การประยุกต์ Card DSP

(การตัดตัวอักษรภาษาไทยออกจากหน้ากระดาษ)

Application of DSP Card

(Cutting Thai Character from paper page)

โดย

นาย ณัฐพงศ์ กมลรัตน์ 34102113

นาย สมนึก แซ่หยี่ 34107393

นาย สมโภชน์ อุทัยวิวัฒน์กุล 34107397

อาจารย์ที่ปรึกษา

อ. เทอดศักดิ์ ลิ่วหาทอง

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมอิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2537

รายงานโครงงานวิจัย ภาคเรียนที่ 2 ปีการศึกษา 2537

เรื่อง การประยุกต์ Card DSP ในการตัดตัวอักษรภาษาไทยออกจากหน้ากระดาษ

จัดทำโดย

นาย ณัฐพงศ์ กมลรัตน์ 34102113

นาย สมนึก แซ่หยี่ 34107393

นาย สมโภชน์ อุทัยวิวัฒน์กุล 34107397

ภาควิชาอิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(อ.เทอดศักดิ์ ลิ่วหาทอง)

อาจารย์ที่ปรึกษา

การประยุกต์ CARD DSP: เรื่องการตัดตัวอักษรภาษาไทยออกจากหน้ากระดาษ

(Application of DSP Card: Cutting Thai Character from paper page)

โดย นาย ณัฐพงษ์ กมลรัตน์ 34102113
นาย สมนึก แซ่หยี่ 34107393
นาย สมโภชน์ อุทัยวิวัฒน์กุล 34107397
อาจารย์ที่ปรึกษา อ.เทอดศักดิ์ ลิ่วหาทอง

บทคัดย่อ

ในปริญญานิพนธ์เล่มนี้ เป็นการประยุกต์ใช้งาน Card ที่สร้างขึ้นคือใช้ Microprocess ทางด้าน Digital Signal Processing เบอร์ TMS320c25 เป็นตัวประมวลผลโดยติดต่อกับ Computer ซึ่งการประยุกต์ใช้งานของ Card นี้ จะเป็นเรื่องเกี่ยวกับ "การแยกตัวอักษรภาษาไทยออกจากหน้ากระดาษที่ Scan เข้ามา " output ที่ได้จากการประมวลผลโดย Card คือ ตัวอักษรไทยที่แยกออกจากกัน (ตัวอักษรแต่ละตัวที่แยกออกจากกันนี้สามารถนำมาวิเคราะห์ เพื่อแยกแยะว่าเป็นตัวอักษรตัวใด แต่วิธีการวิเคราะห์ไม่ได้รวมอยู่ในวัตถุประสงค์ของวิทยานิพนธ์ฉบับนี้) ซึ่งการประมวลผลใน Card นี้จะช่วยแบ่งภาระในการทำงานของ CPU ของ Computer เพราะการทำงานของ TMS320c25จะมีความเร็วในการทำงานสูงทำให้ช่วยประหยัดเวลาในการทำงานไปได้มาก วิธีการส่งที่จะไปให้ Card ประมวลผลนั้นจะส่งผ่านโดยใช้หลักของMap Memory ซึ่งเหมือนกับการอ้าง Address ธรรมดาการติดต่อโดยวิธีนี้จะสะดวกและส่งข้อมูลได้เร็วกว่าการส่งข้อมูลผ่านทาง Port

ABSTRACT

This Thesis talk about the application of the DSP Card (TMS320c25)that connect to IBM PC by slot. The application of this card is " Cutting Thai character from paper page " the output of this process is divided characters from paper page (these

characters from this process are recognized to know that what is ASCII Code of these characters but this process dosen't aim of this thesis).This card helps IBM PC to the process of Cutting Thai Character from paper page because CPU TMS320c25 is high speed CPU to save time for processing. The method of transfer data between card and IBM PC by point to normal address, this process is conventional and high speed than transfer data by port.

สารบัญ

บทที่ 1
บทนำ

ในปริญญานิพนธ์เล่มนี้เป็นเรื่องเกี่ยวกับการประยุกต์ใช้งาน Card DSP ที่มี CPU TMS320C25 เป็นตัวประมวลผล ซึ่งการประยุกต์ใช้งาน Card นี้จะเป็นเรื่องเกี่ยวกับ "การแยกตัวอักษรภาษาไทยออกจากหน้ากระดาษที่ Scan เข้ามา " (วิธีการแยกตัวอักษรแต่ละตัวออกจากกันนี้จะมีประโยชน์ในด้านที่จะนำไปวิเคราะห์ เพื่อที่จะได้ทราบอักษรที่เราแยกได้นั้นเป็นตัวอะไร แต่ในปริญญานิพนธ์เล่มนี้จะไม่กล่าวถึงวิธีการวิเคราะห์ดังกล่าว) รายละเอียดก่อนที่จะทำการแยกตัวอักษรนั้นมีรายละเอียดย่อยๆดังต่อไปนี้

1.1 การเตรียมภาพตัวอักษร ภาพตัวอักษรโดยจะเตรียมได้จากเครื่อง Scanner ซึ่งภาพที่ได้จากเครื่อง Scanner นี้จะเก็บอยู่ในรูปของ Bit-Map File โดย File ของภาพที่ได้จากเครื่อง Scanner นี้จะมีส่วนที่เรียกว่า Header ทำหน้าที่ในการเก็บรายละเอียดของภาพที่ Scan เข้ามาเช่นภาพที่ Scan มามีขนาดกว้างเท่าไร ,ยาวเท่าไร ,ขนาดของข้อมูลภาพจริงๆเท่าไร, ขนาดของ Header ของภาพมีขนาดเท่าไรเป็นต้น รูปแบบของ Header จะกล่าวในภายหลัง

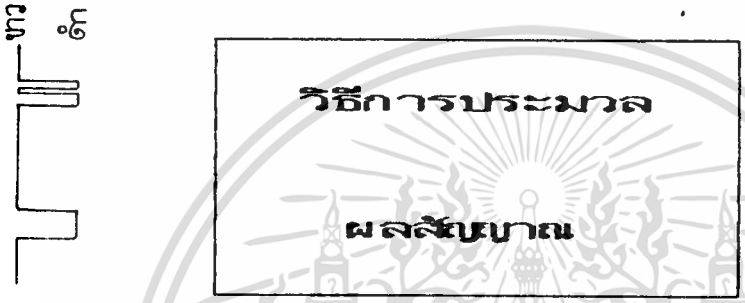
1.2 การลดขนาดของภาพ ขั้นตอนนี้เป็นการลดขนาดกว้าง-ยาวของภาพลง เพื่อให้ขนาดของภาพเหมาะสมที่จะนำมาประมวลผล ในปริญญานิพนธ์เล่มนี้ขนาดของภาพที่เหมาะสมที่เลือกใช้มีขนาด 128 จุด * 128 จุด(เหมาะสมกับขนาด RAM ที่ใช้บน Card)

1.3 ทำการ Filter ภาพขนิดของการ Filter ที่ใช้จะเป็น High Pass Filter เพื่อจะทำให้ File ภาพตัวอักษรที่ได้มีความคมชัดยิ่งขึ้น โดยวิธีของการ Filter จะแสดงรายละเอียดในภายหลัง

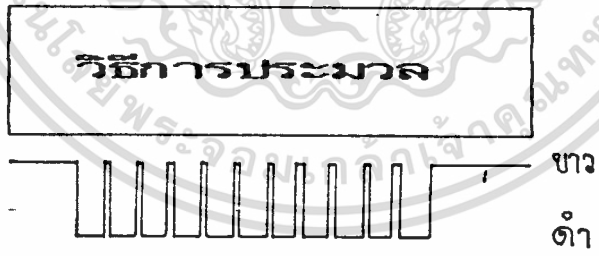
1.4 การทำ Threshold เหตุผลที่ทำ Threshold นี้เพื่อที่จะใช้แยกสีของตัวอักษรให้แตกต่างจากสีของพื้นออกอย่างเด่นชัด โดยใช้หลักการนับจำนวนจุดของสีของสีแต่ละสีบนภาพว่าแต่ละสีมีกี่จุดแล้วก็ใช้สีค่าหนึ่งเป็นตัวแบ่งว่า สีที่มีความเข้มมากกว่าสีดังกล่าวให้มีค่าเป็น 1 และสีที่มีความเข้มต่ำกว่าให้มีค่าเป็น 0

จากรายละเอียดที่ผ่านมา เป็นขั้นตอนก่อนการแยกตัวอักษร ขั้นตอนของการแยกตัวอักษรนั้นมี Algorithm ดังนี้

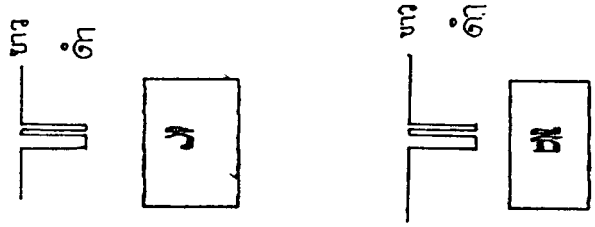
1) นำ File ภาพที่ได้จากการ Scan (และผ่านขั้นตอนต่างๆดังที่ได้กล่าวมาแล้ว) มาแยกแต่ละบรรทัดให้ออกจากกันก่อน โดยใช้หลักการของ Histogram กล่าวคือ ถ้าบรรทัดใดของภาพมีสีดำ (ถ้าภาพมีสีดำ พื้นสีขาว) นั่นหมายความว่า บรรทัดนั้นมีตัวอักษรอยู่ ในทำนองเดียวกัน ถ้าแต่ละบรรทัดมีแต่สีขาว ก็หมายความว่า บรรทัดนั้นไม่มีตัวอักษรอยู่เลย ดังนั้น เราก็ใช้หลักการดังกล่าวแยกตัวอักษรแต่ละบรรทัดออกจากกันได้ ดังรูป



2) นำบรรทัดใดบรรทัดหนึ่งที่ได้จากการแยกแต่ละบรรทัดของขั้นตอนที่ 1 นำมาแยกตัวอักษรแต่ละตัวให้ออกจากกัน ดังรูป โดยวิธีในการแยกนั้นใช้วิธีเดียวกับการแยกในขั้นตอนที่ 1



3) นำตัวอักษรที่ได้จากขั้นตอนที่ 2 มาทำการแยกตัวอักษรในแนวบรรทัดอีกครั้ง ซึ่งขั้นตอนนี้เป็นการแยกตัวอักษรให้ออกจากกันโดยสิ้นเชิง



บทที่ 2

TMS320C25

TMS320C25 เป็น CPU ที่ปรับปรุงขึ้นจากรุ่นเดิมคือ TMS320C20 จึงทำให้ชีดความสามารถของ TMS320C25 นี้เพิ่มขึ้นอีกมากคือ

- คำสั่งแต่ละคำสั่งที่ทำงานจะใช้เพียง 100 nS และคำสั่งส่วนใหญ่จะใช้เพียงหนึ่งCycle เท่านั้น ดังนั้น TMS320C25 จึงมีความเร็วในการทำงานได้สูงกว่า 10 ล้านคำสั่งในเวลา 1 วินาที

- มีคำสั่งที่จะโอนย้ายข้อมูล ระหว่างหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้

- ในการทำงานแต่ละคำสั่ง TMS320C25 จะทำงาน 3 จังหวะแบบ pipeline คือ FETCH,DECODE,EXECUTE โดยการทำงานนี้จะไม่สามารถเห็นได้โดยผู้ใช้งาน เพราะเป็นการทำงานที่เกิดขึ้นพร้อมกันเพื่อเพิ่มความเร็ว

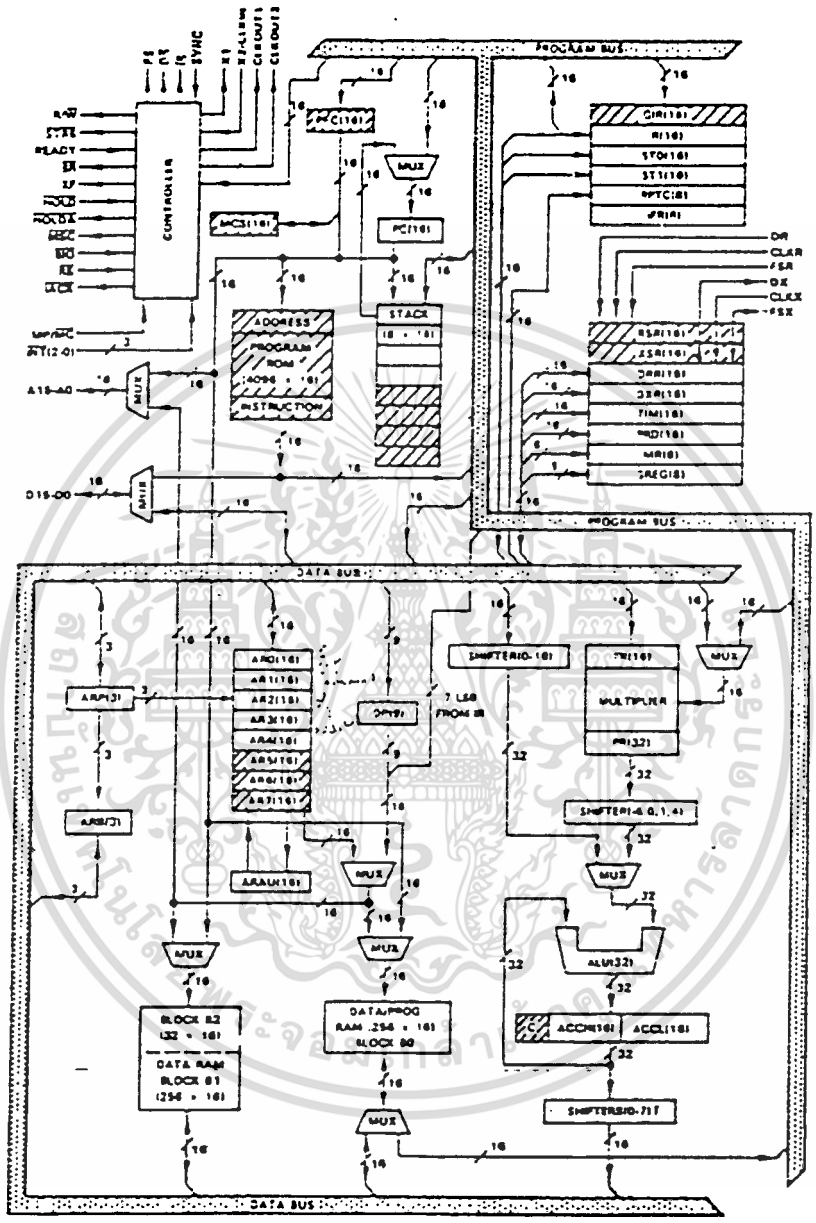
- ได้เพิ่มจำนวนหน่วยความจำขึ้นอีก โดยมี RAM ภายในถึง 544 Words(1 Word มี 16 bit) หน่วยความจำภายนอกที่ใช้สามารถต่อได้ถึง 64K Wordsซึ่งมีจำนวนสูงมาก จึงเป็น DSP ที่มีการต่อหน่วยความจำภายนอกได้สูง

- มี ROM ให้ใช้ความผู้สั่งทำอยู่บนชิพอีก 4K Words การทำงานของชิพนี้ให้ข้อเด่นอีกข้อคือ สามารถLoadโปรแกรมจากหน่วยความจำภายนอกเข้าไปเก็บใน RAM ภายใน เพื่อให้การทำงานทำได้เร็วยิ่งขึ้น

- ชีดความสามารถที่เพิ่มเติมมาอีกประการหนึ่งคือ การมี HARDWARE TIMER และชีดความสามารถในการโอนย้ายข้อมูลเป็น Block

Diagramของ TMS320C25 แสดงได้ดังรูปที่ 2.1

สถาปัตยกรรมของ TMS320C25 สร้างขึ้นมาเพื่อความเร็วในการทำงานและโครงสร้างที่ทำงานได้ด้วยชีดความสามารถที่สูงขึ้น และเพื่อให้การทำงานของ Busไม่ขึ้นต่อกัน จึงแยกBusเป็น Busของโปรแกรมและ Busของข้อมูลจากกัน โดย Busของโปรแกรมจะเป็นทาง เข้าออกรหัสคำสั่ง และOperandของคำสั่ง ส่วนBusข้อมูลจะเชื่อมต่อโดยตรงกับวงจรการทำงานการประมวลผล เช่น CALU (Central Arithmetic Login Unit) และRegisterที่เป็นFileข้อมูลย่อย AR0-AR7 และยังมี ARAU ซึ่งเป็น Auxiliary Register Arithmetic Unit การทำโครงสร้างการคำนวณทางคณิตศาสตร์นี้ยึดหลักการให้ทำงานด้วยประสิทธิภาพ เช่น การเลื่อนbit(shift) การคูณหารหรือคำสั่งทางลอจิก



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

รูปที่ 2.1 โครงสร้างของ TMS320C25

2.1 การจัดหน่วยความจำ

TMS320C25 มีการแบ่งหน่วยความจำเป็นส่วน program และ data ตามรูปที่2.2

2.1.1 data memory

Block B0 256 Words(1 Word มี 16 bit) B1 256 Words B2 32 Wordsเป็นหน่วยความจำที่อยู่ในชิป (on-chip Ram) ซึ่งสามารถทำงานได้ด้วยความเร็วเต็มที่ โดยสามารถโปรแกรมให้เป็น Data Memory หรือ Program Memory ได้โดยคำสั่ง CNFD หรือ CNFP รูปที่ 2.2a แสดง Memory Map หลังจากใช้คำสั่ง CNFD ส่วนรูปที่ 2.2b แสดง Memory Map หลังจากใช้คำสั่ง CNFP ส่วนBlock B1 และ B2 จะเป็น Data Memory เสมอ

TMS320C25 สามารถอ้างอิงหน่วยความจำข้อมูลได้ทั้งหมด 64 Kwords หากเราใช้หน่วยความจำข้อมูลบนชิปด้วย ตำแหน่งของมันจะถูก map ลงในตำแหน่งที่ต่ำกว่า 1 Kwords ของ data memory space และหน่วยความจำข้อมูลสามารถขยายเพิ่มขึ้นโดยตรงจนมีขนาด 64 Kwords ขณะที่ยังคงทำงานด้วยความเร็วเต็มที่

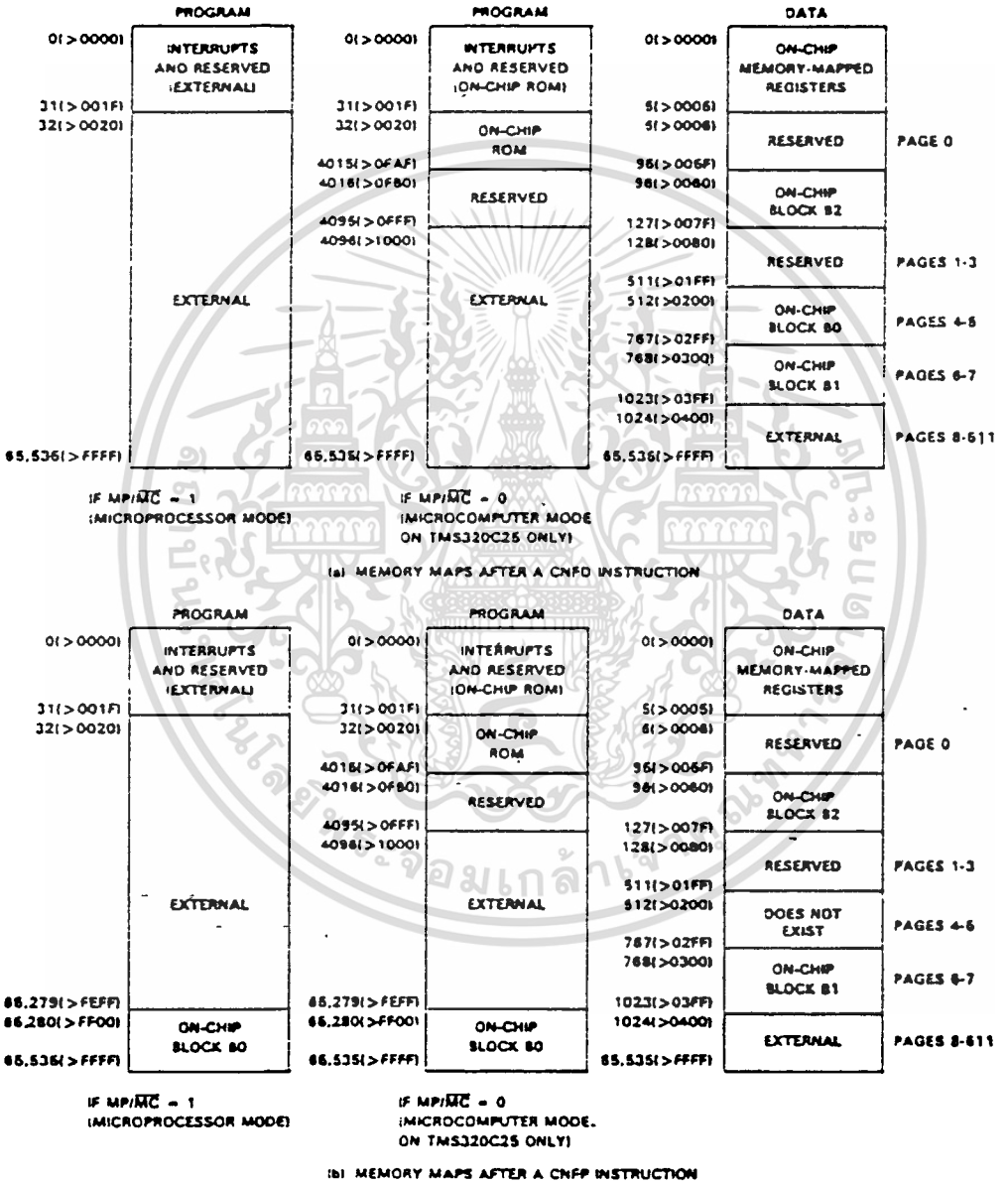
2.1.2 program memory

โปรแกรม RAM และ ROM บนชิปหรือหน่วยความจำภายนอกความเร็วสูงสามารถถูกใช้ด้วยความเร็วเต็มที่โดยไม่มีสภาวะรอ (wait state) หรือมิฉะนั้นขา READY ก็สามารถติดต่อกับ TMS320C25 ให้ช้าลงหากใช้หน่วยความจำภายนอกที่ทำงานช้า TMS320C25 สามารถมีหน่วยความจำโปรแกรมได้ทั้งหมด 64 Kwords โดย RAM ภายใน (block B0) สามารถกำหนดให้เป็นหน่วยความจำโปรแกรมได้ โดยใช้ software ซึ่งการ Executeจาก block B0 สามารถเริ่มได้หลังจาก Memory space ถูกกำหนดใหม่

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

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

ภายใน TMS320C25 ยังมี Register file 8 ตัว คือ AR0- AR7 ไว้สำหรับการทำการอ้าง Address โดยอ้อม หรือเป็นที่เก็บข้อมูลชั่วคราว Register 8 ตัวนี้สามารถเรียกใช้โดยตรงจากคำสั่งหรือจากการอ้างอิงเพียง 3 bit จาก ARP (Auxiliary Register Pointer) ARP นี้รับข้อมูลได้โดยตรงจากหน่วยความจำข้อมูลหรือมาจาก Operandของคำสั่ง



รูปที่ 2.2 การจัดหน่วยความจำของ TMS320C25

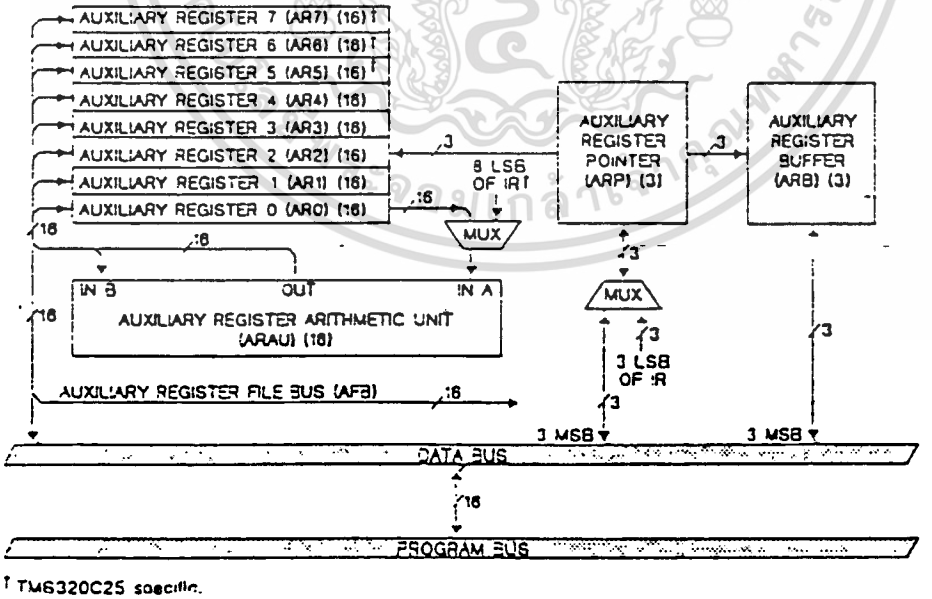
Register fileนี้ต่อโดยตรงเข้ากับ ARAU ดังแสดงในรูปที่ 2.3 ARAU สามารถทำการเพิ่มค่าหรือลดค่าเพื่อทำการ Auto index ซึ่งทำให้การอ้างAddressเพื่อนำข้อมูลจากตารางทำได้ง่ายและรวดเร็ว ดังนั้น ARAU จึงมีหน้าที่สำคัญสำหรับการสนับสนุน การอ้างAddressให้ได้ขีดความสามารถสูงขึ้น โครงสร้างของ ARAU แสดงได้ดังรูปที่ 2.3

นอกจากนี้ ARAU ยังสามารถเป็นหน่วยคำนวณทางคณิตศาสตร์ได้ด้วย โดยสามารถทำการคำนวณตัวเลขขนาด 16 bit ในขณะที่ CALU คำนวณตัวเลขขนาด 32 bit

2.2 หน่วยคำนวณทางคณิตศาสตร์และลอจิกกลาง :CALU

ประกอบด้วยวงจรการเลื่อนบิตขนาด 16 bit, วงจรคูณเลขขนาด 16 x 16 bit และ ALU ขนาด 32 bit โดยมีAccumulatorรับข้อมูลขนาด 32 bit วงจรเลื่อนBitจะทำการเลื่อนBitไปทางซ้ายได้ตั้งแต่ 0 - 16 bitโดยโปรแกรมได้จากคำสั่ง เมื่อเลื่อนbitแล้วbitทางขวามือจะได้รับการเติม 0 ให้

