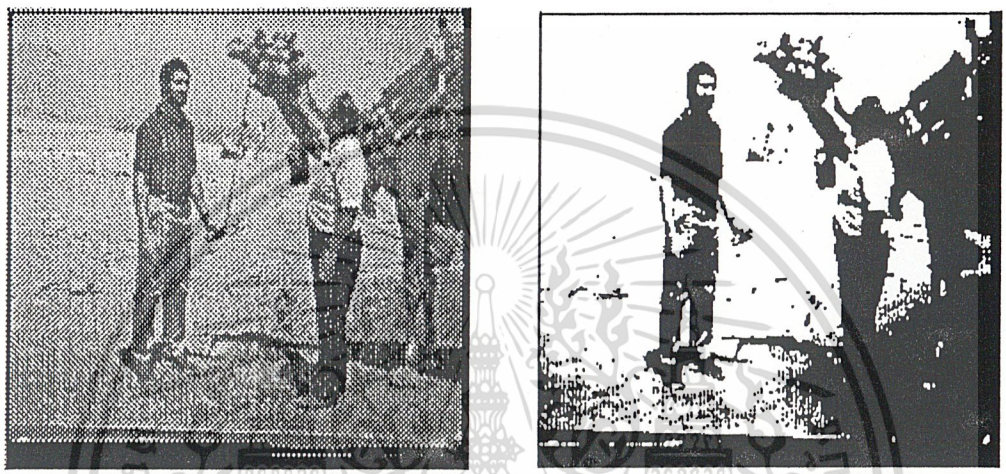


รูปที่ 3.6 แสดงการทำการกลับสีภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เทรสโฮล (thresholding)

การทำเทรสโฮลมีประโยชน์ในการการทำภาพขาวดำ การแปลงใช้ฟังก์ชันการแปลงอย่างง่ายดังรูปที่ 3.7 คือค่าระดับสีเทาของจุดใดที่มีค่าเกิน R (ค่าเทรสโฮล) จะมีค่าเท่ากับ L-1 เมื่อ L เป็นจำนวนระดับสีเทา ค่าที่ระดับสีเทาของจุดภาพใดมีค่าน้อยกว่า R เท่ากับ 0



(ก) (ข)

รูปที่ 3.7 แสดงการทำเทรสโฮล (ก) ภาพต้นฉบับ (ข) ภาพที่ผ่านการทำเทรสโฮล

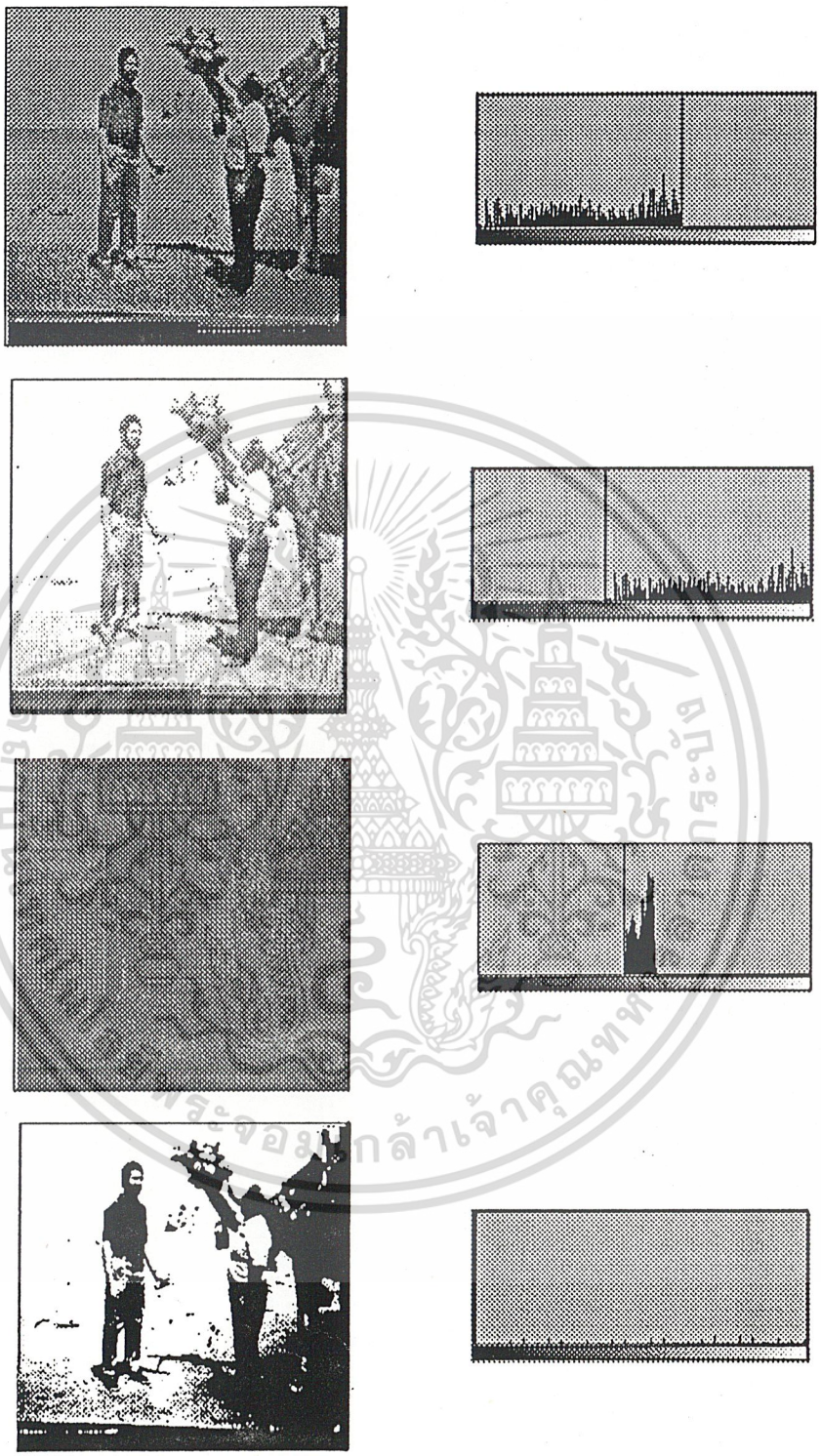
- ฮิสโตแกรม (Histogram)

ฮิสโตแกรมของภาพระดับสีเทา จะอยู่ในช่วง 0 ถึง L-1 เมื่อ L เป็นจำนวนระดับสีเทานิยามไว้ว่าเป็นฟังก์ชันเชิงเลขเต็มหน่วย (discrete function) $P(r_k) = n_k/n$ เมื่อ r_k เป็นระดับสีเทาที่ k n_k คือจำนวนจุดทั้งหมดบนภาพที่มีระดับสีเทา n_k n คือจำนวนจุดทั้งหมดบนภาพ และ k = 0,1,2,...,L-1

ถ้ากล่าวกันอย่างง่ายๆแล้ว $P(r_k)$ จะให้ความน่าจะเป็นของโอกาสที่จะมีระดับสีเทาที่ r_k เมื่อนำฟังก์ชันนี้มาพลอตกราฟตลอดทุกค่า k แล้วเราจะได้กราฟแสดงลักษณะของภาพโดยรวม อย่างเช่นรูปที่ 3.9ข แสดงฮิสโตแกรมของภาพ 4 แบบฮิสโตแกรมของ รูปที่ 3.9 ก แสดงให้เห็นว่ามีจำนวนจุดภาพ ที่มีระดับสีเทาอยู่ในช่วงสีเข้มมาก ซึ่งจะสอดคล้องกับภาพที่มีลักษณะมืด ซึ่งจะตรงข้ามกับรูปที่ 4.9ข

ฮิสโตแกรมที่เสดงในรูปที่3.9ค จะเห็นว่ามีการกระจายที่แคบ แสดงว่าภาพนี้มีไดนามิกเรนจ์ (dynamic range) ที่แคบซึ่งเป็นภาพที่คอนทราสต่ำ (low contrast) ภาพนี้จะมีลักษณะสีทึบๆ เนื่องจากมีระดับสีเทาในช่วงกลาง ๆ มาก ในรูปที่ 3.9 ง แสดงการกระจ่ายกันอย่างทั่วถึงของฮิสโตแกรม ซึ่งจะเป็นภาพที่คอนทราสสูง (high contrast)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 แสดงลักษณะของภาพกับฮิสโตแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 การหาขอบภาพโดยหลักการของ SOBEL

การหาขอบภาพโดยวิธีการของ SOBEL จะเป็นวิธีการพิจารณาหาขอบภาพเป็นส่วนๆไปจนครบตามพื้นที่ที่ต้องการหาขอบภาพ โดยมีหลักการดังนี้ คือ จะใช้หน้าต่าง (window) 3 x 3 เป็นขอบเขตในการหาความแตกต่างของข้อมูลภาพในหน้าต่างนี้ โดยความแตกต่างของข้อมูลภาพหมายถึงจะให้ความสนใจกับความแตกต่างที่เกิดจากการเปลี่ยนจากสีขาวเป็นสีดำ หรือ จากสีดำเป็นสีขาวของข้อมูลภาพ บริเวณนั้นก็จะได้ขอบของข้อมูลภาพออกมา ซึ่งโดยทั่วไปแล้ว การหาขอบภาพโดยวิธีของ SOBEL นี้จะแยกการหาขอบภาพเป็น 2 ทิศทาง คือการหาขอบภาพทางแนวนอน และ การหาขอบภาพทางแนวตั้ง ซึ่งการหาขอบภาพของทั้ง 2 ทิศทางจะมี ค่าประจำตำแหน่งของแต่ละช่องของหน้าต่างที่ใช้ในการพิจารณาดังนี้

การหาขอบภาพทางแนวตั้ง (MASK I)

1	0	-1
2	0	-2
1	0	-1

การหาขอบภาพทางแนวนอน (MASK II)

-1	-2	-1
0	0	0
1	2	1

หลักการของการหาขอบภาพด้วยวิธีการของ SOBEL มีดังนี้คือ

1. ทำการรวมค่าที่อยู่ในขอบเขตของวินโดวส์ โดยสมการที่ใช้ในการรวมค่าเป็นดังนี้

การหาขอบภาพทางแนวตั้ง

$$S(x,y) = \left| -S(x-1,y-1) - 2S(x,y-1) - S(x+1,y-1) + S(x-1,y+1) + 2S(x,y+1) + S(x+1,y+1) \right| \quad (3.4.1)$$

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การหาขอบภาพทางแนวนอน

$$S(x,y) = \left| S(x-1,y-1) + 2S(x-1,y) + S(x-1,y+1) + -S(x+1,y-1) - 2S(x+1,y) - S(x+1,y+1) \right| \quad (3.4.2)$$

เมื่อ x คือ พิกัดทางแนวนอนของข้อมูลรูปภาพ (1 - 512)

y คือ พิกัดทางแนวตั้งของข้อมูลรูปภาพ (1 - 512)

เช่น จากข้อมูลภาพ รูปที่ 3.10 (การหาขอบภาพทางแนวตั้ง) ถ้าต้องการหาผลรวมของข้อมูลตำแหน่งที่ (x=1,y=1) จะได้ดังนี้

$S(1,1) = S(0,0) + 2S(0,1) + S(0,2) - S(2,0) - 2S(2,1) - S(2,2)$

$= 1 + 2(1) + 1 - 1 - 2(1) - 1$

$= 0$

ดังนั้น $S(1,1) = 0$

ดังปรากฏใน รูปที่ 3.11

1	0	-1
2	0	-2
1	0	-1

1	1	1	2	2	2	1	1
1	2	1	200	201	200	2	1
1	1	1	201	200	200	1	2
1	1	2	200	201	201	2	2
2	2	1	201	201	200	1	1
1	1	1	200	200	200	1	1
2	1	1	201	201	200	2	2
1	1	1	2	2	2	1	1

รูปที่ 3.9 แสดงการหาผลรวมของข้อมูลภาพตำแหน่งที่ (x=1,y=1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X	x	x	...
X	0	...	...
X	...	...	...
...	...	...	...

รูปที่ 3.10 แสดงผลรวมของข้อมูลภาพตำแหน่งที่ (x=1,y=1)

หมายเหตุ x หมายถึง ข้อมูลภาพที่ไม่สามารถนำมาประมวลผลได้ (ไม่นำมาใช้ในการพิจารณา)

2. ทำการรวมค่าของข้อมูลภาพดังที่ได้กล่าวมาแล้วข้างต้นจนครบทุกจุดภาพ

โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการรวมค่าของข้อมูลภาพจนครบทุกจุดภาพ จะมีข้อมูลปรากฏดังรูปที่ 3.12

X	x	X	x	x	x	x	x
X	0	597	598	3	598	597	x
X	1	797	797	3	796	794	x
X	3	797	797	4	797	795	x
X	3	796	798	3	798	796	x
X	2	797	798	2	797	795	x
X	1	600	600	2	598	596	x
X	x	X	X	x	x	x	X

รูปที่ 3.11 แสดงผลลัพธ์การรวมค่าของข้อมูลภาพครบทุกจุดภาพ

3. ทำการปรับค่าของข้อมูลที่ได้จากการรวมค่าของข้อมูลภาพ

โดยถ้าข้อมูลที่ได้มีค่าเกิน 255 เราจะได้ค่าของข้อมูลมีค่าสูงสุดเท่ากับ 255 เท่านั้น โดยจากข้อมูลภาพที่ยกตัวอย่างมาข้างต้น เมื่อทำการปรับค่าของข้อมูลภาพจนครบทุกจุดภาพ จะมีข้อมูลปรากฏดังรูปที่ 3.13

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

x	x	x	X	x	x	x	x
x	0	255	255	3	255	255	x
x	1	255	255	3	255	255	x
x	3	255	255	4	255	255	x
x	3	255	255	3	255	255	x
x	2	255	255	2	255	255	x
x	1	255	255	2	255	255	x
x	x	x	x	x	x	x	x

รูปที่ 3.12 แสดงผลลัพธ์การปรับค่าของข้อมูลภาพ

4. ตัวอย่างของข้อมูลภาพขนาด 512x512 pixels ที่ใช้จริงในการหาขอบภาพ โดยวิธีการของ SOBEL



(ก) แสดงรูปต้นแบบ (ข) แสดงผลลัพธ์ที่ได้

รูปที่ 3.13 แสดงการหาขอบภาพด้วยวิธีการของ SOBEL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 หลักการฟิลเตอร์ภาพโดยการกรองความถี่ต่ำ (Low-Pass Filter)

ผลที่เกิดขึ้นจากการกรอความถี่ต่ำนั้นจะทำการปรับปรุงสีของภาพให้มีความนุ่มนวลขึ้น และเรียบขึ้นเพื่อลดส่วนผิดพลาดที่มีอยู่ในภาพวิธีนี้จะเป็นประโยชน์อย่างมากที่จะใช้กำจัดจุดที่มีค่าระดับสีแตกต่างจากจุดรอบข้างมาก ๆ เนื่องจากจุดเหล่านี้ จะทำให้เกิดความผิดพลาด ในขบวนการอิมเมจเซกเตชั่นอย่างยิ่งทั้งวิธี Edge Detection และ Region Clustering เนื่องจากขบวนการ Edge Detection นั้นการที่จุดดังกล่าวนี้มีความแตกต่างกับจุดรอบข้างมาก ๆ ทำให้มันถูกมองว่าเป็นส่วนหนึ่งของขอบภาพ และในวิธี Region Clustering มันจะมีค่าแตกต่างจาก Region ที่อยู่รอบ ๆ มากมันจึงไม่ถูกรวมเข้าไปใน Region ใด ๆ ทำให้ Region ซึ่งมีความแตกต่างกันน้อยจำเป็นต้องถูกรวมเข้าเป็น Region เดียวกันเพื่อให้เหลือจำนวน Region เท่าที่ต้องการ

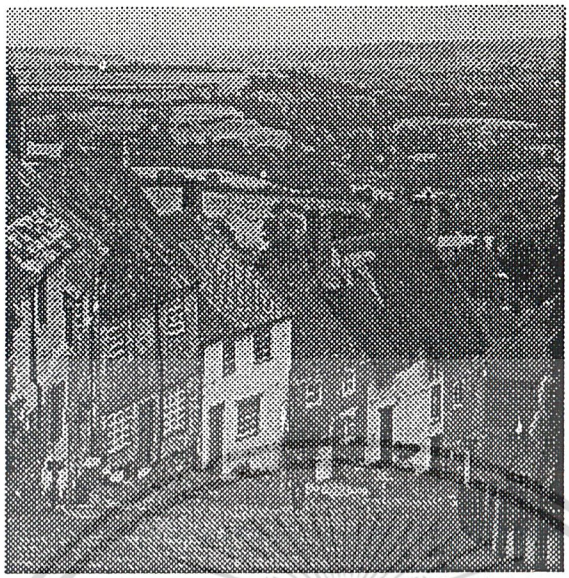
สำหรับการฟิลเตอร์ภาพโดยการกรองความถี่ต่ำนี้หรืออาจจะเรียกว่า Image Smoothing จะมีหน้าที่ในการกรองข้อมูลภาพในส่วนที่มีการเปลี่ยนของระดับสีสูง (high frequency) ให้มีการเปลี่ยนแปลงน้อยลง และคงสภาพส่วนที่มีการเปลี่ยนแปลงต่ำให้คงที่อยู่ ซึ่งวิธีการทำงานของ low-pass นี้จะทำโดยการเปลี่ยนค่าของจุดที่พิจารณาให้เป็นค่าเฉลี่ยระหว่างตัวมันเองกับจุดรอบ ๆ ข้างซึ่งรายละเอียดจะได้กล่าวต่อไป

ในการกรองความถี่ต่ำนั้นจะมีมาสก์ (mask) ขนาด 3x3 เป็นตัวทำการประมวลผลภาพ ซึ่งมาสก์แบบ Low-Pass นี้มีสัมประสิทธิ์เป็นตัวเลขที่มีค่าทุกตัวเท่ากันหมดคือ 1/9 ซึ่งหน้าตาของมาสก์จะแสดงดังรูป

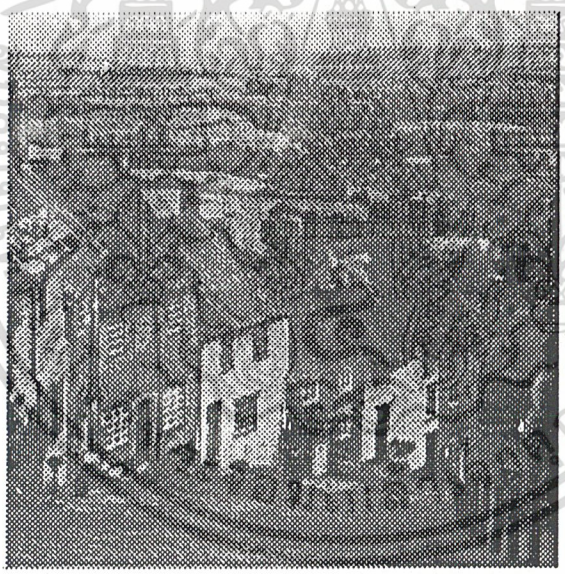
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

เมื่อมาสก์แบบ Low-Pass เคลื่อนที่ผ่านจุดภาพใด ๆ มันจะทำการเฉลี่ยลดความแตกต่างของระดับสีของจุดภาพนั้น ๆ กับจุดภาพข้างเคียง เรียกมาสก์แบบ Low-Pass นี้ว่าเป็น บล็อกฟิลเตอร์ (box filter) โดยที่ผลรวมของสัมประสิทธิ์ทั้งหมดในมาสก์มีค่าเท่ากับ 1 (9 x 1/9 = 1) และมาสก์แบบ Low-Pass จะมีค่าสัมประสิทธิ์ทั้งหมดเป็นบวก ซึ่งผลที่ได้ของภาพเมื่อผ่านการกรองแบบ Low-Pass แล้วนั้นจะแสดงดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.14 แสดงภาพต้นแบบก่อนการกรองแบบ Low-Pass



รูปที่ 3.15 แสดงภาพต้นแบบหลังการกรองแบบ Low-Pass

จากที่กล่าวมาแล้วข้างต้นว่าการกรองความถี่ต่ำจะสนใจเฉพาะส่วนที่มีการเปลี่ยนแปลงระดับสีของจุดภาพสูงและจะไม่สนใจส่วนที่มีการเปลี่ยนแปลงระดับสีของจุดภาพต่ำ เพื่อให้เข้าใจหลักการทำงานของการกรองความต่ำดีขึ้นจะขออธิบายควบคู่ไปกับรูปที่ 3.16 และ 3.17

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

164	164	164	164	164
164	164	164	164	164
164	164	164	164	164
164	164	164	164	164
164	164	164	164	164

รูปที่ 3.16 แสดงส่วนของจุดภาพภาพที่มีการเปลี่ยนแปลงของระดับสีต่ำ
ที่ถูกมาสก์ด้วยการกรองแบบ low-pass

164	164	164	164	164
164	164	164	164	164
164	164	164	164	164
164	164	164	164	164
164	164	164	164	164

รูปที่ 3.17 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีต่ำที่ผ่านการกรองแบบ low-pass

จากรูปจะพบว่าเมื่อมาสก์แบบกรองความถี่ต่ำผ่านไปยังส่วนใด ๆ ของจุดภาพที่ไม่มีการเปลี่ยนแปลงระดับสีของจุดภาพเลย(เทียบระดับสีกับจุดภาพข้างเคียง) ภาพที่ได้จะไม่เกิดผลใด ๆ เลย หรือไม่มีการเปลี่ยนแปลงนั่นเอง เราเรียกส่วนของจุดภาพที่ไม่มีการเปลี่ยนแปลงนี้ว่ามีความถี่เท่ากับศูนย์ ซึ่งหากพบว่าส่วนใดมีความถี่ของการเปลี่ยนแปลงเท่ากับศูนย์ ส่วนๆ นั้นเมื่อผ่านมาสก์การกรองแบบความถี่ต่ำแล้วจะไม่เกิดการเปลี่ยนแปลง ดังนั้น ส่วนของจุดภาพอินพุทจึงเหมือนกับส่วนของเอาต์พุทแต่ถ้าเมื่อใดก็ตามที่ส่วนของจุดภาพนั้นๆ มีความถี่ในการเปลี่ยนแปลงระดับสีสูงผลที่ได้คือค่าของระดับความเข้มจะเกิดการเฉลี่ยกันระหว่างที่จุดของมันเองกับที่จุดรอบ ๆ ดังแสดงให้เห็นดังรูป

42	231	42	231	42
231	42	231	42	231
42	231	42	231	42
231	42	231	42	231
42	231	42	231	42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.18 แสดงส่วนของจุดภาพภาพที่มีการเปลี่ยนแปลงของระดับสีสูงที่ถูกมาสก์ด้วยการกรองแบบ low-pass

126	147	126	147	126
147	126	147	126	147
126	147	126	147	126
147	126	147	126	147
126	147	126	147	126

รูปที่ 3.19 แสดงส่วนของจุดภาพที่มีการเปลี่ยนแปลงของระดับสีสูงที่ผ่านการกรองแบบ low-pass

จากรูปจะพบว่าส่วนของจุดภาพในรูปที่มีความแตกต่างของระดับสีค่อนข้างมากหรือมีความถี่ในการเปลี่ยนแปลงสูง ดังนั้นมาสก์แบบกรองความต่ำจะทำการเปรียบเทียบค่าความเข้มของจุดกึ่งกลางกับจุดต่าง ๆ ที่อยู่รอบ ๆ ซึ่งค่าของจุดกึ่งกลางที่ได้ใหม่นี้เกิดจากการนำค่าทุกค่าที่ถูกมาสก์แบบกรองความถี่ต่ำทับอยู่ โดยจะนำค่าสัมประสิทธิ์ทุกค่าที่อยู่ในมาสก์คูณกับค่าของความเข้มของจุดภาพที่จุดนั้น ๆ แล้วนำค่าที่ได้จากการคูณทั้งหมด นั้นมาบวกกันและผลลัพธ์ที่ได้ก็จะเก็บเป็นค่าใหม่ของจุดภาพ ดังนั้นจะพบว่าค่าของจุดต่าง ๆ ที่ได้จะขึ้นอยู่กับค่าของจุดรอบข้างด้วย และผลที่ได้จะพบว่าค่าความเข้มของจุดที่มีความแตกต่างกันมากจะลดความแตกต่างนั้นลงให้มีค่าเกือบใกล้เคียงกันทั้งนี้ขึ้นอยู่กับค่าที่แตกต่างกันของจุดภาพด้วย คือว่าถ้าค่าของจุดภาพแตกต่างกันน้อยเมื่อผ่านการกรองความถี่ต่ำค่าที่ได้เกือบจะเท่ากันแต่ถ้าค่าแตกต่างกันมากค่าที่ได้ก็จะลดความใกล้เคียงลงไปตามค่าที่ต่างกันของจุดภาพ จึงทำให้รูปภาพที่ได้จากการกรองแบบความถี่ต่ำลดความคมชัดเจนของภาพลงไป รายละเอียดบางอย่างของภาพอาจจะหายไปจึงทำให้ภาพที่ได้นั้นออกเบลอ ๆ (blur) ไม่ค่อยชัดเหมือนภาพต้นฉบับ

3.6 หลักการฟิลเตอร์ภาพโดยการกรองความถี่สูง (High-Pass Filter)

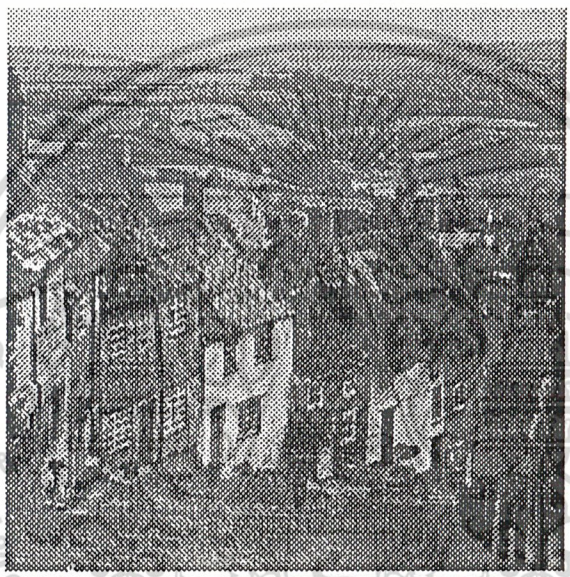
ผลที่เกิดขึ้นของภาพที่ผ่านการกรองแบบความถี่สูงจะตรงข้ามกับผลที่เกิดในการกรองแบบความถี่ต่ำ กล่าวคือ ภาพที่ได้จะถูกเน้นให้มีความชัดเจนมากยิ่งขึ้นในส่วนที่มีการเปลี่ยนแปลงระดับสีสูงและในส่วนที่มีการเปลี่ยนแปลงระดับสีต่ำจะถูกลดทอนหายไป

ในการกรองแบบความถี่สูงนั้นจะมีมาสก์ (mask) ขนาด 3x3 เป็นตัวทำการประมวลผลภาพ ซึ่งมาสก์แบบ High-Pass นี้มีค่าสัมประสิทธิ์ที่ตรงส่วนกึ่งกลางของมาสก์เท่ากับ 9 และส่วนค่าที่อยู่รอบ ๆ จุดกึ่งกลางจะมีค่าเท่ากันหมดคือมีค่าเท่ากับ -1 ซึ่งผลรวมของสัมประสิทธิ์ทุกค่าในมาสก์ก็จะมีค่าเท่ากับ 1 เหมือนกับแบบ Low-Pass ซึ่งหน้าตาของมาสก์จะแสดงดังรูป

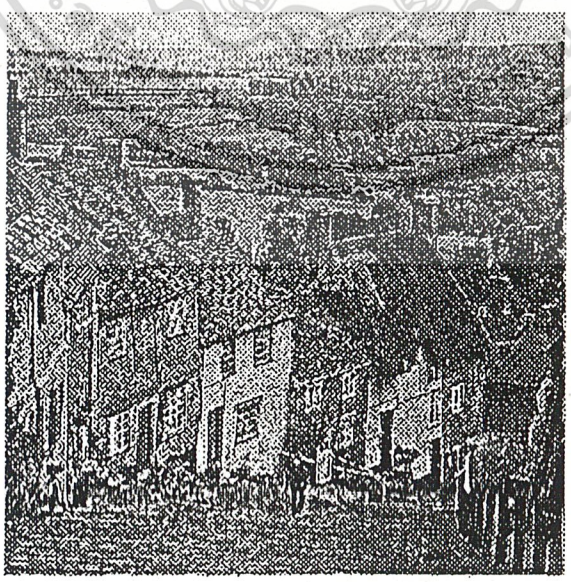
-1	-1	-1
-1	9	-1
-1	-1	-1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากมาสก์แบบกรองความถี่สูงพบว่าตรงกึ่งกลางของมาสก์มีค่าท่ากับ 9 ซึ่งเป็นค่าสัมประสิทธิ์ที่มีค่ามาก เมื่อเปรียบเทียบกับค่าสัมประสิทธิ์ที่อยู่รอบ ๆ จึงทำให้เมื่อมาสก์แบบกรองความถี่สูงเคลื่อนที่ผ่านส่วนของจุดภาพใด ๆ ที่มีความถี่ของการเปลี่ยนแปลงระดับสีสูงมันจะทำการยกระดับค่าของจุด ๆ นั้นให้มีความแตกต่างสูงขึ้นทำให้ภาพที่ได้สว่างขึ้นกล่าวคือ จุดที่สว่างอยู่แล้วก็จะสว่างขึ้นไปอีกส่วนจุดที่มืดอยู่หากมีรายละเอียดซ่อนอยู่ก็จะเห็นได้ชัดเจนขึ้น สว่างขึ้น ทำนองเดียวกันถ้าจุดที่มาสก์แบบกรองความถี่สูงผ่านนั้น ผ่านส่วนของภาพที่มีความถี่ของการเปลี่ยนแปลงระดับสีต่ำก็จะถูกทำให้ลดทอนหายไป ซึ่งผลที่ได้ของภาพเมื่อผ่านการกรองแบบ High-Pass แล้วนั้นจะแสดงดังรูป



รูปที่ 3.20 แสดงภาพต้นแบบก่อนการกรองแบบ High-Pass



รูปที่ 3.21 แสดงภาพต้นแบบหลังการกรองแบบ High-Pass

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สิ่งที่น่าสนใจคือว่าการกรองแบบ High-Pass นั้นมันสามารถทำให้ภาพที่เบลอ (blur) ภาพที่ขาดความคมชัด ภาพที่มีรายละเอียดบางอย่างขาดหายไปให้กลับคืนมาได้ ให้ภาพมีความคมชัดมากขึ้น
และทำให้ภาพที่ได้ดูเด่นขึ้น

ขบวนการฟิลเตอร์แบบ High-Pass นั้นเป็นขบวนการวิเคราะห์ความแตกต่างของกลุ่มข้อมูลรอบๆ กับจุดที่พิจารณา โดยฟิลเตอร์นี้ จะทำหน้าที่กรองข้อมูลในส่วนที่มีการเปลี่ยนแปลงของระดับสีต่ำ (low frequency) ให้ถูกลดทอนหายไป และคงสภาพส่วนที่มีการเปลี่ยนสูงให้คงอยู่ โดยใช้วิธี high-pass filter นี้จะใช้วิธีการปรับค่าจุดที่อยู่ตรงกลางให้มีค่าเท่ากับค่าผลรวมของผลต่างระหว่างตัวมันกับจุดรอบ ๆ

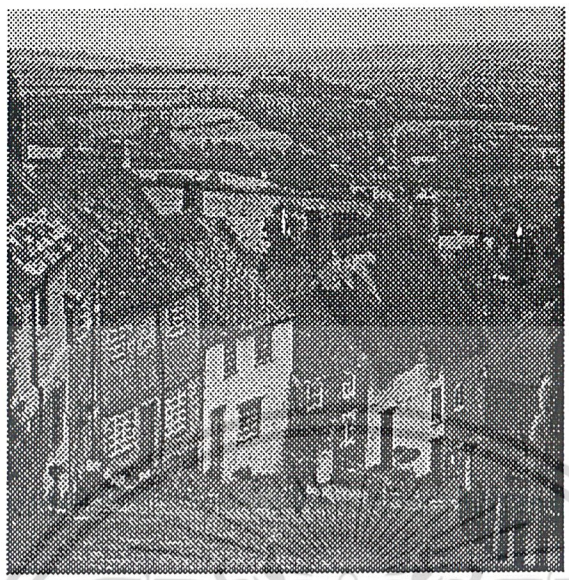
3.7 การหาขอบภาพโดยใช้หลักการของลาปาเซียล (Laplacian)

การหาขอบภาพโดยใช้หลักการของลาปาเซียลนี้จะทำให้ส่วนที่เป็นขอบภาพนั้นสว่างขึ้นโดยมาสก์ของวิธีลาปาเซียลนี้จะมีจะมีสัมประสิทธิ์ที่เป็นตัวเลขต่าง ๆ ดังรูป

-1	-1	-1
-1	8	-1
-1	-1	-1

วิธีนี้จะทำให้ขอบภาพที่ได้นั้นคมชัดขึ้น สว่างขึ้นและมันจะแบ่งชั้นกับส่วนที่ไม่ใช่ขอบภาพ โดยมันจะทำที่ทุก ๆ ส่วนของจุดภาพที่มีความแตกต่างของระดับสีภาพอย่างชัดเจน ซึ่งผลที่ได้จากการหาขอบภาพโดยวิธีลาปาเซียลนี้จะแสดงดังรูปที่ 3.22

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.22 แสดงภาพต้นแบบก่อนการหาขอบภาพโดยใช้วิธี Laplacian



รูปที่ 3.23 แสดงภาพต้นแบบหลังการ หาขอบภาพโดยใช้วิธี Laplacian

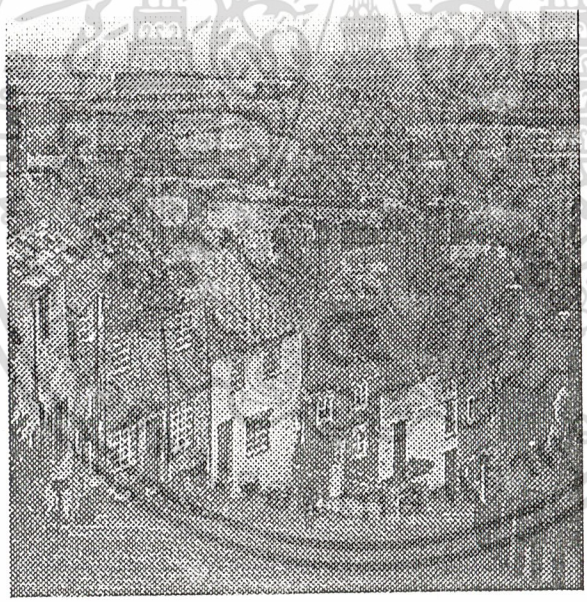
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8 การหาขอบภาพโดยใช้หลักการ Edge Enhancement

การหาขอบภาพโดยใช้หลักการ Edge Enhancement นี้จะทำให้ส่วนที่เป็นขอบภาพนั้นคมชัดขึ้น โดยมาสก์ของ Edge Enhancement จะมีจะมีสัมประสิทธ์ที่เป็นตัวเลขต่าง ๆ ดังรูป

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

ซึ่งผลที่ได้จากการหาขอบภาพโดยหลักการ Edge Enhancement นี้จะมีความละเอียดมากกว่าแบบอื่นๆ จึงทำให้ภาพที่ออกมานั้นคมชัดกว่าภาพต้นแบบ ดังแสดงในรูปที่ 3.24



รูปที่ 3.24 แสดงภาพที่ได้จากการหาขอบภาพโดยหลักการ Edge Enhancement

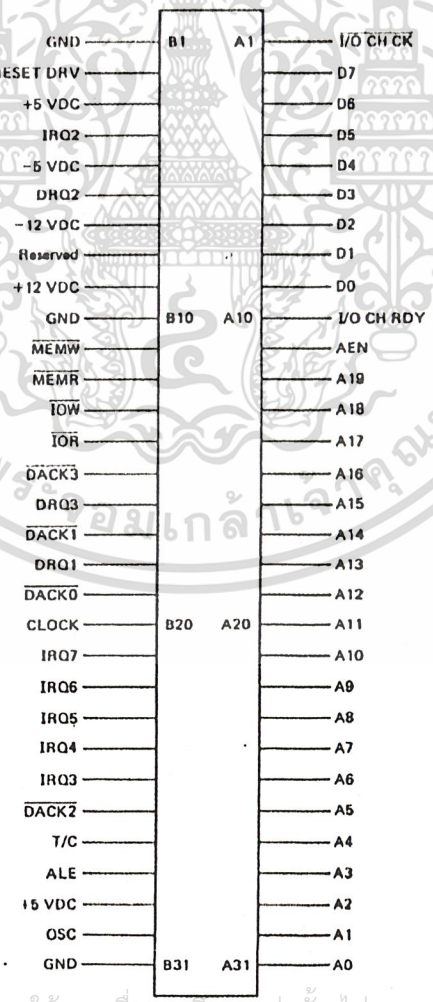
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
IBM PC กับการอินเตอร์เฟส

ในตอนนี้จะขอแนะนำส่วนต่าง ๆ ของ IBM PC ที่เกี่ยวข้องกันการอินเตอร์เฟสซึ่งต้องต่อโดยตรงกับระบบบัสของ IBM PC ให้เข้าใจเสียก่อน

4.1 AT SLOT และ XT SLOT ของ IBM PC

เนื่องจากเครื่องรุ่น AT ใช้ CPU เบอร์ 80286 ซึ่งเป็น CPU ขนาด 16 บิตแท้ คือมีการประมวลผลภายในเป็น 16 บิต ในขณะที่รุ่น XT ใช้ CPU เบอร์ 8088 ซึ่งเป็น 16 บิตเทียม คือมีการประมวลผลภายในเป็นแบบ 16 บิตแต่ดาต้าบัสมีเพียงแค่ 8 บิตเท่านั้น ดังนั้นการจัดการเกี่ยวกับข้อมูลภาพ 16 บิตจึงต้องกระทำ 2 ครั้ง ครั้งละ 8 บิตจะเห็นได้ว่าระบบบัสของ XT เป็นซับเซตของระบบบัสของ AT



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.1 แสดงตำแหน่งของสัญญาณบนสล็อต IBM

บนเมนบอร์ดของ AT จะมีสล็อตอยู่ 2 ชนิด คือ สล็อตสั้นและสล็อตยาว ซึ่งสล็อตยาวจะมีจำนวนขาสัญญาณและตำแหน่งของขาสัญญาณบนสล็อต เหมือนกับ XT ส่วนขาสัญญาณที่อยู่บนสล็อตสั้นที่เพิ่มขึ้น จะเป็นสัญญาณที่มีแต่เฉพาะบน AT เท่านั้นประกอบด้วยข้อมูลครึ่งบน 8 บิตแอดเดรสที่เพิ่มขึ้นมาอีก 4 บิต และขาสัญญาณควบคุมที่เพิ่มขึ้นจาก XT สำหรับรูปของสล็อตและตำแหน่งของสัญญาณต่างๆ แสดงไว้ในรูปที่ 4.1

สำหรับเครื่องคอมพิวเตอร์ที่ใช้ CPU สูงกว่า 80286 นั้น เช่น 80386 ที่เป็น CPU ขนาด 32 บิต ก็ยังคงใช้สล็อตแบบ AT อยู่ ซึ่งหลักการทำงานก็ยังคงเหมือนรุ่นก่อนๆ เช่นกัน

4.2 รายละเอียดของสัญญาณต่าง ๆ บนสล็อต

(I) , (O) และ (I/O) หมายถึง ทิศทางของขาสัญญาณเมื่อเทียบกับเมนบอร์ดโดยที่

(I) หมายถึง ขาสัญญาณอินพุท

(O) หมายถึง ขาสัญญาณเอาต์พุท

(I/O) หมายถึง ขาสัญญาณอินพุทขาสัญญาณที่เป็นได้ทั้งอินพุทและเอาต์พุท

(*I/O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณเอาต์พุทแต่จะเป็นอินพุทในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้า จะหมายถึงขาสัญญาณที่แอคตีฟที่ ลอจิก "0"และขาสัญญาณที่ไม่มี หรือมีเครื่องหมายบวกนำหน้าอยู่จะหมายถึง ขาสัญญาณลอจิกที่แอคตีฟ "1"สัญญาณที่ต่ออยู่บนสล็อตนี้สามารถขับ ไอซี TTL ชนิด low power ได้ สองตัว โดยที่ไม่ทำให้เกิดการโหลดหรือการเพี้ยนของสัญญาณ ขาสัญญาณต่าง ๆ บนสล็อตของ XT และ AT สามารถแบ่งออกเป็นกลุ่ม ๆ ได้ดังนี้

4.2.1 เพาเวอร์ซัพพลาย

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

4.2.2 แอดเดรสบัส และ สัญญาณต่าง ๆ ที่เกี่ยวข้อง

SA0-SA19 (*I/O)	เป็นแอดเดรสบิตที่ 0 ถึง 19 โดยที่ SAO มีนัยสำคัญต่ำสุด ขาสัญญาณนี้จะแอคตีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็น "1" และจะถูกแลตช์ไว้ตอนขอบขาลงของขาสัญญาณ BALE แอดเดรสทั้ง 20 บิตนี้สามารถอ้างหน่วยความจำได้ถึง 1 เมกกะไบต์ XT สำหรับ AT เมื่อใช้ร่วมกับ LA17-LA23 จะอ้างได้ถึง 16 เมกกะไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

LA17-LA23 (*I/O)	(เฉพาะรุ่น AT) ขาสัญญาณนี้จะแอคตีฟ เมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก "1" แต่จะไม่มีการแลตช์ไว้ ตอนขอบขาลงของขาสัญญาณ BALE ดังนั้นถ้าอุปกรณ์ I/O ไม่มีการอ้างแอดเดรสเกิน 1 เมกกะไบต์ ขาสัญญาณนี้ก็ไม่จำเป็นต้องใช้ แต่ถ้ามีการอ้างแอดเดรสเกินอุปกรณ์ I/O จะต้องทำการแลตช์สัญญาณนี้ไว้ โดยใช้ขอบขาล่างของสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR
AEN (0)	(Addres (Address Enable) ขาสัญญาณนี้จะแอคตีฟ เมื่อตัวควบคุม DMA ได้ทำการควบคุมบัสต่าง ๆ ของระบบแล้ว ดังนั้นการอ้างพอร์ตของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ในการดีโค้ดด้วย เพื่อที่จะไม่ทำให้เกิดการติดต่อระหว่างระบบกับอุปกรณ์ I/O ตัวอื่นยกเว้นตัวที่กำลังทำขบวนการ DMA อยู่
BALE (0)	(Address Latch Enable) ขาสัญญาณนี้ใช้ในการแสดงเริ่มต้นของกระบวนการต่าง ๆ ที่มีการติดต่อกับหน่วยความจำ โดยจะแอคตีฟเมื่อค่าแอดเดรสที่ CPU ต้องการติดต่อด้วย อยู่บนแอดเดรสบัสเรียบร้อยแล้ว ตามปกติขอบขาลงของสัญญาณนี้ จะทำให้เกิดการแลตช์สัญญาณ SA0-SA19 และถ้ามีการอ้างแอดเดรสเกิน 1 เมกกะไบต์ (AT) จะใช้ขอบขาลงของสัญญาณนี้ในการแลตช์สัญญาณ LA17-LA23 ด้วยเช่นกัน แต่สำหรับในกระบวนการ DMA สัญญาณนี้จะมีสถานะเป็น "1" ตลอด
SBHE (*I/O)	(เฉพาะรุ่น AT) (Bus High Enable) เป็นขาสัญญาณที่ใช้แสดงว่ามีการรับส่งข้อมูลในบิตที่ SD8-SD15 ดาต้าบัส
SD0-SD7 (I/O)	สำหรับรุ่น AT จะมี SD8-SD15 เพิ่มขึ้นมาด้วย คือ ดาต้าบิต 0-7 สำหรับรุ่น XT และสำหรับรุ่น AT คือดาต้าบิต 0-15 โดยที่ SD0 มีนัยสำคัญต่ำสุด สำหรับ AT ถ้ามีการติดต่อกับบิตที่ SD8-SD15 สามารถตรวจสอบได้จากขาสัญญาณ SBHE