ALU ขนาด 32 bit จะทำงานตามคำสั่งทางคณิตศาสตร์และลอจิก สังเกตว่าการทำงานของ ALU นี้จะมีวงจรผ่านค่า เช่น ผ่านค่าการ shift bit แล้วมาทำใน ALU(ดู Diagram ของรูปที่ 2.3)
ซึ่งทำให้การทำงานแต่ละคำสั่งทำงานได้เร็วมาก โดยเฉพาะคำสั่งการคูณบวกสำหรับการประมวลผลสัญญาณดังที่กล่าวมาแล้ว



รูปที่ 2.3 โครงสร้างของ ARAU

2.3 วงจรคูณ

TMS320C25 มีการคูณตัวเลขขนาด 16 x 16 bit ซึ่งให้Outputตัวเลข เป็น 32 bit การคูณจะนำข้อมูลจาก TR(Temporary Register)มาคูณ ผลลัพธ์จะได้ตัวเลข 32 bitเก็บไว้ที่ PR(Product Register) Outputจากทุก Registerนี้สามารถส่งต่อไปทำอย่างอื่นอีก เช่น การเลื่อนbitหรือการปัดจุดทศนิยม นอกจากนี้ยังทำการเลื่อนbitไปทางขวา 6 bitหรือกระทำในรูปแบบทางคณิตศาสตร์อื่นได้

การคูณทำได้ทั้งแบบคิดเครื่องหมายและไม่คิดเครื่องหมาย คำสั่งที่น่าสนใจ เช่น MAC-multiply / accumulate เป็นคำสั่งที่คูณแล้วบวกสะสม หรือคำสั่ง MACD-multiply/ accumulate and Data move ซึ่งเป็นการใช้คำสั่งเดียวแต่ทำงานได้หลายอย่างและทำได้รวดเร็ว เพราะโครงสร้างเป็นแบบ pipelining ทั้งสองคำสั่งนี้จะใช้ Operand 2 ตัว โดยอาจจะนำมาจากโปรแกรมหรือจากข้อมูลก็ได้

2.4 การควบคุมการทำงาน

TMS320C25 มีส่วนการควบคุมการทำงานที่สำคัญภายใน ซึ่งได้แก่ วงจรตั้งเวลา วงจรนับให้เกิดการทำงาน Markable-Interruptจากภายนอก 3 ตัว และ Interrupt-ภายในซึ่งเกิดจากพอร์ตอนุกรมหรือวงจรตั้งเวลา

วงจรตั้งเวลาประกอบด้วยRegisterวงจรนับขนาด 16 bit ซึ่งเป็นวงจรนับลง(down-counter) การทำงานจะทำการนับสัญญาณนาฬิกาจาก CLKOUT1 และเมื่อวงจรนับเวลาลดค่า Registerลงมาเหลือ 0 ก็จะทำให้เกิดการInterrupt (TINT) และจากนั้นวงจรตั้งเวลาก็จะ Load ค่าจาก PRD เข้ายังRegisterของวงจรนับใหม่เป็นCycleต่อไป ดังนั้นช่วงเวลาที่เกิด Interruptแต่ละครั้งจึงเท่ากับ (PRD+1)*CLKOUT ซึ่งทำให้เกิดประโยชน์ในการกำหนดการ Synchronizeกับอุปกรณ์ภายนอก

ค่าของ Register RPTC (repeat counter) เป็นตัวกำหนดจำนวนวงรอบของการทำงานของคำสั่งซึ่งนำค่ามาจากหน่วยความจำข้อมูลหรือจากคำสั่งโดยตรง การกำหนด RPTC จะทำให้คำสั่งได้รับการทำซ้ำเป็นจำนวนครั้งเท่ากับค่าที่Loadไว้ ซึ่งจะทำการซ้ำได้ถึง 256 ครั้ง

การInterrupt CPUจากภายนอกจะมีขาInterrupt 3 ขา คือ INT0 ถึง INT2

2.5 พอร์ตอนุกรม

พอร์ตอนุกรมเป็นพอร์ตที่ใช้เชื่อมโยงกับวงจรภายนอกโดยการใช้ข้อมูลแบบอนุกรม การเชื่อมโยงข้อมูลนี้อาจจะเชื่อมต่อกับตัวอุปกรณ์ A/D และ D/A โดยวงจรเชื่อมต่อภายนอกจะต้อง

เป็นHardwareที่รับสัญญาณนี้ได้ ซึ่งภายในชิพมี Register2 ตัวที่ทำหน้าที่เก็บข้อมูลโดยRegister ตัวหนึ่งเป็นRegisterส่งข้อมูล อีกตัวหนึ่งเป็นRegisterรับข้อมูล การรับข้อมูลจะรับข้อมูลแบบ 8 bitหรือ 16 bitก็ได้ โดยแยกModeการรับส่งเป็นแบบ byte หรือ word การส่งรับข้อมูลนี้ได้รับการสร้างสัญญาณSyncจากภายในซึ่งสามารถรับส่งสัญญาณข้อมูลได้สูงถึง 5 MHz

ข้อปรับปรุงพอร์ตอนุกรมของ TMS20C25 ได้แก่ การสร้างBufferของตัวรับและตัวส่งข้อมูล และ CLKR/CLKX คือ สัญญาณนาฬิกาสำหรับการรับส่งข้อมูล สามารถใช้สัญญาณนาฬิกาได้ตั้งแต่ 0 Hz เป็นต้นไป นอกจากนี้ยังเพิ่ม FSM-Fram Sync Mode เพื่อให้ใช้กับระบบโทรศัพท์แบบ PCM ตามมาตรฐาน AT&T T-1 และ CCITT G.711/712 จึงทำให้การประยุกต์ TMS320C25 ในระบบโทรศัพท์ทำได้ง่าย

2.6 การเชื่อมต่อกับอินพุต-เอาท์พุต

TMS320C25 มีอินพุต-เอาท์พุตพอร์ตอย่างละ 16 พอร์ต แต่ละพอร์ตเป็นพอร์ตแบบขนาน 16 bit โดยเชื่อมต่อกับBusข้อมูล คำสั่งที่เกี่ยวกับ IN กับ OUT จะใช้เวลา 2 Cycle แต่ถ้าให้ทำงานแบบซ้ำโดยกำหนดค่าเข้าไปในRegisterทำซ้ำแล้ว คำสั่ง IN หรือ OUT นี้จะใช้เวลาได้เพียงCycleเดียว การเชื่อมโยงทาง I/O จะแยกออกจากระบบเพราะไมโครโปรเซสเซอร์ DSP นี้มีช่องทางสำหรับโปรแกรมข้อมูลและ I/O นอกจากนี้ยังสามารถมอง I/O เหมือนเป็นหน่วยความจำได้

บนตัวไอซีจะประกอบด้วย D0-D15 เป็นBusข้อมูล A0-A15 เป็นBusของAddressและยังประกอบด้วยขาไอซี 3 ขา เพื่อเลือกว่าเป็นโปรแกรม, ข้อมูลหรืออินพุตเอาท์พุต (DS, PS และ IS) การกำหนดทิศทางจะใช้ R/W และมีสัญญาณ STRB เป็นตัวควบคุมการรับส่งข้อมูล

TMS320C25 มีขา HOLD เพื่อใช้ทำ DMA โดยเมื่อขานี้ activeจะทำให้สถานะของ Address Bus, Busข้อมูล และสายสัญญาณควบคุมอยู่ในสภาวะHigh-Impedance เพื่อให้อุปกรณ์ภายนอกติดต่อกับ ROM และ RAM ได้ต่อไป

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

ตารางที่ 2.1 Interupt Locations และ Priorities

2.7 INTERRUPT

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

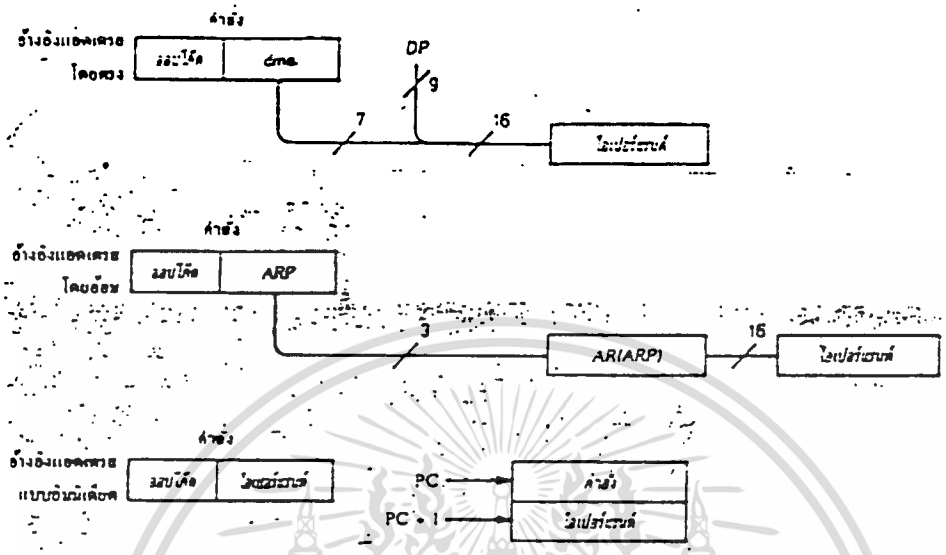
2.8 โหมดการอ้างแอดเดรสของ TMS320C25

คำสั่งส่วนใหญ่ของTMS320C25เป็นคำสั่งที่ใช้รหัส 16 bitเพื่อว่าการFetchและExecute จะกระทำได้เพียงcycle Address 8 bit การอ้างอิงAddressมีอยู่ 3 mode ดังรูปที่ 2.4 คือ แบบอ้างอิงAddressโดยตรง (Direct) แบบอ้างอิงAddressโดยอ้อม(Indirect) และแบบImmediate การอ้างอิงAddressโดยตรงและโดยอ้อมเป็นการต้องการติดต่อกับหน่วยความจำ ส่วนการอ้างอิงAddress แบบImmediateเป็นการนำเอา Operand มาใช้โดยตรง

การอ้างอิงAddressโดยตรงจะใช้คำสั่ง 16 bit โดยแยกออกเป็นOpcode 9 bitและอีก 7 bitเป็น Operand ในการทำงานจะใช้ 7 bitร่วมกับ 9 bitจาก Page-pointer (DP)เพื่อรวมให้เป็น 16 bit สำหรับอ้างอิงหน่วยความจำ 64K ดังนั้นการจัดหน่วยความจำจึงแบ่งเป็นเพจๆ ละ 128 Wordและมีจำนวน 512 Page

การอ้างAddressแบบทางอ้อมจะอ้างผ่าน Register(AR_0 - AR_7) Registerเหล่านี้เป็นRegisterตัวนับLoopหรือสำหรับเก็บข้อมูลชั่วคราวก็ได้ รูปที่ 2.5เป็นการใช้ Register AR_3 เป็นตัวอ้าง Addressไปยังหน่วยความจำ การกำหนด Registerทำได้โดยการใช้ Registerช่วย (Auxiliary Register Pointer) ARP ขนาด 3 bit ซึ่งมีค่าได้จาก 0-7 เพื่อแทน AR_0-AR_7 ค่าใน ARP นี้สามารถLoadเข้าโดยตรงจากข้อมูลในหน่วยความจำ หรือจากคำสั่งที่มีการกำหนดค่าแบบ Immediate operand และเราสามารถข้อมูลจาก Registerช่วย

เหล่านี้รีโหลดกลับเข้าหน่วยความจำได้เช่นกัน

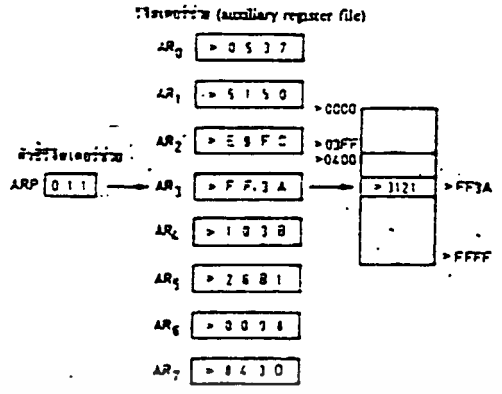


รูปที่ 2.4

จากตารางที่ 2.2 จะเห็นว่ามีการอ้างAddressแบบทางอ้อม 7 แบบ คือ

- การIndexด้วยการเพิ่มค่า
- การIndexด้วยการลดค่า
- การIndexโดยการบวกข้อมูลของ AR_0
- การIndexโดยการลบข้อมูลของ AR_0
- การIndexโดยการบวกข้อมูลของ AR_0 สำหรับการทำ bit-reversing ใน Algorithm FFT
- การIndexโดยการลบข้อมูลของ AR_0 สำหรับการทำ bit-reversing ใน Algorithm FFT
- ไม่มีการIndex

การอ้างAddressแบบImmediateจะมีส่วนของOperandประกอบอยู่ในคำสั่ง โดยค่าตัวเลขนี้จะเป็นไปได้ 2 ขนาด คือ แบบ 8 bitหรือแบบ 13 bi t และสำหรับคำสั่งแบบที่มีขนาด 2 WordจะมีขนาดของตัวเลขOperand ที่ใช้งานขนาด 16 bit



รูปที่ 2.5

โหมดแอดเดรส	ลักษณะการอ้าง	การทำงาน
OP A	แอดเดรสโดยทางตรง	
OP * (,NARP)	แอดเดรสโดยทางอ้อม	ไม่มีการเปลี่ยนค่าใน AR
OP * + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR เดิมเพิ่มขึ้น 1
OP * - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR เดิมลดค่าลง 1
OP * 0 + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR_0 บวกกับค่าของ AR เดิม
OP * 0 - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR_0 ลบกับค่าของ AR เดิม
OP * BRO + (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR_0 บวกกับค่าของ AR เดิม โดยมีการให้บิตทดกระจายไว้
OP * BRO - (,NARP)	แอดเดรสโดยทางอ้อม	ค่าของ AR_0 ลบจากค่าของ AR เดิม โดยมีการให้บิตทดกระจายไว้

ตารางที่ 2.2

2.8 ชุดคำสั่งของ TMS320C25

สำหรับ TMS320C25 นี้ ได้ปรับปรุงโครงสร้างทางสถาปัตยกรรมเพิ่มขึ้นอีก ดังนั้นจึงมีชุดคำสั่งที่เพิ่มเติมขึ้นอีกหลายคำสั่งและบางคำสั่งก็มีขีดความสามารถสูงขึ้น สัญลักษณ์ที่ใช้แทนคำสั่งมีดังนี้

ACC - Accumulator

ARB - Registerช่วยชี้บัฟเฟอร์

AR_n - Registerช่วย (auxiliary register) ตัวที่ n

ARP - ตัวชี้Registerช่วย (auxiliary register pointer)

BIO - Branch control input

C - bitตัวทด

CM - 2 bitที่ใช้สำหรับการกำหนดโหมดการเปรียบเทียบ (compare mode)

CNF - bitที่ใช้สำหรับการควบคุม RAM บนชิพ (configulation control bit)

dma - Addressหน่วยความจำข้อมูล

DP - PointerบอกPageข้อมูล (data page pointer)

FO - bitบอกสถานะformat (format status bit)

FSM - bitบอกโหมด Frame synchronize

HM - bitบอกโหมดการ hold

INTM - bit flagโหมด interrupt

>nn - บอกว่า nn คือ เลขฐาน 16

OV - Overflow flag

OVM - bitโหมด Overflow

P - Registerผลคูณ

PA - Address พอร์ต โดยใช้ PA0 หมายถึง พอร์ต 0

PC - Program counter

PM - 2 bitที่บอก P Registerให้outputเลื่อนbit

pma - Addressหน่วยความจำโปรแกรม

Preg - Registerผลคูณ

RPTC - Counterนับการซ้ำ

ST_n - Registerสถานะ (ST0 หรือ ST1)

SXM - bitบอกโหมด sign extension

T - Registerชั่วคราว

TC - bitทดสอบการควบคุม (test control register)

TOS - จุดสูงสุดของStack (top of stack)

Treg - Registerชั่วคราว

TXM - bitบอกโหมดส่งข้อมูลอนุกรม

Usgn - ค่าที่ไม่คิดเครื่องหมาย

XF - bitบอกสถานะของขา XF

ตารางที่ 2.3 เป็นชุดคำสั่งของ TMS320C25 ซึ่งได้ให้รายละเอียดเป็นกลุ่มคำสั่งไว้

ACCUMULATOR MEMORY REFERENCE INSTRUCTIONS

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

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

ตารางที่ 2.3 ชุดคำสั่งของ TMS320C25

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

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

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

ตารางที่ 2.3 ชุดคำสั่งของ TMS320C25 (ต่อ)

BRANCH/CALL INSTRUCTIONS							
Mnemonic and Description		Words	16-Bit Opcode MSB				LSB
B	Branch unconditionally	2	1111	1111	1DDD	DDDD	
BACC†	Branch to address specified by accumulator	1	1100	1110	0010	0101	
BANZ	Branch on auxiliary register not zero	2	1111	1011	1DDD	DDDD	
BBNZ†	Branch if TC bit ≠ 0	2	1111	1001	1DDD	DDDD	
BBZ†	Branch if TC bit = 0	2	1111	1000	1DDD	DDDD	
BC‡	Branch on carry	2	0101	1110	1DDD	DDDD	
BGEZ	Branch if accumulator ≥ 0	2	1111	0100	1DDD	DDDD	
BGZ	Branch if accumulator > 0	2	1111	0001	1DDD	DDDD	
BIOZ	Branch on I/O status = 0	2	1111	1010	1DDD	DDDD	
BLEZ	Branch if accumulator ≤ 0	2	1111	0010	1DDD	DDDD	
BLZ	Branch if accumulator < 0	2	1111	0011	1DDD	DDDD	
BNC‡	Branch on no carry	2	0101	1111	1DDD	DDDD	
BNV†	Branch if no overflow	2	1111	0111	1DDD	DDDD	
BNZ	Branch if accumulator ≠ 0	2	1111	0101	1DDD	DDDD	
BV	Branch on overflow	2	1111	0000	1DDD	DDDD	
BZ	Branch if accumulator = 0	2	1111	0110	1DDD	DDDD	
CALA	Call subroutine indirect	1	1100	1110	0010	0100	
CALL	Call subroutine	2	1111	1110	1DDD	DDDD	
RET	Return from subroutine	1	1100	1110	0010	0110	
TRAP†	Software interrupt	1	1100	1110	0001	1110	

AUXILIARY REGISTERS AND DATA PAGE POINTER INSTRUCTIONS							
Mnemonic and Description		Words	16-Bit Opcode MSB				LSB
ADRK‡	Add to auxiliary register short immediate	1	0111	1110	KKKK	KKKK	
CMPR†	Compare auxiliary register with auxiliary register AR0	1	1100	1110	0101	00KK	
LAR	Load auxiliary register	1	0011	0RRR	1DDD	DDDD	
LARK	Load auxiliary register short immediate	1	1100	0RRR	KKKK	KKKK	
LARP	Load auxiliary register pointer	1	0101	0101	1000	1RRR	
LDP	Load data memory page pointer	1	0101	0010	1DDD	DDDD	
LDPK	Load data memory page pointer immediate	1	1100	100K	KKKK	KKKK	
LRLK†	Load auxiliary register long immediate	2	1101	0RRR	0000	0000	
MAR	Modify auxiliary register	1	0101	0101	1DDD	DDDD	
SAR	Store auxiliary register	1	0111	0RRR	1DDD	DDDD	
SBRK‡	Subtract from auxiliary register short immediate	1	0111	1111	KKKK	KKKK	

T REGISTER, P REGISTER, AND MULTIPLY INSTRUCTIONS							
Mnemonic and Description		Words	16-Bit Opcode MSB				LSB
APAC	Add P register to accumulator	1	1100	1110	0001	0101	
LPH†	Load high P register	1	0101	0011	1DDD	DDDD	
LT	Load T register	1	0011	1100	1DDD	DDDD	
LTA	Load T register and accumulate previous product	1	0011	1101	1DDD	DDDD	
LTD	Load T register, accumulate previous product, and move data	1	0011	1111	1DDD	DDDD	
LTP†	Load T register and store P register in accumulator	1	0011	1110	1DDD	DDDD	
LTS†	Load T register and subtract previous product	1	0101	1011	1DDD	DDDD	
MAC†	Multiply and accumulate	2	0101	1101	1DDD	DDDD	
MACD†	Multiply and accumulate with data move	2	0101	1100	1DDD	DDDD	
MPY	Multiply (with T register, store product in P register)	1	0011	1000	1DDD	DDDD	
MPYA‡	Multiply and accumulate previous product	1	0011	1010	1DDD	DDDD	
MPYK	Multiply immediate	1	101K	KKKK	KKKK	KKKK	
MPYS‡	Multiply and subtract previous product	1	0011	1011	1DDD	DDDD	
MPYU‡	Multiply unsigned	1	1100	1111	1DDD	DDDD	
PAC	Load accumulator with P register	1	1100	1110	0001	0100	
SPAC	Subtract P register from accumulator	1	1100	1110	0001	0110	
SPH‡	Store high P register	1	0111	1101	1DDD	DDDD	
SPL‡	Store low P register	1	0111	1100	1DDD	DDDD	
SPM†	Set P register output shift mode	1	1100	1110	0000	10KK	
SQRA†	Square and accumulate	1	0011	1001	1DDD	DDDD	
SQRS†	Square and subtract previous product	1	0101	1010	1DDD	DDDD	

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

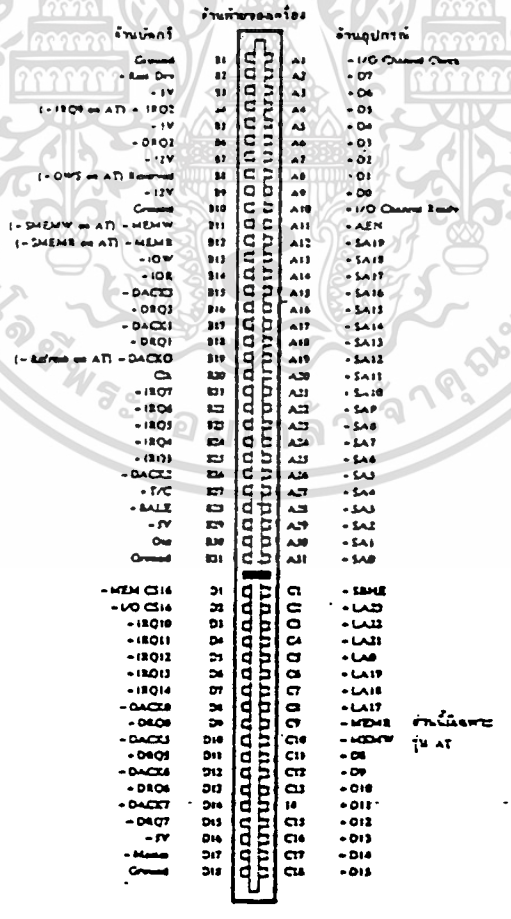
ตารางที่ 2.3 ชุดคำสั่งของ TMS320C25 (ต่อ)

บทที่ 3
IBM PC กับการอินเตอร์เฟส

ในตอนนี้จะขอแนะนำส่วนต่างๆของ IBM PC ที่เกี่ยวข้องกับการอินเตอร์เฟส ซึ่งต้องต่อโดยตรงกับระบบBusของ IBM PC ให้เข้าใจก่อน

3.1 AT SLOT และ XT SLOT ของ IBM PC

เนื่องจากเครื่องรุ่น AT ใช้ CPU เบอร์ 80286 ซึ่งเป็น CPU ขนาด 16 bitแท้ คือมีการประมวลผลภายในเป็น 16 bit ในขณะที่รุ่น XT ใช้ CPU เบอร์ 8088 ซึ่งเป็น 16 bitเทียม คือมีการประมวลผลภายในเป็นแบบ 16 bitแต่ Data Busมีเพียงแค่ 8 บิตเท่านั้น ดังนั้นการจัดการเกี่ยวกับข้อมูลขนาด 16 bitจึงต้องกระทำ 2 ครั้ง ครั้งละ 8 bit จะเห็นได้ว่าระบบBusของ XT เป็นSubsetของระบบ Busของ AT



รูปที่ 3.1 แสดงตำแหน่งของสัญญาณบน Slot IBM

บนเมนบอร์ดของ AT จะมีSlotอยู่ 2 ชนิด คือSlotสั้นและSlotยาว ซึ่งSlotยาวจะมีจำนวนขาสัญญาณและตำแหน่งของขาสัญญาณบนslotเหมือนกับ XT ส่วนขาสัญญาณที่อยู่บนSlotสั้นที่เพิ่มขึ้น จะเป็นสัญญาณที่มีแต่เฉพาะบน AT เท่านั้น ประกอบด้วยข้อมูลครึ่งบน 8 bit Address ที่เพิ่มขึ้นมาอีก 4 bit และขาสัญญาณควบคุมที่เพิ่มขึ้นจาก XT สำหรับรูปของSlotและตำแหน่งของสัญญาณเสดงไว้ในรูปที่ 2.1

สำหรับเครื่องคอมพิวเตอร์ที่ใช้ CPU สูงกว่า 80286 นั้น เช่น 80386 ที่เป็น CPU ขนาด 32 bit ก็ยังคงใช้สล็อตแบบ AT อยู่ซึ่งมีการทำงานเหมือนเดิมในรุ่นก่อนๆเช่นกัน

3.2 รายละเอียดของสัญญาณต่าง ๆ บนสล็อต

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

(I) หมายถึง ขาสัญญาณinput

(O) หมายถึง ขาสัญญาณoutput

(I/O) หมายถึง ขาสัญญาณที่เป็นได้ทั้งinputและoutput

(*I/O) หมายถึง ในช่วงการทำงานปกติจะเป็นขาสัญญาณoutput แต่จะเป็นinput ในช่วงที่เกิดขบวนการ DMA

สำหรับขาสัญญาณที่มีเครื่องหมายลบนำหน้าจะหมายถึงขาสัญญาณที่activeที่ลอจิก "0" และขาสัญญาณที่ไม่มี หรือมีเครื่องหมายบวกนำหน้าอยู่จะหมายถึง ขาสัญญาณที่activeที่ลอจิก "1" สัญญาณที่ต่ออยู่บนSlotนี้สามารถขับไอซี TTL ชนิดLow Powerได้สองตัว โดยที่ไม่ทำให้เกิดการโหลดหรือการเพี้ยนของสัญญาณ ขาสัญญาณต่างๆ บนSlotของ XT และ AT สามารถแบ่งออกเป็นกลุ่ม ๆ ได้ดังนี้