4.2.3 <u>สัญญาณอินเตอร์รัพต์</u>

IRQ2-IRQ7 (I)	(Interrupt Request) (สำหรับรุ่น AT จะเป็น IRQ3-7,9-12,14,15) เป็นขาสัญญาณอินเตอร์รัพต์ CPU สำหรับ AT ลำดับความสำคัญของสัญญาณอินเตอร์รัพต์ IRQ เป็นดังนี้คือ 9,10,11,12,14,15,3,4,5,6 และ 7 โดย IRQ9 มีลำดับความสำคัญน้อยที่สุด สำหรับ XT IRQ2 จะมีลำดับความสำคัญมากที่สุดรอง ๆ ลงไปคือ IRQ 3,4,5,6,7 โดยปกติสัญญาณนี้จะมีสถานะเป็น "0" เสมอ เมื่อต้องการอินเตอร์รัพต์ CPU ให้ส่งพัลส์ที่เป็นลอจิก " 1 " ให้กับมันโดยที่ไม่จำเป็นต้องคำนึงถึงคาบเวลาของพัลส์ ทั้งนี้เพราะระบบของ IBM ตัวอินเตอร์รัพต์คอนโทรลเลอร์ (8259 Interrupt Controller) จะถูกโปรมแกรมให้ตรวจสอบสัญญาณอินเตอร์รัพต์ โดยใช้ขอบขาลงของสัญญาณนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-I/O CH CK (I)	(I/O Channel Check) เป็นสัญญาณที่บอกถึงความผิดพลาดในการรับส่งข้อมูลซึ่งตรวจจากพาริตี้บิต ถ้าพาริตี้บิตที่อ่านจากหน่วยความจำกับพาริตี้ที่สร้างขึ้นจากระบวนการรับส่งข้อมูล สัญญาณนี้จะทำให้เกิดการอินเตอร์รัพต์ CPU แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิด Parity Error ขึ้น CPU จะแสดงข้อความบอกความผิดพลาดขึ้นและจะหยุดการทำงาน (Halt) เพื่อให้ผู้ใช้ตรวจสอบหาสาเหตุของการผิดพลาด

4.2.4 สัญญาณที่ใช้ในกระบวนการ DMA

DRQ1-DRQ3 (I)	(DMA Request) (สำหรับรุ่น AT จะเป็น DRQ0-3,5-7) เป็นขาสัญญาณที่ใช้ในการขอทำกระบวนการ DMA โดยที่ DRQ0 มีลำดับความสำคัญมากที่สุด และ DRQ3 มีลำดับความสำคัญน้อยที่สุดสำหรับรุ่น XT และสำหรับรุ่น AT ขา DRQ7 จะมีลำดับความสำคัญน้อยที่สุด
-DACK03 (0)	(DMA Acknowledge) (สำหรับรุ่น AT จะเป็น -DACK0-3,5-7) เป็นสัญญาณตอบสนองของการขอทำ DMA ของอุปกรณ์ I/O เพื่อให้อุปกรณ์ I/O ทราบว่า ก่อนขอทำกระบวนการ DMA นั้นได้รับการตอบสนองแล้ว เช่น ถ้ามีการขอทำ DMA ผ่านทาง DRQ2 และเมื่อ CPU รับรู้แล้วจะทำให้สัญญาณ DACK2 แอคตีฟ
Refrech (0)	(เฉพาะรุ่น AT) (Memory Refaesh) มีหน้าที่เหมือนกับขาสัญญาณ DACKO ในรุ่น XT คือใช้แสดงขบวนการรีเฟรชหน่วยความจำเพราะว่าในรุ่น AT จะมีวงจรที่ใช้ในการรีเฟรชหน่วยความจำโดยตรงอยู่แล้ว ดังนั้นจึงไม่จำเป็นต้องใช้ขาสัญญาณ DRQ0 และ DACK0
-Master (O)	(เฉพาะรุ่น AT) (Master) ขาสัญญาณนี้จะใช้ร่วมกับ DMA Request ในการเข้าควบคุมระบบบัส ในกระบวนการ DMA โดยที่ตัว DMA คอนโทรลเลอร์จะส่งสัญญาณ DMA Request แล้วรอจนกระทั่งได้รับการตอบสนอง โดยสัญญาณ DACK เกิดการแอคตีฟขึ้นแล้วจึงส่งค่าสัญญาณนี้ ให้กับCPU ทำให้ แอดเดรสบัส ดาต้าบัส และคอนโทรลบัส เข้าสู่สถานะไตรสเตต หรือ ไฮอิมพีเดนซ์ หลังจากนั้นตัว DMA คอนโทรลเลอร์จะต้องรออีกหนึ่งคาบสัญญาณคล็อก ก่อนที่จะเข้าควบคุมบัสต่าง ๆ และจะต้องรออีก 2 ไซเคิล ก่อนที่จะทำการอ่านหรือเขียนข้อมูล ช่วงเวลาที่สัญญาณนี้แอคตีฟไม่ควรเกิน 15 ไมโครวินาทีมิฉะนั้น ข้อมูลในหน่วยความจำจะสูญหายไปได้ เนื่องจากขาดสัญญาณรีเฟรชหน่วยความจำ
T/C (0)	(Terminal count) เป็นขาสัญญาณที่บอกอุปกรณ์ I/O ที่ทำ DMA ให้ทราบว่าจำนวนข้อมูลที่รับส่งในกระบวนการ DMA นี้ครบจำนวนแล้วโดยจะส่งสัญญาณนี้เป็นพัลส์ให้กับอุปกรณ์ I/O

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.5 สัญญาณควบคุมต่างๆ

-MEMR (*I/O)	(Memory Read) [สำหรับรุ่น AT คือ ขาสัญญาณ-SMEMR (System Memory Read)] ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำส่งข้อมูลออกมาที่ดาต้าบัส แต่สำหรับ AT สัญญาณ -SMEMR จะแอคตีฟ เมื่อการอ่านข้อมูลจากหน่วยความจำที่อยู่ภานใน 1 เมกกะไบต์แรกเท่านั้น
MEMR (0)	(เฉพาะรุ่น AT) (Memory Read) ขาสัญญาณนี้ไม่ใช่ขาสัญญาณเดียวสัญญาณ-MEMR ใน XT มันจะมีการแอคตีฟในกระบวนการอ่านข้อมูลที่เกิดขึ้น ไม่ว่าอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์หรือไม่
MEMW (*I/O)	(Memory Write) [สำหรับรุ่น AT คือ ขาสัญญาณ –SMEMW (System Memory Write)] ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำเก็บข้อมูลจากดาต้าบัสแต่สำหรับ AT สัญญาณ –SMEMW จะแอคตีฟ เมื่อการเก็บข้อมูลจากหน่วยความจำที่อยู่ภานใน 1 เมกกะไบต์แรกเท่านั้น
MEMW (0)	(เฉพาะรุ่น AT) (Memory Write) ขาสัญญาณนี้ไม่ใช่ขาสัญญาณเดียวกันกับสัญญาณ-MEMR ใน XT มันจะมีการแอคตีฟในทุก ๆ กระบวนการเก็บข้อมูลที่เกิดขึ้น ไม่ว่าอยู่ในช่วงหน่วยความจำ 1 เมกกะไบต์แรกหรือไม่
-IOR (*I/O)	(I/O Read) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการส่งข้อมูลมาที่ดาต้าบัส
-IOW (*I/O)	(I/O Write) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการเก็บข้อมูลจากดาต้าบัสเข้าไป
RESET DRV (0)	(Reset Driver) เป็นขาสัญญาณที่แอคตีฟตอนช่วงที่เราเริ่มจ่ายไฟให้กับระบบเพื่อใช้ในการรีเซต CPU และ อุปกรณ์ต่าง ๆ ในระบบคอมพิวเตอร์ รวมทั้งอุปกรณ์ I/O ที่ต่ออยู่ด้วย
-MEM CS16 (I)	(เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่าต้องการรับส่งข้อมูลกับหน่วยความจำที่ละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือทำการรับส่งข้อมูลที่ละ 8 บิต สองครั้งทำให้ได้ 16 บิต
-I/O CS16 (I)	(เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่าต้องการรับส่งข้อมูลกับอุปกรณ์ I/O ที่ละ 16 บิต ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือ ทำการรับส่งข้อมูลที่ละ 8 บิต สองครั้งทำให้ได้ 16 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.6 สัญญาณที่ใช้สร้าง Wait States

I/O CH RDY (I)	(I/O Channel Ready) ขาสัญญาณนี้จะทำให้แอคตีฟโดยอุปกรณ์ I/O หรือ หน่วยความจำที่ไม่สามารถทำงานให้ทันกับระบบ ดังนั้น จะต้องทำการหน่วงระบบให้ทำงานช้าลงด้วยการเพิ่ม Wait States โดยการทำให้สัญญาณนี้แอคตีฟในช่วงเวลาที่ I/O ได้รับสัญญาณจากการดีโค้ดแอดเดรสสัญญาณ -MMER, สัญญาณ -MMEW, สัญญาณ -IOR, สัญญาณ -IOW
OWS (I)	(เฉพาะรุ่น AT) (Zero Wait States) การแอคตีฟของขาสัญญาณนี้จะบังคับไม่ให้เกิดการสร้าง Wait States โดยอัตโนมัติ นั้นคือการที่จะเกิด Wait States ขึ้นได้จะต้องขึ้นอยู่กับสัญญาณนี้ เช่น การทำงานในกระบวนการอ่านเขียนข้อมูลขนาด 16 บิต โดยไม่ใช้ Wait States ทำได้โดยการสร้างสัญญาณ OWS จากสัญญาณการดีโค้ดแอดเดรส และสัญญาณที่ใช้ในการอ่านเขียน หรือ การลด Wait States ในกระบวนการอ่านเขียนข้อมูลขนาด 8 บิต ให้เหลือเพียง 2 Wait States ทำได้โดยการให้สัญญาณ OWS แอคตีฟหลังการอ่านหรือเขียนไปแล้ว 1 คล็อก โดยปกติการขับสัญญาณนี้ควรใช้เกจที่มีเอาท์พุทเป็นแบบ Open Collector ที่ทนกระแสได้ 20 mA (Sinking Current)

4.2.7 สัญญาณนาฬิกา

CLK (0)	(System Clock) สำหรับ XT ขานี้จะมีความถี่ประมาณ 4.77 MHz หรือ อาจสูงกว่านี้ก็ได้ สำหรับรุ่นใหม่ ๆ และสำหรับ AT จะมีความถี่ประมาณ 6 MHz หรือในรุ่นใหม่อาจจะมีความถี่สูงถึง 15 MHz โดยปกติ ขาสัญญาณนี้มีดิวตี้ไซเคิล 50% สำหรับ CPU เบอร์ 80286 ตัวกำเนิดสัญญาณนาฬิกาที่ป้อนให้จะมีความถี่เป็น 2 เท่าของความถี่ CPU ทำงาน แต่ ขาสัญญาณนี้ ก็จะยังคงมีความถี่เท่ากับความถี่ที่ CPU ทำงานเสมอ
OSC (0)	(Oscilator) เป็นขาสัญญาณที่มีความถี่สูงคือ 14.31818 MHz ความถี่ของสัญญาณนี้จะคงที่เสมอและจะไม่ซิงโครนัสกับสัญญาณอื่นๆในระบบดังนั้นจึงไม่ควรนำสัญญาณนี้ไปใช้เป็นคล็อกของอุปกรณ์I/O ที่ต่ออยู่กับระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3 การจัดการ Address ของ Memory และ Port

การควบคุมอุปกรณ์ I/O ที่ต่ออยู่กับ IBM PC จะกระทำผ่านพอร์ต โดยการอ้างถึงแอดเดรสของพอร์ตที่อุปกรณ์นั้นต่ออยู่โดยตรง ดังนั้นการที่จะใช้งาน หรือควบคุมอุปกรณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ตใน IBM PC พอร์ต และหน่วยความจำจะแยกจากกันโดยเด็ดขาด ถึงแม้ว่าการอ้างถึงจะใช้สัญญาณจากแอดเดรสบัสเหมือนกันก็ตาม แต่สัญญาณที่ใช้ในการอีเนเบิ้ลในการอ่านและเขียนข้อมูลจะต่างกัน ดังนั้นการติดต่อกับพอร์ตจึงมีคำสั่งแยกต่างหากออกจากคำสั่งที่ใช้ติดต่อกันหน่วยความจำคือ IN และ OUT ด้วยเหตุนี้ การจัดแอดเดรสของหน่วยความจำและแอดเดรสของพอร์ต I/O จึงแยกออกจากกัน ใน IBM PC

การจัดการแอดเดรสของหน่วยความจำแสดงได้ในตารางที่ 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 418K
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	Expanded 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 (8808) แต่ในรุ่นAT (80286) ใช้แอดเดรส 24 เส้น คือ A0-A23 การอ้างแอดเดรสของพอร์ตสำหรับ CPU เบอร์ 8088 และ 80286 สามารถอ้างได้ถึง 64K พอร์ต แต่ใน IBM PC ทั้งในรุ่น AT และ XT ออกแบบให้ใช้แอดเดรสเพียง 10 เส้นเท่านั้น คือ A0-A9 ดังนั้นจำนวนพอร์ตสูงสุดที่สามารถอ้างได้คือ 1024 พอร์ต ในจำนวนทั้งหมดนี้ ยังแบ่งออกเป็น 2 กลุ่ม คือกลุ่มพอร์ตที่มีแอดเดรสอยู่ในช่วง 000H-0FFH จะใช้บนเมนบอร์ดสำหรับชิพซัพพอร์ตเท่านั้น เช่น 8259 (Interrupt Controller) , 8257 (DMA Controller) , 8253 (Time & Counter) และกลุ่มที่มีแอดเดรสอยู่ในช่วง 100H-3FFH จะใช้งานกับการ์ดขยายต่าง ๆ ที่เสียบในสล็อต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ADDRESS	DESCRIPTION	NOTE
1F0-1F8	Fixed disk	1
200-20F	Games adapter	
210-217	Expansion print	2
278-27F	2nd parallel printer port	1
2B0-2DF	Alternate EGA	
2FB-2FF	2nd serial port	
2E1	GPIB (0)	4
2E2-2E3	Data acquisition	4
300-31F	Prototype card	
320-32F	Fixed disk	2
360-36F	PC network	
378-37F	1st parallel print 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		
		Device on main board not included
	1	AT only
	2	PC only
	3	2nd Bisynchronous on AT only
	4	These device 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

อินเตอร์รัพต์ผ่านทางสล็อตก็พจากตารางที่ 4.2 จะเห็นได้ว่า แอดเดรสของพอร์ตถูกแบ่งออกเป็นช่วงย่อย ๆ ซึ่งจะกำหนดไว้ให้ใช้กับอุปกรณ์ I/O เฉพาะอย่างถ้าในระบบของเราไม่ได้ใช้งานอุปกรณ์นั้น เราสามารถนำเอาแอดเดรสช่วงนั้นมาใช้งานได้ เช่น ถ้าในระบบของเราไม่ได้ใช้จอยสติก (Joystick) เราสามารถใช้งานพอร์ตในช่วง (200H-20FH) , ได้แต่อย่าไรก็ตามการเลือกใช้งานพอร์ตที่ไม่ได้ถูกกำหนดให้ใช้กับอุปกรณ์อื่นจะดีกว่า ทำให้อุปกรณ์ที่ใช้งานผ่านพอร์ตนี้สามารถใช้ได้กว้างขวางยิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 อินเตอร์รัพต์ (Interrupt)

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

การอินเตอร์รัพต์แบบ Maskable มีจำนวน 256 เวกเตอร์ (เวกเตอร์ในที่นี้ คือ แอดเดรสเริ่มต้นของโปรแกรมย่อย (Interrupt Service Routine) ที่ถูกกระทำเมื่อได้รับการอินเตอร์รัพต์) ในจำนวนทั้งหมด 256 เวกเตอร์นี้ จะรวมถึงซอฟแวร์อินเตอร์รัพต์ (B DOS Call) , สัญญาณอินเตอร์รัพต์ที่เกิดจากตัว CPU เองด้วย เช่น การหารด้วยศูนย์ (Divide By Zero) , การทำทีละคำสั่งที่ใช้ในโปรมแกรมดีบัก๊ (Single Step) และฮาร์ดแวร์อินเตอร์รัพต์ สำหรับฮาร์ดแวร์อินเตอร์รัพต์สามารถแบ่งออกได้ความสำคัญดังนี้ สำหรับรุ่น XTแบ่งได้ 8 ระดับ และ 15 ระดับในรุ่น AT ตารางที่ 4.3 แสดงฮาร์ดแวร์อินเตอร์รัพต์ใน IBM PC อินเตอร์รัพต์บางระดับจะถูกใช้ในเมนบอร์ด เช่น อินเตอร์รัพต์ระดับ 0 เป็นต้น ส่วนที่เหลือจะถูกใช้ในการ์ดขยายต่างๆ เช่น ตัวควบคุมดิสต์ (Floppy & Hard Disk Controller)

ใน IBM PC การขอทำฮาร์ดแวร์อินเตอร์รัพต์แบบ Markable เราไม่จำเป็นต้องให้ค่าเวกเตอร์ เพราะว่า 8259 (Interrupt Controller) จะเป็นตัวจัดการเองทั้งหมด เราเพียงแค่ ให้สัญญาณขอทำอินเตอร์รัพต์ผ่านทางสล็อตก็พอ

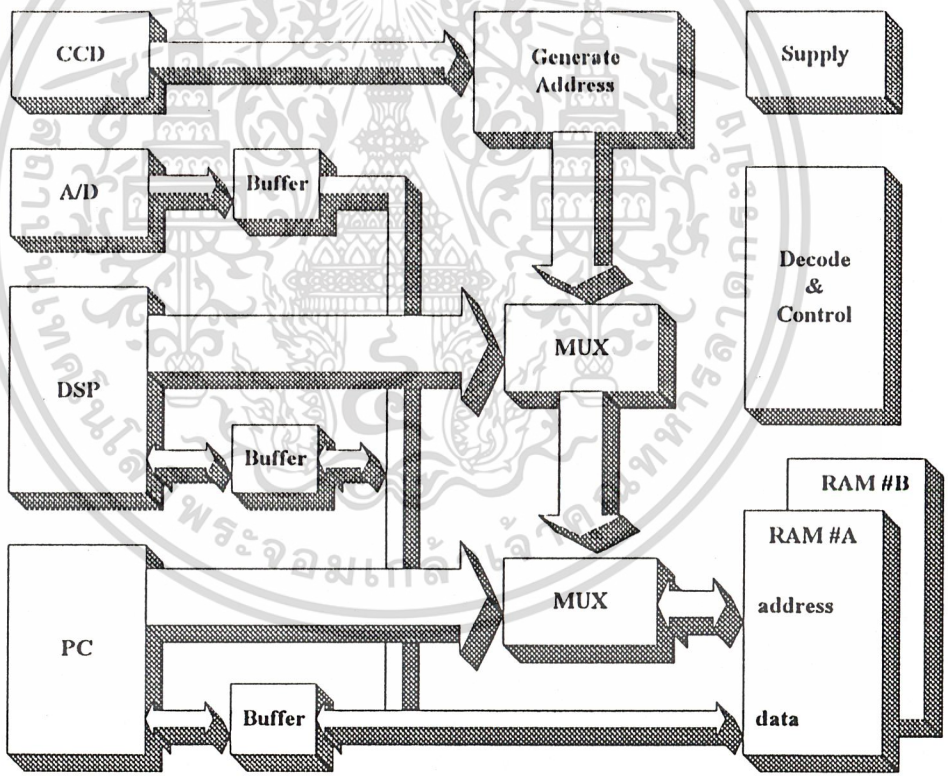
Interrupt Signal		Hardware Interrupt Level	Program Interrupt Number (HEX)	Comment
PC	AT			
		IRQ 0	08	Timer output 0
		IRQ 1	09	Keyboard
//		IRQ 2	0A	Int. Ctrl 2 on AT
		IRQ 8	70	Real time Clock
	//	IRQ 9	71	S/W To IOQ2
	//	IRQ 10	72	Reserved
	//	IRQ 11	73	Reserved
	//	IRQ 12	74	Reserved
		IRQ 13	75	Co-processer
	//	IRQ 14	76	Hard disk
	//	IRQ 15	77	Reserved
//	//	IRQ 3	0B	Serial port 2
//	//	IRQ 4	0C	Serial port 1
//	//	IRQ 5	0D	Hard disk
//	//	IRQ 6	0E	Floppy disk
//	//	IRQ 7	0F	Printer port

ตารางที่ 4.3 แสดงการจัดฮาร์ดแวร์อินเตอร์รัพต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การออกแบบวงจร

การแปลงสัญญาณภาพจากกล้อง CCD ให้เป็นข้อมูลภาพ แล้วนำไปประมวลผลด้วยการ์ด TMS 320C31 (DSP) และนำผลที่ได้ไปแสดงที่จอคอมพิวเตอร์ (PC) นั้นสามารถแบ่งออกเป็นส่วนใหญ่ๆได้ 3 ส่วน คือ การเก็บข้อมูลภาพ,การประมวลผลภาพและการแสดงผลภาพ ซึ่งแต่ละส่วนจะมีสัญญาณแอดเดรส (address buses) และสัญญาณข้อมูล (data buses) เป็นของตัวเอง ดังนั้นจะต้องออกแบบให้ทั้ง 3 ส่วนนี้เข้ากันได้ (compatible) ในหน่วยความจำชั่วคราว (RAM) ที่ต้องใช้ร่วมกัน

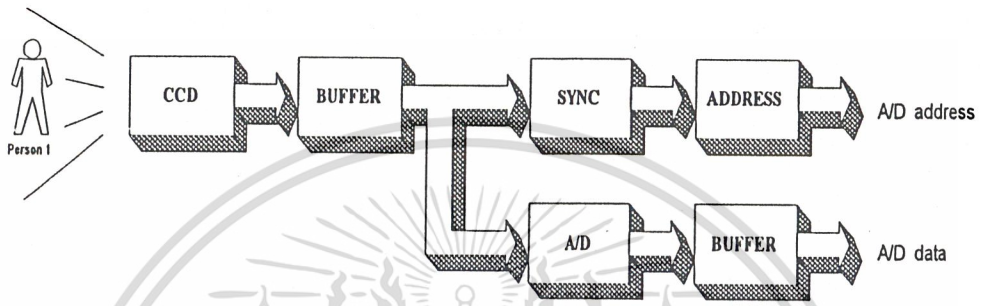


รูปที่ 5.1 แสดงบล็อกไดอะแกรมของทั้ง 3 ส่วนที่ต่อเข้าด้วยกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.1 การแปลงสัญญาณภาพ

เริ่มจากการนำสัญญาณภาพจากกล้อง CCD ที่เป็นสัญญาณอนาล็อกมาแปลงเป็นสัญญาณดิจิตอล และนำสัญญาณอีกส่วนหนึ่งไปสร้างสัญญาณแอดเดรส เพื่อนำไปอ้างอิงหน่วยความจำขนาด 256 Kbyte



รูปที่ 5.2 แสดงบล็อกไดอะแกรมของส่วนการแปลงสัญญาณภาพ

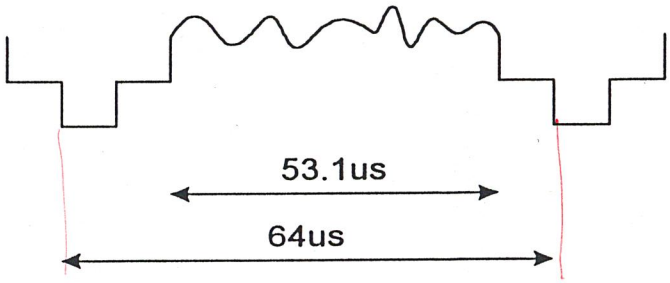
5.1.1 การคำนวณความถี่ที่ใช้ในการสุ่มสัญญาณภาพ

ก่อนอื่นต้องพิจารณาถึงเวลาที่ใช้ในการสแกนภาพทางแนวนอนว่าใช้เวลาเท่าไหร่ จากนั้นก็นำเวลาดังกล่าว มาหารด้วยจำนวนจุดภาพที่ต้องการ เพื่อที่จะได้เวลาในการสุ่ม (sample time) แต่ละจุด ซึ่งสัญญาณที่ได้จากกล้อง CCD นั้นเป็นระบบ PAL ซึ่งมีพารามิเตอร์ของการสแกนภาพได้แสดงไว้ในตารางที่ 5.1

Scanning	NTSC	PAL	SECAM
Lines/Frame	525	625	625
Frame/second	30	25	25
Interlace ratio	2:1	2:1	2:1
Aspect ratio	4:3	4:3	4:3
Color subcar. (Hz)	3579545	4433619	4406250

ตารางที่ 5.1 แสดงค่าพารามิเตอร์ของระบบต่างๆที่มีใช้กันอยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.3 แสดงเวลาที่ใช้ในการสแกนภาพของสัญญาณทางแนวนอน

จากตารางที่ 5.1 จะเห็นว่าความถี่ที่ใช้ในการสแกนทางแนวนอนของระบบ PAL เท่ากับ 15,625 Hz (625x25) หรือเท่ากับ 64 μS/scanline แต่มีช่วงเวลาของการเกิดสัญญาณภาพจริงเพียง 53.1 μS จากเวลาที่ได้นี้สามารถนำไปคำนวณหาความถี่ของการสุ่ม (sample frequency) ได้ด้วยสมการ

$$T_s = T_{hot} / N = 53.1/512 = 0.103\ \mu S \text{ หรือ } 9.642\ MHz$$

เมื่อ T_s คือ เวลาในการสุ่มสัญญาณภาพแต่ละจุด

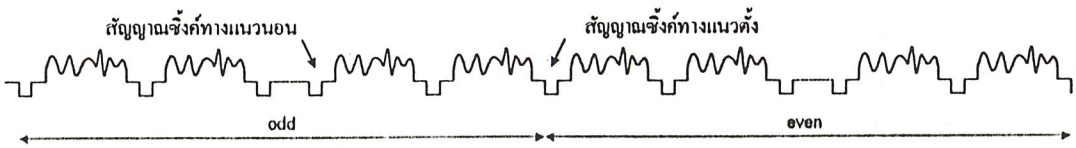
T_{hot} คือ เวลาในการสแกนภาพทางแนวนอน

N คือ จำนวนจุดภาพที่ต้องการในแนวนอน

จากความถี่ที่คำนวณได้ในการสุ่มขนาด 512 จุดภาพ/scanline ที่ต้องใช้ความถี่ 9.642 MHz นั้นทำได้ยาก ดังนั้นจึงเลือกใช้ความถี่ที่สูงขึ้นและสามารถหาได้ง่ายคือ ความถี่ขนาด 12 MHz

- <u>การแปลงสัญญาณภาพ</u>

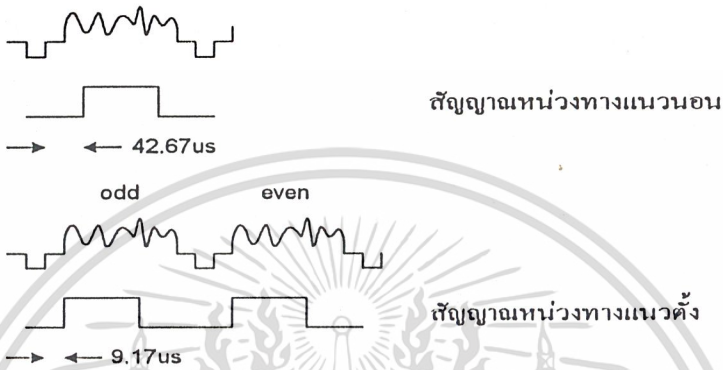
สัญญาณภาพที่เป็นระบบPALนั้น จะประกอบด้วยสัญญาณภาพที่มีสัญญาณซิ้งค์ทางแนวนอนและทางแนวตั้ง เป็นตัวกำหนดการสแกนจุดภาพ โดยแบ่งการสแกนเป็น 2 ฟิลด์ คือ ฟิลด์คู่ (odd) กับฟิลด์คี่ (even)



รูปที่ 5.4 แสดงส่วนประกอบของสัญญาณภาพระบบ PAL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการสุ่มสัญญาณภาพ (sampling) ทางแนวนอน ให้ได้ข้อมูลดิจิตอลขนาด 512 จุดนั้น ย่อมมีสัญญาณส่วนที่ไม่ได้ถูกสุ่มอยู่ด้วย เพราะสัญญาณภาพที่ได้จากกล้อง CCD นั้นจะมีการเผื่อสัญญาณไว้ให้สามารถสุ่มสัญญาณได้ถึง 1024 จุดภาพ ดังนั้นจึงต้องมีการหน่วงเวลา เพื่อให้สามารถสุ่มสัญญาณภาพที่อยู่กึ่งกลางภาพได้



จากความถี่ที่ใช้ในการสุ่ม = 12 MHz หรือ 0.083 µS
ดังนั้นเวลาที่ใช้ในการสุ่ม = (0.083)(512) = 42.67
และเวลาที่ใช้ในการหน่วง = (61 - 42.67)/2 = 9.17

รูปที่ 5.5 แสดงการสุ่มสัญญาณภาพและการคำนวณหาค่าเวลาหน่วง

5.2 การกำเนิดสัญญาณแอดเดรส

ในการสร้างสัญญาณแอดเดรส เพื่อใช้ในการอ้างตำแหน่งหน่วยความจำในการเก็บข้อมูลภาพ แบ่งออกได้เป็นสองส่วนคือ ส่วนของสัญญาณฟิลด์คู่ (odd) กับฟิลด์คี่(even) ในการอ้างตำแหน่งหน่วยความจำ RAM#A สามารถทำได้ดังนี้คือ

- <u>ตัวนับทางแนวนอน</u>

ในการเก็บภาพแนวนอนขนาด 512 จุดภาพ/scanline ทำให้ต้องมีการอ้างตำแหน่งของหน่วยความจำ 512 ตำแหน่ง ด้วยความถี่ 6 MHz ที่สำคัญก็คือ การเริ่มต้นนับจะต้องสัมพันธ์กับสัญญาณภาพทางแนวนอน จึงต้องใช้สัญญาณพัลซ์ที่มาจากการหน่วง มาใช้ในการกำหนดตำแหน่งการเริ่มเก็บข้อมูลภาพ แล้วหลังจากทำการนับครบ 512 จุดภาพแล้วตัวนับก็จะส่งพัลซ์ออกมาหยุดการนับพร้อมกับรีเซ็ตให้แอดเดรสช่วง 9 บิตล่างมีค่าเป็นศูนย์ใหม่ เพื่อรอการกระตุ้นจากสัญญาณภาพทางแนวนอนต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

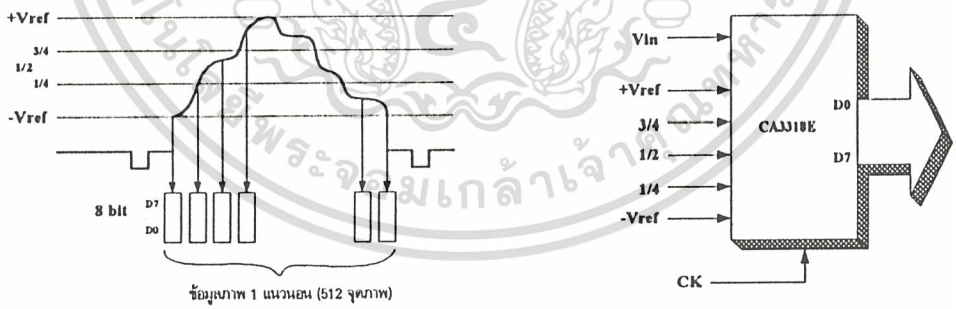
- ตัวนับทางแนวตั้ง

เป็นการกำเนิดสัญญาณแอดเดรสเพื่อเก็บภาพจำนวน 256 scanline/ฟิลด์ภาพ ก็มีการออกแบบเช่นเดียวกันกับการสร้างแอดเดรสทางแนวนอน โดยจะใช้สัญญาณพัลซ์ที่เกิดจากตัวนับทางแนวนอนมาเป็นอินพุทของตัวนับทางแนวตั้ง แล้วสร้างเป็นแอดเดรสช่วง 8 บิตบนต่อไปและเมื่อนับครบ 256 scanline ของฟิลด์คู่แล้วก็จะทำการหยุดและรีเซ็ตตัวเองเพื่อรอการกระตุ้นจากสัญญาณทางแนวตั้งต่อไป

เมื่อนำแอดเดรสที่ได้จากตัวนับทั้งสองชุดนี้มารวมกันก็จะได้แอดเดรส 17 เส้นสำหรับใช้อ้างตำแหน่งของหน่วยความจำได้ 128 kbyte แต่เนื่องจากต้องการเก็บข้อมูลภาพทั้งสองฟิลด์ ดังนั้นจึงต้องมีสัญญาณแอดเดรสอีกเส้น เพื่อแยกการเก็บภาพของทั้งสองฟิลด์ เมื่อรวมแล้วจะมีแอดเดรส 18 เส้น ซึ่งสามารถอ้างหน่วยความจำได้ 256 kbyte ก็เพียงพอสำหรับภาพขนาด 512x512 จุดภาพ

5.3 การแปลงสัญญาณอนาล็อกเป็นดิจิตอล

สัญญาณภาพที่มาจากกล้อง CCD จะถูกเปลี่ยนให้เป็นสัญญาณดิจิตอล โดยวงจร A/D converter โดยมีสัญญาณควบคุมการทำงาน(select) ที่มาจากส่วนควบคุม เพื่อควบคุมจังหวะการติดต่อกับหน่วยความจำ ซึ่งมีสัญญาณนาฬิกา 12 MHz เป็นตัวขับจังหวะการสุ่มของ A/D converter นอกจากนั้นยังมีส่วนของแรงดันอ้างอิงที่จ่ายให้กับตัว A/D converter เพื่อให้สามารถปรับค่าให้เหมาะสมกับสัญญาณที่เข้ามาได้

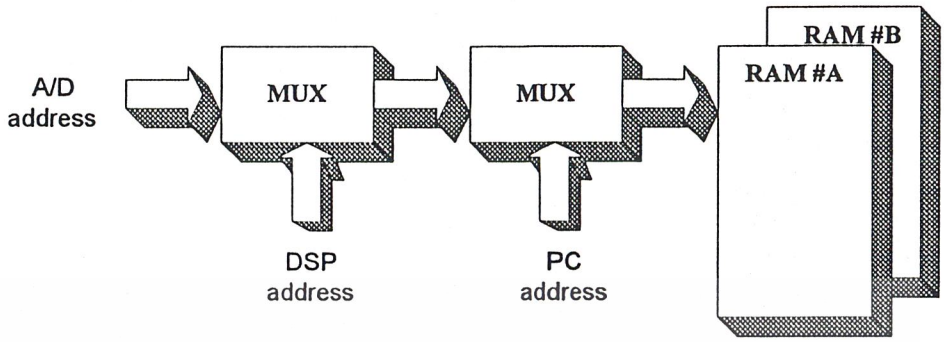


รูปที่ 5.6 แสดงสัญญาณภาพกับแรงดันอ้างอิงและการต่อวงจร A/D converter

5.4 การอ้างตำแหน่งหน่วยความจำ

การอ้างตำแหน่งหน่วยความจำนี้ ประกอบด้วยหน่วยความจำชั่วคราว (RAM) จำนวนสองชุดคือ RAM#A กับ RAM#B ซึ่งเป็นหน่วยความจำร่วมกันระหว่าง หน่วยเก็บข้อมูลภาพ (A/D) หน่วยประมวลผล (DSP) และหน่วยแสดงผล (PC) ดังนั้นจึงต้องจัดการอ้างตำแหน่งหน่วยความจำให้ทั้ง 3 ส่วนเข้ากันได้ โดยใช้วงจร multiplex มาช่วยในการสวิทช์แอดเดรสของแต่ละส่วน

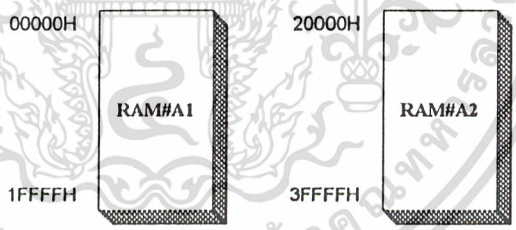
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ ดัดแปลง เผยแพร่ หรือแสวงหาประโยชน์ทางการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.7 แสดงบล็อกไดอะแกรมของวงจร multiplex

- การอ้างหน่วยความจำของวงจรเก็บข้อมูลภาพ

จะเริ่มเก็บสัญญาณภาพฟิลด์คู่ก่อนที่แอดเดรส 00000H - 1FFFFH และ สัญญาณภาพฟิลด์คี่ที่แอดเดรส 20000H - 3FFFFH สำหรับอ้างหน่วยความจำ RAM#A

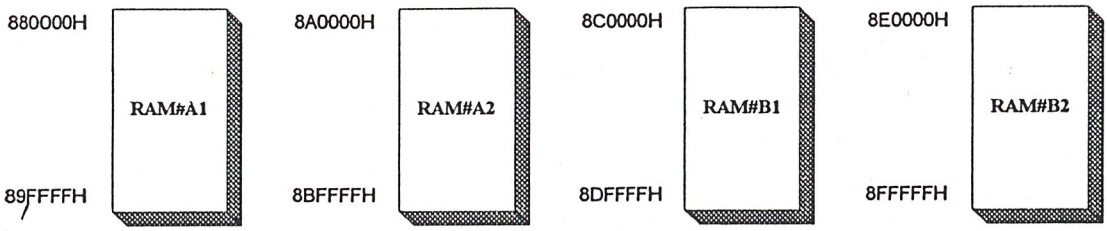


รูปที่ 5.8 แสดงการอ้างหน่วยความจำของวงจรเก็บข้อมูลภาพ

- การอ้างหน่วยความจำของหน่วยประมวลผลภาพ (DSP)

ใน memory map ของการ์ด TMS320C31 จะมีช่วงแอดเดรสอยู่ช่วงหนึ่งที่สามารถต่อขยายไปยังหน่วยความจำภายนอกได้คือช่วงแอดเดรส 80A000H - BFFFFFH ซึ่งอ้างตำแหน่งได้ถึง 4 Mbyte แต่เลือกใช้ที่แอดเดรส 880000H - 8BFFFFH สำหรับ RAM#A และแอดเดรส 8C0000H - 8FFFFFH สำหรับ RAM#B ทั้งนี้เพื่อความสะดวกในการออกแบบวงจร

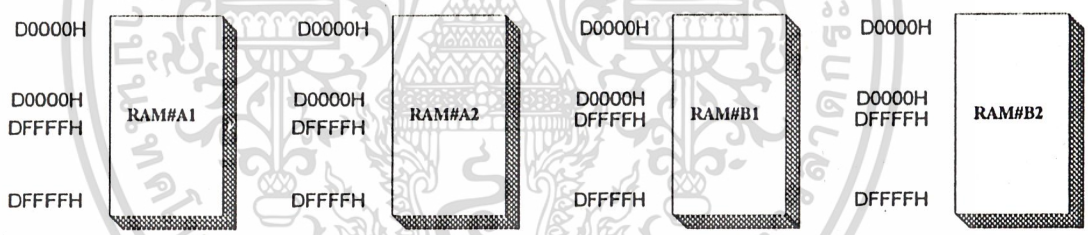
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.9 แสดงการอ้างหน่วยความจำของหน่วยประมวลผลภาพ (DSP)

- การอ้างหน่วยความจำของคอมพิวเตอร์(PC)

ใน memory map ของเครื่องคอมพิวเตอร์นั้น จะมีแอดเดรสอยู่ช่วงหนึ่งที่สามารถต่อขยายไปยังหน่วยความจำภายนอกได้คือช่วงแอดเดรส D0000H - DFFFFH ซึ่งอ้างตำแหน่งหน่วยความจำได้เพียง 64 Kbyte ดังนั้นต้องมีการแบ่งการอ่านข้อมูลของ RAM แต่ละตัวออกเป็นสองช่วง เพื่อที่จะได้สามารถอ่านข้อมูลภาพใน RAM ได้ครบทุกตัว



รูปที่ 5.10 แสดงการอ้างหน่วยความจำของคอมพิวเตอร์(PC)

5.5 ชุดควบคุม

ส่วนนี้จะเป็นตัวควบคุมการทำงานของแผงวงจรเก็บข้อมูลภาพทั้งหมด โดยคำสั่งต่างๆจะถูกสั่งออกมาจากซอฟแวร์ ด้วยการส่งค่าออกมาที่พอร์ตเอาท์พุท โดยตำแหน่งของพอร์ตที่จะส่งค่าออกมานั้นจะเลือกที่ตำแหน่ง 300H ค่าที่ได้นี้จะนำไปควบคุมการเก็บภาพควบคุมวงจรmultiplex การเลือกหน่วยความจำ เป็นต้น ส่วนค่าที่ออกมาควบคุมการทำงานต่างๆนั้น สามารถดูได้จากตารางที่ 5.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณควบคุม				ค่าของพอร์ต	การทำงาน
E3	E2	E1	E0		
1	0	X	X	0x08	เริ่มเก็บข้อมูลภาพ
1	1	X	X	0x12	เริ่มทำการประมวลผล
0	0	0	0	0x00	
0	0	0	1	0x01	อ่านข้อมูล
0	0	1	0	0x02	จากRAM#A
0	0	1	1	0x03	
0	1	0	0	0x04	
0	1	0	1	0x05	อ่านข้อมูล
0	1	1	0	0x06	จากRAM#B
0	1	1	1	0x07	

ตารางที่ 5.2 แสดงสัญญาณในการควบคุมต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

ผลการทดลอง

จากการที่ใช้ CARD TMS320C31 (DSP) ในการประมวลผลภาพจะต้องมีส่วนประกอบทางด้านฮาร์ดแวร์และทางซอฟต์แวร์เข้ามาช่วยในการประมวลผล โดยในโครงงานนี้ได้นำข้อมูลภาพที่จะมาประมวลผลจาก 2 แหล่งภาพคือนำข้อมูลภาพมาจากกล้อง CCD และนำข้อมูลภาพมาจากไฟล์อิมเมจ(File Image) โดยทำการโหลดลงไปในหน่วยความจำของการ์ดที่ได้ออกแบบเอง (RAM#A) จากนั้นก็จะเลือกว่าจะทำการประมวลผลด้วย algorithm แบบใด เช่น Sobel Enhancement , Edge Enhancement , Laplacian Enhancement , High-Pass Filter , Low-Pass Filter และเลือกว่าจะใช้ PC หรือ DSP