เพาเวอร์ซัพพลาย

Ground	ขาสัญญาณนี้ต่ออยู่กับGroundของระบบRegulator
+5 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC Regulator +5 V
-5 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC Regulator -5 V
+12 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC Regulator +12 V
-12 V	ขาสัญญาณนี้ต่ออยู่กับไฟ DC Regulator -12 V

Address Bus และ สัญญาณต่าง ๆ ที่เกี่ยวข้อง

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

สัญญาณ BALE Addressทั้ง 20 bitนี้ สามารถอ้างหน่วยความจำได้ถึง 1 Mbyte สำหรับเครื่องXT สำหรับ AT เมื่อใช้ร่วมกับ LA17-LA23 จะอ้างได้ถึง16 Mbyte

LA17-LA23 (เฉพาะรุ่น AT) ขาสัญญาณนี้จะactiveเมื่อขาสัญญาณ BALE มีสถานะเป็นลอจิก
(*I/O) "1" แต่จะไม่มีการLatchไว้ที่ขอบขาลงของสัญญาณ BALE ดังนั้นถ้าอุปกรณ์ I/O ไม่มีการอ้างAddressเกิน 1 Mbyte ขาสัญญาณนี้ก็ไม่จำเป็นต้องใช้ แต่ถ้ามีการอ้างAddressเกิน อุปกรณ์ I/O จะต้องทำการLatchสัญญาณนี้ โดยใช้ขอบขาลงของสัญญาณ BALE ร่วมกับขาสัญญาณ -MEMW และ -MEMR

AEN (Address Enable) ขาสัญญาณนี้จะactiveเมื่อตัวควบคุม DMAได้ทำการควบคุม
(O) บัสต่าง ๆ ของระบบแล้ว ดังนั้นการอ้างพอร์ตของอุปกรณ์ I/O จะต้องใช้สัญญาณนี้ในการDecodeด้วย เพื่อที่จะไม่ทำให้เกิดการติดต่อระหว่างระบบกับอุปกรณ์ I/O ตัวอื่น ยกเว้นตัวที่กำลังทำขบวนการ DMA อยู่

BALE (Address Latch Enable) ขาสัญญาณนี้ใช้ในการแสดงการเริ่มต้นของขบวนการ
(O) ต่าง ๆ ที่มีการติดต่อกับหน่วยความจำ โดยจะactiveเมื่อค่าAddressที่ CPU ต้องการติดต่อด้วยอยู่บนAddress Busเรียบร้อยแล้ว ตามปกติขอบขาลงสัญญาณนี้จะทำให้เกิดการLatchสัญญาณ SA0-SA19 และถ้ามีการอ้างAddressเกิน 1 Mbyte ใน AT จะใช้ขอบขาลงของสัญญาณนี้ในการLatchสัญญาณ LA17-LA23 ด้วย เช่นกัน แต่สำหรับในขบวนการ DMA สัญญาณนี้จะมีสถานะเป็น "1" ตลอด

SBHE (เฉพาะรุ่น AT) (Bus High Enable) เป็นขาสัญญาณที่ใช้แสดงว่ามีการรับส่ง
(*I/O) ข้อมูลในbitที่ SD8-SD15

ขาข้อมูลบัส

SD0-SD7 สำหรับรุ่น AT จะมี SD8-SD15 เพิ่มขึ้นมาด้วย คือ Data bit0 ถึง 7 สำหรับรุ่น
(I/O) XT และสำหรับรุ่น AT คือ Data bit 0 ถึง 15 โดยที่ SD0 มีนัยสำคัญต่ำสุด สำหรับ AT ถ้ามีการติดต่อกับbitที่ SD8-SD15 สามารถตรวจสอบได้จากขาสัญญาณ SBHE

สัญญาณอินเตอร์รัพท์

IRQ2-IRQ7 (Interrupt Reguest) (สำหรับรุ่น AT จะเป็น IRQ3-7,9-12,14,15) เป็น
(I) ขาสัญญาณInterrupt CPU สำหรับ AT ลำดับความสำคัญของสัญญาณ Interrupt IRQ เป็นดังนี้ คือ 9,10,11,12,14,15,3,4,5,6 และ 7 โดย IRQ9 มีลำดับความสำคัญน้อยที่สุด สำหรับ XT IRQ2 จะมีลำดับความสำคัญมากที่สุด รองลงไป คือ IRQ 3,4,5,6,7 โดยปกติสัญญาณนี้จะมีสถานะเป็น "0" เสมอ ถ้าต้องการ

Interrupt CPU ให้ส่งPulseที่เป็นลอจิก "1" ให้กับมัน โดยไม่จำเป็นต้องคำนึงถึงคาบเวลาของPulse ทั้งนี้เพราะระบบของ IBM ตัวInterrupt Controller (8259 Interrupt controller)จะถูกโปรแกรมให้ตรวจสอบสัญญาณInterrupt โดยใช้ขอบขาลงของสัญญาณนี้

-I/O CH CK (I/O Channel Check) เป็นขาสัญญาณที่บอกถึงความผิดพลาดในการรับส่งข้อมูล

(I) ซึ่งตรวจสอบจาก Parity bit ถ้า Parity bitอ่านจากหน่วยความจำกับ Parity bit ที่สร้างขึ้นจากขบวนการรับส่งข้อมูล สัญญาณนี้จะทำให้เกิดการ Interrupt CPU แบบ NMI เพื่อบอกให้ CPU ทราบว่าเกิด Parity Error ขึ้น CPU จะแสดงข้อความบอกความผิดพลาดขึ้นและจะหยุดการทำงาน (Halt) เพื่อให้ผู้ใช้ตรวจสอบหาสาเหตุของการผิดพลาด

สัญญาณที่ใช้ในขบวนการ DMA

DRQ1-DRQ3 (DMA Request)(สำหรับรุ่น AT จะเป็น DRQ0-3,5-7) เป็นขาสัญญาณใช้ในการ

(I) ขอทำขบวนการ DMA โดยที่ DRQ0 มีลำดับความสำคัญมากที่สุด และ DRQ3 มีลำดับความสำคัญน้อยที่สุดสำหรับรุ่น XT และสำหรับรุ่น AT ขา DRQ7 จะมีลำดับความสำคัญน้อยที่สุด

-DACK03 (DMA Acknowledge) (สำหรับรุ่น AT จะเป็น -DACK0-3 , 5-7) เป็นสัญญาณ

(O) ตอบสนองการขอทำ DMA ของอุปกรณ์ I/O เพื่อให้อุปกรณ์ I/O ทราบว่าการขอทำขบวนการ DMA นั้นได้รับการตอบสนองแล้ว เช่น ถ้ามีการขอทำ DMA ผ่านทาง DRQ2 และเมื่อ CPU รับรู้แล้ว จะทำให้สัญญาณ DACK2 active

Refresh (เฉพาะรุ่น AT) (Memory Refresh) มีหน้าที่เหมือนกับขาสัญญาณ DACK0 ในรุ่น

(O) XT คือ ใช้แสดงขบวนการRefreshหน่วยความจำ เพราะว่าในรุ่น AT จะมีวงจรที่ใช้ในการRefreshหน่วยความจำโดยตรงอยู่แล้ว ดังนั้นจึงไม่จำเป็นต้องใช้ขาสัญญาณ DRQ0 และ DACK0

-Master (เฉพาะรุ่น AT) (Master) ขาสัญญาณนี้จะใช้ร่วมกับ DMA Request ในการเข้าควบคุมระบบBusในขบวนการ DMA โดยที่ตัว DMA Controllerจะส่งสัญญาณ DMA

(O) Request แล้วรอจนกระทั่งได้รับการตอบสนองโดยสัญญาณ DACKเกิดการactive ขึ้น แล้วจึงจะส่งสัญญาณนี้ให้กับ CPU จะทำให้ Adress Bus, Data Busและ Control Busเข้าสู่สถานะTri-state หรือ High-Impedance หลังจากนั้นตัว DMA Controllerจะต้องรออีกหนึ่งคาบสัญญาณClock ก่อนที่จะเข้าควบคุมBus ต่างๆ และจะต้องรออีก 2 Cycle ก่อนที่จะทำการอ่านหรือเขียนข้อมูล ช่วงเวลา

ที่สัญญาณนี้activeไม่ควรเกิน 15 us มิฉะนั้นข้อมูลภายในหน่วยความจำจะสูญหายไปได้ เนื่องจากขาดสัญญาณRefreshหน่วยความจำ

T/C (O) (Terminal Count) เป็นขาสัญญาณที่บอกอุปกรณ์ I/O ที่ทำ DMA ให้ทราบว่าจำนวนข้อมูลที่รับส่งในขบวนการDMA นี้ครบจำนวนแล้วโดยจะส่งสัญญาณนี้เป็นPulseให้กับอุปกรณ์ I/O

สัญญาณควบคุมต่างๆ

-MEMR (*I/O) (Memory Read) [สำหรับรุ่น AT คือ ขาสัญญาณ -SMEMR (System Memory Read)] ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำส่งข้อมูลออกมาที่Data Bus แต่สำหรับ AT สัญญาณ -SMEMR จะactive เมื่อการอ่านข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 Mbyteแรกเท่านั้น

MEMR (O) (เฉพาะรุ่น AT) (Memory Read) ขาสัญญาณนี้มิใช่สัญญาณเดียวกันกับสัญญาณ -MEMR ใน XT มันจะactiveก็ในทุกขบวนการอ่านข้อมูลที่เกิดขึ้น ไม่ว่าอยู่ในช่วงหน่วยความจำ 1 Mbyteแรกหรือไม่

MEMW (*I/O) (Memory Write) [สำหรับรุ่น AT คือ ขาสัญญาณ -SMEMW (System Memory Write)] ขาสัญญาณนี้จะเป็นตัวบอกให้หน่วยความจำเก็บข้อมูลจากData Bus แต่สำหรับ AT สัญญาณ -SMEMW จะactive เมื่อเกิดการเก็บข้อมูลจากหน่วยความจำที่อยู่ภายใน 1 Mbyteแรกเท่านั้น

MEMW (O) (เฉพาะรุ่น AT) (Memory Write) ขาสัญญาณนี้ไม่ใช่สัญญาณเดียวกันกับสัญญาณ -MEMW ใน XT มันจะActiveในทุกขบวนการเก็บข้อมูลที่เกิดขึ้น ไม่ว่าอยู่ในช่วงหน่วยความจำ 1 Mbyteแรกหรือไม่

-IOR (*I/O) (I/O Read) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการส่งข้อมูลลงมาที่ Data Bus

-IOW (*I/O) (I/O Write) เป็นขาสัญญาณที่บอกให้อุปกรณ์ I/O ที่ต่ออยู่ ทำการเก็บข้อมูลจาก Data Busเข้าไป

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

-MEM CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่า ต้องการรับส่งข้อมูลกับหน่วยความจำทีละ 16 bitถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือ ทำการรับส่งข้อมูลทีละ 8 bit สองครั้งเพื่อให้

ได้ข้อมูลขนาด 16 bit

-I/O CS16 (I) (เฉพาะรุ่น AT) (Memory 16 Chip Select) เป็นขาสัญญาณที่ใช้บอกระบบให้ทราบว่า ต้องการรับส่งข้อมูลกับอุปกรณ์ I/O ทีละ 16bit ถ้าไม่ป้อนสัญญาณนี้ การรับส่งข้อมูลจะทำเหมือนกับ XT คือ ทำการรับส่งข้อมูลทีละ 8 bit 2 ครั้ง เพื่อให้ได้ข้อมูลขนาด 16 bit

สัญญาณที่ใช้สร้าง Wait States

I/O CH RDY (I) (I/O Channel Ready) ขาสัญญาณนี้จะถูกทำให้activeโดยอุปกรณ์ I/O หรือหน่วยความจำที่ไม่สามารถทำงานได้ทันกับระบบ ดังนั้น จะต้องทำการหน่วงระบบให้ทำงานช้าลง ด้วยการเพิ่มWait States โดยการทำให้สัญญาณนี้activeในช่วงเวลาที่ I/O ได้รับสัญญาณจากการDecode Adress, สัญญาณ -MEMR , สัญญาณ -MRMW , สัญญาณ -IOR , สัญญาณ -IOW

OWS (I) (เฉพาะรุ่น AT) (Zero Wait State) activeของขาสัญญาณนี้จะบังคับไม่ให้เกิดการสร้าง Wait States โดยอัตโนมัติ นั่นคือ การที่จะเกิด Wait States ขึ้นได้ จะต้องขึ้นอยู่กับสัญญาณนี้ เช่น การทำงานในขบวนการอ่านเขียนข้อมูลขนาด 16 bit โดยไม่ใช้ Wait States ทำได้โดยการสร้างสัญญาณ OWS จากสัญญาณการDecode Addressและสัญญาณที่ใช้ในการอ่านเขียน หรือการลด Wait States ในขบวนการอ่านเขียนข้อมูลขนาด 8 bitให้เหลือเพียง 2 Wait States ทำได้โดยให้สัญญาณ OWS active หลังจากการอ่านหรือเขียนไปแล้ว 1 Clock โดยปกติการขับสัญญาณนี้ควรใช้ Gate ที่มี outputเป็นแบบ Open Collector ที่ทนกระแสได้ 20 mA (Sinking Current)

สัญญาณนาฬิกา

CLK (O) (System Clock) สำหรับ XT ขาสัญญาณนี้จะมีความถี่ประมาณ 4.77 MHz หรืออาจจะสูงกว่านี้ก็ได้สำหรับรุ่นใหม่ๆ และสำหรับ AT จะมีความถี่ประมาณ 6 MHz หรือในรุ่นใหม่ๆ อาจจะมีความถี่สูงถึง 15 MHz

โดยปกติขาสัญญาณนี้มี Duty Cycle 50 % สำหรับ CPU เบอร์ 80286 ตัวกำเนิดสัญญาณนาฬิกาที่ป้อนให้จะมีความถี่เป็น 2 เท่าของความถี่ที่ CPU ทำงาน แต่ขาสัญญาณนี้ก็จะยังคงมีความถี่เท่ากับความถี่ที่ CPU ทำงานอยู่เสมอ

OSC (O) (Oscillator) เป็นขาสัญญาณที่มีความถี่สูง คือ 14.31818 MHz ความถี่ของสัญญาณนี้จะคงที่เสมอ และจะไม่Synchronousกับสัญญาณอื่นๆในระบบ ดังนั้นจึงไม่ควรนำสัญญาณนี้ไปใช้เป็นสัญญาณClockของอุปกรณ์ I/O ที่ต่ออยู่กับระบบ

3.3 การจัด Address ของ Memory และ Port

การควบคุมอุปกรณ์ I/O ที่ต่ออยู่กับ IBM PC จะกระทำผ่านพอร์ต โดยการอ้างถึงAddress ของพอร์ตที่อุปกรณ์นั้นต่ออยู่โดยตรง ดังนั้นการที่จะใช้งานหรือควบคุมอุปกรณ์เหล่านี้จึงจำเป็นต้อง ศึกษาถึงวิธีการควบคุมพอร์ต ใน IBM PC พอร์ตและหน่วยความจำจะแยกจากกันโดยเด็ดขาด ถึง แม้ว่าการอ้างถึงจะใช้สัญญาณจากAddress Busเหมือนกันก็ตาม แต่สัญญาณที่ใช้ในการEnableใน การอ่านและเขียนข้อมูลจะต่างกัน ดังนั้นการติดต่อกับพอร์ตจึงมีคำสั่งแยกต่างหากออกจากคำสั่งที่ ใช้ติดต่อกับหน่วยความจำ คือ IN และ OUT ด้วยเหตุนี้ การจัดAddressของหน่วยความจำ และ Addressของพอร์ต I/O จึงแยกออกจากกัน ใน IBM PC การจัดAddressของหน่วยความจำ แสดงได้ในตารางที่ 3.1 และในตารางที่ 3.2 แสดงการจัดแอดเดรสของพอร์ต I/O

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

ตารางที่ 3.1 แสดงการจัดแอดเดรสของหน่วยความจำบน IBM

การอ้างAddressของหน่วยความจำ จะใช้Addressทั้งหมด 20 เส้น คือ A0-A19 ในรุ่น XT (8088) แต่ในรุ่น AT (80286) ใช้Address 24 เส้น คือ A0-A23 การอ้างAddress ของพอร์ตสำหรับ CPU เบอร์ 8088 และ 80286 สามารถอ้างได้ถึง 64 k port แต่ใน IBM PC ทั้งในรุ่น AT และ XT ออกแบบให้ใช้Addressเพียง 10 เส้น เท่านั้น คือ A0-A9 ดังนั้น จำนวนพอร์ตสูงสุดที่สามารถอ้างได้คือ 1024 พอร์ต ในจำนวนทั้งหมดนี้ ยังแบ่งออกเป็น 2 กลุ่ม คือกลุ่มพอร์ตที่มีแอดเดรสอยู่ในช่วง 000H-0FFH จะใช้บนเมนบอร์ดสำหรับชิพชัพพอร์ตเท่านั้น เช่น 8259 (Interrupt Controller) , 8257 (DMA Controller) ,8253 (Timer & Counter) และกลุ่มที่มีAddressอยู่ในช่วง 100H-3FFH จะใช้งานกับการ์ดขยายต่างๆที่เสียบใน

Address	Description	Note
1F0-1F8	Fixed disk	1
200-20F	Games adapter	
210-217	Expansion unit	2
278-27F	2nd parallel printer port	1
280-2DF	Alternate EGA	
2F8-2FF	2nd serial port	
2E1	GPIB (0)	4
2E2-2E3	Data Acquisition (0)	4
300-31F	Prototype card	
320-32F	Fixed disk	2
360-36F	PC Network	
378-37F	1st parallel printer port	
380-38F	SDLC/2nd Bisynchronous	3
390-393	Cluster (0)	4
3A0-3AF	1st Bisynchronous	1
3B0-3BF	Monochrome display/printer	
3C0-3CF	EGA	
3D0-3DF	CGA	
3F0-3F7	Floppy disk	
3F8-3FF	1st serial port	

Notes:
Devices on main board not included
1 AT only
2 PC only
3 2nd Bisynchronous on AT only
4 These devices decode the full 16 address bits thereby allowing further devices in the same category above 3FF (eg GPIB (1) = 2E1 etc)

ตารางที่ 3.2 แสดงการใช้งานของพอร์ตบน IBM

จากตารางที่ 3.2 จะเห็นได้ว่าแอดเดรสของพอร์ตถูกแบ่งออกเป็นช่วงย่อยๆ ซึ่งจะกำหนดไว้ให้ใช้กับอุปกรณ์ I/O เฉพาะอย่าง ถ้าในระบบของเราไม่ได้ใช้งานอุปกรณ์นั้น เราสามารถนำAddressของพอร์ตในช่วงนั้นมาใช้งานได้ เช่น ถ้าระบบเราไม่ได้ใช้จอยสติก (Joystick) เราสามารถใช้งานพอร์ตในช่วง (200H-20FH) ได้ แต่อย่างไรก็ตาม การเลือกใช้งานพอร์ตที่ไม่ได้ถูกกำหนดให้ใช้กับอุปกรณ์อื่นจะดีกว่า ทำให้อุปกรณ์ที่ใช้งานผ่านพอร์ตนี้สามารถใช้ได้กว้างขวางยิ่งขึ้น

3.4 (Interrupts)

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

การอินเตอร์รัพท์แบบ Maskable มีจำนวน 256 Vector (Vector ในที่นี้คือ Address

เริ่มต้นของโปรแกรมย่อย (Interrupt Service Routine) ที่ถูกกระทำเมื่อได้รับการ Interrupt) ในจำนวนทั้งหมด 256 Vectorนี้ จะรวมถึงSoftware Interrupt(B DOS Call) , สัญญาณInterruptที่เกิดจากตัว CPU เองด้วย เช่น การหารด้วยศูนย์ (Divide By Zero) , การทำทีละคำสั่งที่ใช้ในโปรแกรมDebug(Single Step)และHardware Interrupt สำหรับHardware Interruptแบ่งออกได้ตามลำดับความสำคัญดังนี้ สำหรับรุ่น XT แบ่งได้ 8 ระดับ และ 15 ระดับในรุ่น AT ตารางที่ 3.3 แสดงHardware Interruptใน IBM PC Interruptบางระดับจะถูกใช้ในMain Boardเช่น อินเตอร์รัพต์ระดับ 0 เป็นต้น ส่วนที่เหลือจะถูกใช้ในการ์ดขยายต่างๆ เช่น ตัวควบคุมดิสก์ (Floppy & Hard Disk Controller)

ใน IBM PC การขอทำแบบ Hardware Interruptแบบ Maskable เราไม่จำเป็นต้องให้ค่าเวกเตอร์ เพราะว่า 8259 (Interrupt Controller) จะเป็นตัวจัดการเองหมด เราเพียงแค่ให้สัญญาณขอทำInterruptผ่านทางSlotก็พอ

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

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

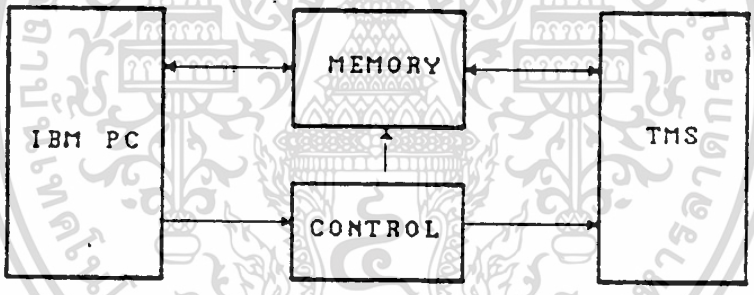
The functions of IRQ2 (or IRQ9) is officially reserved and would not be a good one to consider using as it is used by EGA cards.

ตารางที่ 3.3 แสดงการจัดHardware Interrupt

บทที่ 4
รายละเอียดการดำเนินงาน

4.1 หลักการเบื้องต้น

จะขอกล่าวถึงจุดประสงค์ของโครงงานนี้อีกครั้ง คือเพื่อทำการเพิ่มประสิทธิภาพในการประมวลสัญญาณเชิงเลขให้แก่คอมพิวเตอร์ โดยจะเน้นในงานด้าน Image Processing โดยหลักการพื้นฐานก็คือการใช้ CPU DSP มาช่วยในการคำนวณให้กับคอมพิวเตอร์ โดย CPU ที่นำมาใช้นี้จะถูกออกแบบมา ณ พาะ มีคุณสมบัติที่สามารถคำนวณตัวเลขจำนวนมากๆได้อย่างมีประสิทธิภาพ เมื่อนำมาช่วยคำนวณแทน CPU ของเครื่องคอมพิวเตอร์ ก็สามารถที่จะช่วยลดเวลาในการคำนวณลงไปได้มาก โดยมีโครงสร้างพื้นฐานเป็นดังรูปด้านล่าง



รูปที่ 4.1 แสดง Block Diagram พื้นฐาน

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

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

ประมวลผลใน Data Memory ไปใช้ต่อ

4.2 แนวทางในการออกแบบ

การทำโครงงานชุดนี้ ได้เลือกคอมพิวเตอร์ที่ใช้ CPU ในตระกูล 80x86 ของ IBM PC หรือCompatibleมาทำการศึกษาทดลอง เนื่องจากเป็นเครื่องที่นิยมใช้กันอย่างกว้างขวาง ดังนั้นในการออกแบบเพื่อทำการ Interface กับคอมพิวเตอร์ จึงต้องพิจารณาระบบการทำงานของคอมพิวเตอร์ก่อนเป็นอย่างแรก

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

หลักการทำงานก็คือ หน่วยความจำที่อยู่บนการ์ด TMS กับ Blockของหน่วยความจำบน PC จะเป็นตัวเดียวกัน เมื่อใดที่ PC ต้องการโหลดข้อมูลภาพไปยังหน่วยความจำหรือดูข้อมูลจากหน่วยความจำ ก็สามารถกระทำบนตัวหน่วยความจำได้เลย และเมื่อการ์ด TMS ต้องการนำข้อมูลภาพไปประมวลผล ก็จะนำเอาข้อมูลที่อยู่ใน RAM ชุดเดียวกับที่ PC ใช้ในการโหลดหรือดึงข้อมูลนั่นเอง ซึ่งในลักษณะนี้จะช่วยลดเวลาในการเคลื่อนย้ายข้อมูลลงได้มาก

เมื่อพิจารณาการจัดหน่วยความจำภายในเครื่องคอมพิวเตอร์ IBM PC จะเห็นได้ว่าในพื้นที่ของหน่วยความจำในคอมพิวเตอร์ PCขนาด 1 Mbyte จะมี Blockว่างที่ไม่ได้ใช้งานอยู่ 2 Block ตามการจัดการหน่วยความจำของDOS คือ Block D (D000:0000-D000:FFFF) และ Block E (E000:0000-E000:FFFF)ว่างอยู่สามารถใช้งานได้ โดยที่แต่ละBlockมีขนาดเท่ากับ 64 Kbyte ดังนั้นจึงได้เลือกใช้วิธีการ map หน่วยความจำบนการ์ดลงไปในระบบของคอมพิวเตอร์ โดยเลือกใช้แค่เพียง 1 Block ดังนั้นคอมพิวเตอร์จะสามารถติดต่อกับหน่วยความจำภายนอกได้เหมือนกับเป็นหน่วยความจำในระบบ ในขณะที่ TMS ก็ยังสามารถติดต่อได้เช่นเดิม นั่นคือวิธีที่ใช้เพิ่มประสิทธิภาพให้กับการโอนข้อมูลระหว่าง PC กับ TMS

ในการใช้งาน เราจะทำการโหลดโปรแกรมจากคอมพิวเตอร์ไปเก็บใน RAM ส่วนของ Program Memory ก่อน ซึ่งโปรแกรมที่โหลดให้ TMS คือโปรแกรมการคำนวณต่าง ๆ ที่ต้องการ

ให้ TMS ประมวลผล จากนั้นเมื่อต้องการให้เริ่มประมวลผล ก็จะโหลดข้อมูลต่างๆไปยัง RAM ส่วน Data Memory (โดยการโหลดค่าต่างๆไปยัง RAM บนการ์ดจะผ่านทางการ map Memory บน PC ทั้งสิ้น) จากนั้นก็ทำการส่งคำสั่งจากพอร์ตของ PC เพื่อไปควบคุมให้ TMS ติดต่อกับ RAM ได้ และเริ่มทำงาน (ให้สัญญาณ reset ของ TMS ไม่ active , เป็น "1") เมื่อ TMS ประมวลผลเสร็จ PC ก็นำข้อมูลที่ได้ไปใช้จาก RAM ในส่วนของ Data Memory โดยการที่ PC จะรู้ได้อย่างไรว่า TMS ประมวลผลเสร็จแล้วนั้น จะใช้วิธีการ interrupt PC โดยให้ TMS out ข้อมูลออกมาทางพอร์ตของ TMS เพื่อเป็นสัญญาณไป interrupt PC เมื่อ PC ได้รับ interrupt ก็จะรู้เองว่า TMS ทำงานเสร็จแล้ว จากนั้นก็ทำการส่งคำสั่งควบคุมให้ RAM ติดต่อกับ PC เพื่อนำข้อมูลมาใช้งานต่อ พร้อมทั้งหยุดการทำงานของ TMS (ป้อน reset ค้างให้กับ TMS)

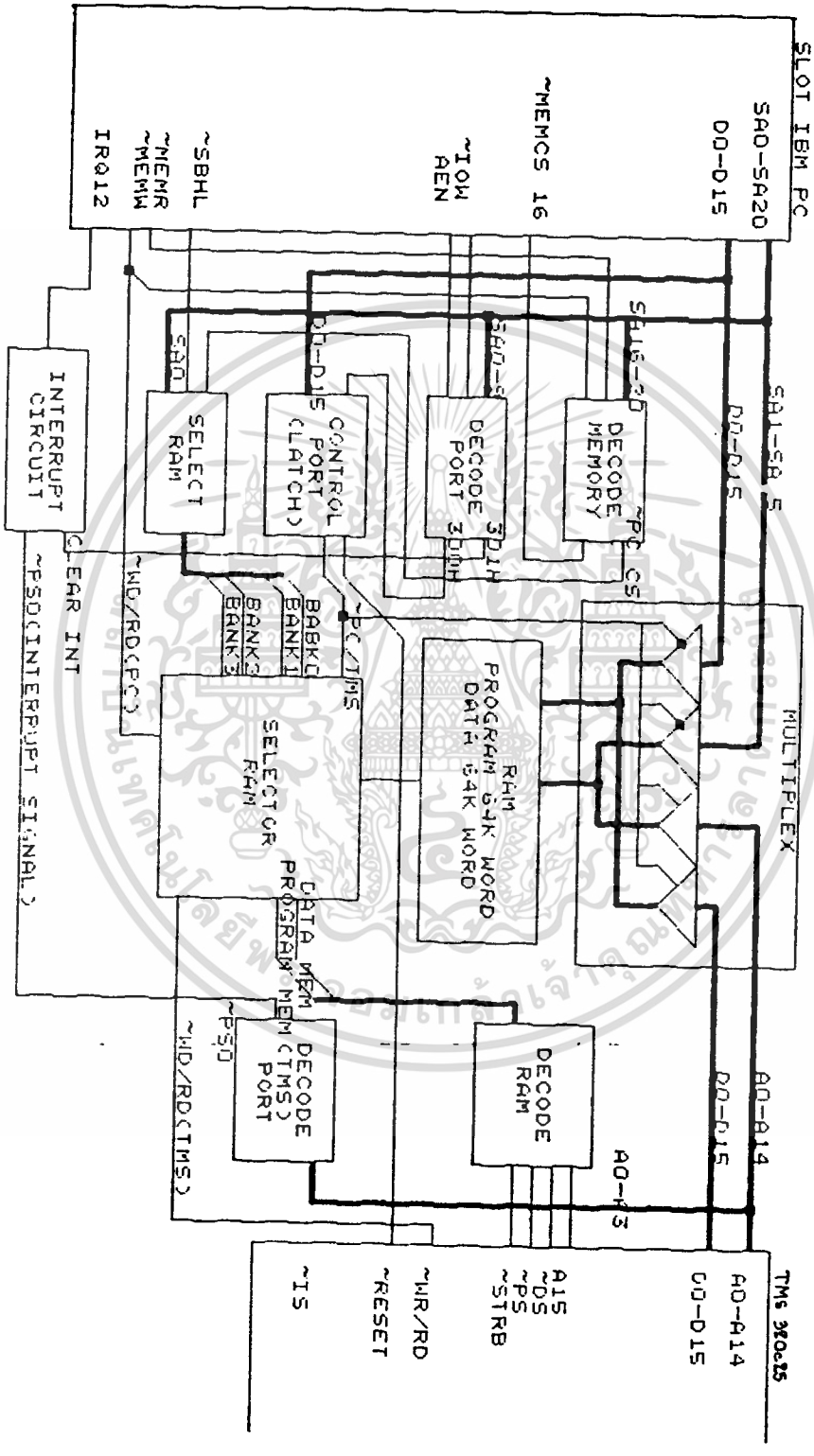
4.3 รายละเอียดของวงจร

โครงงานนี้เป็นการสร้างวงจรขึ้นมาชุดหนึ่ง โดยจะทำการติดต่อกับ PC ผ่านทางSlot ของMainboard ดังนั้นจึงมีลักษณะเป็นการ์ดสำหรับเสียบเข้ากับSlotนอกจากนี้ การที่จะให้การ์ดสามารถทำงานได้ยังต้องประกอบด้วยโปรแกรมการประมวลผลสำหรับ TMS และโปรแกรมควบคุมการทำงานของการ์ดด้วย เพื่อให้วงจรสามารถทำงานประมวลผลได้ตามต้องการ ซึ่งจากหลักการและแนวทางการออกแบบที่ได้กล่าวไปแล้ว สามารถออกแบบวงจรได้ดังในรูปวงจรแผ่นที่ 1-6 โดยมี Block Diagram ที่สมบูรณ์ในรูปที่ 4.2

หลักการทำงานของวงจร

จาก Block diagram ของวงจร วงจรนี้จะประกอบด้วย RAM จำนวน 128 k word โดยจะแบ่งเป็นส่วนของ Program Memory 64 k word และ Data Memory 64 k word ซึ่ง RAM ชุดนี้สามารถติดต่อได้ 2 ทาง คือ จาก PC และ TMS ดังนั้นจึงต้องมีวงจรสำหรับทำการสวิทช์เลือกให้ RAM ติดต่อกับทางใด ซึ่งได้แก่ วงจรใน Block Bus Selector และ Control Signal Selector สำหรับส่วนสำคัญในการประมวลผลก็คือ วงจรของ CPU TMS 320C25 ซึ่งจะประกอบด้วยวงจรในการ Decodeเลือก RAM สำหรับติดต่อกับ RAM และวงจร Decodeพอร์ตเพื่อใช้สร้างสัญญาณInterruptให้กับ PC นอกจากนี้ก็เป็นส่วนของวงจรทาง PC ได้แก่ส่วนของการDecode Address ติดต่อกับ RAM และวงจรควบคุมการทำงานต่างๆ บนการ์ด

ผ่านทางพอร์ต I/O ของ PC

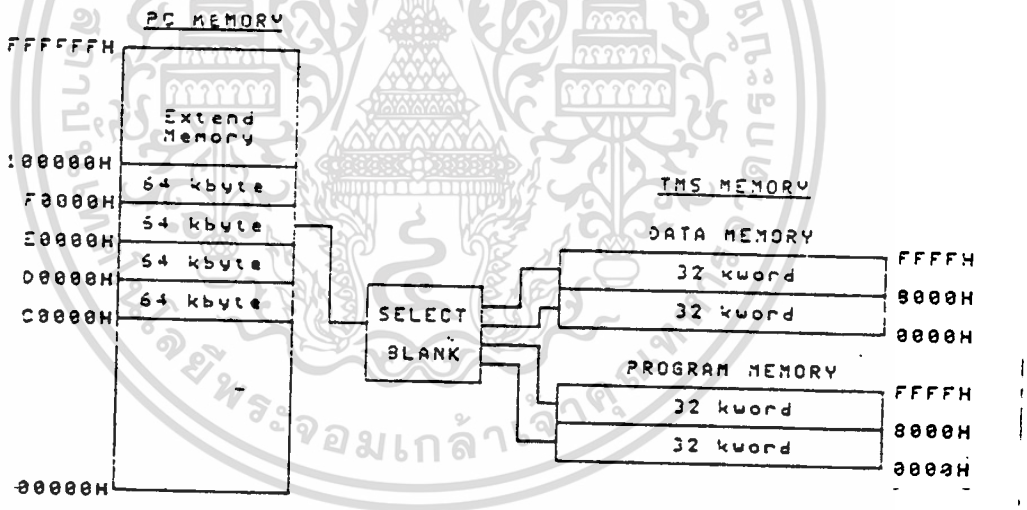


รูปที่ 4.2 Block Diagram ของการ์ด

การจัดการ map Memory บน Card

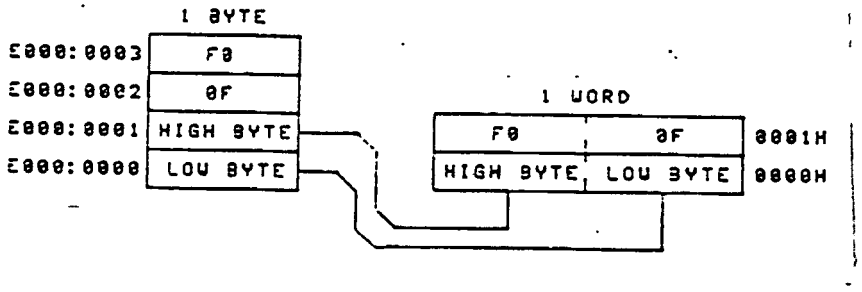
ขนาดของ RAM ที่ใช้นั้น ถูกเลือกใช้เท่ากับขนาดของหน่วยความจำที่ TMS ต้องการ ซึ่งก็คือแบ่งเป็นส่วนของ Program memory 65,536 ตำแหน่ง และ Data memory อีก 65,536 ตำแหน่ง (อย่างละ 64 k ตำแหน่ง) โดยที่ TMS มีขา Data Bus ทั้งสิ้น 16 เส้น นั้นคือ TMS ประมวลผลครั้งละ 16 bit (2 byte หรือ 1 word) ดังนั้นจึงต้องการหน่วยความจำทั้งหมดเป็นจำนวน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

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



รูปที่ 4.3 แสดงการ map RAM บนการ์ดกับ memory ของ PC

สำหรับวงจรที่ออกแบบ จะใช้หน่วยความจำบน PC เพียง 1 Blockเท่านั้น (ปกติจะใช้บล็อค E แต่ก็สามารถใช้ Block D ได้เช่นกันโดยจะต้องแน่ใจว่าหน่วยความจำส่วนนั้นไม่มีการใช้งาน และต้องว่างไม่มี RAM ในส่วนนั้น) ดังนั้น RAM บนการ์ดจะถูกแบ่งเป็น 4 ส่วน ได้แก่ Program Memory ส่วนล่างและส่วนบน กับ Data Memory ส่วนล่างและส่วนบน จะได้ส่วนละ 64 K byte (32 K word) เท่ากับ 1 บล็อค ใน PC พอดี จากการแบ่งจะได้เป็น Blank0 (BK0) DATA LOW MEMORY , Blank1 (BK1) DATA HIGH MEMORY , Blank2 (BK2) PROGRAM LOW MEMORY , Blank3 (BK3) PROGRAM HIGH MEMORY



รูปที่ 4.4 แสดงการ map memory ระหว่าง 1 word กับ 1 byte

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

<u>การทำงานของวงจร</u>

การใช้งานของการ์ดคือ การโหลดโปรแกรมและข้อมูลต่างๆ ลงบน RAM จากนั้นก็ให้ TMS ทำการประมวลผล เมื่อเสร็จสิ้นการประมวลผล TMS ก็จะส่งสัญญาณกลับมาInterrupt PC เพื่อเป็นการบอกให้รู้ จาก Block diagram ส่วนที่เป็นตัวเชื่อมระหว่าง RAM กับ PC และ TMS ได้แก่ วงจรใน Block Selector และ Control Signal Selector ซึ่งเป็นส่วนที่เลือกให้สัญญาณจากด้านใดติดต่อกับ RAM โดย Bus Selector เป็นตัวสวิทช์เลือก Address bus และ Data bus ส่วนBlock Control Signal Selector จะสวิทช์เลือกสัญญาณ ~WR และ ~CS ของ RAM โดยที่ทั้งสองBlockจะถูกควบคุมโดยส่วนBlock Control port ซึ่งสัญญาณจากBlock Control port ก็ถูกส่งมาจาก PC อีกต่อหนึ่ง

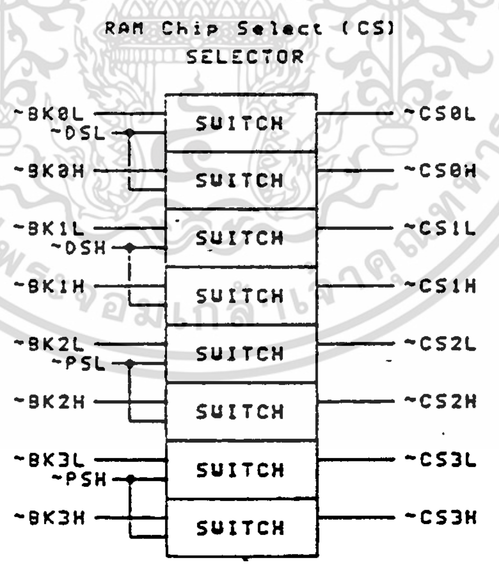
ในการติดต่อระหว่าง TMS กับ RAM จะมีวงจร decode address (ในBlock decode RAM) เพื่อสร้างสัญญาณ enable RAM (Chip select -CS) ได้สัญญาณออกมา 4 เส้น คือ ~DSL , ~DSH , ~PSL และ ~PSH โดย ~DSL และ ~DSH เป็นสัญญาณติดต่อกับ Data Memory ส่วนล่างและบน ส่วน ~PSL และ ~PSH ใช้ติดต่อกับ Program Memory ส่วนล่างและบนเช่นกัน นอกจากนี้ก็ยังมีวงจร decode port ซึ่งใช้สำหรับสร้างสัญญาณInterruptให้ PC หลังจากประมวลผลเสร็จ ซึ่งจะทำงานร่วมกับBlock Interrupt circuit อีกต่อหนึ่ง

ด้านของ PC ก็ประกอบด้วย วงจร decode memory สำหรับ Decode หน่วยความจำ

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

เมื่อเปรียบเทียบสัญญาณ Chip Select ระหว่างทางด้าน TMS กับ PC จะเห็นว่าต่างกัน 1 เท่า เนื่องจาก TMS เป็น CPU 16 bit แท้ 1 Addressมี Data 16 bit ในขณะที่ทาง PC ไม่ใช่ CPU 16 bit โดยตรง คือสามารถทำงานได้ทั้ง 8 bit และ 16 bit แต่หน่วยความจำ 1 AddressจะมีDataอยู่เพียง 8 bit ในกรณีที่ต้องการให้ทำงานแบบ 16 bit จะแบ่งเป็นByteคู่และByteคี่ หรือByteล่างและByteบน ดังเช่น ในโครงงานนี้จะให้ทำงานแบบ 16 bit ซึ่งจะมีสัญญาณควบคุมแยกกัน คือ A0 สำหรับไบท์ล่าง(คู่) และ SBHL สำหรับไบท์บน(คี่) ทำให้ PC ต้องใช้สัญญาณมากกว่า 1 เท่า

สัญญาณ Chip Select ทั้งหมด จะถูกส่งให้ส่วน Control Signal Select ซึ่งจะทำการจัดการเลือกสัญญาณให้ RAM อีกที



รูปที่ 4.5 แสดงช่องทางงานภายในของส่วน Control Signal Select

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

รายละเอียดของวงจรแต่ละส่วน

วงจร CPU

แสดงได้ในรูปวงจรแผ่นที่ 1 มี CPU คือ TMS 320C25 ซึ่งมี Address และ Data bus อย่างละ 16 เส้น ส่วนประกอบที่ทำให้ CPU ทำงานได้ คือ Crystal ซึ่งต่ออยู่ระหว่างขา X2 และ X7 ของ TMS ใช้สร้างสัญญาณ Clock ภายในตัว TMS โดยเลือกใช้งานที่ความถี่เพียง 20 MHz เท่านั้น จากความสามารถเต็มที่ซึ่งทำงานได้ที่ 40 MHz เนื่องจากโครงการนี้เป็นการเริ่มต้นศึกษาทดลองใช้งาน CPU เบอร์นี้เป็นครั้งแรก ดังนั้นถ้าหากเลือกใช้ความถี่ 40 MHz ที่ความถี่สูง การเดินสายจะมีปัญหามากขึ้น เพื่อลดปัญหาจึงใช้ที่ความถี่ต่ำลง ซึ่งการเดินสายไม่มีผลมากนัก

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

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

วงจร RAM

แสดงได้ในรูปวงจรแผ่นที่ 2 ซึ่งประกอบด้วย RAM จำนวน 8 ตัว เบอร์ MT5C2568 เป็น Static RAM ขนาด 32K * 8 bit ซึ่งเป็น RAM ชนิดพิเศษที่มีเวลาในการเข้าถึงต่ำ เพื่อลดความยุ่งยากในส่วนของวงจร CPU ที่จะต้องมีวงจรสร้าง Wait State ถ้าใช้ RAM ที่มีเวลาเข้าถึงมาก

ลักษณะการต่อ RAM ขา Address และขา ~WE/RD จะต่อร่วมกันหมด ส่วน Data bus จะแบ่งเป็น 2 ส่วน แถวบน U3-U6 จะต่อกับ data Bus D8-D15 ใช้เก็บ Data Byte บน ส่วนแถวล่าง U7-U10 จะต่อกับ Data Bus D0-D7 ใช้เก็บ Data Byte ล่าง สำหรับขา ~CS ของแต่ละตัวจะแยกกัน ซึ่งเป็นผลจากวิธีการติดต่อระหว่าง RAM กับ PC ซึ่งแบ่งเป็นBlank 4 Blank โดยแต่ละBlankแบ่งเป็นByteคู่และByteคี่อีก ทำให้ RAM แต่ละตัวต้องมี ~CS แยกกัน

วงจรจัดการสัญญาณ ~CS จะอยู่ในรูปวงจรแผ่นที่ 4 มี IC 74LS157 2 ตัว ทำหน้าที่สวิทช์เลือกสัญญาณจาก PC และ TMS จากรูปวงจรจะเห็นว่าสัญญาณจาก PC จะมีด้วยกัน 8 เส้น ในขณะที่จาก TMS จะมีเพียง 4 เส้น โดยInputของ IC 74LS157 ทางด้าน TMS จะมีการต่อขา inputเป็นคู่ ทำให้กรณี TMS ติดต่อ RAM จะ active RAM 2 ตัวพร้อมกัน คือ ByteบนและByteล่างทำงานพร้อมกัน ส่วนทางPCการ active Byteบน กับ Byteล่างจะแยก

สัญญาณ select RAM ของ TMS มาจาก IC 74LS139 2 ชุด (IC 1 ตัวมี Decoder 2-4 อยู่ 2 ชุด) ทำการ Decode สัญญาณจาก TMS โดยชุดแรกสำหรับ Decode เลือก Data Memory ซึ่งใช้สัญญาณ ~DS , A15 และ ~STRB ร่วมกัน อีกชุดหนึ่งสำหรับ Decode เลือก Program Memory ใช้สัญญาณ ~PS, A15 ~STRB โดย A15 ใช้สำหรับ Decode Address ส่วนบนกับส่วนล่าง ~STRB เป็นสัญญาณที่Activeเมื่อเป็นการติดต่อกับหน่วยความจำ สำหรับ ~DS และ ~PS เป็นสัญญาณที่activeเมื่อ CPU ติดต่อกับหน่วยความจำส่วน Data และProgram ตามลำดับ จากการ decode จะได้เป็นสัญญาณ ~DSL , ~DSH , ~PSL และ ~PSH คือ สัญญาณเลือก RAM Data Memory ส่วนล่าง , Data Memory ส่วนบน , Program Memory ส่วนล่างและ Program Memory ส่วนบน ตามลำดับ

ส่วนทางด้าน PC สัญญาณ select ทั้ง 8 เส้น ได้จากวงจร Gate ในรูปวงจรแผ่นที่ 4 ซึ่งประกอบด้วย OR Gate 8 ตัว แบ่งเป็น 2 ชุด ชุดหนึ่งจะมีinputหนึ่งต่อกับขา A0 ของ PC สำหรับสร้างสัญญาณติดต่อ RAM ส่วนByteล่าง (Byteคู่) อีกชุด inputจะต่อกับสัญญาณ ~SBHL จาก PC เพื่อเลือกติดต่อกับByteบน (Byteคี่) (สัญญาณ ~SBHL นี้จะมีเฉพาะในSlotแบบ AT เป็นส่วนที่ขยายจากแบบ XT จะ active กรณีที่ต้องการใช้งานแบบ 16 bit เท่านั้นคือจะต้องมี

สัญญาณลอจิก "0" ป้อนให้กับขา ~MEMCS16 ก่อนในช่วงCycleแรกๆ ของการอ่าน/เขียนหน่วยความจำหรือพอร์ต โดยสัญญาณนี้จะActiveเมื่อมีการอ่านหรือเขียนข้อมูลทาง Data Bus (Byte บน D8-D15) ส่วนinputอีกขาหนึ่งของ Gate แต่ละตัวจะต่อกับ IC 74LS138 ซึ่งทำหน้าที่ Decode เลือกBlankโดยสัญญาณเลือกBlank BKEN,BKSEL0 และ 1 มาจาก Control port outputของ 74LS138 จะใช้เพียง4 เส้นเท่านั้นคือมี 4 Blank แต่ละขาจะต่อกับ Gate 2 ตัว คือ ตัวที่ Decode Byteล่าง และ Byteบนอีกตัว เพราะฉะนั้น OR Gate แต่ละตัวจะเลือก active(Low) ต่างๆกัน คือ ~BK0L-~BK3L สำหรับ Blank0-3 Low Byte และ ~BK0H-~BK3H สำหรับ Blank0-3 High Byte

<u>วงจร PC Decode</u>

ส่วนนี้จะเป็นส่วนที่ติดต่อกับ PC โดยตรง คือ ทำการ Decode memory และ Port ของ PC ซึ่งวงจรแสดงในรูปวงจรแผ่นที่ 5 จากรูปวงจร Decode port ได้แก่ IC 74LS688 และ 74LS138 โดย 74LS688 จะทำการเปรียบเทียบขา Address A3-A9 กับค่าตั้งไว้ที่ Dip Switch ตามเบอร์พอร์ตที่ใช้ จากนั้นoutputของ 74LS688 จะไปเข้า 74LS138 เพื่อ Decode ร่วมกับ A0-A2 อีกครั้งได้เป็นสัญญาณ Select port ในปริญญานิพนธ์นี้เลือกใช้ที่พอร์ต 300H กับ 301H โดยพอร์ต 300H จะเป็น Control port สัญญาณ Select port จาก 74LS138 ขาแรกจะต่อเข้าวงจร Latch IC 74LS273 เพื่อ Latch ข้อมูลจาก D0-D8 ใช้เป็นสัญญาณควบคุมวงจรส่วนอื่นต่อไป ที่ขา CLR ของ 74LS273 นั้นจะต่อกับสัญญาณ Rest DRV เพื่อให้ขณะที่ Reset PC ก็จะ Reset ตัว 74LS273 ด้วย ทำให้สัญญาณ ~BKEN เป็น 0 นั่นคือเป็นสภาวะที่ไม่มีส่วนใดติดต่อกับ RAM บนการ์ดได้ และสัญญาณ Reset TMS เป็น 0 ด้วย หรือกล่าวได้ว่า เป็นการ Reset การ์ดไปด้วยพร้อมกับ PC

การ Decode หน่วยความจำจะใช้ขา Addressเพียง 4 เส้น คือ A16-A19 เพราะว่าเป็นการ Decode Blockของหน่วยความจำที่จะใช้เท่านั้น โดยใช้ IC Comparator 4 bit 74LS85 เป็นตัวเปรียบเทียบกับ Dip switch ที่ set ค่าของBlockที่ต้องการไว้ สัญญาณ outputจาก 74LS85 จะต่อเข้าวงจร Gate ร่วมกับสัญญาณอ่าน/เขียนหน่วยความจำ (~MEMR & ~MEMW) สร้างเป็นสัญญาณ ~PC CS ป้อนให้กับวงจรอื่นต่อไป สำหรับ active RAM

จากวงจรแผ่นที่ 5 ยังมีสัญญาณจาก PC ที่สำคัญอีก 2 เส้น คือ AEN และ ~MEM CS16 สัญญาณ AEN เป็นสัญญาณที่บอกถึงกรณีมีการทำ DMA ซึ่งจะต้อง Disable พอร์ตอื่นๆ ที่ไม่เกี่ยวให้หมด ดังนั้นในวงจร Decode port จึงต้องมีการ Decode สัญญาณ AEN ไว้ด้วย ส่วน

สัญญาณ ~MEM CS16 เป็น input ของ PC ใช้เพื่อบอกให้ PC ทำการอ่าน/เขียนหน่วยความจำแบบ16 bit ซึ่งถ้าไม่มีสัญญาณนี้ป้อนให้ ก็จะไม่มีการส่ง Data ออกมาบน Data Bus D8-D15 สัญญาณ ~SBHL ก็ไม่activeด้วย คือจะเป็นการทำงานแบบ 8 bit ตามธรรมดา จึงต้องมีการป้อนสัญญาณนี้ให้ด้วย โดยสัญญาณนี้จะต้องป้อนให้-PC ในช่วงเวลาแรกของCycleการอ่าน/เขียนข้อมูล คือหลังจากที่ค่า Address ถูกปล่อยลง Bus ต่อจากนั้นจะต้องมีการป้อนinputให้กับขา ~MEM CS16 ทันที จึงจะสามารถทำงานแบบ 16 bit ได้ ดังนั้นในวงจรจึงใช้สัญญาณจาก 74LS85 มาทำการ enable Buffer เพื่อป้อนลอจิก "0" ให้กับขา ~MEM CS16 สาเหตุที่ต้องมีBufferเนื่องจากขา ~MEM CS16 หรือขาinputทั่วไปของSlotจะต้องการกระแส sink ประมาณ 20 mA จึงต้องต่อ Buffer และต้องเป็นBufferที่มีoutputเป็น Tri-State หรือ Open Collector ด้วย