ฟังก์ชั่นต่าง ๆ ของโปรแกรมมีรายละเอียดดังนี้

F1	: CCD (นำภาพจากกล้อง CCD มาแสดงที่หน้าจอ)
F2, CTRL S	: SAVE IMAGE
F3, CTRL O	: OPEN IMAGE
F4	: CLEAR SCREEN
F5	: DSP PROCESSING
F6	: PC PROCESSING
F7	: HISTOGRAM
F8	: RUN CCD
F9	: HELP
F10, ALT F4	: QUIT
CTRL F1	: SOBEL ENHANCEMENT
CTRL F2	: LOWPASS ENHANCEMENT
CTRL F3	: HIGHPASS ENHANCEMENT
CTRL F4	: LAPLACIAN ENHANCEMENT
CTRL F5	: EDGE ENHANCEMENT
A	: SHOW RAM#A
B	: SHOW RAM#B
CTRL A	: CLEAR RAM#A
CTRL B	: CLEAR RAM#B

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ALT A : LOAD SCREEN TO RAM#A

ALT B : LOAD SCREEN TO RAM#B

CTRL I : INVERT IMAGE

CTRL R : BRIGHTNESS IMAGE

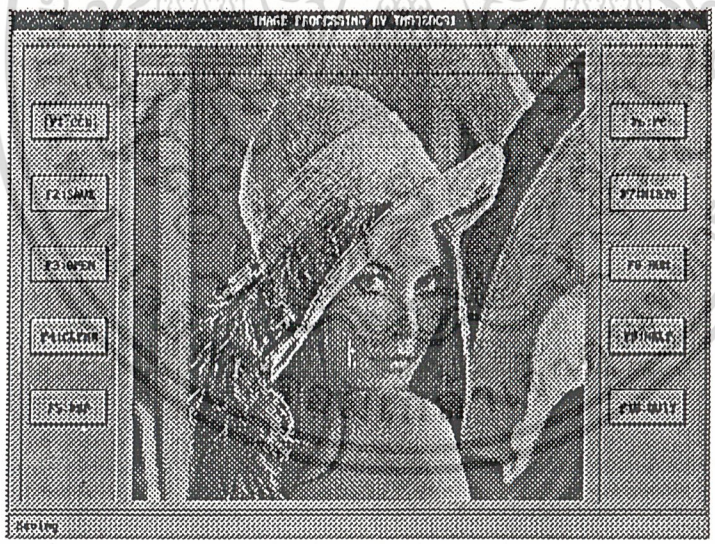
CTRL D : SWITCH TO DSP

CTRL F : SAVE FULL SCREEN

ซึ่งผลการทดลองที่ได้จะแบ่งออกเป็นสองส่วน คือส่วนที่ได้จากกล้อง CCD และส่วนที่ได้จากไฟล์อิมเมจ

6.1 ผลการทดลองกับภาพที่ได้จาก File Image

6.1.1 ข้อมูลภาพต้นแบบที่ได้จาก File Image ดังแสดงในรูปที่ 6.1

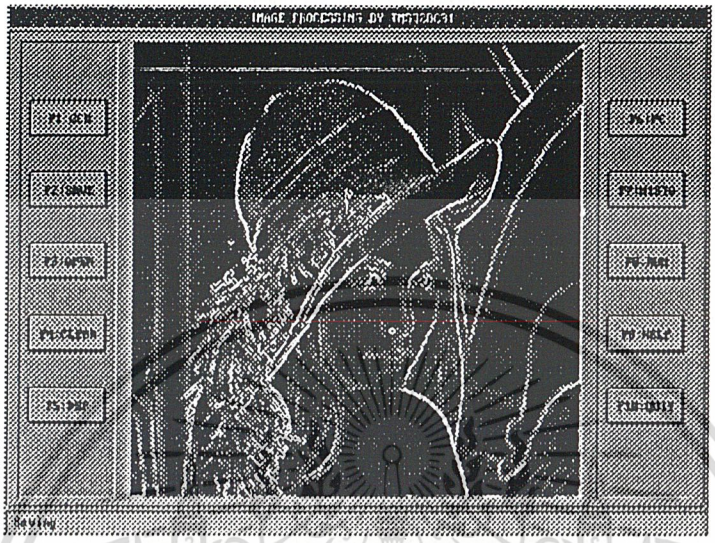


รูปที่ 6.1 แสดงภาพ Lena ที่ได้จากไฟล์ข้อมูลภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

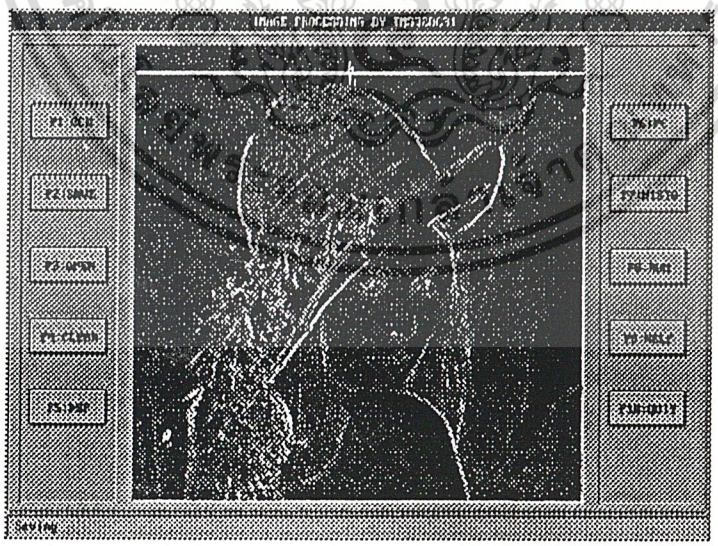
6.1.2 ประมวลผลภาพด้วย DSP

- การหาขอบภาพด้วยวิธีของ Sobel Enhancement



รูปที่ 6.2 แสดงผลของภาพที่ผ่านการประมวลผลโดยใช้วิธี Sobel Enhancement

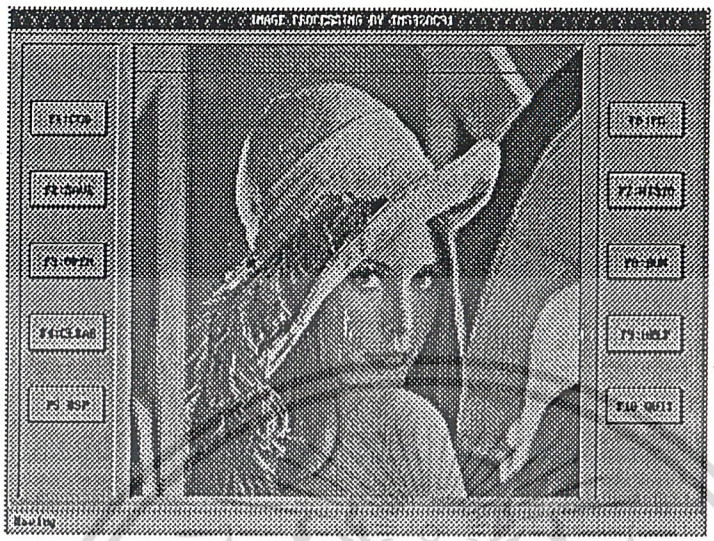
- การหาขอบภาพด้วยวิธีของ Laplacian Enhancement



รูปที่ 6.3 แสดงผลของภาพที่ผ่านการประมวลผลโดยใช้วิธี Laplacian Enhancement

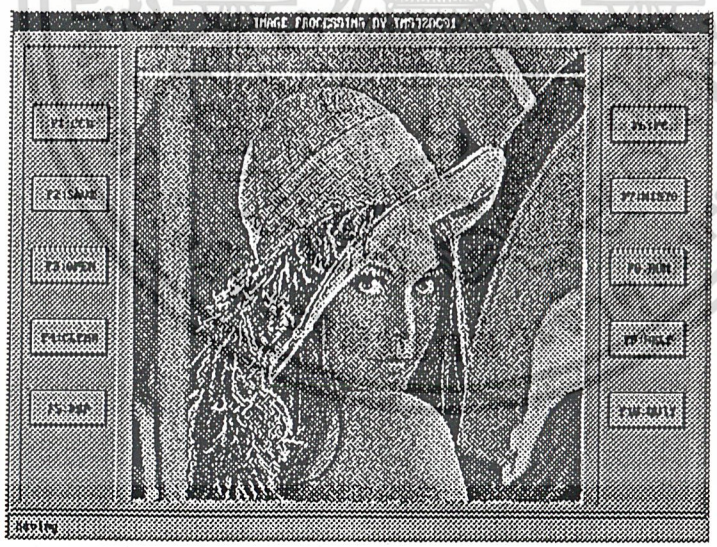
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การทำ Lowpass Filter



รูปที่ 6.4 แสดงผลของภาพที่ผ่านการประมวลผลโดยใช้วิธี Lowpass Filter

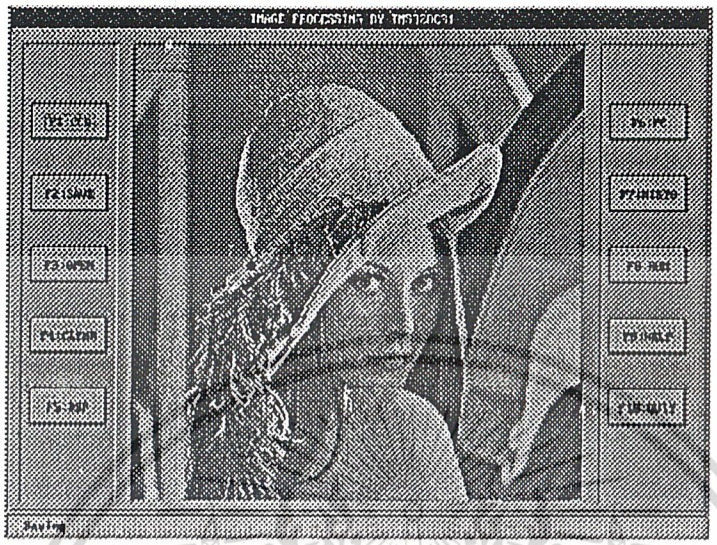
- การทำ Highpass Filter



รูปที่ 6.5 แสดงผลของภาพที่ผ่านการประมวลผลโดยใช้วิธี HighPass Filter

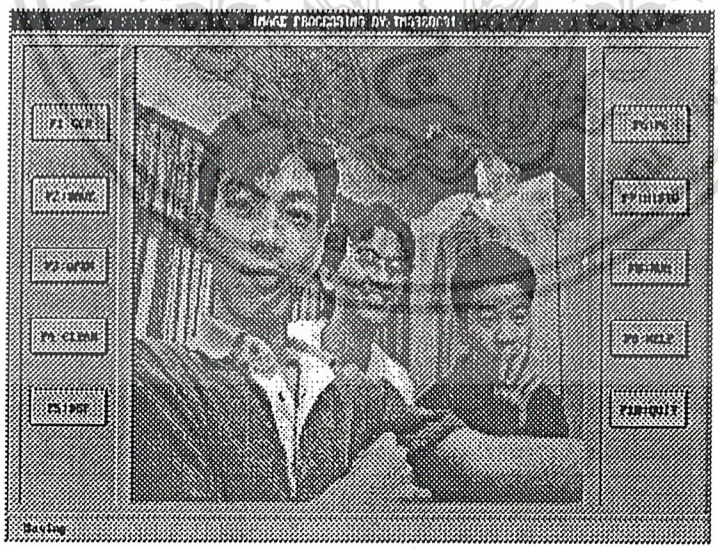
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- การทำ Edge Enhancement



รูปที่ 6.6 แสดงผลของภาพที่ผ่านการการประมวลผลโดยใช้วิธี Edge Enhancement

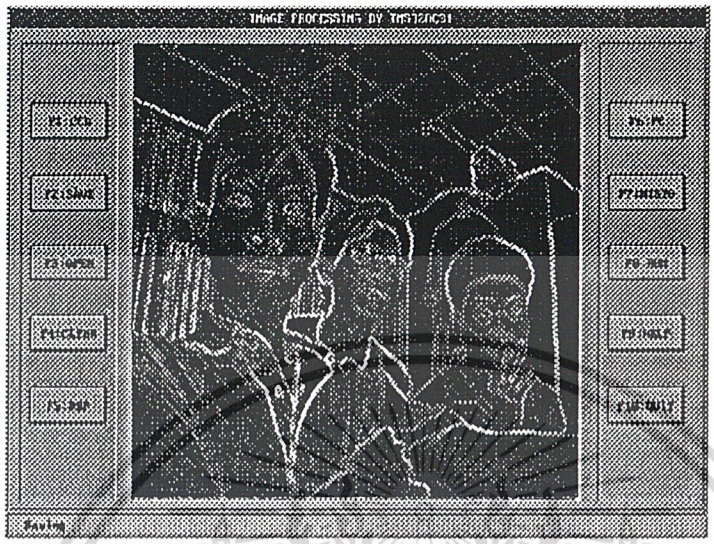
6.2 การทดลองกับภาพที่ได้จากกล้อง CCD



รูปที่ 6.7 แสดงภาพที่ได้จากกล้อง CCD

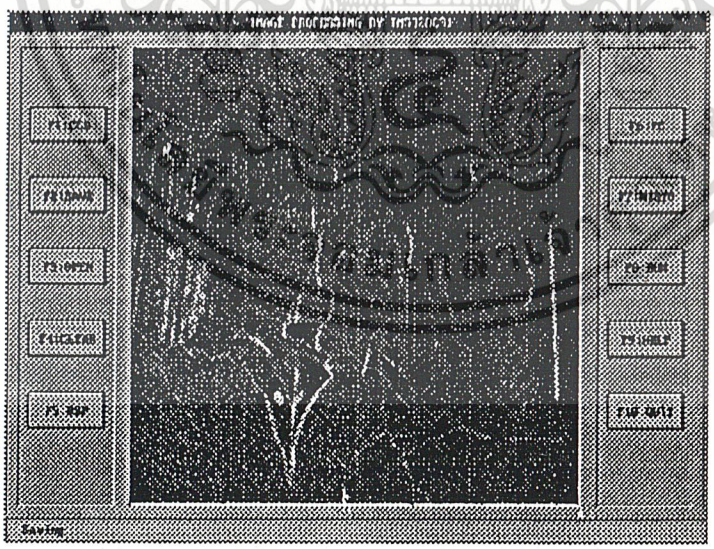
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **การหาขอบภาพด้วยวิธีของ Sobel Enhancement**



รูปที่ 6.8 แสดงภาพที่ผ่านการประมวลผลโดยใช้วิธี Sobel Enhancement

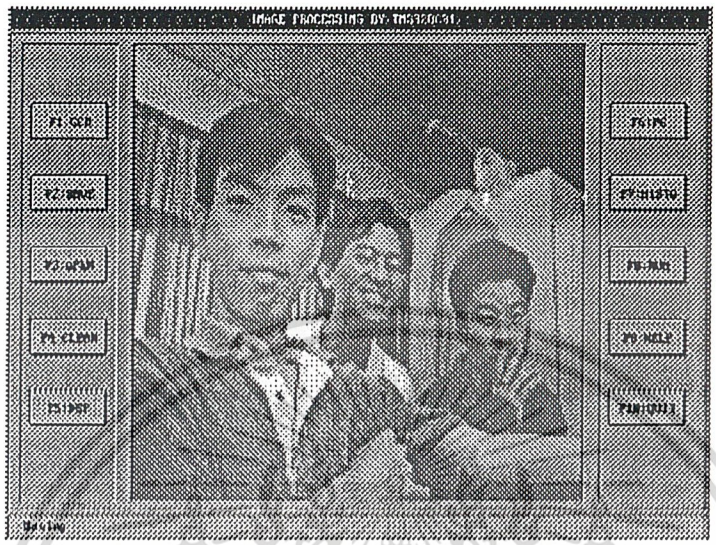
- **การหาขอบภาพด้วยวิธีของ Laplacian Enhancement**



รูปที่ 6.9 แสดงภาพที่ผ่านการประมวลผลโดยใช้วิธี Laplacian Enhancement

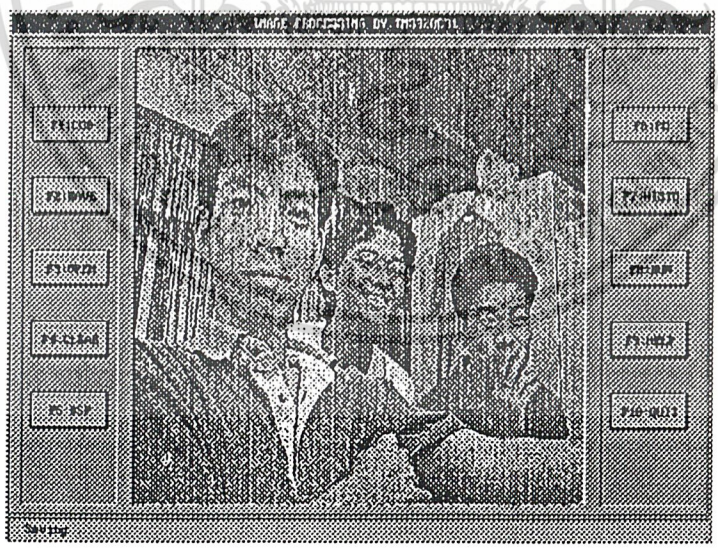
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **การหาขอบภาพด้วยวิธีของ Low-Pass Fileter**



รูปที่ 6.10 แสดงภาพที่ผ่านการประมวลผล**โดยใช้วิธี** Low-Pass Filter

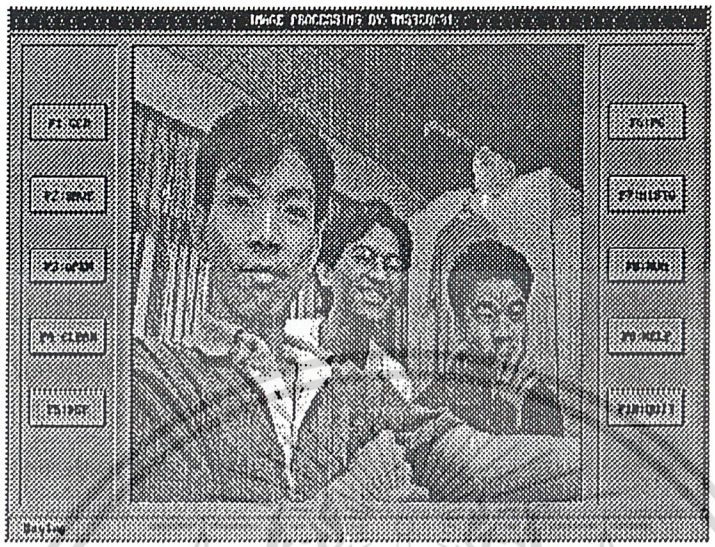
- **การหาขอบภาพด้วยวิธีของ High-Pass Fileter**



รูปที่ 6.11 แสดงภาพที่ผ่านการประมวลผล**โดยใช้วิธี** High-Pass Filter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **การหาขอบภาพด้วยวิธีของ Edge Enhancement**

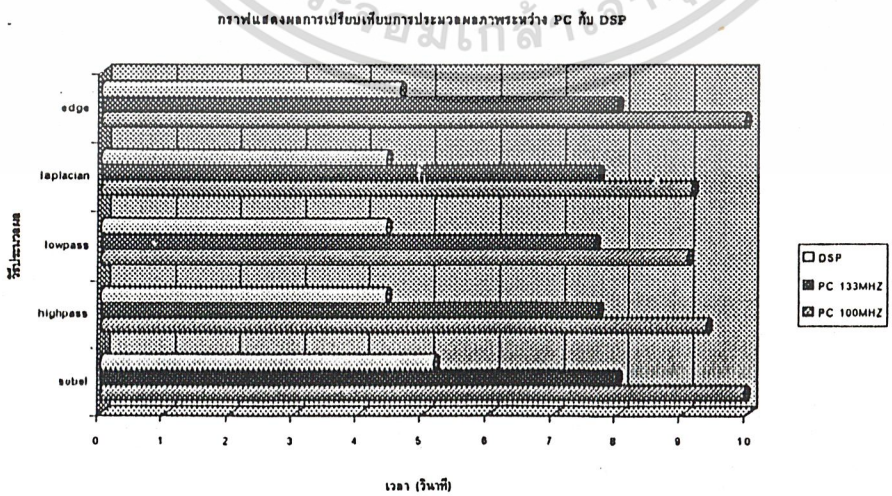


รูปที่ 6.12 แสดงภาพที่ผ่านการประมวลผล**โดยใช้วิธี** Edge Enhancement

6.3 การเปรียบเทียบการประมวลผลระหว่าง PC กับ DSP

ในการประมวลผลภาพ สำหรับการประมวลภาพทั้งสองแบบ จะมีข้อแตกต่างกันเล็กน้อย คือ ในการประมวลผลด้วยเครื่อง PC นั้นข้อมูลภาพที่ได้จากการประมวลผลนั้นจะนำมาแสดงที่หน้าจอทันที โดยไม่ต้องรอให้ทำการประมวลผลเสร็จหมดทั้งภาพ แต่สำหรับการประมวลผลโดยใช้ DSP นั้นข้อมูลภาพที่ได้จากการประมวลผลนั้นจะนำมาเก็บไว้ใน RAM#B ก่อน แล้วจากนั้นจึงนำข้อมูลภาพที่ได้จากการประมวลผลมาแสดงที่หน้าจอ