Bus selector

วงจร Bus Selector เป็นส่วนที่สวิทช์เลือก Address Bus และ Data Bus ระหว่าง PC กับ TMS ให้ต่อกับ RAM วงจรจริงแสดงในรูปวงจรแผ่นที่ 3 วงจรนี้จะใช้Buffer 2 ทาง 74LS245 จำนวน 8 ตัว เป็นตัวจัดการ แบ่งเป็นสำหรับ Data Bus และ Address อย่างละ 4 ตัว โดยโครงสร้างการต่อของ Data Bus และ Address Bus จะคล้ายกัน คือ ทางด้านขา B ของ 74LS245 จะต่อกับ Bus ของ PC และของ TMS แต่ที่ชุดขา A จะมีการต่อถึงกันเป็นคู่ระหว่างชุดของ PC กับของ TMS แล้วไปต่อเข้าขา RAM ต่อไป ลักษณะการต่อของขา A นั้น สำหรับ Data Bus จะต่อแบบให้ D0 ของ PC ต่อกับ D0 ของ TMS ไล่ไปจนถึง D15 ของ PC กับ D15 ของ TMS แต่สำหรับ Address Bus จะต่างจาก Data Bus เล็กน้อย คือขา A0 ของ TMS จะต่อกับขา A1 ของ PC ไล่ไปจนถึง A14 ของ TMS ซึ่งจะต่อกับ A14 ของ PC สำหรับการที่ PC ไม่ใช้ A0 เนื่องจาก A0 ถูกใช้สำหรับเลือก Address คู่ร่วมกับ ~SBHL สำหรับเลือก Address คี่ ซึ่งจะพิจารณาอีกแบบก็คือ 1 Address ของ TMS จะเท่ากับ 2 Address ของ PC นั่นคือ สัญญาณ A1 ของ PC จึงตรงกับ A0 ของ TMS

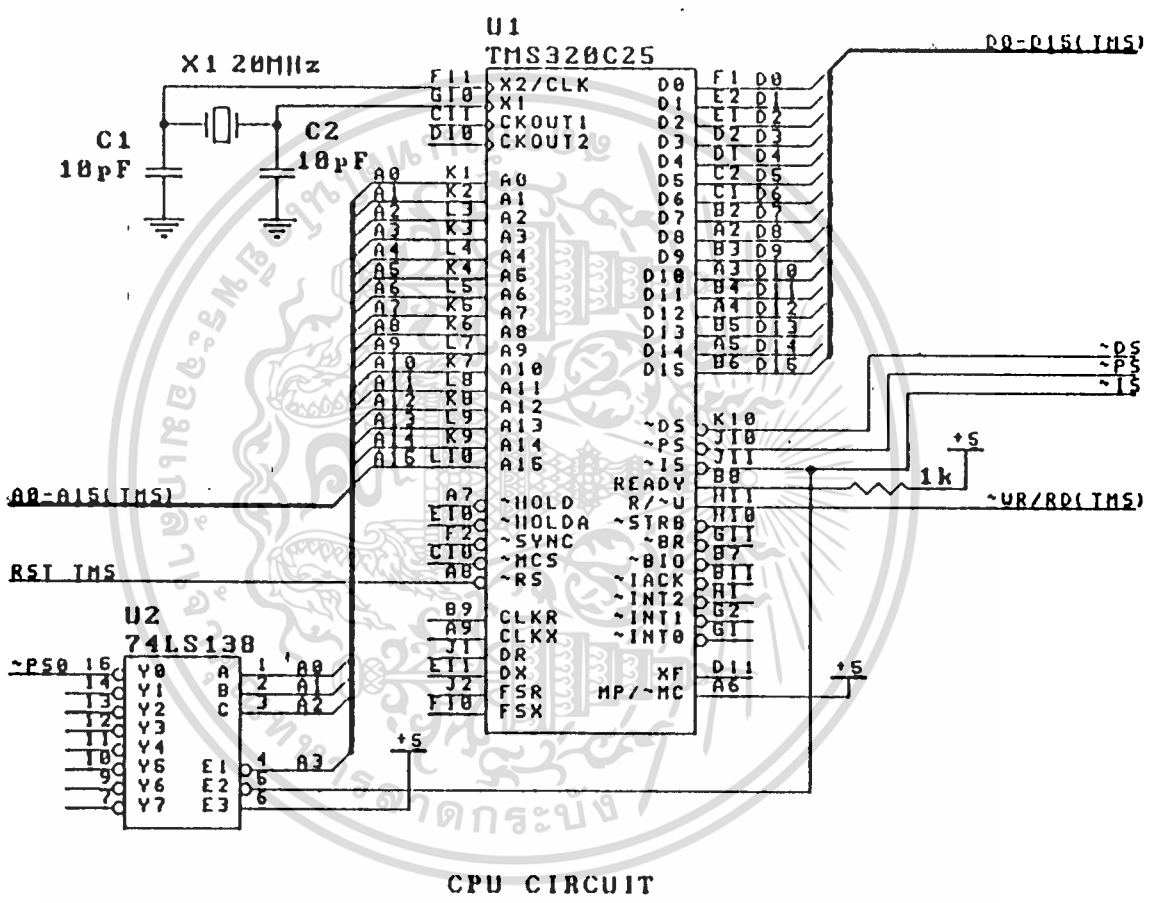
สำหรับการควบคุมการสวิทช์เลือกนั้นใช้สัญญาณ ~PC/TMS จาก Control port ซึ่งจะป้อนเข้ากับขา ~E ของ 74LS245 เพื่อ Enable Buffer โดยที่ชุดBufferของ TMS นั้น สัญญาณ ~PC/TMS จะผ่าน Inverter ก่อน เพื่อให้ทำงานที่สัญญาณ ~PC/TMS เป็นลอจิก "1" ส่วนBufferด้าน PC นั้น สัญญาณ ~PC/TMS จะถูกต่อเข้า OR Gate ก่อน ร่วมกับ สัญญาณ Decode หน่วยความจำ ~PC CS เพื่อที่ให้Bufferทำงานเฉพาะกรณีที่อ้าง Address ตรงBlock

เท่านั้น

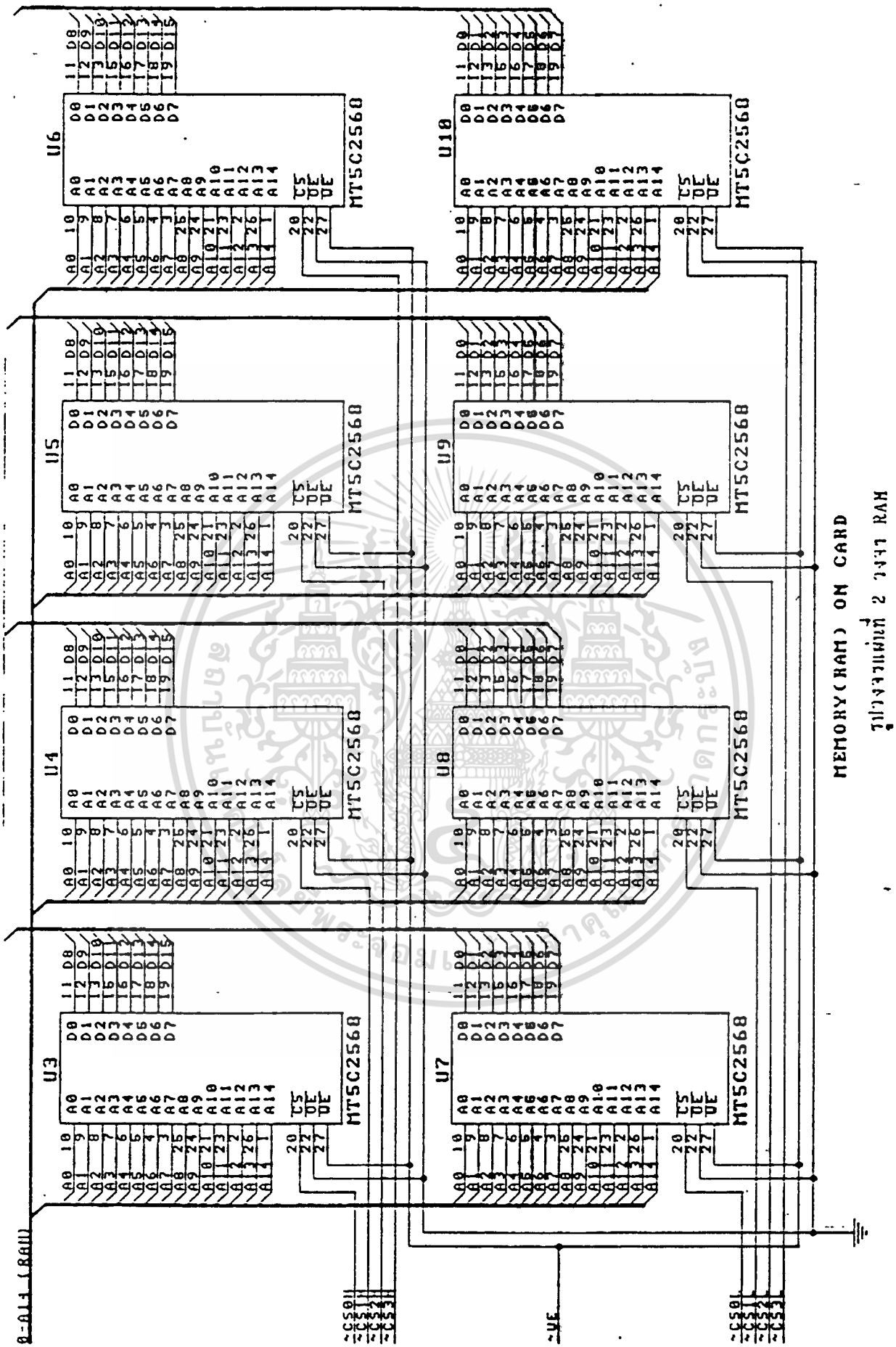
นอกจากนี้ ในส่วน Data selector นั้น ขา DIR ของ 74LS245 จึงจะต้องถูกควบคุมด้วยเช่นกัน ซึ่งต่างจากส่วน Address ที่ขา DIR จะต่อลง Ground เพราะเป็นการติดต่อทางเดียวแต่สำหรับ data bus ต้องติดต่อสองทาง โดยสัญญาณที่ใช้ควบคุมได้แก่ สัญญาณ ~WE ของ RAM นั่นเอง ซึ่งสัญญาณนี้ก็ผ่านการ select มาแล้วอีกต่อเช่นกัน จากรูปในวงจรแผ่นที่ 3 จะเห็นว่าสัญญาณ ~WE จะถูก Select ร่วมมากับ Address bus เนื่องจากBufferในตัว 74LS245 เหลืออีก 1 คู่ ซึ่งสัญญาณ ~WE ที่ได้ จะถูกนำไปต่อเข้าที่ RAM ทุกตัว เพื่อบอกสภาวะการเขียนหรืออ่าน RAM และยังต่อเข้า Bufferของ Data Selectorเพื่อควบคุมทิศทางของ Buffer 2 ทาง ตามการเขียนหรืออ่าน RAM

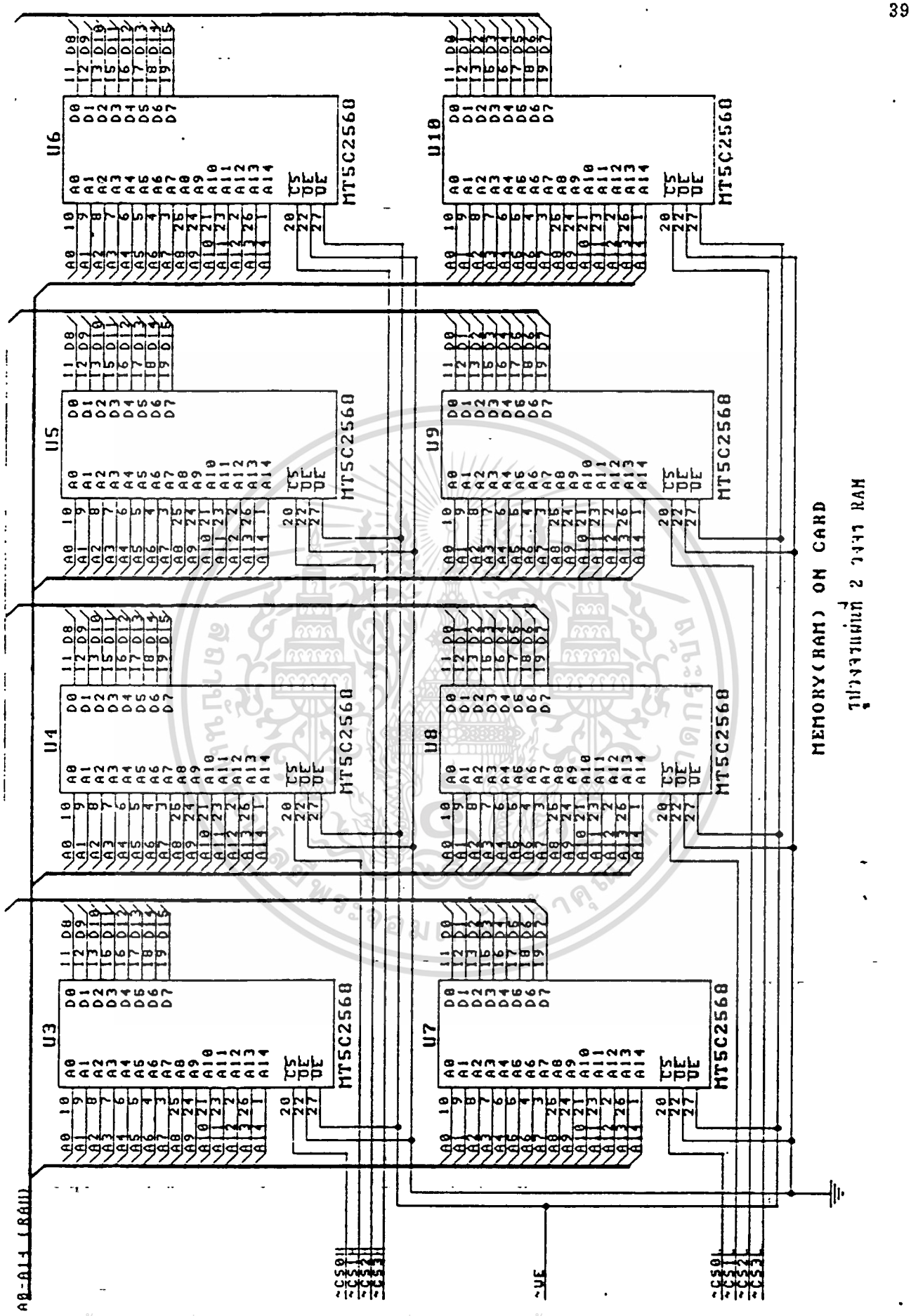
<u>วงจรสร้างสัญญาณอินเตอร์รัพต์ PC</u>

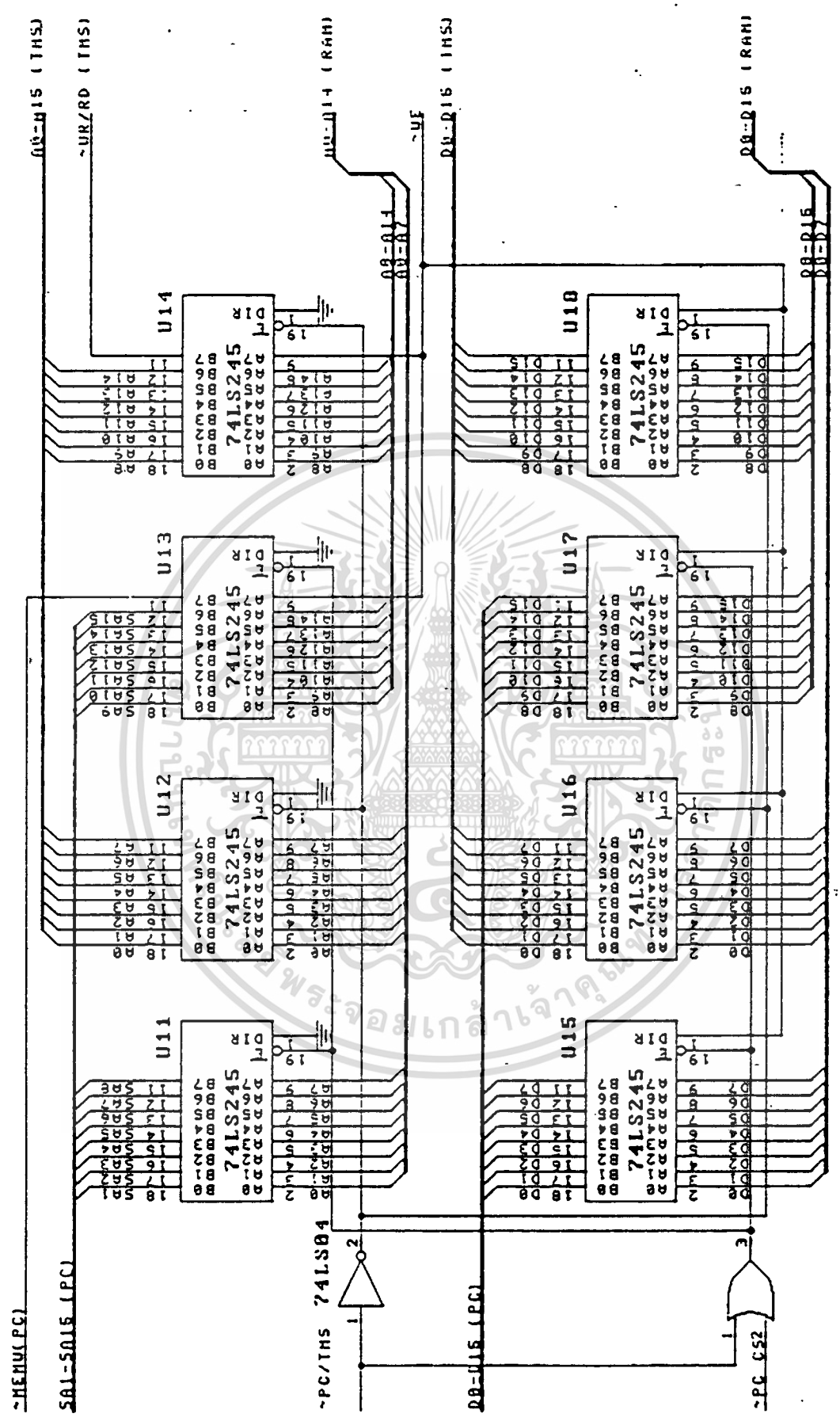
วงจรนี้ได้แสดงในรูปวงจรแผ่นที่ 6 อุปกรณ์ที่ทำหน้าที่หลักคือ D-FLIP FLOP 74LS74 โดยจะรับสัญญาณจากทาง TMS ได้แก่สัญญาณ ~WE Or กับ สัญญาณ ~PS0 (สัญญาณจากการ Decode port 0) ซึ่งactiveที่ขอบขาขึ้นของสัญญาณ เมื่อมีสัญญาณจาก TMS ที่ขา Q ของ D-FLIP FLOP ซึ่งปกติจะเป็น 0 อยู่ ก็จะเปลี่ยนเป็น 1 ส่งผ่านBuffer เพื่อไปInterrupt PC สำหรับที่ขา CLR ของ D-FF นั้นจะเป็นวงจร reset เพื่อ CLR สัญญาณInterrupt โดยใช้สัญญาณ reset DRV จากSlotสำหรับ Crear ตอนเริ่มต้นทุกครั้ง และสัญญาณ ~CLR INT outจากการ Decode หมายเลข 301H เพื่อใช้ Clear Interrupt หลังจาก PC รับรู้การInterrupt



รูปวงจรแผ่นที่ 1 วงจร CPU

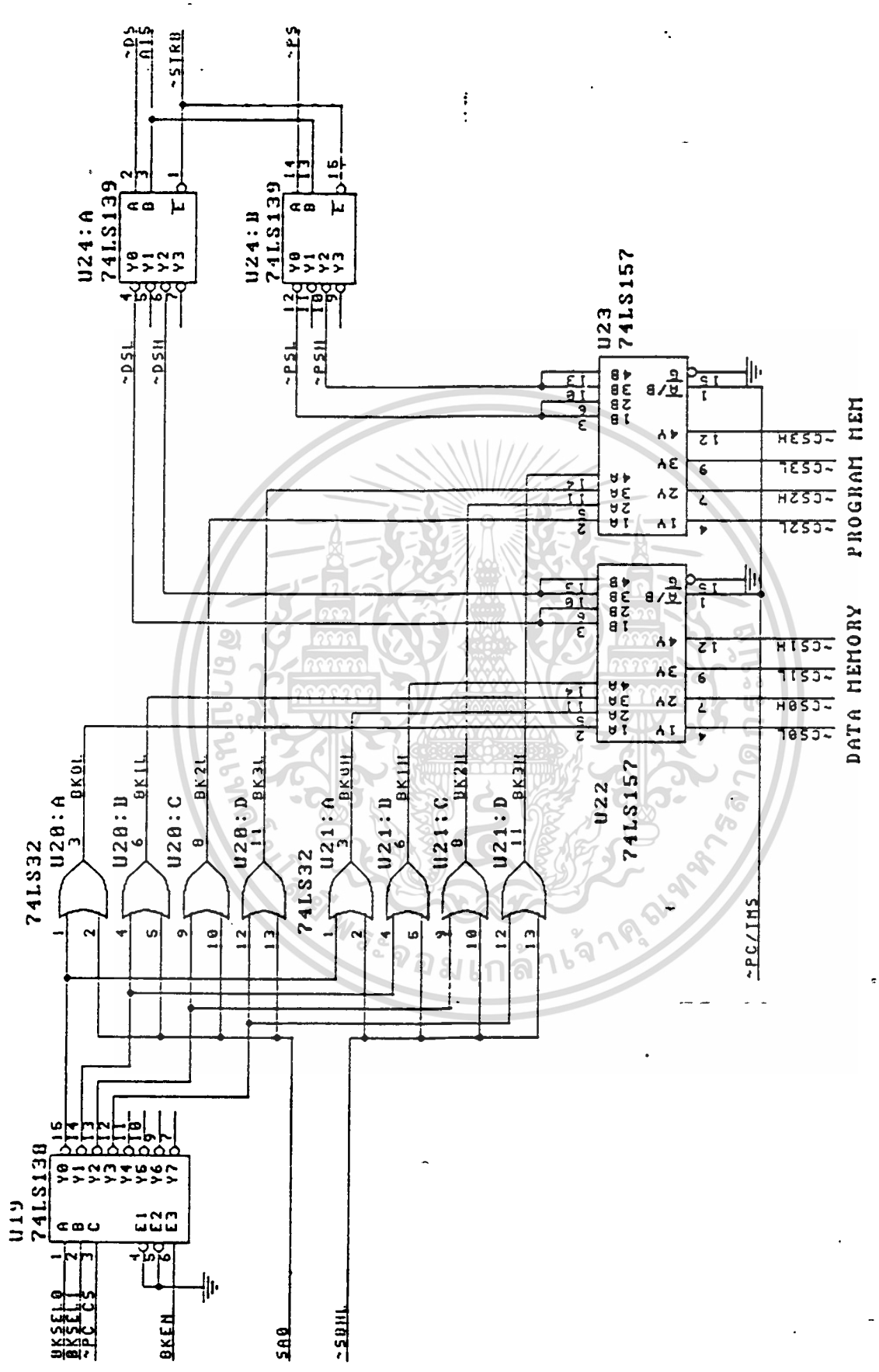




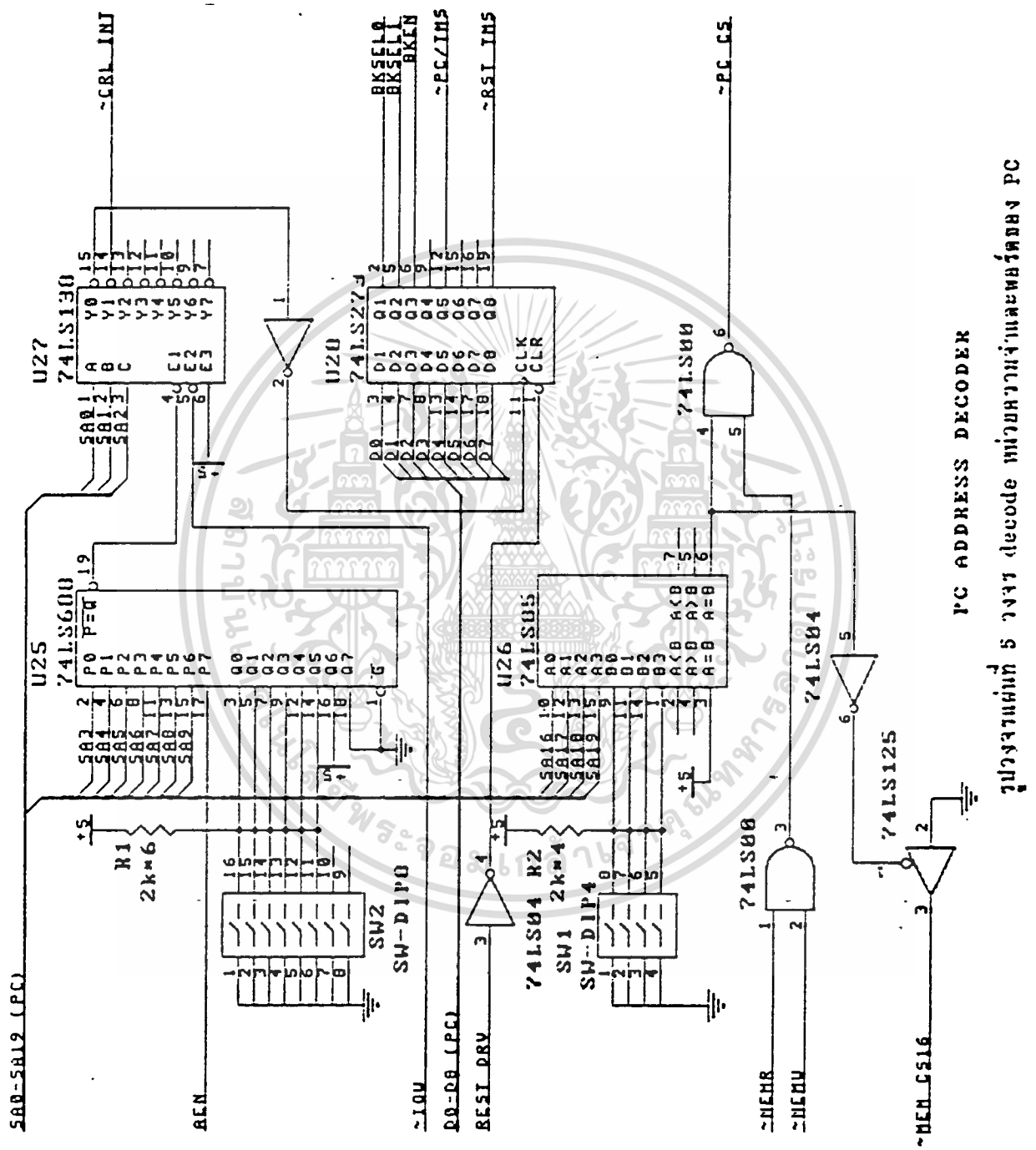


ADDRESS AND DAT BUS SELECTOR

รูปวงจรแผ่นที่ 3 วงจรสวิทช์เลือก Address และ Data Bus

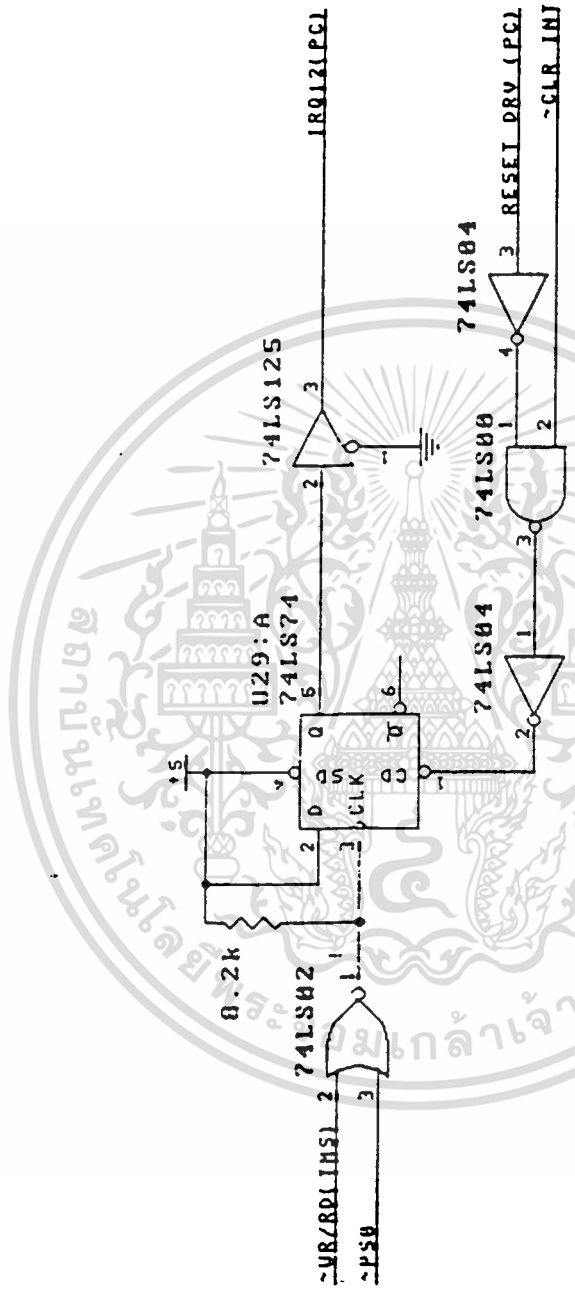


รูปวงจรแผ่นที่ 4 วงจรถอดรหัสหน่วยความจำ RAM



รูปวงจรแสดง decode หน่วยความจำและพอร์ตของ PC

PC ADDRESS DECODER



รูปวงจรแผ่นที่ ๑ แสดงการสร้างสัญญาณอินเตอร์รัพต์

4.4 การประยุกต์ใช้งานการ์ด

จากบทที่ผ่านมาแล้วทั้งหมดนั้น จะเห็นได้ว่าเรามอง Image Processing Card เป็นเพียง Co-processor ให้กับ PC เท่านั้น ดังนั้นการประยุกต์ใช้งานของการ์ดนี้เราจะทำการเปรียบเทียบความเร็วที่ได้จากการประมวลผล 2 ทางคือ

- โดยการใช้ PC ประมวลผลอย่างเดียวโดยเขียนโปรแกรมบนภาษา C
- ใช้ภาษา C ควบคุมการประมวลผลของการ์ด โดยในส่วนของการ์ดจะใช้ภาษา Assembly ของ CPU TMS320C25 ในการประมวลผล

ซึ่งทั้ง 2 ทางนั้น จะประมวลผลในหัวข้อเรื่องเดียวกันคือ การวิเคราะห์แยกตัวอักษรออกจากกัน เป็นตัวๆ จากหน้ากระดาษหนึ่งเพื่อส่งตัวอักษรที่ได้ให้อีกกลุ่มหนึ่งเพื่อทำการวิเคราะห์ว่าเป็นตัวอักษรอะไรโดยขั้นตอนการทำงานของโปรแกรมต่างๆจะเป็นดังนี้

4.4.1 การประมวลผลด้วย PC ล้วนๆ (ภาษา C อย่างเดียว)

จุดประสงค์หลักของการประมวลผลนี้ ก็คือการแยกตัวอักษรออกเป็นตัวๆ ซึ่งจะใช้หลักการของการเขียน Histogram เพื่อทำการวิเคราะห์แยกความแตกต่างระหว่างสีขาวกับสีดำ ซึ่งขั้นตอนการประมวลผลต่างๆ จะเป็นดังนี้คือ

- ทำการ scan ภาพ จากเครื่อง scanner แล้วเก็บภาพลงใน file รูป Bitmap format และทำการแสดงผลออกทางจอภาพด้วยโปรแกรม DISPLAY.C
- เนื่องจาก file ภาพที่เรา scan มาเป็น file Bitmap format ซึ่งจะประกอบด้วยส่วนหัว (Header) และตัวข้อมูลจริง (Data 256 gray file) ซึ่งส่วนหัวจะเป็นการบอกลักษณะของภาพทั้งขนาด,ความสูง,ความกว้าง ฯลฯ ส่วนข้อมูลที่ได้จากการ scan จะเป็นภาพซึ่งดูเหมือนมองสะท้อนกระจก ดังนั้น จึงต้องทำการตัดข้อมูลเฉพาะ ที่เราต้องการ (ขนาด 128 x 128)ด้วยโปรแกรม 128.C ซึ่งเราสามารถดูภาพผ่านจอภาพที่ได้จากการตัดขนาด 128 x 128 ด้วยโปรแกรม LOADPIC.C
- เราจะเห็นว่าภาพที่ได้จากการ scan นั้นมีสัญญาณรบกวน (noise) มากเกินไป และ แต่ละจุดของภาพก็มีค่าตั้งแต่ 0-255 ซึ่งไม่เหมาะแก่การวิเคราะห์ Histogram ดังนั้นเราจึงต้องทำการ High Pass filter ภาพ เพื่อให้ได้ภาพที่คมชัดด้วยโปรแกรม FILTER.C และทำการเขียน Histogram ของภาพรวมทั้งหมดด้วยโปรแกรม HISTRO.C เพื่อทำการ

ตัดภาพให้เหลือบิตเดียวคือสีขาวกับสีดำเท่านั้นด้วยค่า threshold value

- เมื่อเราได้ภาพขนาด 128 x 128 และแต่ละจุดภาพมีแค่สีขาวกับสีดำเท่านั้นแล้ว เราก็พร้อมที่จะทำการวิเคราะห์เพื่อแยกตัวอักษรออกเป็นตัวๆด้วยโปรแกรม CLIP.C

ซึ่งรายละเอียดของโปรแกรมต่างๆ เป็นดังนี้คือ

1) การแสดงผลออกทางจอภาพ

ในส่วนของโปรแกรมนี้ เป็นการ display ออกจอภาพ ซึ่งเป็น Bitmap format โดยส่วนหัวจะเริ่มตั้งแต่ bit ที่ 1 ถึงค่า offset ต่อจากนั้นจะเป็น dataจริง ซึ่งจะกลับจากซ้ายมาขวาและบนมาล่าง ซึ่งเป็นภาพที่เสมือนมองจากกระจก โดยส่วนของ Header จะมี format ดังนี้

- 1-14 bit แรก เป็นส่วนของBITMAPFILEHEADER ซึ่งจะประกอบด้วย
  bf Type 2 บิต ใช้บอกชนิดของภาพ
  bf Size 4 บิต ใช้บอกขนาด file รวม
  bf off Bits 4 บิต ใช้บอกค่า offset ของ file
- 15-51 bit ต่อมาเป็นส่วนของ BITMAPINFOHEADER ซึ่งจะประกอบด้วย
  bi Size 4 บิต ใช้บอกขนาด file ไม่รวมค่า offset
  bi Width 4 บิต ใช้บอกขนาดความกว้างของภาพ
  bi Height 4 บิต ใช้บอกขนาดความสูงของภาพ
- 52-60 bit ต่อมา เป็นส่วนของ tag RGBQUAD ซึ่งใช้บอกว่าเป็นภาพขาวดำ 256 gray level

ซึ่งเมื่อเราทำการอ่านค่า header file แล้ว จะทำการเก็บค่า data แต่ละจุดที่ละจุดลงใน memory ที่จองไว้ในเครื่อง PC จนครบขนาดภาพแล้วทำการเปลี่ยนจาก mode graphic เป็น mode text โดยผ่าน bios ของ DOS แล้วทำการ plot ค่า ทีละจุดออกจอภาพจนครบขนาด file ซึ่งขั้นตอนนี้อาจทำการเก็บค่าข้อมูลใหม่ลงใน file ชื่อใหม่ก็ได้ แล้วทำการเปลี่ยนจาก mode graphic เป็น mode text ดังเดิม ดัง flowchart

2. ทำการ กรองภาพ (High pass filter) ให้ได้ภาพคมชัดขึ้น โดยการแปลงแบบ FFT (fast fourier transfer) แบบ 2 มิติ โดยทำตามแนวแกน x

ขนาด128บรรทัดก่อนแล้วจึงแปลงFFTตามแนวแกนy อีกครั้ง เมื่อแปลงภาพเป็นโดเมนความถี่แล้วก็จะทำการตัดภาพ (HPF) โดยใช้ค่าที่ 3 เป็นตัวตัดแล้วแปลง IFET 2 มิติให้เป็นโดเมนเวลาอีกที ดัง flow chart

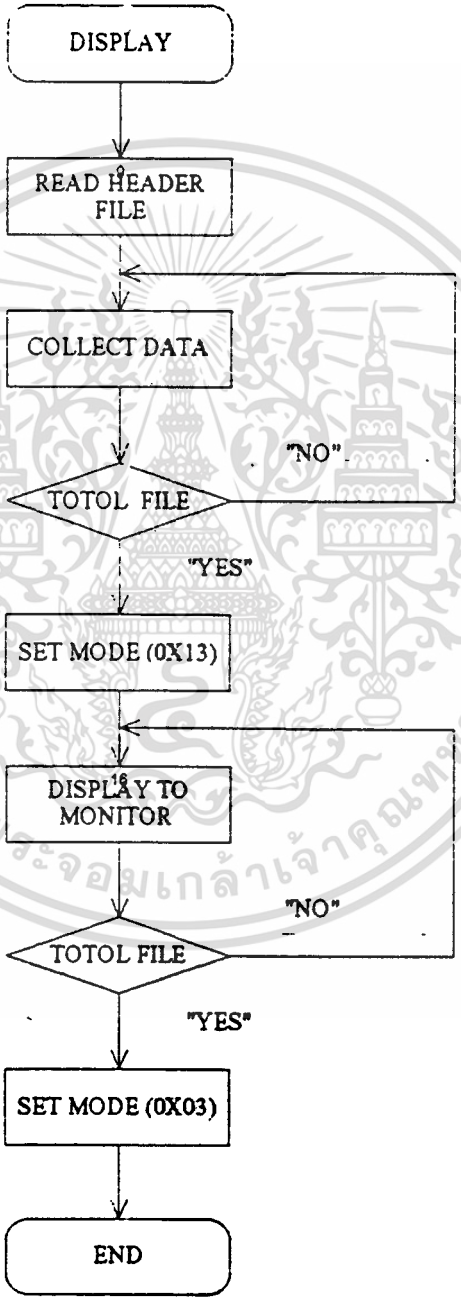
3. เมื่อทำการ Filterให้ได้ภาพที่คมชัดแล้วก็จะทำการเขียน HISTOGRAM ของภาพ เพื่อหาค่า theshold ของภาพ แล้วทำการตัดให้เหลือบิตเดียวที่ค่า theshold

4. จากนั้นทำการ clip ภาพ ให้เหลือแค่ตัวอักษรเดียว โดยการเขียน Histogram เฉพาะแกนตั้งของภาพ แล้วทำการแยกบรรทัดตรงบริเวณรอยต่อระหว่างสีขาวกับสีดำ จากนั้นก็จะทำการตัด column ของภาพโดยการเขียน Histogram เฉพาะแกนนอนของภาพ แล้วทำการแยก column ตรงบริเวณรอยต่อระหว่างสีขาวกับสีดำ ในแต่ละบรรทัด แล้วทำการแยกบรรทัดอีกครั้งในแต่ละ column ที่ตัดไว้แล้วดัง flowchart

TopChart

Friday, March 10, 1995

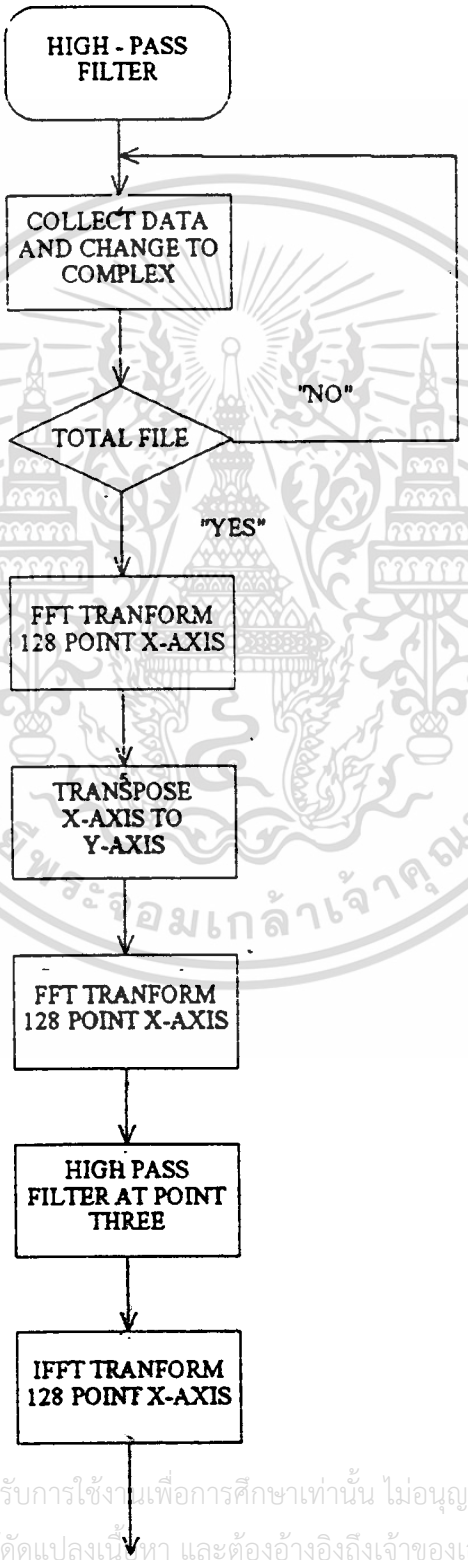
3:00 PM



TopChart

Friday, March 10, 1995

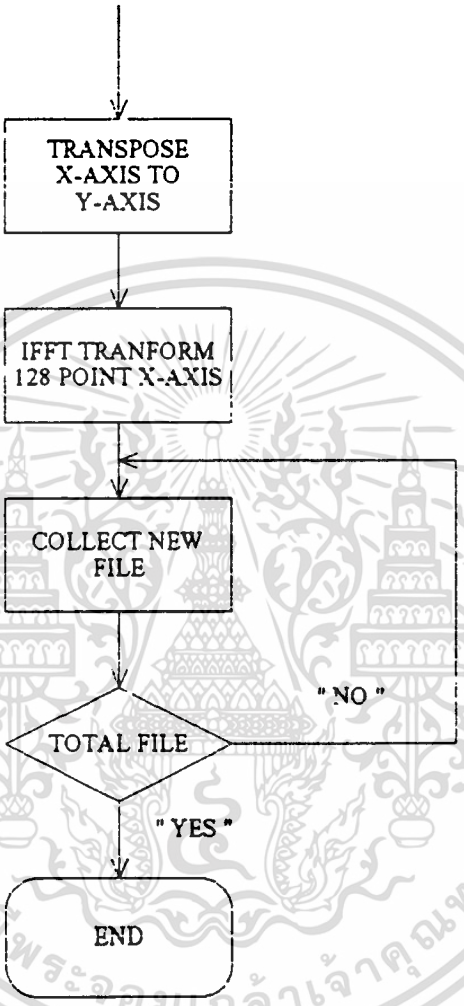
4:33 PM



TopChart

Friday, March 10, 1995

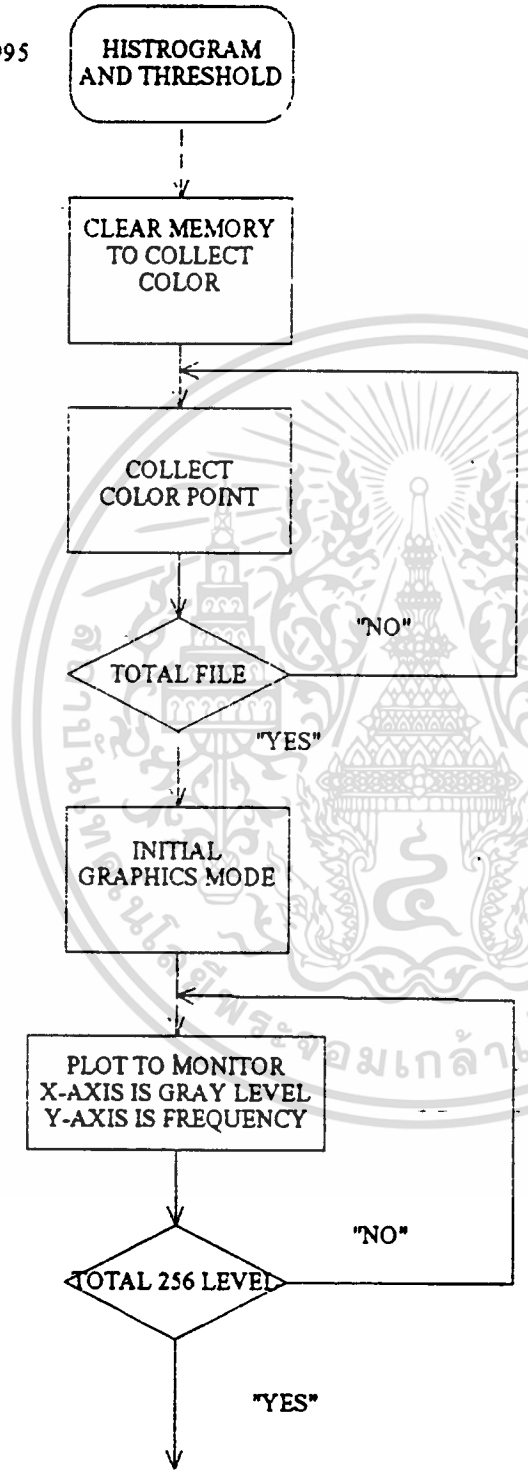
5:01 PM



TopChart

Friday, March 10, 1995

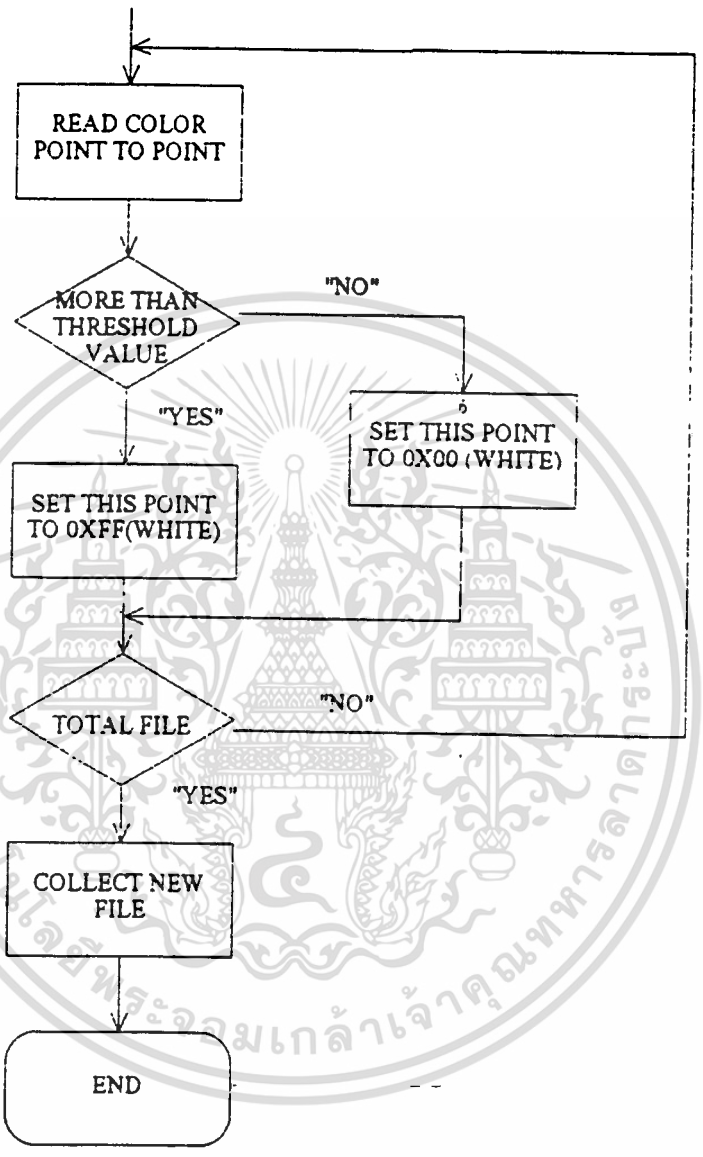
5:30 PM



TopChart

Friday, March 10, 1995

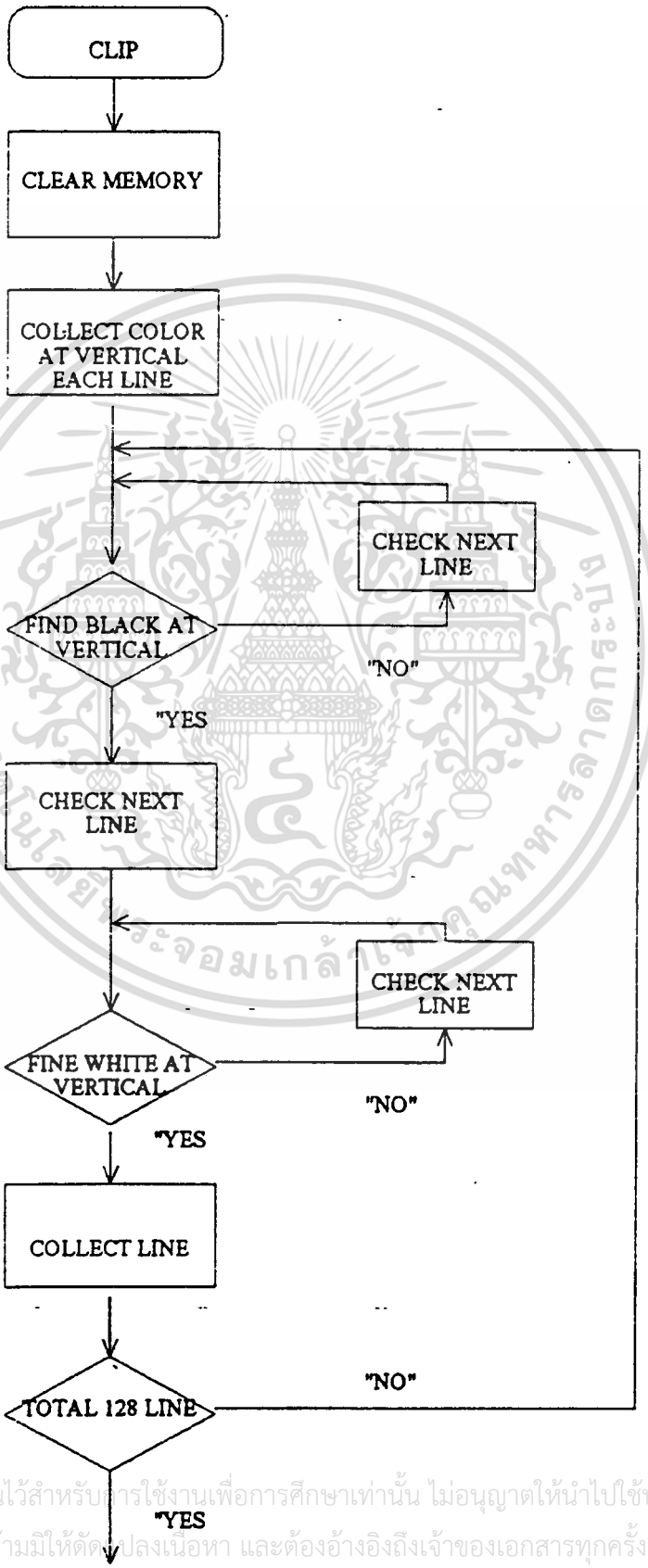
7:41 PM



TopChart

Friday, March 10, 1995

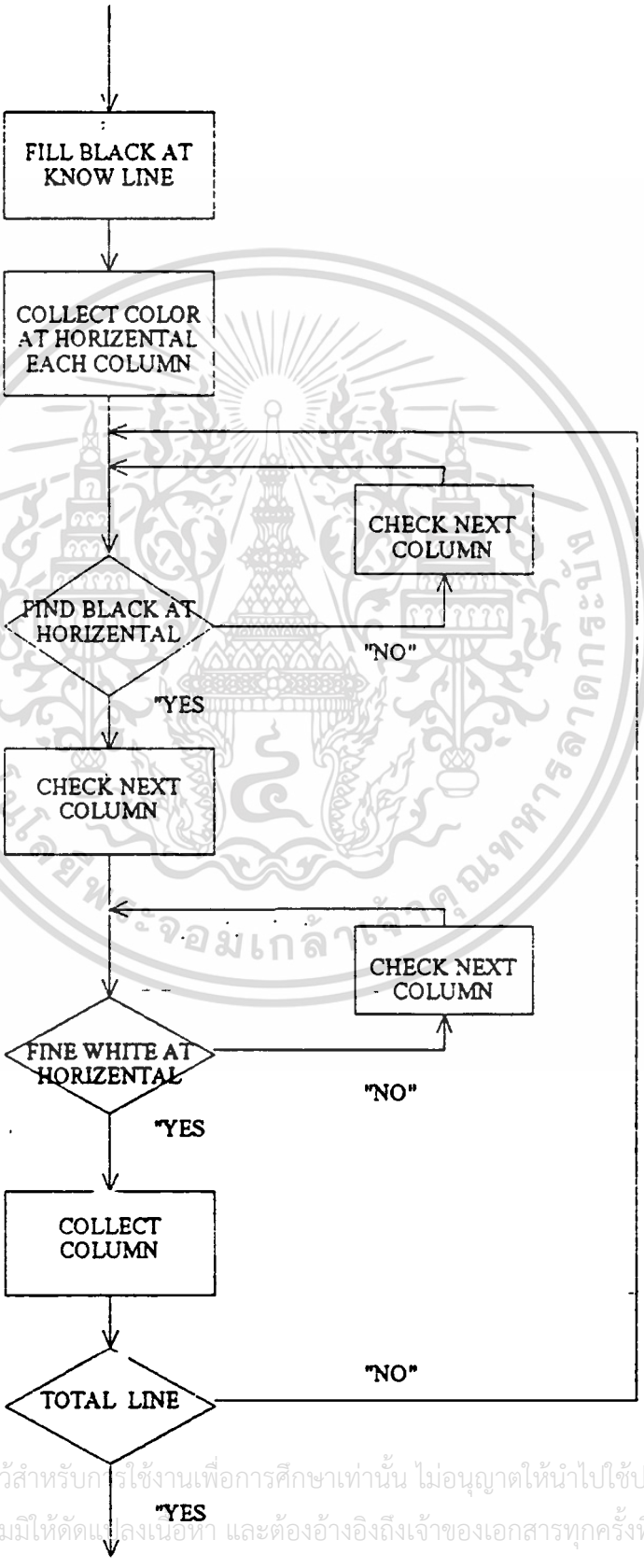
8:12 PM



TopChart

Friday, March 10, 1995

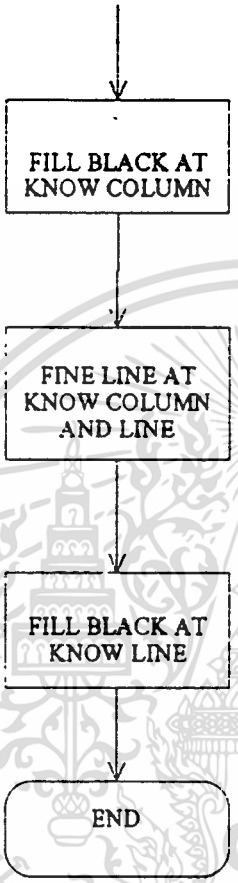
8:12 PM



TopChart

Friday, March 10, 1995

9:38 PM



4.4.2 ใช้ภาษาC ควบคุมการประมวลผล

จากที่เคยกล่าวไว้ว่า " เราใช้ Card Image เป็น Co-Processor กับ PC เท่านั้น " ดังนั้นตัว Card ย่อมมี Program (ที่เป็น Assembly)ที่ทำหน้าที่ประมวลผลของมันเอง แต่ถูกควบคุมการทำงานโดย PC ซึ่งในตอนแรกนี้จะอธิบายโปรแกรมการประมวลผลของตัว TMS บน Card และต่อไปจะอธิบายโปรแกรมควบคุมการทำงานโดย PC