รูปที่ 6.13 กราฟแสดงการเปรียบเทียบการประมวลผลภาพระหว่าง PC กับ DSP



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
บทสรุปและวิจารณ์

7.1 สรุป

สรุปผลงานการวิจัย การประมวลผลภาพด้วยการ์ด TMS 320C31 สำหรับการประมวลผลภาพด้วยกระบวนการสปาเชียลโดเมน (Spatial Domain) ที่ทดสอบการทำงานโดยการหาขอบภาพด้วยวิธีการ Sobel, LowPass, HighPass, Laplacian, Edge Enhancement ซึ่งผลที่ออกมาเป็นที่น่าพอใจ คือสามารถประมวลผลภาพขนาด 512x512 Pixel ได้อย่างรวดเร็ว เมื่อเทียบกับการประมวลผลภาพด้วยเครื่องคอมพิวเตอร์

ข้อจำกัดของวิธีการหาขอบภาพคือ ถ้าหากความแตกต่างของของระดับสีเทาของภาพนั้นมีค่าใกล้เคียงกันมากๆ ก็จะทำให้การหาขอบภาพนั้นไม่สมบูรณ์ ดังนั้นต้องเลือกสถานที่ในการถ่ายภาพที่เหมาะสม เพื่อให้การประมวลผลภาพเป็นไปอย่างสมบูรณ์แบบ

นอกจากการประมวลผลภาพด้วยวิธีการของ Sobol, LowPass, HighPass, Laplacian, Edge Enhancement แล้ว ยังสามารถนำงานวิจัยนี้ไปประยุกต์ใช้ในการประมวลผลภาพด้วยวิธีการอื่น ๆ ได้อีกด้วย เพื่อในอนาคตจะได้มีการนำ DSP ไปศึกษาและประยุกต์ใช้งานในด้านอื่น ๆ ต่อไป ซึ่งงานวิจัยนี้ถูกออกแบบให้สามารถเก็บภาพขนาด 512x512 Pixel ได้ 2 ภาพ เพื่อเพิ่มลูกเล่นในการประมวลผลภาพด้วย DSP ให้มีความคล่องตัวเหมาะสำหรับการนำไปพัฒนาต่อไป

7.2 ข้อจำกัดของระบบ

1. โปรแกรมทำงานบน DOS ได้อย่างเดียว
2. ประมวลผลได้เฉพาะไฟล์ .img เท่านั้น
3. ข้อมูลภาพที่ใช้เป็น Grayscal 256 ระดับ
4. ขนาดของข้อมูลภาพที่ใช้คือ 512x512 Pixel

7.3 ปัญหาและอุปสรรคที่พบ

1. ปกติภายใน การ์ด TMS 320C31 จะมี A/D ในตัวอยู่แล้วแต่เนื่องจากข้อมูลภาพต้องการความถี่ในการแซมปลิ้ง 10 MHz ขึ้นไป ซึ่ง A/D ของการ์ด TMS 320C31 หรือ TLC 32C40 ที่อยู่ในตัวการ์ดนั้นมีความถี่ในการ sampling สูงสุดไม่เกิน 19.2 KHz และข้อมูลที่ได้มีขนาด 10 bits ซึ่งใช้ไม่ได้ ดังนั้นจึงต้องทำการต่อ A/D ภายนอกแล้วต่อขนานบัสข้อมูลกับ การ์ด TMS 320C31 แทน

2. ในการ โหลดข้อมูลจาก A/D converter เข้าไปใน CARD TMS 320C31 แล้วทำการประมวลผลและเก็บในหน่วยความจำนั้น ที่มีความเร็วในการ access time ไม่เท่ากัน คือหน่วยความจำที่ใช้ในการเก็บข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาพมี access time 70 ns แต่การ์ด TMS 320C31 มี access time ที่เร็วกว่า คือ 40 ns ดังนั้นจึงต้องมีการเขียนโปรแกรมสร้าง Wait State ในการ์ด TMS 320C31 เพื่อให้สามารถอ่านเขียนในหน่วยความจำได้

3. เนื่องจากแผงวงจรหลักจะต้องทำการติดต่อกับอุปกรณ์ภายนอก (Interface) ด้วยกันสามส่วนคือ A/D Converter , TMS 320C31 (DSP) และ Computer (PC) ซึ่งแต่ละส่วนจะต้องมีการอ้างตำแหน่งหน่วยความจำชั่วคราวที่ใช้ร่วมกัน (RAM#A, RAM#B) ดังนั้นต้องมีวงจร Multiplex ในการจัดการหน่วยความจำที่ใช้ร่วมกันนี้

4. ในการใช้ DSP ทำการประมวลผลภาพนั้น จะทำการควบคุมการทำงานจาก PC โดยจะทำการโหลดไฟล์โปรแกรม .ASM ที่เขียนขึ้น โหลดส่งผ่าน parallel port ไปยังหน่วยความจำส่วนของโปรแกรมของการ์ด DSP ซึ่งในการโหลดไฟล์โปรแกรมจะต้องมีการตรวจเช็ค parallel port ว่าพร้อมหรือไม่ที่จะทำการส่งข้อมูล และคำสั่งเพื่อจะให้ DSP ทำการประมวลผลตามที่เราต้องการ

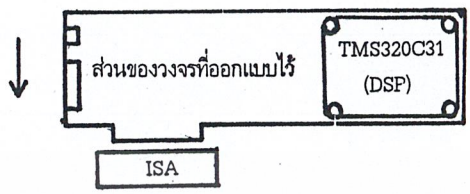
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

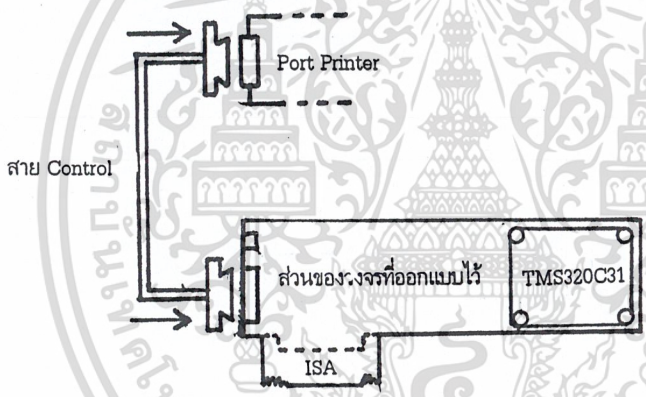
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับขั้นการทดลอง

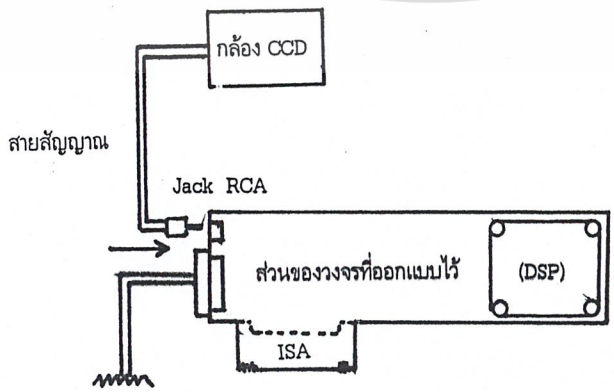
1. นำการ์ดที่ออกแบบไว้สำหรับการ์ด TMS320C31 มาใส่ในสล็อต ISA ของคอมพิวเตอร์



2. นำสายคอนโทรลที่มากับตัวการ์ด TMS320C31 ต่อเข้ากับ Port Printer

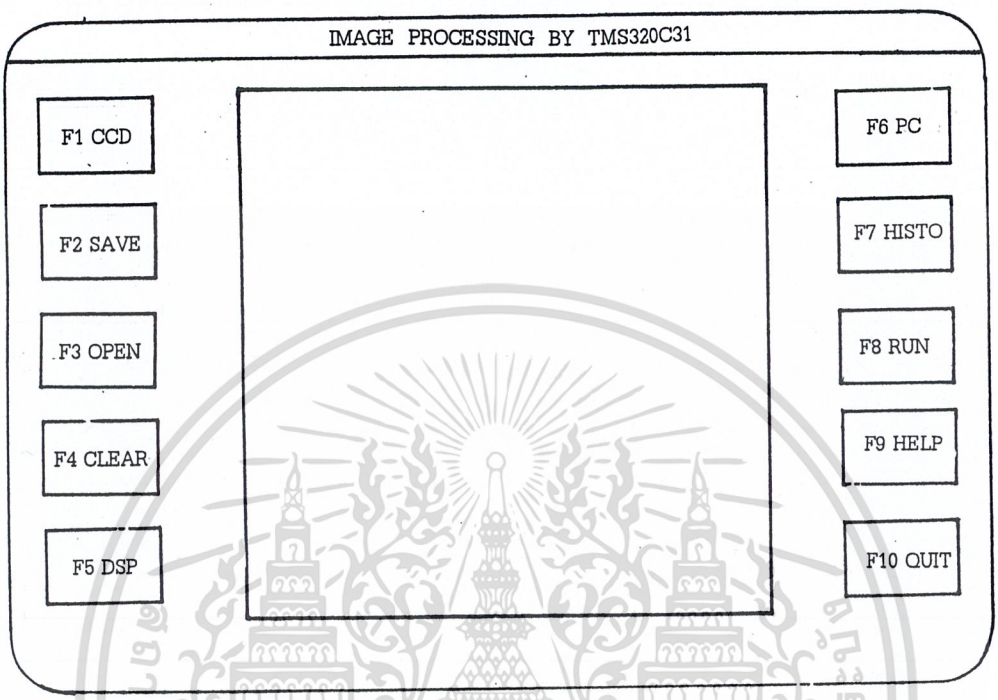


3. นำสายสัญญาณจากกล้อง CCD ต่อเข้ากับ Input ของการ์ด



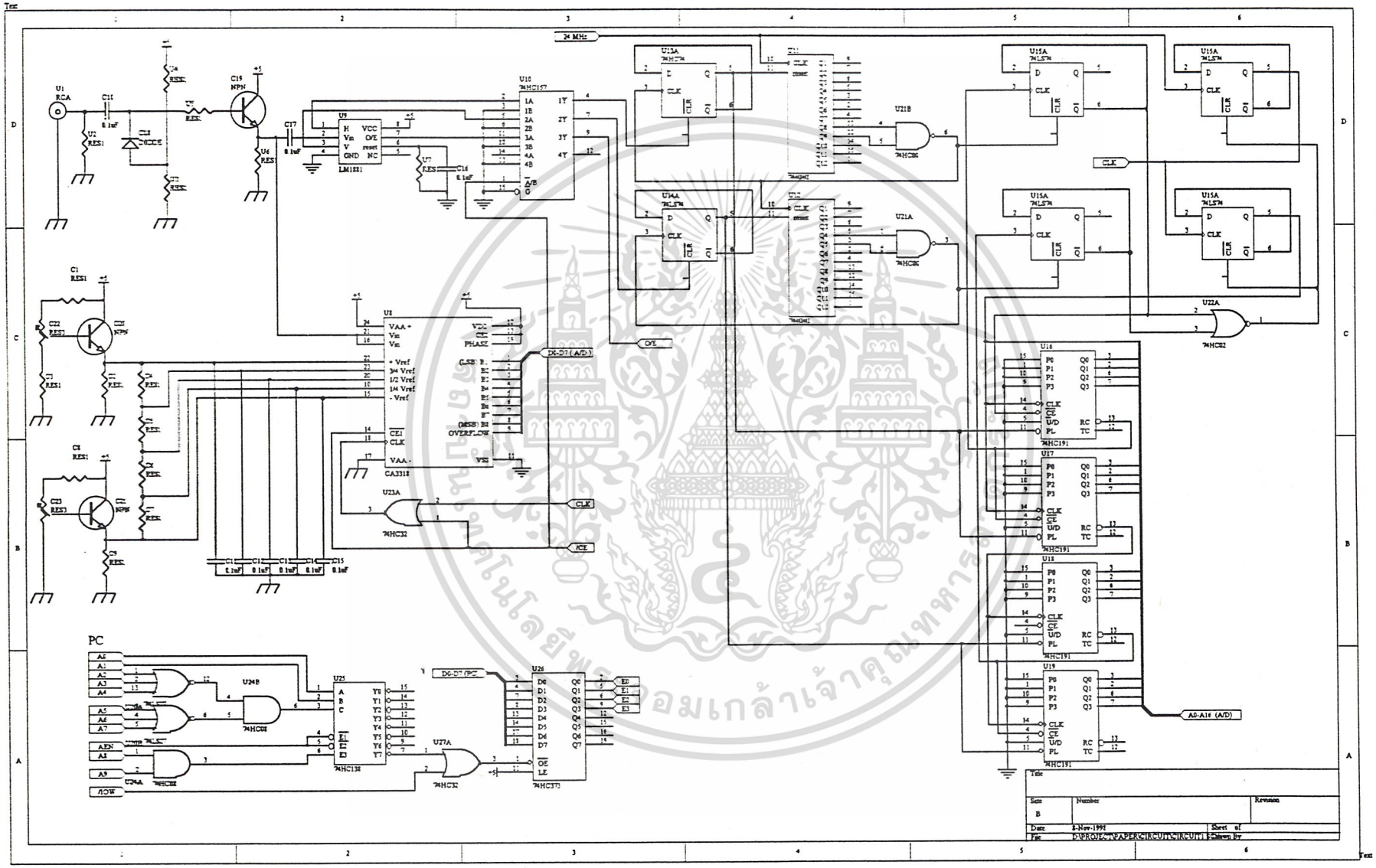
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

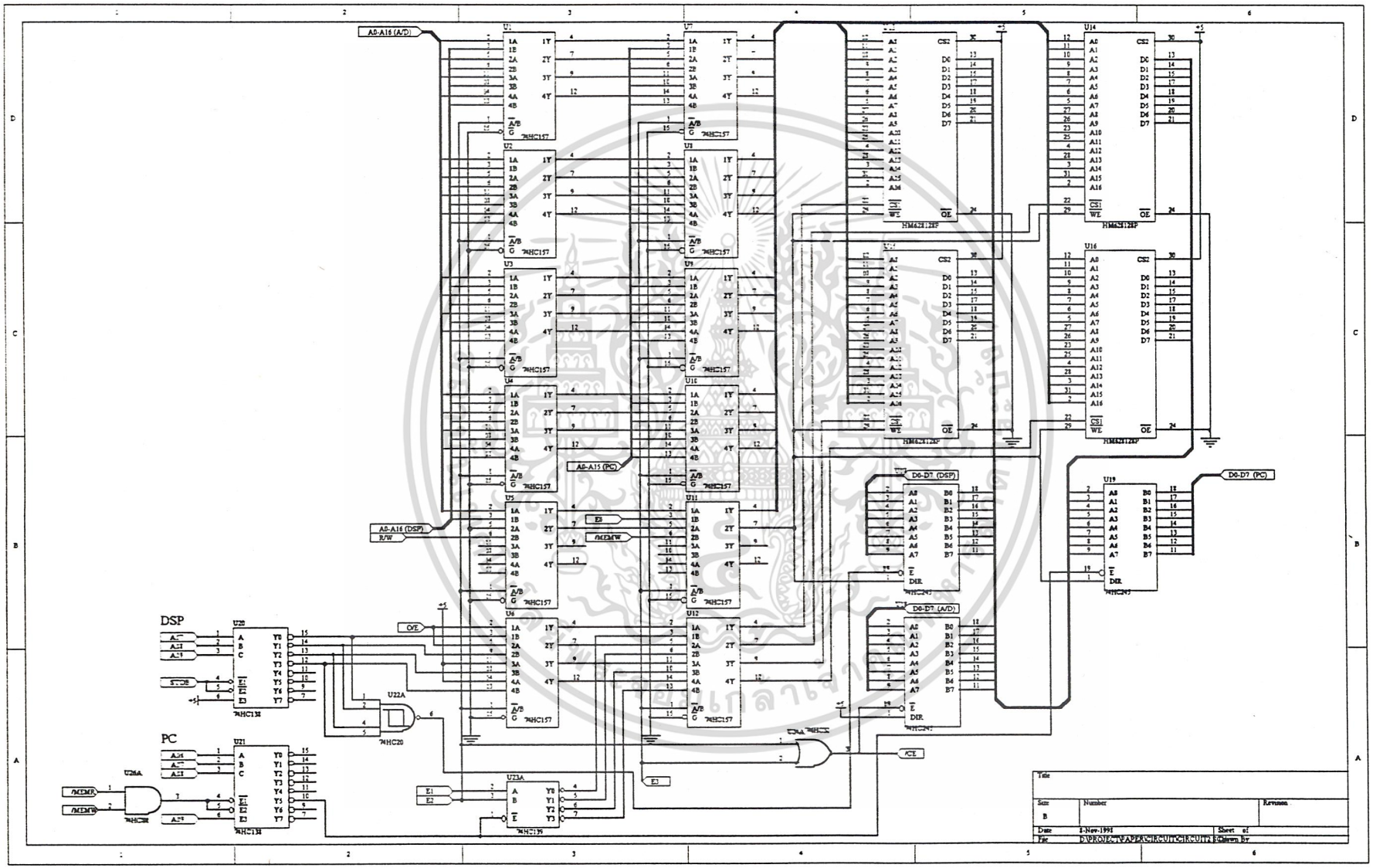
4. เมื่อติดตั้งสายสัญญาณต่างๆ เรียบร้อยแล้วก็ทำการเปิดเครื่องคอมพิวเตอร์เข้าไปสู่การทำงานใน DOS Mode
5. จากนั้นก็เข้าสู่โปรแกรมการใช้งานชื่อ c31_prj.exe ซึ่งภาพที่ปรากฏหน้าจอจะเป็นดังรูป