-โปรแกรมการประมวลผลของ TMS

เนื่องจากบน Card นั้นจะมี CPU ซึ่งทำหน้าที่ในการประมวลผลอยู่ การที่จะให้ CPU ใดๆ ก็ตามสามารถทำงานได้ หัวใจสำคัญก็คือ โปรแกรมการทำงานของ CPU ดังนั้นการที่จะให้ CPU ทำงานประมวลผลอะไรก็ต้องเขียนโปรแกรมขึ้นมาสำหรับงานนั้นๆ สำหรับปริญญานิพนธ์นี้ใช้ CPU เบอร์ TMS 320C25 การเขียนโปรแกรมต้องเขียนด้วยภาษา Assembly และด้วยลักษณะพิเศษเฉพาะด้านของ CPU เบอร์นี้ ซึ่งมีการทำงานของแต่ละคำสั่งแปลกไปจาก CPU ทั่วไป การออกแบบโปรแกรมจึงค่อนข้างยาก เพื่อความสะดวกในการพัฒนาโปรแกรมต่อไป จึงได้ออกแบบให้หน่วยความจำบน Card เป็น RAM ทั้งหมด คือ ทั้ง Data และ Program Memory แทนที่จะเป็นเฉพาะ Data Memory โดยโปรแกรมของ TMS จะโหลดผ่านทาง PC ได้ ซึ่งจะเป็นการพัฒนาโปรแกรมของ TMS บน PC ได้ ในการพัฒนาโปรแกรมก็จะเขียนโปรแกรมในรูปของ Text File และใช้โปรแกรมEditorใดๆ จากนั้นก็ใช้โปรแกรมCompilerทำการ Compileให้เป็น Binary File ซึ่งเป็นFileที่เก็บข้อมูลในรูปของภาษาเครื่องของ TMS ในการใช้งานก็จะโหลด Binary Fileเหล่านี้ลงในหน่วยความจำ Program Memory เมื่อ TMS ประมวลผลก็จะอ่านคำสั่งจาก Program Memory ไปทำงาน จะเห็นได้ว่าใช้ PC เป็นตัวจัดการทั้งหมด ซึ่งสะดวกมากและสามารถแก้ไขโปรแกรมได้ง่าย สำหรับCompiler ที่ใช้ได้แก่ C32 ซึ่งจะแปลได้เป็นFile .HEX จากนั้นก็ใช้โปรแกรมอื่นเช่น HEXBIN2.EXE แปลงไฟล์ .HEX เป็น Binary File .BIN

สำหรับตัวโปรแกรมของ TMS นั้น ซึ่งในโครงงานนี้จะเป็นการแยกตัวหนังสือ แต่ละตัวออกจากกัน ซึ่ง Program ที่จะให้ TMS ทำงานคือ มีวิธีการแยกตัวอักษร 2 แนว คือ แยกทางแนวนอน (Cut Row) และแยกทางแนวตั้ง (Cut Column) จะตัดแนวไหนขึ้นอยู่กับการคำสั่งของโปรแกรมควบคุมการทำงานบน PC และสิ่งหนึ่งที่ต้องมีคือในตอนท้ายของโปรแกรมหลังประมวลผลเสร็จ จะต้องทำคำสั่ง OUT PORT 0 ด้วยค่าอะไรก็ได้ เพื่อบอกแก่ PC ว่าเสร็จแล้ว

การทำงานของ Program ภาษา Assembly

การทำงานของ Program ภาษา Assembly ของการ์ด CPU TMS320C25 ซึ่งจะมีส่วนของ Program ภาษา C ในการควบคุมการทำงานของ Card นี้ ดังได้กล่าวมาแล้ว ในส่วนของภาษา Assembly จะมีส่วนของโปรแกรมย่อยอยู่ 3 ส่วนคือ

1. หัด Threshold
2. Scan and Detect (ROW)
3. Scan and Detect (COLUNM)

1. หัด Threshold

คือส่วนโปรแกรมที่ทำหน้าที่เปลี่ยนระดับความเข้มของข้อมูลที่ Scan เข้ามาให้เป็น 2 ระดับ คือ ระดับ 1 กับระดับ 0 จากระดับทั้งหมด 256 ระดับ ทั้งนี้เพื่อเป็นการแยกส่วนของตัวอักษร และ พื้นให้มีระดับต่างกันอย่างเห็นได้ชัด ซึ่งค่า Threshold นี้จะได้มาจากการ Plot ค่า Histogram ระหว่างความเข้มและจำนวนจุดทั้งหมดที่ระดับความเข้มนั้น ๆ

2. Scan and Detect Row

จะเป็นโปรแกรมในส่วนที่ Scan ข้อมูลแต่ละจุดบนแถว ๆ หนึ่งโดยจะตรวจจับระดับความเข้มของสีตัวอักษร ก็จะบันทึกตำแหน่งของบรรทัดนั้น ๆ ที่เจอ เสร็จแล้วก็ไป Scan บรรทัดต่อไปจนครบ ซึ่งบรรทัดที่ตรวจพบนี้คือขอบเขตของตัวอักษรที่จะแยกในแนวนอน

3. Scan and Detect Column

จะเป็นโปรแกรมที่ Scan ข้อมูลแต่ละจุดในแต่ละแถวในแนวดิ่งจะตรวจจับระดับความเข้มของสีตัวอักษรว่ามีหรือไม่ ถ้ามีการตรวจพบระดับความเข้มของตัวอักษรก็จะบันทึกตำแหน่งของ Column ที่ตรวจเจอระดับของสีตัวอักษร จากนั้นก็ไป Scan Column ต่อไปจนครบ ซึ่ง Column ที่ตรวจเจอนี้คือขอบเขตของตัวอักษรที่จะแยกในแนวดิ่ง

เมื่อ Scan และ Detect ค่าได้แล้ว ก็จะทำการเก็บค่าต่าง ๆ เหล่านี้ไว้ใน Memery ที่กำหนดไว้ เพื่อส่งค่าให้แก่ Program ภาษา C เพื่อประมวลผลต่อไป

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

BIT 0,1 Blank Select 0,1 เป็นbitเลือกBlankของ RAM บนการ์ด

bit 0	bit 1	
0	0	Data Memory ส่วนล่าง 0000-7FFFH
0	1	Data Memory ส่วนบน 8000-FFFFH
1	0	Program Memory ส่วนล่าง 0000-7FFFH
1	1	Program Memory ส่วนบน 8000-FFFFH

BIT 2 Blank Enable ทำหน้าที่กำหนดให้ PC ติดต่อกับ RAM ได้หรือไม่ Setเป็น "1" คือ ให้ติดต่อได้ เป็น "0" ไม่ให้ติดต่อ

BIT 4 PC/TMS เป็น bit ที่ทำหน้าที่ switch เลือกระหว่าง PC กับ TMS ให้ติดต่อกับ RAM Setเป็น "1" คือ ติดต่อกับ TMS เป็น "0" ติดต่อกับ PC

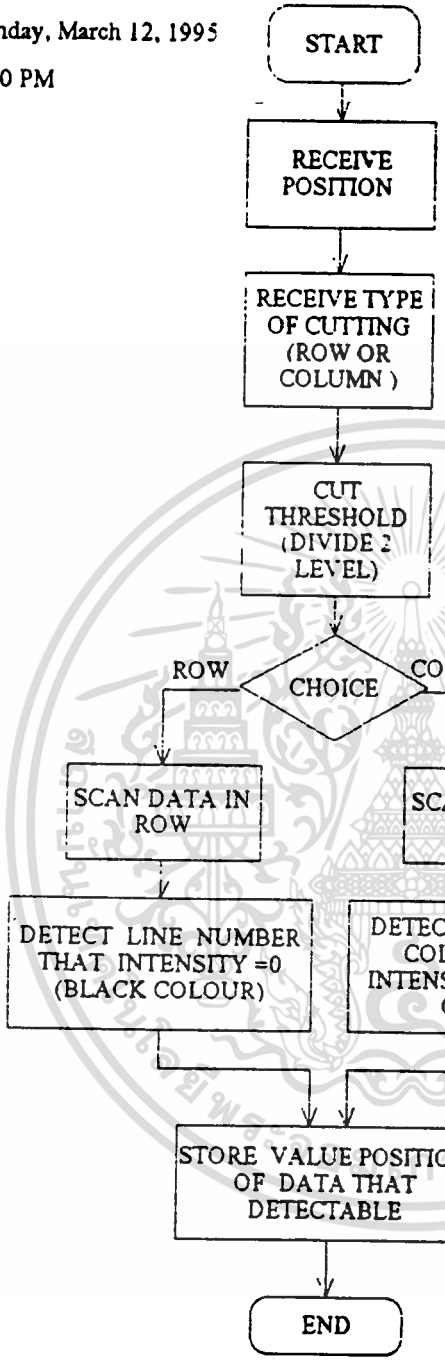
BIT 7 Reset TMS เป็น bit ควบคุม TMS ให้ทำงานหรือไม่รู้จะใช้ขา Reset เป็นสัญญาณควบคุม ถ้าต้องการให้ทำงานก็Setเป็น "1" ถ้าไม่ต้องการให้ทำงานก็Setเป็น "0"

ตารางที่ 4.1 แสดงหน้าที่ของแต่ละ bit ใน Control Port (Port 300H)

TopChart

Sunday, March 12, 1995

3:40 PM



-โปรแกรมควบคุมการทำงานโดย PC

จากที่เคยกล่าวไว้ว่า Card จะทำงานอะไรนั้น (แยกทางแนว ROW หรือ แยกทางแนว Column) ทุก ๆ อย่างจะถูกควบคุมโดย PC ทั้งหมดซึ่ง PC จะทำงานดังต่อไปนี้

1) ทำการ Clearram ที่จะใช้ทั้งหมดให้เป็น 0

2) ทำการ Load Program และ Load Data(เป็นโปรแกรมย่อยซึ่งจะอธิบายรายละเอียดของโปรแกรมที่หลัง)

3) Set แนวตัด(แนวนอนหรือแนวตั้ง) ซึ่งจะให้ตัดในแนวนอนก่อน เพื่อแยกแต่ละบรรทัดให้ออกจากกัน

4) ส่งให้ TMS ทำงานโดยใช้ Program Run (เป็นโปรแกรมซึ่งจะอธิบายรายละเอียดที่หลัง) ให้เริ่มการตัดในแนวนอน

5) ทำการเก็บตำแหน่งของแต่ละบรรทัดไว้

6) Set แนวตัดเป็นแนวตั้ง และ Set แนวบรรทัดที่ต้องการ เพื่อที่จะแยกตัวอักษรแต่ละตัวในบรรทัดนั้นออกมา

7) ส่งให้ TMS ทำงานโดยใช้โปรแกรม Run ให้เริ่มการตัดในแนวตั้ง

8) เก็บตำแหน่งของตัวอักษรที่ตัดได้

9) ทำงานตามข้อ 3),4) และ 5) อีกครั้ง

-รายละเอียดโปรแกรมย่อย

ซึ่งในที่นี้มีรายละเอียดของ 2 โปรแกรม

1) การ Load Data และ Load Program ระหว่างหน่วยความจำบนการ์ดกับ PC

2) โปรแกรม Run

1)การ Load Program และ Load Data ระหว่างหน่วยความจำบนการ์ดกับ PC

ซึ่งเป็นการ Load Data และ Load Program ลง RAM ซึ่งมีขั้นตอนดังนี้

1.1 Switch ให้ RAM ติดต่อกับ PC และเลือก Blank ที่ต้องการติดต่อด้วยโดยส่งคำสั่งควบคุมไปที่ Control port ซึ่งแต่ละ bit ทำหน้าที่อย่างไร แสดงในตารางที่ 5.1 สำหรับขั้นตอนนี้จะเซ็ตให้ bit 7 เป็น "0" คือ Reset TMS อยู่ ,bit 4 เป็น "0" เพื่อเลือกให้ RAM ติดต่อกับ PC bit 2 เป็น "1" เป็นการ Enable ให้ PC ติดต่อกับ RAM ได้ และ bit 0 และ 1 จะเซ็ตตาม Blank ที่ต้องการจะติดต่อด้วย คือ ถ้าต้องการติดต่อกับหน่วยความจำส่วนใดก็เซ็ต Blank ให้ตรงกัน สำหรับ bit อื่นไม่ได้ใช้จะเซ็ตเป็นอะไรก็ได้ แต่สำหรับกรณีนี้จะเซ็ตเป็น 0 หมดดังนั้น ค่าที่จะส่งไปก็คือ 000001xxb โดยค่า xx เป็นอะไรก็ได้ แล้วแต่จะเซ็ตเป็น Blank ใด

1.2 หลังจากผ่านขั้นตอนแรกแล้ว ก็จะสามารถติดต่อกับ RAM บน Card ได้โดยผ่านทางหน่วยความจำของ PC ในบล็อคที่ map ไว้ เช่น map ที่บล็อค E ก็ติดต่อผ่านทาง address E000:0000H-E000:FFFFH ในการจะ Load ข้อมูลลง RAM ก็โหลดข้อมูลไปยังหน่วยความจำช่วงบล็อค E หรือถ้าจะอ่านข้อมูลก็ Load Data จากหน่วยความจำบล็อค E

ในการ Load Data มีสิ่งที่จะต้องระวังคือ ตำแหน่งของข้อมูลใน PC กับตำแหน่งของข้อมูลใน TMS คือการย้ายข้อมูลต่างๆจะต้องสัมพันธ์กับตำแหน่งที่ TMS จะทำงาน เพื่อที่จะได้ทำงานได้ถูกต้อง จึงต้องเข้าใจความสัมพันธ์ระหว่างตำแหน่งข้อมูลใน PC กับ TMS ก่อน

จากที่ทราบแล้วว่า ข้อมูลใน PC 1 ตำแหน่ง เป็นแบบ 8 bit ในขณะที่ TMS เป็น 16 bit ดังนั้นในการอ้างถึงข้อมูลใน TMS 1 ตำแหน่ง PC จะต้องใช้ 2 ตำแหน่ง สามารถเขียนความสัมพันธ์ได้ดังนี้

กรณีเป็นหน่วยความจำส่วนล่าง

Address บน PC เก็บ low byte = E0000H + Address บน TMS * 2

Address บน PC เก็บ high byte = E0001H + Address บน TMS * 2

กรณีเป็นหน่วยความจำส่วนบน

Address บน PC เก็บ low byte = E0000H + (Address บน TMS - 8000H) * 2

Address บน PC เก็บ high byte = E0001H + (Address บน TMS - 8000H) * 2

จากสมการ คือ Address 0000-FFFFH จะซ้ำกับ 8000H-FFFFH แต่อยู่ต่าง Blankกัน

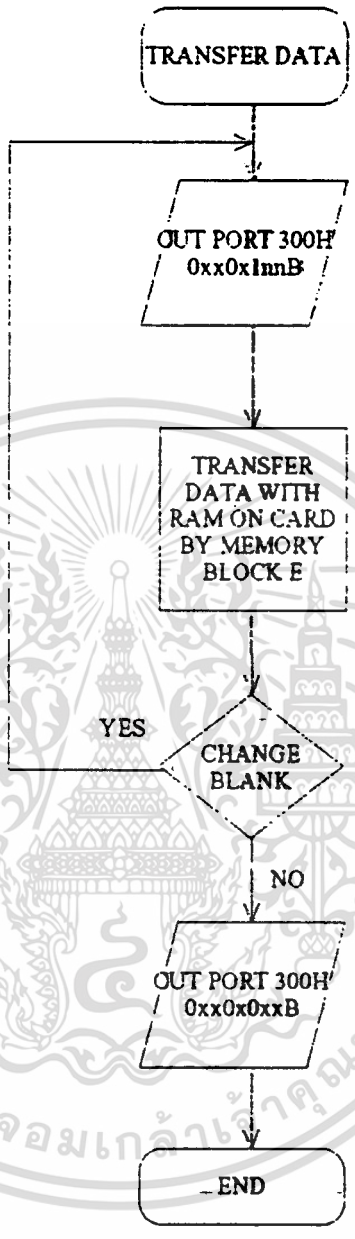
1.3 หลังจากทำการ Load Data เสร็จ ถ้าต้องการเปลี่ยน Blank ก็ส่งคำสั่งไปที่ Control port ด้วยคำสั่งเดียวกับในข้อ 1.1 เพียงแต่เปลี่ยนค่า bit 0 และ 1 ตามแบงค์ที่ต้องการ ติดต่อใหม่

1.4 หลังจากสิ้นสุดการ Load Data บน RAM แล้ว ให้ส่งค่า 00000000B ไปยัง Control port คือ เซ็ตบิต 3 ให้เป็น "0" เพื่อยกเลิกมิให้ PC ติดต่อกับ RAM บนการ์ด

TopChart

Sunday, March 12, 1995

12:45 PM



FLOW CHART ของการโหลดข้อมูลและโปรแกรมลง RAM

2. การ RUN โปรแกรมบน Card

ในการ RUN โปรแกรมจะกระทำหลังจากที่มีการโหลดตัวโปรแกรมและข้อมูลต่างๆลง RAM เสร็จสิ้นแล้ว ซึ่งการ RUN Program นอกจากจะมีการควบคุมตัว Card แล้ว ยังต้อง Set Interrupt ของ PC ด้วย เพื่อให้รับรู้การ Interrupt มีขั้นตอนดังนี้

2.1 จัดการเกี่ยวกับ Interrupt บน PC คือ

- Enable Hardware Interruptเบอร์ 12 ที่ตัว Interrupt Controler โดยการ set ค่า bit ที่ 4 ของ port A1H ให้เป็น "0"(port ของInterrupt Control ตัวที่ 2)

- Set Interrupt vector ของ Hardware Interrupt เบอร์ 12 ซึ่งตรงกับ Interrupt ที่ 74 บน PC ซึ่งมี vector อยู่ที่ Address 1D0-1D3H โดย setไปที่โปรแกรมจัดการ Interrupt ที่เขียนขึ้นใหม่

2.2 ส่งคำสั่งไปที่ Control port เพื่อให้ CPU เริ่มประมวลผล ซึ่งคำสั่งที่ส่งไป คือ 10010000B มีความหมายดังนี้ ให้ bit 7 ซึ่งเป็นสัญญาณ Reset TMS เป็น "1" คือ เริ่มให้ TMS ประมวลผล ขณะเดียวกัน bit 4 ซึ่งเป็นตัวสวิทช์ RAM ระหว่าง PC กับ TMS ก็จะต้องเป็น "1" เช่นกัน คือให้สวิทช์ไปที่ TMS สำหรับ bit อื่นๆจะไม่มีผลสำหรับกรณีนี้ ซึ่งปกติจะเซ็ตเป็น "0" หมด

2.3 หลังจากเริ่มให้ TMS ประมวลผล ทางด้าน PC ก็จะคอยสัญญาณตอบกลับจาก TMS ซึ่งจะอยู่ในรูปของ Interrupt เมื่อใด PC ได้รับ Interrupt ก็หมายถึง TMS ประมวลผลเสร็จแล้ว ก็จะทำการหยุดการทำงานของ TMS โดยส่งคำสั่งควบคุมการ์ไปที่ Control port set ให้ bit ที่ 7 และ 4 เป็น "0"

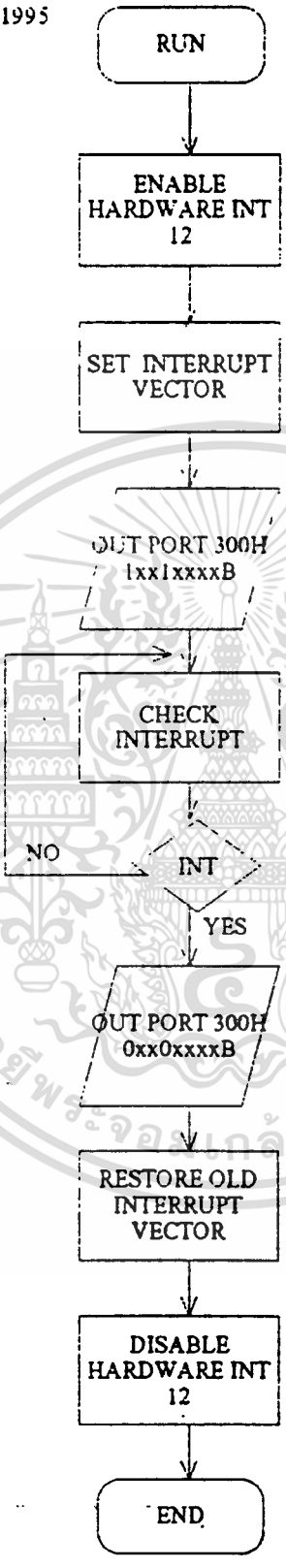
2.4 จัดการ Interrupt ของ PC ให้เหมือนเดิมได้แก่ ค่าที่ Interrupt Controler ซึ่งปกติจะ Disable อยู่ และคืนค่า Interrupt vector เก่าด้วย

2.5 หลังจากขั้นตอนที่ 2.4 ก็เป็นการจบขั้นตอนที่จะให้ TMS ประมวลผล จากนั้นการนำข้อมูลที่ได้ไปใช้ ก็จะอยู่ในหัวข้อแรก คือ การ Load Data

TopChart

Sunday, March 12, 1995

1:37 PM



FLOW CHART ของการสั่งให้การ์ดประมวลผล

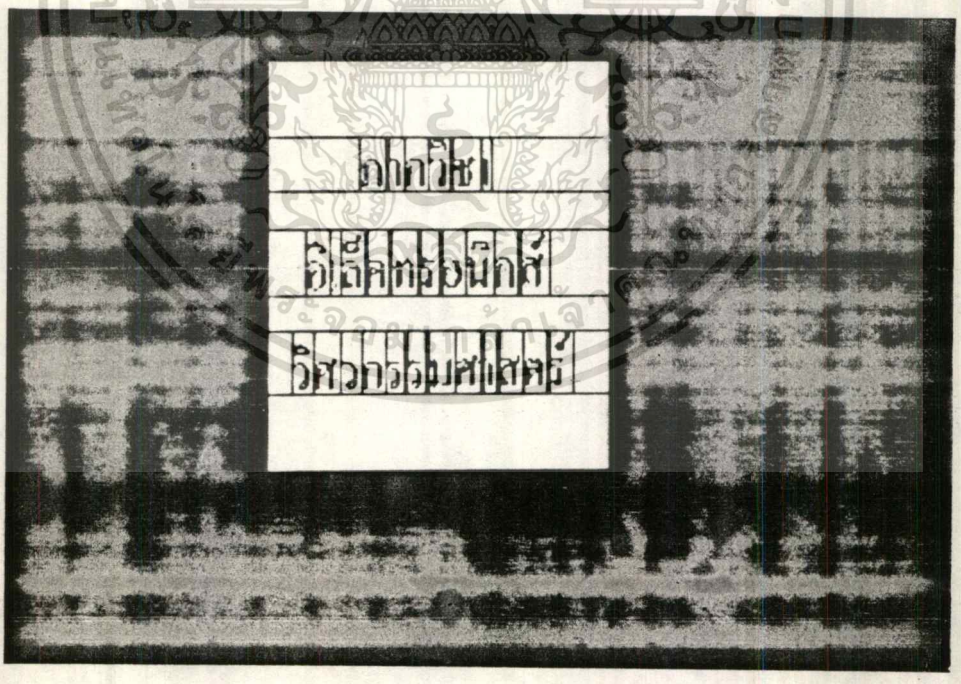
บทที่ 5
ผลการทดลอง

บทนี้เป็นบทที่แสดงผลการทดลองโดยแสดงผลการทำงานที่ใช้การภาษา C ล้วน ๆ และ ผลการทำงานที่ใช้ CPU TMS320c25 เป็นตัวประมวลผล ซึ่งแสดงดังรูป โดยรูปที่ 5.1 - 5.5 แสดงผลการทำงานโดยใช้ภาษา C ล้วน ๆ และรูปที่ 5.6 แสดงผลการทำงานที่ใช้ CPU TMS320c25 เป็นตัวประมวลผล