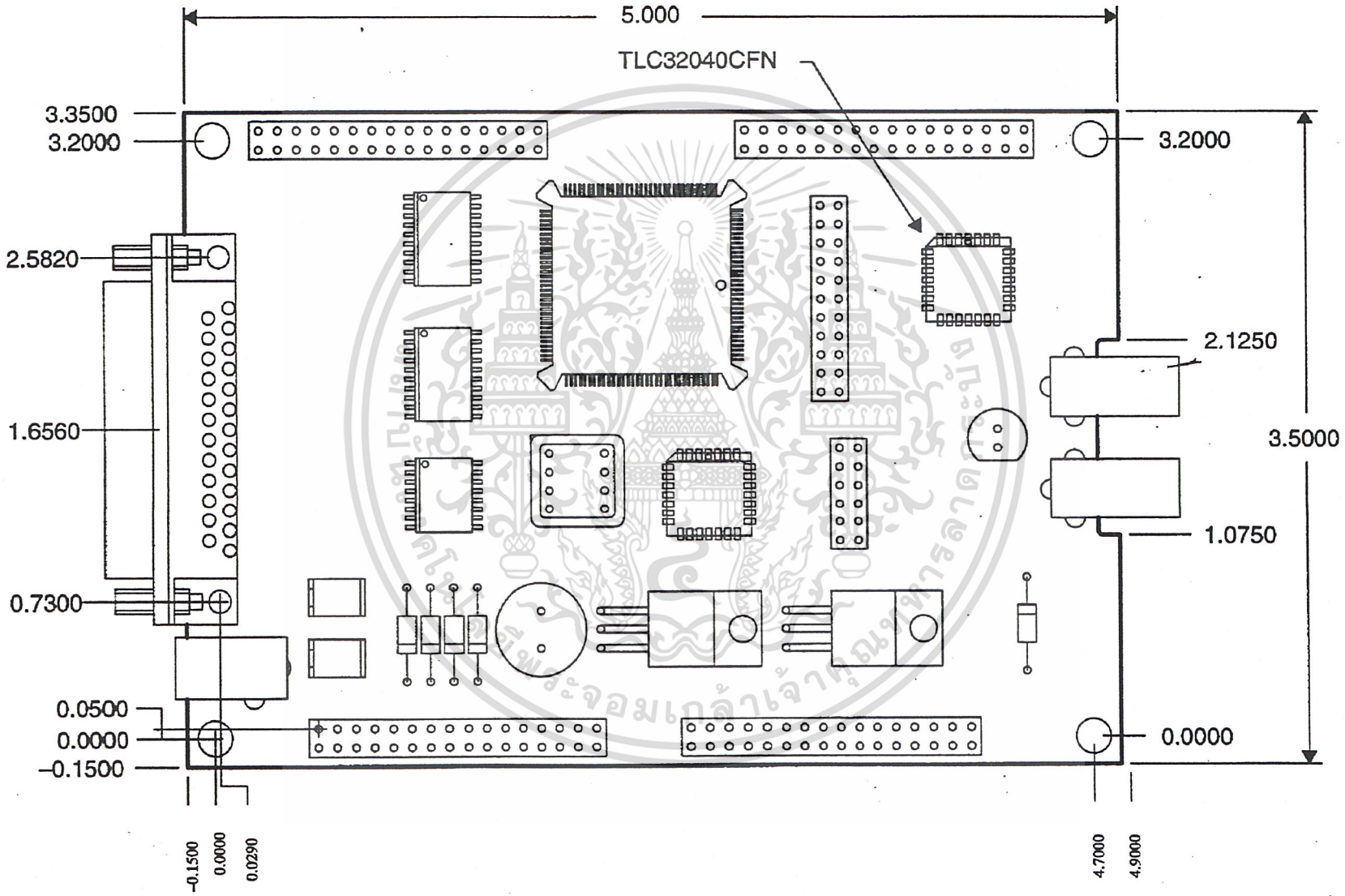


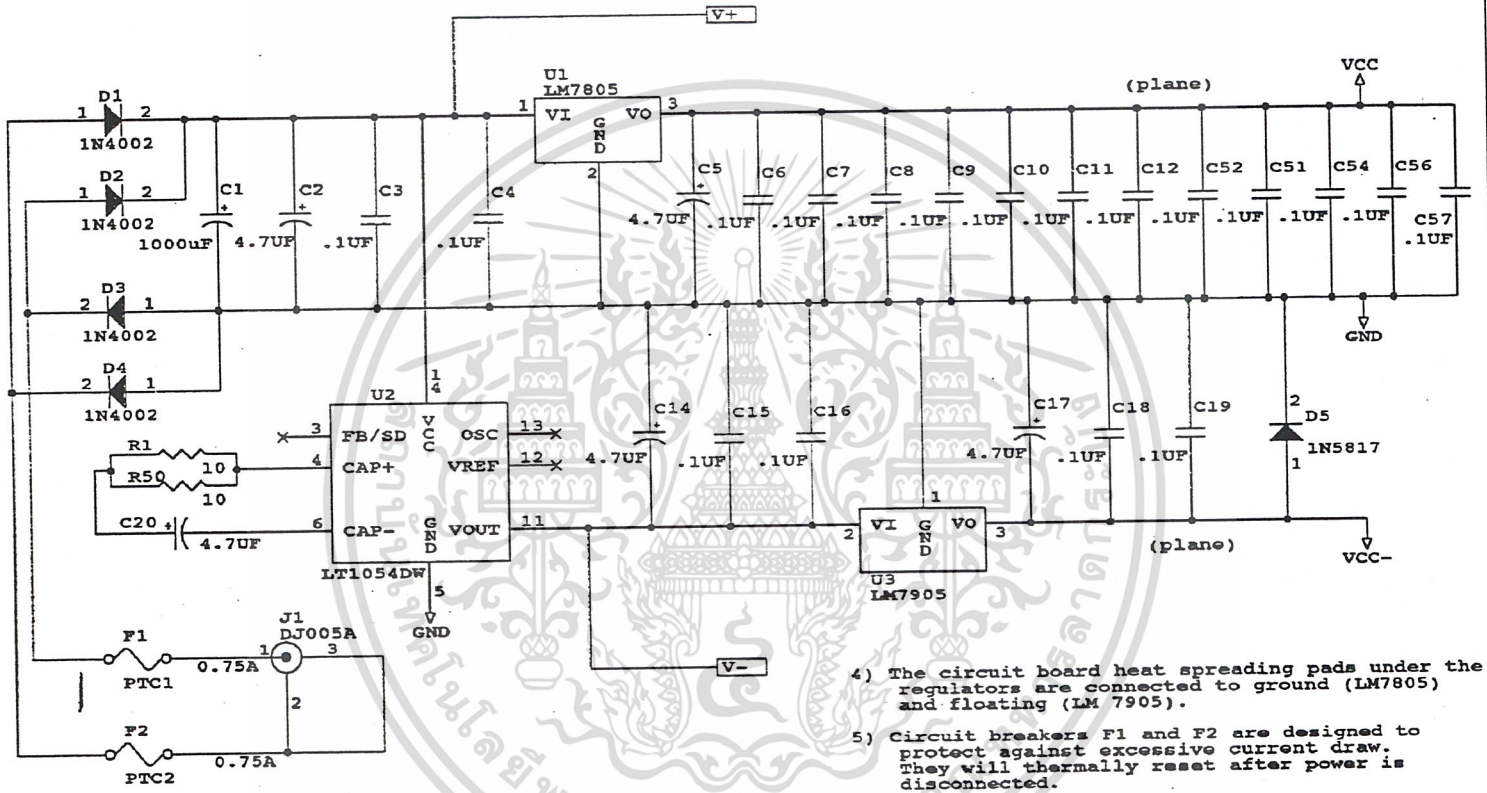
6. ส่วนฟังก์ชันการใช้งานต่างๆ สามารถดูได้ที่ บทที่ 6 ผลการทดลอง หรือ กด F9 (help) ก็จะแสดงฟังก์ชันการใช้งานต่างๆ ของโปรแกรมออกมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


24 MHz
PC
CLK
OE
CE
D0-D7 (A/D)
A0-A16 (A/D)
74HC191
74HC02
74HC86
74HC32
74HC138
74HC373
CA3318
LM1881
Title
Size
B
Number
Revision
Date
8-Nov-1991
Sheet of
File
D:\PROJECT\PAPER\CIRCUIT\CIRCUIT1.ddb
Drawn by


A0-A16 (A/D)
A0-A15 (PC)
A0-A16 (DSP)
R/W
D0-D7 (DSP)
D0-D7 (A/D)
D0-D7 (PC)
DSP
PC
OE
E3
ACE
Title
Size
B
Number
Revision
Date: 8-Nov-1991
Sheet of
File: D:\PROJECT\PAPER\CIRCUIT\CIRCUIT2.SCH
Drawn By:


5.000
TLC32040CFN
3.3500
3.2000
3.2000
2.5820
1.6560
0.7300
2.1250
3.5000
1.0750
0.0500
0.0000
–0.1500
0.0000
–0.1500
0.0000
0.0290
4.7000
4.9000



4) The circuit board heat spreading pads under the regulators are connected to ground (LM7805) and floating (LM 7905).

5) Circuit breakers F1 and F2 are designed to protect against excessive current draw. They will thermally reset after power is disconnected.

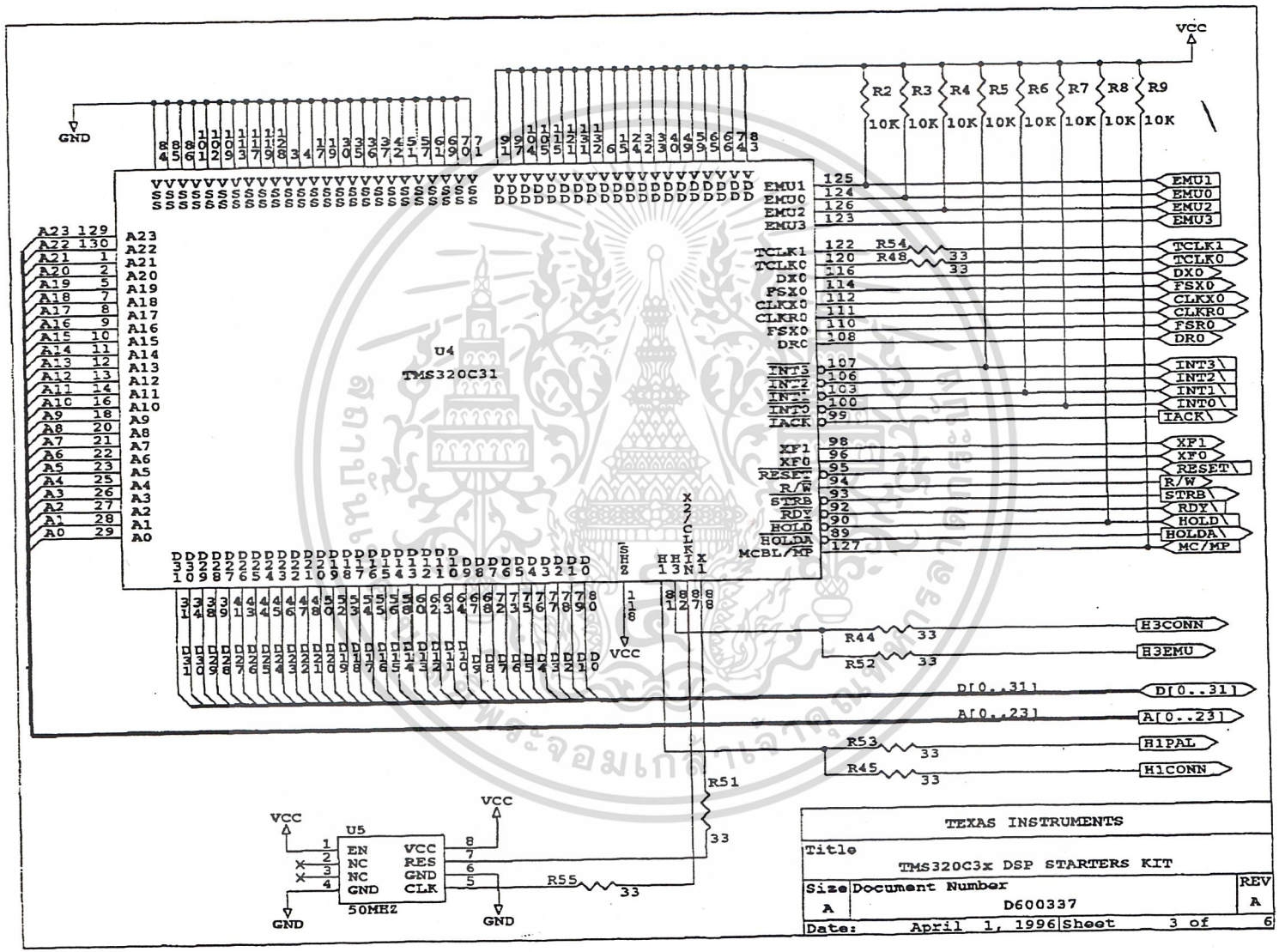
6) R44 and R45 (H1 and H3 clock signals) are not installed.

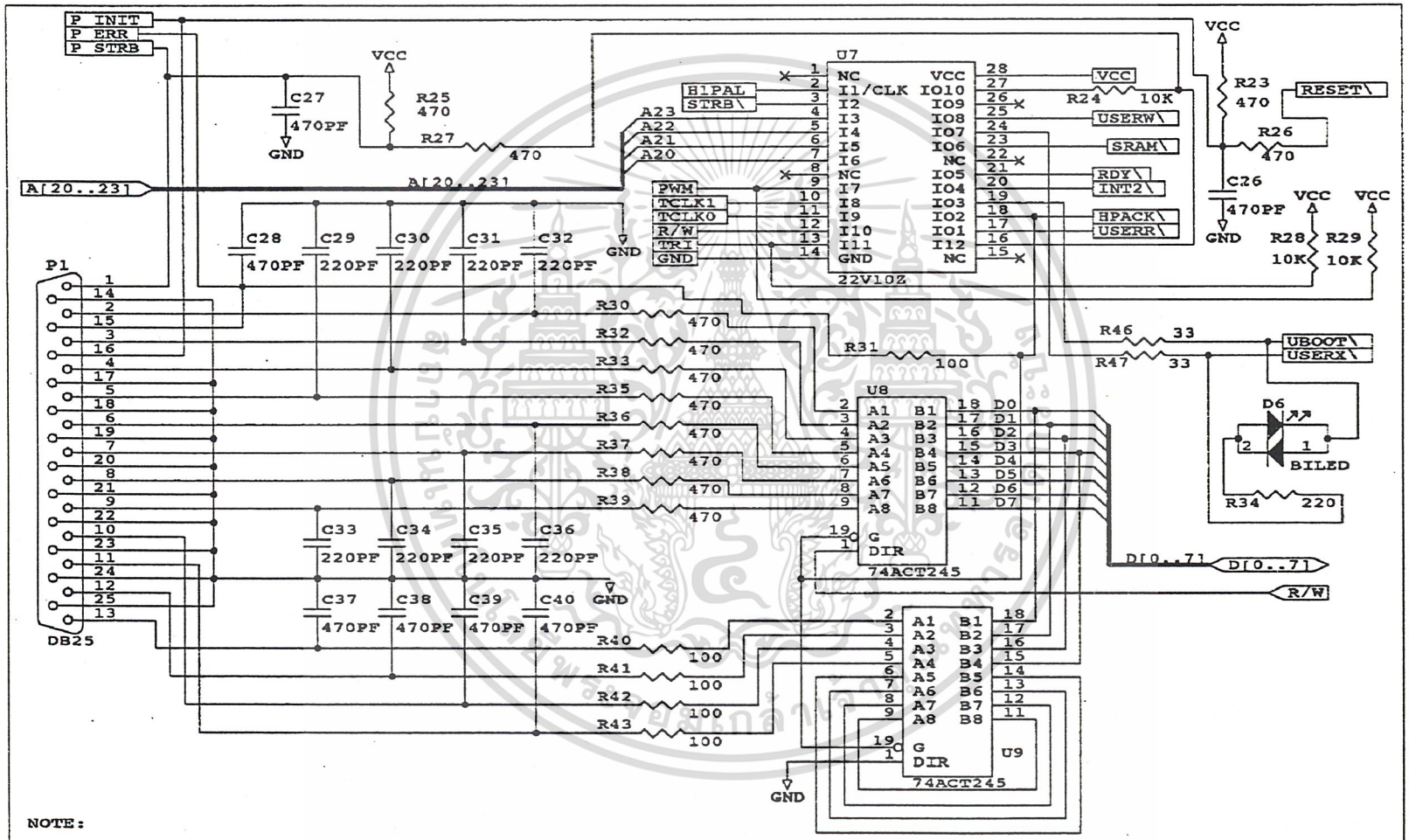
NOTES: 1) Input Power Requirements
7-12VDC or 6-9VAC >500mA, UL CLASS II ONLY

2) To minimize heating in the power supply components it is recommended that the external current on any supply should not exceed 50mA. Heating can also be minimized by using a lower supply voltage or by heatsinking the regulator.

3) The heatsink tabs of the LM7085 and LM7905 regulators are internally connected to GND and V- respectively. DO NOT ELECTRICALLY CONNECT THE HEATSINK TABS TOGETHER.

TEXAS INSTRUMENTS		
Title: TMS320C3x DSP STARTERS KIT		
Size: A	Document Number: D500337	REV: A
Date: March 26, 1996	Sheet 2 of 6	

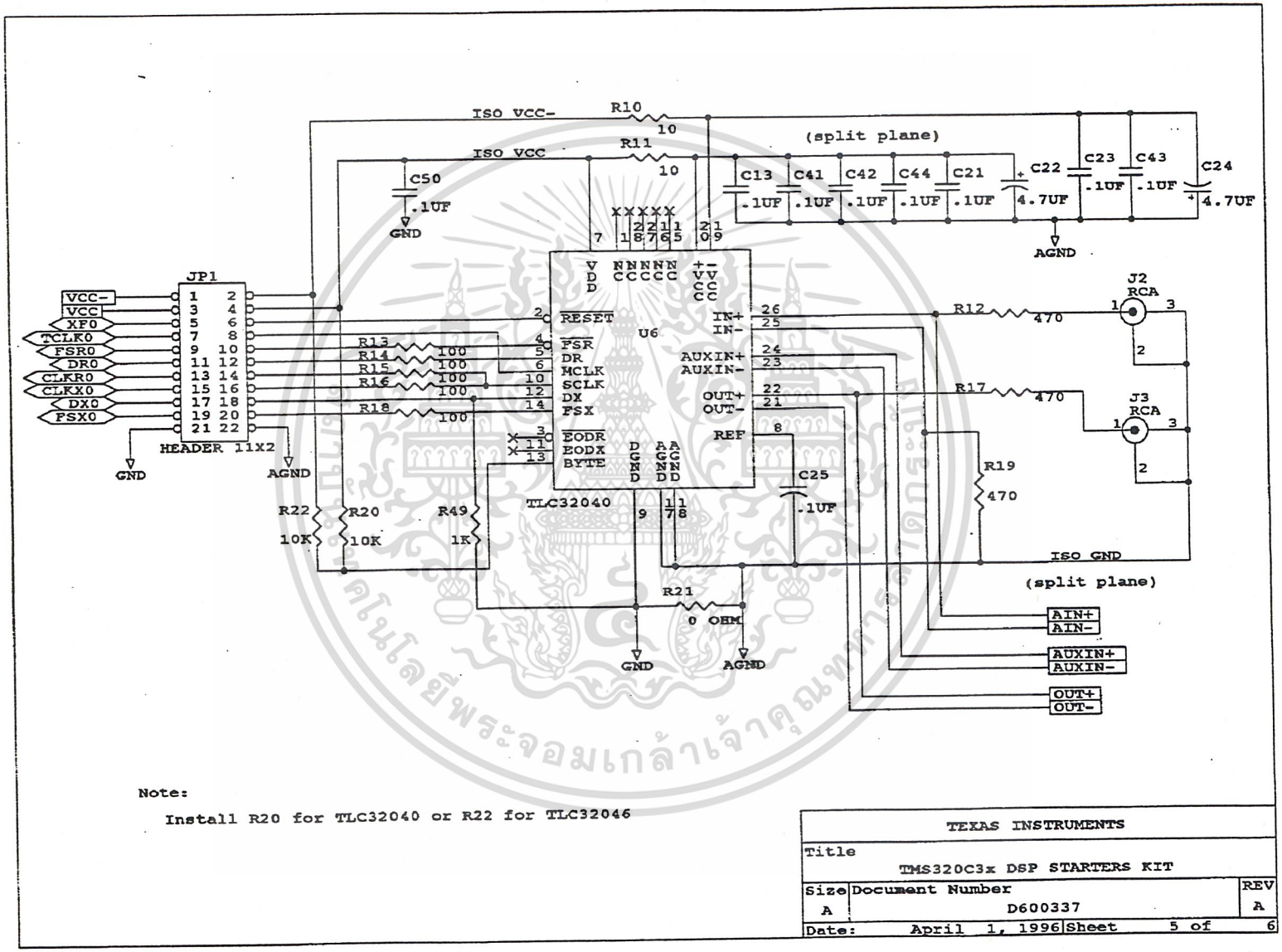

VCC
GND
R2 10K
R3 10K
R4 10K
R5 10K
R6 10K
R7 10K
R8 10K
R9 10K
U4
TMS320C31
EMU1
EMU0
EMU2
EMU3
TCLK1
TCLK0
DX0
FSX0
CLKX0
CLKR0
FSR0
DR0
INT3
INT2
INT1
INT0
IACK
XF1
XF0
RESET
R/W
STRB
RDY
HOLD
HOLDA
MC/MP
R54 33
R48 33
R44 33
R52 33
R53 33
R45 33
R51 33
R55 33
H3CONN
H3EMU
H1PAL
H1CONN
D[0..31]
A[0..23]
U5
EN
NC
NC
GND
VCC
RES
GND
CLK
50MHZ
TEXAS INSTRUMENTS
TMS320C3x DSP STARTERS KIT
Size A
Document Number D600337
REV A
Date: April 1, 1996
Sheet 3 of 6



NOTE:

By configuring TCLK0 and TCLK1 as clock outputs and driving PWM high (default) the output pins UBOOT\ and USERX\ drive the LED with a PWM signal. The PWM voltage is the phase difference between the two clocks and can be configured to be a DC output or a triangle wave (see DSK documentation).

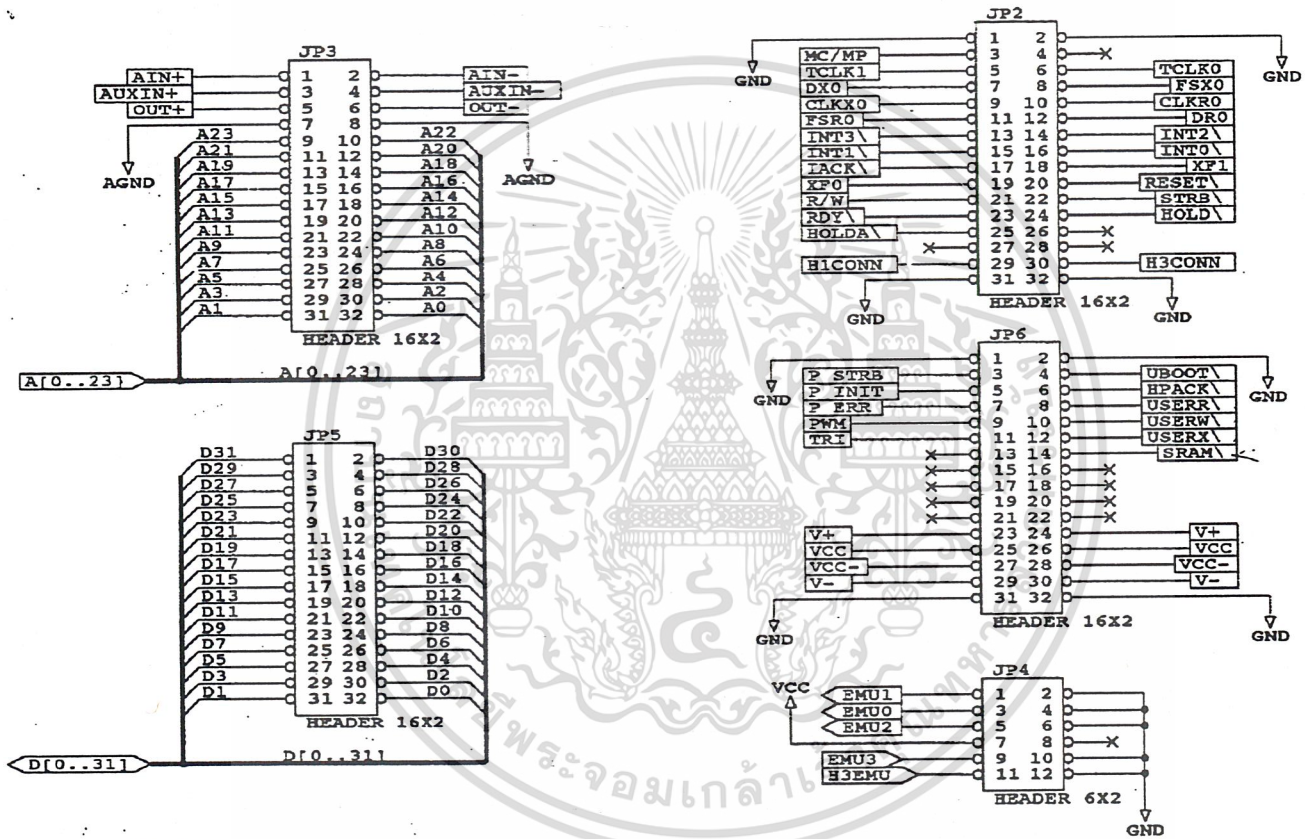
TEXAS INSTRUMENTS		
Title: TMS320C3x DSP STARTERS KIT		
Size: A	Document Number: D600337	REV: A
Date: April 1, 1996	Sheet 4 of 6	



Note:

Install R20 for TLC32040 or R22 for TLC32046

TEXAS INSTRUMENTS		
Title: TMS320C3x DSP STARTERS KIT		
Size: A	Document Number: D600337	REV: A
Date: April 1, 1996	Sheet 5 of 6	



NOTES: RESET\, RDY\, and INT2\ are driven by the host interface and should not be driven by external signals.

Most signals at the expansion headers are unbuffered. When building external circuits which will use these signals, care should be taken to meet all I/O requirements and timings to ensure proper circuit operation and avoid damage.

TEXAS INSTRUMENTS		
Title: TMS320C3x DSP STARTERS KIT		
Size: A	Document Number: 'D600337	REV: A
Date: March 21, 1996	Sheet 6 of 6	

```
//-------------------------------------------------------
// C31_PRJ.CPP
//-------------------------------------------------------
#include<stdio.h>
#include<string.h>
#include<bios.h>
#include<time.h>
#include<stdlib.h>
#include<dos.h>
#include<alloc.h>
#include<conio.h>
#include<math.h>
#include<io.h>
#include<ctype.h>
#include<graphics.h>
#include "stdfon.h"
#include "objects.h"
#include "algorith.h"
#include "keydef.h"
#include "dsk.h"
#include "dsk_coff.h"
#include "c3xmmrs.h"

char DSK_SOBEL[]="SOBEL.DSK";
char DSK_LOWPASS[]="LOWPASS.DSK";
char DSK_HIGHPASS[]="HIGHPASS.DSK";
char DSK_LAPLACIAN[]="LAPLACE.DSK";
char DSK_EDGE[]="EDGE.DSK";
char DSK_EXE[]="C31_EXE.EXE";

void main(void)
{
    ptr = (unsigned char*)MK_FP(0xd000,0);
    clrscr();
    Scan_Command_line(DSK_EXE);
    for(;;)
    {
        if(Init_Communication(10000) == NO_ERR)  //Init DSP
            break;
    }
    InitGraph(); //Init graph mode 800*600*256
    GrayLevel();
    Menu();
    CheckKey();
    CloseGraph();
    printf("\nThank You for using program.");
}
/*************************************************************************/
void Menu()
{
    Window("IMAGE PROCESSING BY TMS320C31",0,0,MAXX,MAXY,_LIGHTGRAY,_GRAYED,_WHITE);
    //menu left
    DeepRectangle(17,ystart-1,21+104,ystart+ImageLength,0);
    TextButton(28,110,86,40,VISIBLE|FOCUSED,"F1:CCD");
    TextButton(28,110+80,86,40,VISIBLE,"F2:SAVE");
    TextButton(28,110+160,86,40,VISIBLE,"F3:OPEN");
    TextButton(28,110+240,86,40,VISIBLE,"F4:CLEAR");
    TextButton(28,110+320,86,40,VISIBLE,"F5:DSP");
    //menu right
    DeepRectangle(MAXX-21-104,ystart-1,MAXX-17,ystart+ImageLength,0);
    TextButton(685,110,86,40,VISIBLE,"F6:PC");
    TextButton(685,110+80,86,40,VISIBLE,"F7:HISTO");
    TextButton(685,110+160,86,40,VISIBLE,"F8:RUN");
    TextButton(685,110+240,86,40,VISIBLE,"F9:HELP");
    TextButton(685,110+320,86,40,VISIBLE,"F10:QUIT");
```

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved
ห้ามมิให้ทำซ้ำ ดัดแปลง หรือนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น โดยไม่ได้รับอนุญาตเป็นลายลักษณ์อักษรจากมหาวิทยาลัยเชียงใหม่ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    //Image Frame 512x512
    DeepRectangle(xstart-1,ystart-1,xstart+ImageWidth,ystart+ImageLength,0);
    //STATUS BAR
    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
    DeepRectangle(4,MAXY-19-2*4,MAXX-4,MAXY-4,0);
    OutTextXY(30,MAXY-4-17,"STATUS BAR",_BLACK);
}
/*************************************************************************/
void CheckKey(void)
{
    int key=0,modifiers=0;
    for(;;)
    {
        key = bioskey(0);
        modifiers = bioskey(2);
        if(key)
        {
            switch(key)
            {
                case _SHFT_A :
                case _A  :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                            OutTextXY(30,MAXY-4-17,"Show RAM#A",_BLACK);
                            CCD();//ShowRAM(RAM_A);
                            break;

                case _SHFT_B :
                case _B  :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                            OutTextXY(30,MAXY-4-17,"Show RAM#B",_BLACK);
                            ShowRAM(RAM_B);
                            break;

                case _F1 :  outportb(0x300,8);
                            TextButton(28,110,86,40,DOWN,"F1:CCD");
                            start=clock();
                            delay(100);
                            CCD(); //Show data from A/D [RAM#A]
                            end = clock();
                            Counter();
                            TextButton(28,110,86,40,UP,"F1:CCD");
                            bar(5,MAXY-19-7,600,MAXY-5);
                            OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
                            break;

                case _F2 :  status=prompt("Enter file name :",SAVE_IMG);
                            if(status) {
                                TextButton(28,110+80,86,40,DOWN,"F2:SAVE");
                                SaveScreen();
                                TextButton(28,110+80,86,40,UP,"F2:SAVE");
                                bar(5,MAXY-19-7,600,MAXY-5);
                                OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
                            }
                            else {
                                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                                OutTextXY(30,MAXY-4-17,"SAVING NO SUCCESS!",_BLACK);
                            }
                            break;

                case _F3 :  status=prompt("Enter file name :",LOAD_IMG);
                            if(status) {
                                TextButton(28,110+160,86,40,DOWN,"F3:OPEN");
                                OpenImage();
                                TextButton(28,110+160,86,40,UP,"F3:OPEN");
                                bar(5,MAXY-19-7,600,MAXY-5);
                                OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
                            }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            else {
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"OPENNING NO SUCCESS!",_BLACK);
            }
            break;

case _F4 :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-17,"Clear Screen",_BLACK);
            TextButton(28,110+240,86,40,DOWN,"F4:CLEAR");
            clear_scr(_GRAYED);
            TextButton(28,110+240,86,40,UP,"F4:CLEAR");
            bar(5,MAXY-19-7,600,MAXY-5);
            OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
            break;

case _F5 :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-17,"DSP PROCESSING",_BLACK);
            TextButton(28,110+320,86,40,DOWN,"F5:DSP");
            start = clock();
            outportb(0x300,12);
            RUN_CPU();
            bar(300,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(360,MAXY-4-17,"[ RUNNING ]",_BLACK);
            if(ENHANCE==SOBEL) delay(4800);
            else if(ENHANCE==LOWPASS) delay(4444);
            else if(ENHANCE==HIGHPASS) delay(4444);
            else if(ENHANCE==LAPLACIAN) delay(4444);
            else if(ENHANCE==EDGE) delay(4619);
            end = clock();
            Counter();
            ShowRAM(RAM_B);
            TextButton(28,110+320,86,40,UP,"F5:DSP");
            bar(5,MAXY-19-7,600,MAXY-5);
            OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
            break;

case _F6 :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-17,"PC PROCESSING",_BLACK);
            TextButton(685,110,86,40,DOWN,"F6:PC");
            OutTextXY(360,MAXY-4-17,"[ RUNNING ]",_BLACK);
            start=clock();
            if(ENHANCE==SOBEL) PC_Sobel();
            else if(ENHANCE==LOWPASS) PC_LowPass();
            else if(ENHANCE==HIGHPASS) PC_HighPass();
            else if(ENHANCE==LAPLACIAN) PC_Laplacian();
            else if(ENHANCE==EDGE) PC_Edge();
            end = clock();
            Counter();
            TextButton(685,110,86,40,UP,"F6:PC");
            bar(5,MAXY-19-7,500,MAXY-5);
            OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
            break;

case _F7 :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-17,"Show Histogram",_BLACK);
            TextButton(685,110+80,86,40,DOWN,"F7:HISTO");
            Histogram();
            TextButton(685,110+80,86,40,UP,"F7:HISTO");
            break;

case _F8 :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-17,"CCD RUNNING",_BLACK);
            OutTextXY(320,MAXY-4-17,"[ PRESSS ESC TO BREAK ]",_BLACK);
            TextButton(685,110+160,86,40,DOWN,"F8:RUN");
            for(;;)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

```
                    {
                        outportb(0x300,8);
                        delay(100);
                        CCD();
                        if(kbhit()) if((key=bioskey(0))==_ESC) break;
                    }
                    TextButton(685,110+160,86,40,UP,"F8:RUN");
                    bar(5,MAXY-19-7,600,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
                    break;

            case _F9  :  bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"HELP",_BLACK);
                    TextButton(685,110+240,86,40,DOWN,"F9:HELP");
                    HelpMenu();
                    TextButton(685,110+240,86,40,UP,"F9:HELP");
                    bar(5,MAXY-19-7,600,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"ENTER NEW COMMAND",_BLACK);
                    break;

            case _AF4 :
            case _F10 :  CloseGraph(); exit(0);

//ZONE CTRL KEY

            case _CTRL_A : bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"Clear RAM#A",_BLACK);
                    ClearRAM(RAM_A);
                    break;

            case _CTRL_B  : bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"Clear RAM#B",_BLACK);
                    ClearRAM(RAM_B);
                    break;

            case _CTRL_D : outportb(0x300,12);
                    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"Switch to DSP",_BLACK);
                    break;

            case _CTRL_F :  status=prompt("Enter file name :",SAVE_IMG);
                    if(status) {
                        start=clock();
                        SaveFullScreen();
                        end = clock();
                        Counter();
                    }
                    else {
                        bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                        OutTextXY(30,MAXY-4-17,"SAVING FULL SCREEN NO
                SUCCESS!",_BLACK);
                    }
                    break;

            case _CTRL_I  : bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"INVERT",_BLACK);
                    start=clock();
                    Invert();
                    end = clock();
                    Counter();
                    break;
            case _CTRL_O  : status=prompt("Enter file name :",LOAD_IMG);
                    if(status) {
                        TextButton(28,110+160,86,40,DOWN,"OPEN");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    OpenImage();
                    TextButton(28,110+160,86,40,UP,"F3:OPEN");
                }
                else {
                    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"OPENNING NO
            SUCCESS!",_BLACK);
                }
                break;

case _CTRL_R : bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                prompt("Enter brightness (integer only) :",BRIGHT);
                start=clock();
                Bright();
                end = clock();
                Counter();
                break;

case _CTRL_S : status=prompt("Enter file name :",SAVE_IMG);
                if(status) {
                    TextButton(28,110+80,86,40,DOWN,"SAVE");
                    SaveScreen();
                    TextButton(28,110+80,86,40,UP,"F2:SAVE");
                }
                else {
                    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                    OutTextXY(30,MAXY-4-17,"SAVING NO SUCCESS!",_BLACK);
                }
                break;

case _CF1 :     ENHANCE=SOBEL;
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"SOBEL ENHANCEMENT",_BLACK);
                LoadDSK(DSK_SOBEL);
                break;

case _CF2 :     ENHANCE=LOWPASS;
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"LOWPASS ENHANCEMENT",_BLACK);
                LoadDSK(DSK_LOWPASS);
                break;

case _CF3 :     ENHANCE=HIGHPASS;
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"HIGHPASS ENHANCEMENT",_BLACK);
                LoadDSK(DSK_HIGHPASS);
                break;

case _CF4 :     ENHANCE=LAPLACIAN;
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"LAPLACIAN
        ENHANCEMENT",_BLACK);
                LoadDSK(DSK_LAPLACIAN);
                break;

case _CF5 :     ENHANCE=EDGE;
                bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"EDGE ENHANCEMENT",_BLACK);
                LoadDSK(DSK_EDGE);
                break;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

```
//ZONE ALT KEY
```

แสวงหากำไรในทุกๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
case _ALT_A :   bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                OutTextXY(30,MAXY-4-17,"Load Screen to RAM#A",_BLACK);
                ScreenToRAM(RAM_A);
```

```
                                break;

                        case _ALT_B :   bar(5,MAXY-19-7,MAXX-5,MAXY-5);
                                        OutTextXY(30,MAXY-4-17,"Load Screen to RAM#B",_BLACK);
                                        ScreenToRAM(RAM_B);
                                        break;

                        default  :      break;
                    }
                }
            }
}
/*************************************************************************/
void OpenImage(void)
{
    for(Y=0; Y<(ImageLength); Y++)
        for(X=0; X<ImageWidth; X++)
        {
            color=getc(LoadImg);
            putpixel(X+xstart,Y+ystart,color);
        }
    fclose(LoadImg);
}
/*************************************************************************/
void ScreenToRAM(int RAM)
{
    if(RAM==RAM_A)  {
        for(ii=0;ii<4;ii++)
        {
            jj=0;
            outportb(0x300,ii);
            for(Y=0; Y<(ImageLength/2); Y+=2)
                for(X=0; X<ImageWidth; X++)
                {
                    switch(ii){
                        case 0 : *(ptr+jj)=getpixel(X+xstart,Y+ystart+1); break;
                        case 1 : *(ptr+jj)=getpixel(X+xstart,Y+ystart+256+1);break;
                        case 2 : *(ptr+jj)=getpixel(X+xstart,Y+ystart); break;
                        case 3 : *(ptr+jj)=getpixel(X+xstart,Y+ystart+256); break;
                    }
                    jj++;
                }
        }
    }
    else {
        ii=0;
        for(Y=0; Y<ImageLength; Y++)
            for(X=0; X<ImageWidth; X++)
            {
                if(Y<ImageLength/4)
                {
                    outportb(0x300,4);
                    *(ptr+ii)=getpixel(X+xstart,Y+ystart);
                    ii++;
                }
                else if(Y<ImageLength/2)
                {
                    outportb(0x300,5);
                    *(ptr+ii)=getpixel(X+xstart,Y+ystart);
                    ii++;
                }
                else if(Y<((ImageLength*3)/4))
                {
                    outportb(0x300,6);
                    *(ptr+ii)=getpixel(X+xstart,Y+ystart);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    ii++;
                }
                else if(Y<ImageLength)
                {
                    outportb(0x300,7);
                    *(ptr+ii)=getpixel(X+xstart,Y+ystart);
                    ii++;
                }
            }
    }
}
/*************************************************************************/
void ShowRAM(int RAM) //show_sobel
{
    if(RAM==RAM_A) CCD();
    else {
        ii=0;
        for(Y=0; Y<ImageLength; Y++)
            for(X=0; X<ImageWidth; X++)
            {
                if(Y<ImageLength/4)
                {
                    outportb(0x300,4);
                    color=(unsigned char)*(ptr+ii);
                    putpixel(X+xstart,Y+ystart,color);
                    ii++;
                }
                else if(Y<ImageLength/2)
                {
                    outportb(0x300,5);
                    color=(unsigned char)*(ptr+ii);
                    putpixel(X+xstart,Y+ystart,color);
                    ii++;
                }
                else if(Y<((ImageLength*3)/4))
                {
                    outportb(0x300,6);
                    color=(unsigned char)*(ptr+ii);
                    putpixel(X+xstart,Y+ystart,color);
                    ii++;
                }
                else if(Y<ImageLength)
                {
                    outportb(0x300,7);
                    color=(unsigned char)*(ptr+ii);
                    putpixel(X+xstart,Y+ystart,color);
                    ii++;
                }
            }
    }
}
/*************************************************************************/
void clear_scr(int bkcolor)
{
    setfillstyle(SOLID_FILL,bkcolor);
    bar(xstart,ystart,ImageWidth-1+xstart,ImageLength-1+ystart);
}
/*************************************************************************/
void ClearRAM(int RAM)
{
    int S;
    if(RAM==RAM_A) S=0;
    else S=4;
    ii=0;
    for(Y=(0+S);Y<(4+S);Y++)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
        ii=0;
        outportb(0x300,Y);
        for(jj=0;jj<65536;jj++)
        {
            *(ptr+ii)=0x77;
            ii++;
        }
    }
}
/*****************************************************************************/
void SaveScreen(void)
{
    for(Y=0;Y<ImageLength; Y++)
        for(X=0;X<ImageWidth;X++)
        {
            color=getpixel(X+xstart,Y+ystart);
            putc(color,SaveImg);
        }
    fclose(SaveImg);
    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
    OutTextXY(30,MAXY-4-17,"SAVING SUCCESS!",_BLACK);
}
/*****************************************************************************/
void SaveFullScreen(void)
{
    for(Y=0;Y<599; Y++)
        for(X=0;X<799;X++)
        {
            color=getpixel(X,Y);
            putc(color,SaveImg);
        }
    fclose(SaveImg);
    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
    OutTextXY(30,MAXY-4-17,"SAVING FULL SCREEN SUCCESS!",_BLACK);
}
/*****************************************************************************/
int prompt(char* command,int method)
{
    char f[2];
    int k,i,x,y;
    int cmd_length;
    //STATUS BAR
    setcolor(_DARKGRAY);
    bar(5,MAXY-19-7,MAXX-5,MAXY-5);
    OutTextXY(30,MAXY-4-18,command,_BLACK);
    cmd_length=strlen(command);
    k=8*cmd_length+30;
    i=0;
    x=k;
    while(1)
    {
        f[0]=(char)getch();
        if(f[0]==27) return(0);
        f[1]='\0';
        if(f[0]==13) break;
        String[i]=f[0];
        if((f[0]==0)||(f[0]==8))
        {
            if(f[0]==0)
                if(getch()!=75) continue;
            i--;
            k-=8;
            x=k;
            del_chr(x,MAXY-26);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            continue;
        }
        i++;
        k+=8;
        OutTextXY(k,MAXY-4-18,f,_BLACK);
    }
    String[i]='\0';
    if(method==SAVE_IMG)
    {
        if((SaveImg=fopen(String,"wb"))!=NULL)
        {
            bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-18,"Saving",_BLACK);
            return(1);
        }
    }
    else if(method==LOAD_IMG){
        if((LoadImg=fopen(String,"rb"))!=NULL)
        {
            //STATUS BAR
            bar(5,MAXY-19-7,MAXX-5,MAXY-5);
            OutTextXY(30,MAXY-4-18,"Loading",_BLACK);
            return(1);
        }
    }
    return(0);
}
/*************************************************************************/
void del_chr(int x,int y)
{
    setcolor(_LIGHTGRAY);
    bar(x,y,x+8,MAXY-5);
    setcolor(_BLACK);
}
/*************************************************************************/
void CCD()
{
    for(ii=0;ii<4;ii++)
    {
        jj=0;
        outportb(0x300,ii);
        for(Y=0; Y<(ImageLength/2); Y+=2)
            for(X=0; X<ImageWidth; X++)
            {
                color=(unsigned char)*(ptr+jj);
                switch(ii){
                    case 0 :  putpixel(X+xstart,Y+ystart+1,color); break;
                    case 1 : putpixel(X+xstart,Y+ystart+256+1,color); //break;
                    case 2 : putpixel(X+xstart,Y+ystart,color); break;
                    case 3 : putpixel(X+xstart,Y+ystart+256,color); break;
                }
                jj++;
            }
    }
}
/*************************************************************************/
void Counter()
{
    gcvt((end-start)/CLK_TCK,10,count);
    strcpy(str,"Time is : ");
    strcat(str,count);
    bar(600,MAXY-19-7,MAXX-5,MAXY-5);
    OutTextXY(610,MAXY-4-17,str,_BLACK);
}
/*************************************************************************/
```

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void HelpMenu()
{
    int lines=15;
    clear_scr(_BLACK);
    setcolor(_WHITE);
    OutTextXY(xstart+220,ystart+lines, "[ MENU ]",_WHITE);
    outtextxy(xstart+120,ystart+lines*4, "F1         : CCD");
    outtextxy(xstart+120,ystart+lines*5, "F2, CTRL S  : SAVE IMAGE");
    outtextxy(xstart+120,ystart+lines*6, "F3, CTRL O  : OPEN IMAGE");
    outtextxy(xstart+120,ystart+lines*7, "F4         : CLEAR SCREEN");
    outtextxy(xstart+120,ystart+lines*8, "F5         : DSP PROCESSING");
    outtextxy(xstart+120,ystart+lines*9, "F6         : PC PROCESSING");
    outtextxy(xstart+120,ystart+lines*10, "F7         : HISTOGRAM");
    outtextxy(xstart+120,ystart+lines*11,"F8         : RUN CCD");
    outtextxy(xstart+120,ystart+lines*12,"F9         : HELP");
    outtextxy(xstart+120,ystart+lines*13,"F10, ALT F4 : QUIT");

    outtextxy(xstart+120,ystart+lines*15,"CTRL F1    : SOBEL ENHANCEMENT");
    outtextxy(xstart+120,ystart+lines*16,"CTRL F2    : LOWPASS ENHANCEMENT");
    outtextxy(xstart+120,ystart+lines*17,"CTRL F3    : HIGHPASS ENHANCEMENT");
    outtextxy(xstart+120,ystart+lines*18,"CTRL F4    : LAPLACIAN ENHANCEMENT");
    outtextxy(xstart+120,ystart+lines*19,"CTRL F5    : EDGE ENHANCEMENT");

    outtextxy(xstart+120,ystart+lines*21,"A          : SHOW RAM#A");
    outtextxy(xstart+120,ystart+lines*22,"B          : SHOW RAM#B");
    outtextxy(xstart+120,ystart+lines*23,"CTRL A     : CLEAR RAM#A");
    outtextxy(xstart+120,ystart+lines*24,"CTRL B     : CLEAR RAM#B");
    outtextxy(xstart+120,ystart+lines*25,"ALT A      : LOAD SCREEN TO RAM#A");
    outtextxy(xstart+120,ystart+lines*26,"ALT B      : LOAD SCREEN TO RAM#B");

    outtextxy(xstart+120,ystart+lines*28,"CTRL I     : INVERT IMAGE");
    outtextxy(xstart+120,ystart+lines*29,"CTRL R     : BRIGHTNESS IMAGE");
    outtextxy(xstart+120,ystart+lines*30,"CTRL D     : SWITCH TO DSP");
    outtextxy(xstart+120,ystart+lines*31,"CTRL F     : SAVE FULL SCREEN");
}
/*****************************************************************/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้