รูปที่ 5.1 แสดงรูปที่ได้จากการใช้เครื่อง Scaner

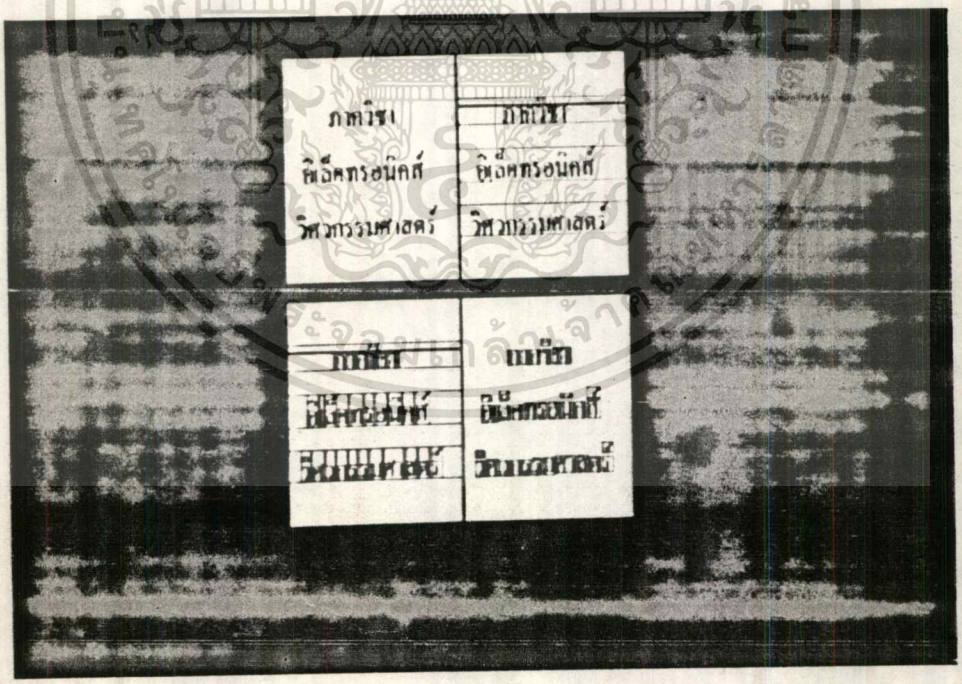
รูปที่ 5.2 แสดงรูปที่ผ่านการ High-Pass Fillter

รูปที่ 5.3 แสดงผลในขั้นตอนที่ 1 ของการแยกตัวอักษร



รูปที่ 5.4 แสดงผลในขั้นตอนที่ 2 ของการแยกตัวอักษร

รูปที่ 5.5 แสดงผลในขั้นตอนที่ 3 ของการแยกตัวอักษร



รูปที่ 5.6 แสดงผลการทำงานโดยใช้ CPU TMSC320c25

สรุปผลการทดลอง

จากการทดลองโดยใช้ภาษา C ล้วน ๆ ในการประมวลผลจะใช้เวลาในการประมวลผลนานกว่าการประมวลผลโดยใช้ CPU TMS320c25 เป็นตัวประมวลผล โดยแสดงเวลาในการใช้งานโดยเปรียบเทียบกับรุ่นของ PC ดังตารางต่อไปนี้

เวลาที่ CPU TMS ใช้	3.03
เวลาที่ CPU 80386 ใช้	8.00
เวลาที่ CPU 80486 ใช้	3.00

จากตารางจะเห็นได้ว่าการใช้ CPU TMS320c25 จะช่วยแบ่งภาระในการทำงานของ CPU ของ Computer เพราะการทำงานของ TMS320c25 จะมีความเร็วในการทำงานสูงกว่าได้ถึงหนึ่งเท่าเมื่อเวลาในการทำงานลงไปได้มาก

กิตติกรรมประกาศ

ขอขอบพระคุณทุกท่านที่มีส่วนช่วยให้ปริญญานิพนธ์นี้สำเร็จลุล่วงด้วยดี

คุณพ่อ คุณแม่ ที่กรุณาอบรมสั่งสอนและสนับสนุนการศึกษาตลอดมา

อ. เทอดศักดิ์ ลิ่วหาทอง อาจารย์ที่ปรึกษา

รศ.ดร มนัส สังวรศิลป์ เอื้อเฟื้ออุปกรณ์

อ. พิชัย คูศิริวณิชกร เอื้อเฟื้ออุปกรณ์

อาจารย์ทุกท่านที่สอนสั่ง

เพื่อนๆ พี่ๆ น้องๆ ทุกคนที่ให้กำลังใจและความช่วยเหลือ

ชุมนุมอิเลคทรอนิคส

บรรณานุกรม

1. TEXAS INSTRUMENT, "SECOND-GENERATION TMS320 USER GUIDE",1987

2. E.ORAN BRIGHAM, "THE FAST FOURIER TRANSFORM AND ITS APPLICATION", PRENTICE-HALL INTERNATIONAL,INC.,1988

3.Rafael c.Gonzales,Richrad E.Wood "Digital-Image Processing" Addison-Wesley Publishing Company,1992 .

4.Paul M.Embree ,Bruce Kimbel "C Langauge Algorithm for Digital Signal Processing" Prentice-Hall International,Inc.,1991

5. ยืน ภู่วรวรรณ, "การประมวลผลสัญญาณดิจิตอล",วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ เล่ม 76-79,ซีเอ็ดยูเคชั่น,2530

6. ไพศาล เลชะรัตนประเสริฐ, "มุม พีซีฮาร์ดแวร์ คอน การ์ดอินเตอร์เฟสอเนกประสงค์", วารสารเซมิคอนดัคเตอร์ อิเลคทรอนิคส์ เล่ม 104,ซีเอ็ดยูเคชั่น,2534

7. บวร ไชยสุขทักษิณ,บุญชัย งามวงศ์วัฒนา,สุรชัย อุวพรพาณิชย์, "การประยุกต์ใช้งานไมโครโปรเซสเซอร์ (DSP) ในงานประมวลผลภาพ",ปริญญานิพนธ์ ปีการศึกษา 2536,ภาควิชาอิเลคทรอนิคส์ วิศวกรรมศาสตร์ สจล.

```
      CPU "320C2X.TBL"  ;CPU TABLE
      HOF   "INT8"      ;HEX OUTPUT FORMAT
      WDLN    2         ;TWO BYTE WORD LENGTH
      AR0: EQU 0
      AR1: EQU 1
      AR2: EQU 2
      AR3: EQU 3
      AR4: EQU 4
      AR5: EQU 5
      AR6: EQU 6
      AR7: EQU 7
     DATA: EQU 400H
THRESHOLD: EQU 09FH
 COUNTER1: EQU 4000H
 COUNTER2: EQU 7FH
       X11: EQU 4504H
       X12: EQU 4506H
       Y11: EQU 4505H
       Y12: EQU 4507H
        X1: EQU 5001H
        X2: EQU 5003H
        Y1: EQU 5002H
        Y2: EQU 5004H
 VIRTUAL1: EQU 4510H
POSITION1: EQU 4600H
POSITION2: EQU 4500H
 VIRTUAL2: EQU 4511H
 VIRTUAL3: EQU 4512H
 VIRTUAL4: EQU 4513H
 VIRTUAL5: EQU 4514H
 VIRTUAL6: EQU 4515H
 VIRTUAL7: EQU 4516H
 VIRTUAL8: EQU 4517H
 VIRTUAL9: EQU 4518H
VIRTUAL10: EQU 4519H
VIRTUAL11: EQU 4520H
VIRTUAL12: EQU 4521H
VIRTUAL13: EQU 4522H
VIRTUAL14: EQU 4523H
VIRTUAL15: EQU 4524H
VIRTUAL16: EQU 4527H
VIRTUAL17: EQU 4526H
VIRTUAL18: EQU 4529H
    VALUE: EQU 5030H
   INIROW: EQU 5030H
  INIROW1: EQU 5031H
;**************************************
      LRLK AR0,X11              ;FILL X1 IN X11
      LRLK AR1,X1
      LARP AR1
      LAR AR3,*
      LARP AR0
      SAR AR3,*
;**************************************
      LRLK AR0,Y12              ;FILL Y1 IN Y12
      LRLK AR1,Y1
      LARP AR1
      LAR AR3,*
      LARP AR0
      SAR AR3,*
;**************************************
      LRLK AR0,X12
      LRLK AR1,X1
      LARP AR1
      LAR AR3,*
      LARP AR0
```

```
        SAR AR3,*
;****************************************
        LRLK AR0,THRESHOLD
        LRLK AR1,DATA
        LRLK AR2,COUNTER1
LOOPF:  LARP AR1
        LAR AR4,*
        LARP AR4
        CMPR 2                  ;COMPARE WITH THRESHOLD
        BBNZ FILLW,*            ;BRANCH IF DATA > THRESHOLD
        B JOB
JOB:    ZAC                     ;DATA < THRESHOLD [FILL 0]
        LARP AR1
        SACL *+
        B LOOP1
FILLW:  ZAC                     ;DATA > THRESHOLD [FILL FF]
        LACK 00FFH
        LARP AR1
        SACL *+
LOOP1:  LARP AR2
        BANZ LOOPF,*-
;************************************************************
;******************** CHOOSE ROW OR COLUMN***********************
;     IF FILL 0                       {CHOOSE COLUMN}
;     IF FILL 1 OR OTHERWISE          {CHOOSE ROW}
        LRLK AR0,5000H
        LARP AR0
        LAC *
        BZ COL,*
;*********************PROGRAM DETECT ROW**************************
        LRLK AR0,Y1
        LRLK AR1,Y2
        LRLK AR3,VIRTUAL1
        CALL SUB1
        LARP AR3
        LAR AR4,*                       ;AR4<- (Y2-Y1)
;************************************************************
        LRLK AR0,X1
        LRLK AR1,X2
        LRLK AR3,VIRTUAL2
        CALL SUB1
        LARP AR3
        LAR AR5,*                       ;AR5<-(X2-X1)
;************************************************************
        LRLK AR2,X1
        LRLK AR7,POSITION2
;**************************************
ROW4:   LRLK AR0,Y1
        LRLK AR1,Y11
        LRLK AR3,VIRTUAL12
        LARP AR0
        LAR AR3,*
        LARP AR1
        SAR AR3,*                       ;PUT Y1 IN Y11
        LRLK AR3,VIRTUAL1
        LARP AR3
        LAR AR4,*                       ;ASSIGN AR4 <= (Y2-Y1)
ROW3:   ZAC
        LRLK AR0,Y11
        LRLK AR1,80H
        LRLK AR3,VIRTUAL3
        CALL SUB2
        LARP AR6
        LAC *
        BZ ROW1,*
        B ROW2
```

```
ROW1:  ZAC
       LACK 0FFH
       LARP AR7
       SACL *
ROW2:  ZAC
       LARP AR0
       LAC *
       ADDK 1H
       LARP AR0
       SACL *
       LARP AR4
       BANZ ROW3,*-
       LARP AR2
       LAC *
       ADDK 1H
       LARP AR2
       SACL *
       LARP AR7
       ADRK 1H
       LARP AR5
       BANZ ROW4,*-
       B SCROW
;***************************************************************
       COL:   NOP
;******************** PROGRAM DETECT COLUMN**********************
       LRLK AR0,Y1
       LRLK AR1,Y2
       LRLK AR3,VIRTUAL4
       CALL SUB1
       LARP AR3
       LAR AR4,*                    ;AR4<-(Y2-Y1)
;***************************************************************
       LRLK AR0,X1
       LRLK AR1,X2
       LRLK AR3,VIRTUAL5
       CALL SUB1
       LARP AR3
       LAR AR5,*                    ;AR5<- X2-X1
;**********CALCULATE NUMBER OF STEP NEW ROW*********************
       LRLK AR0,Y1
       LRLK AR7,POSITION2
COL4:  LRLK AR2,X1                  ;PUT X1 => X11
       LRLK AR1,X11
       LRLK AR3,VIRTUAL14
       LARP AR2
       LAR AR3,*
       LARP AR1
       SAR AR3,*
;***************************************************************
       LRLK AR3,VIRTUAL5
       LARP AR3
       LAR AR5,*                    ;AR5 <=(X2-X1)
COL3:  ZAC
       LRLK AR2,X11
       LRLK AR1,80H
       LRLK AR3,VIRTUAL6
       CALL SUB2
       LARP AR6
       LAC *
       BZ COL1,*
       B COL2
COL1:  ZAC
       LACK 0FFH
       LARP AR7
       SACL *
COL2:  ZAC
```

```
        LARP AR2
        LAC *
        ADDK 1H
        LARP AR2
        SACL *
        LARP AR5
        BANZ COL3,*-
        LARP AR0
        LAC *
        ADDK 1H
        LARP AR0
        SACL *
        LARP AR7
        ADRK 1H
        LARP AR4
        BANZ COL4,*-
;***************************************************************
SCROW: NOP
;*********************PROGRAM SCAN ROW & COLUMN****************
        LRLK AR2,POSITION2
        LRLK AR6,INIROW1
        LRLK AR0,0000H
        LRLK AR1,0000H
        LRLK AR3.COUNTER2
        LRLK AR4,VIRTUAL7
        LRLK AR7,VIRTUAL8
COMP:   ZAC
        LARP AR7
        SAR  AR1,*
        LARP AR7
        LAC *                           ;PUT COMPLEMENT IN ACC
        CMPL                            ;COMPLEMENT ACC
        ANDK 000FH                      ;DETECT LSB
        LARP AR7
        SACL *
        LARP AR7
        LAR AR1,*
        BZ LOOPB,*                      ;JUMP IF ACC==0(BLACK)
LOOPB2: ZAC
        LARP AR2
        LAC *+
        BNZ LOOPB1,*
        LARP AR0
        ADRK 1H                         ;INCREASE AR0
        LARP AR3
        BANZ LOOPB2,*-
        B PLOT
LOOPB1: ZAC
        LARP AR6
        SAR AR0,*+                      ;LINE NUMBER STORE IN INROW
        LARP AR0
        ADRK 1h
;/////////////////////////////////////////////////////////////
        LRLK AR5,VALUE                  ;FIND NUMBER OF CUTTING
        LARP AR5
        LAC *
        ADDK 1H
        LARP AR5
        SACL *
;***************************************************************
        LARP AR3
        BANZ COMP,*-
        B PLOT
;***************************************************************
;*********************DETECT BLACK*****************************
LOOPB: NOP
```

```
LOOPA2: ZAC
        LARP AR2
        LAC *+
        BZ LOOPA1,*                     ;DETECT BLACK ..
        LARP AR0
        ADRK 1H                         ;INCREASE INDEX(AR0)
        LARP AR3
        BANZ LOOPA2,*-
        B PLOT
LOOPA1: ZAC
        LARP AR6
        SAR AR0,*+                      ;LINE NUMBER STORE IN INROW
        LARP AR0
        ADRK 1H                         ;INCREASE INDEX(AR0)
;***************************************************************
        LRLK AR5,VALUE                  ;FIND NUMBER OF CUTTING
        LARP AR5
        LAC *
        ADDK 1H
        LARP AR5
        SACL *
;///////////////////////////////////////////////////
        LARP AR3
        BANZ COMP,*-                    ;LOOP COUNTER
;***************************************************************
PLOT:   NOP
;***************************************************************
G01:    LRLK AR4,5000H
        LARP AR4
        LAC *
        BZ G05,*
;***************************************************************
        LRLK AR2,>12
        LRLK AR0,VALUE                  ;ADD X11(REFERRENCE)
        LRLK AR1,INIROW1
        B G06
G05:    LRLK AR2,>12
        LRLK AR0,VALUE
        LRLK AR1,INIROW1
G06:    LRLK AR3,VIRTUAL13
        LARP AR0
        LAR AR3,*
G04:    LARP AR3
        BANZ G02,*-
        B G03
G02:    LARP AR1
        LAC *
        LARP AR2
        ADD *
        LARP AR1
        SACL *+
        B G04
G03:    LRLK AR0,5000H                  ;CHOOSE ROW OR COL
        LARP AR0
        LAC *
        BZ PLOTCL,*
        B G0
          ;********************** PLOT ROW**************************
        LRLK AR1,80H
        LRLK AR2,INIROW1
        LRLK AR3,VIRTUAL9
        LRLK AR6,COUNTER2
PLOT2:  LRLK AR4,COUNTER2
        LARP AR3
        SAR AR1,*
        LARP AR3
```

```
        LT *                    ;LOAD 80H IN T REGISTER
        LARP AR2
        MPY *+                  ;80*(INIROW)=>P REGISTER
        ZAC
        PAC                     ;(P REGISTER)=>ACC
        ADLK 400H               ;400+(ACC)=>ACC
        LARP AR3
        SACL *
        LARP AR3
        LAR AR5,*
PLOT1:  ZAC
        LARP AR5
        SACL *+
        LARP AR4
        BANZ PLOT1,*-
        LARP AR6
        BANZ PLOT2,*-
        B GO
;***************************************************************
PLOTCL: NOP
        B GO
;**********************PLOT COLUMN******************************
        LRLK AR0,INIROW1
        LRLK AR2,COUNTER2
PLOTC2: LRLK AR1,COUNTER2
        LALK 3FFH
        LARP AR0                ;ADD 400H IN MEMORY[FIND LOCATION]
        ADD *
        LARP AR3
        SACL *+                 ;STORE LOCATION IN MEMORY
        LARP AR0
        LAR AR3,*               ;(AR0)=>AR3
PLOTC1: ZAC
        LARP AR3
        SACL *                  ;PLOT [FILL 01-]
        ADLK 80H                ;INCREASE NEW ROW
        LARP AR1
        BANZ PLOTC1,*-
        LARP AR2
        BANZ PLOTC2,*-
        B GO
;***************************************************************
;SUBROUTINE1
SUB1:   LARP AR1
        LAC *
        LARP AR0
        SUB *
        LARP AR3
        SACL *
        RET
;***************************************************************
;SUBROUTINE2
SUB2:   LARP AR3
        SAR AR1,*
        LARP AR3
        LT *                    ;FILL 80H IN T REGISTER
        LARP AR2
        MPY *                   ;80X1 IN P REGISTER
        ZAC
        PAC                     ;(P REGISTER)=> IN ACC
        ADLK 400H               ;400+(ACC)
        LARP AR0
        ADD *                   ;Y1+400H+(ACC)
        LARP AR3
        SACL *
        LARP AR3
```

```
        LAR AR6,*
        RET
;-------------------------------------------------------------
GO:     OUT *,0
        END
```

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

void clearram(void);


void clearram(void)
{
    long i,j;
    unsigned far *cardmem = (unsigned far *)MK_FP(0xe000,0);

    for(j=0;j<4;j++)
        {
        outportb(0x300,0x04+j);          /* 0 - - 0  - 1 0 0*/
        for(i=0;i<0x8000;i++)
            {
            *(cardmem+i)=0;
            }
        }

}
```

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#include <math.h>

#define CARD_PORT   0x300
#define off_set     0x400

int ldpic(char *tex);

int ldpic(char *tex)
{

    FILE *fp;
    long i,j,status,file_size;
    unsigned char temp;
    unsigned int dim;

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

    if((fp=fopen(tex,"rb"))==NULL)
    {
        printf("Can't open file\n");
        getch();
        return -1;
    }
    fseek(fp,0L,SEEK_END);
    file_size=ftell(fp);
    fseek(fp,0l,SEEK_SET);

    if(file_size>0x8000)
         {
               printf("I can't write data into Ram\n");
               getch();
               fclose(fp);
               return -2;
         }

    dim=(unsigned)sqrt((double)file_size);
    outportb(CARD_PORT,0x04);
    for(i=0;i<file_size;i++)
        {
        status = fread(&temp,sizeof(char),1,fp);
        if(status == 0)
         {
               printf("I can't write data into Ram\n");
               getch();
               fclose(fp);
               return -2;
         }
        *(cardmem+i+off_set)=(unsigned)temp;
        }
    fclose(fp);
    outportb(CARD_PORT,0x00);     /* disable card */
    return 0;

}
```

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>

#define DATA_LOW    0x00
#define DATA_HIGH   0x01
#define PROG_LOW    0x02
#define PROG_HIGH   0x03
#define CARD_PORT   0x300

typedef struct {

     unsigned int b0_7  : 8;
     unsigned int b8_15 : 8;
}binary;

typedef union {

     unsigned int hex;
     binary   bin;
}bin2hex;

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

int loadprg(char *tex);

int loadprg(char *tex)
{
    FILE *fp;
    long file_size,i,j;
    unsigned int temp1;
    unsigned char status,k;
    bin2hex  change;

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


    if ((fp= fopen(tex,"rb")) == NULL)
       {
          printf(" Cannot open input file \n");
          return -1;
       }

    fseek(fp,0L,SEEK_END);
    file_size=ftell(fp);
    fseek(fp,0l,SEEK_SET);
//  printf("File size = %u \n",file_size);

    if(file_size>0x20000)
        {
        printf("File too big , can't load \n");
        fclose(fp);
        return -2;
        }

    if(file_size < 0x10000)
        {
        outportb(CARD_PORT,0x06);     /* first program memory block*/
        for(i=0;i<file_size/2;i++)
            {
            status = fread(&temp1,sizeof(unsigned int),1,fp);
            if(status == 0)
                {
```

```
            printf("I can't write program to Ram \n");
            fclose(fp);
            return -2;
              }

             change.hex = temp1;
             k = change.bin.b8_15;
             change.bin.b8_15 = change.bin.b0_7;
             change.bin.b0_7 = k;
             temp1 = change.hex;

             *(cardmem+i)=(unsigned int)temp1;
        }
        }
    else
    {
        outportb(CARD_PORT,0x06);

        for(i=0;i<0x8000;i++)
        {
             status = fread(&temp1,sizeof(unsigned int),1,fp);
             if(status == 0)
              {
            printf("I can't write program to Ram \n");
            fclose(fp);
            return -2;
              }

             change.hex = temp1;
             k = change.bin.b8_15;
             change.bin.b8_15 = change.bin.b0_7;
             change.bin.b0_7 = k;
             temp1 = change.hex;

             *(cardmem+i)=(unsigned int)temp1;


        outportb(CARD_PORT,0x07);
        for(i=0;i<( (file_size/2)-0x8000);i++)
        {
            status = fread(&temp1,sizeof(unsigned int),1,fp);
            if(status == 0)
             {
            printf("I can't write program to Ram \n");
            fclose(fp);
            return -2;

             }

             change.hex = temp1;
             k = change.bin.b8_15;
             change.bin.b8_15 = change.bin.b0_7;
             change.bin.b0_7 = k;
             temp1 = change.hex;

             *(cardmem+i)=(unsigned int)temp1;
        }
    }
    fclose(fp);
    outportb(CARD_PORT,0x00);      /* disable card */
    return 0;
//  printf("Program are already load \n");
}
```

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#include <time.h>

#define CARD_PORT   0x300
#define INT12BIT    0xef        /* 1 1 1 0 1 1 1 1 */
#define INT_CTRL1   0xa0
#define INT_CTRL2   0xa1
#define INT_NO      0x74      /* interrupr No. 12 */


#define DATA_LOW    0x00
#define DATA_HIGH   0x01
#define PROG_LOW    0x02
#define PROG_HIGH   0x03


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


void interrupt (*oldvec)(void);
void init_card(void);             /* initial card before used */
void end_usecard(void);           /* disable card after used */
void interrupt newisr12(void);    /* interrupt service rutine */
int run();

int run()
{
    unsigned long b;
    unsigned x,y;
    unsigned char intmask;
    unsigned char old_intmask;

    b=0;

    init_card();

    intcode = 1;

    outportb(CARD_PORT,0x10);        /*  0 0 0 1  0 0 0 0  */

    /* Enable interrupt IRQ12 */

    old_intmask = inportb(INT_CTRL2);    /* keep old interrupt mask register*/

    intmask = old_intmask & INT12BIT;    /* enable inturupt 12 bit */

    outportb(INT_CTRL2,intmask);

    outportb(0x301,0);              /* Clear interrupt */

    outportb(CARD_PORT,0x90);      /*  1 0 0 1  0 0 0 0  */

    for(x=0;(x<90)&&intcode;x++)
     for(y=0;(y<30)&&intcode;y++)
     {
       b++;
       delay(2);
     }

    outportb(CARD_PORT,0x00);         /*  0 0 0 0  0 1 0 1 */
```

```
    outportb(INT_CTRL2,old_intmask);    /* Disable interrupt */

    if(intcode == 0)
    {
      return 0;
    }
    else return -1;     /* program load O.K.  return 1 */
    end_usecard();
}

void interrupt newisr12(void)
{
    outportb(0x301,0);
    intcode=0;                      /* interrupt to tell TMS finish */
    outportb(INT_CTRL1,0x64);       /* end of interrupt */
    oldvec();
}

void init_card(void)
{
    unsigned char intmask;

    outportb(CARD_PORT,0x00);           /* 0 - - 0  - 0 0 0 */
    oldvec = getvect(INT_NO);
    setvect(INT_NO,newisr12);

}

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

```
#include <time.h>
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <process.h>

#define CARD_PORT   0x300
#define offset      0x400

//void mode(int mode_code);
//void putpoint(int x,int y,int color);
//void bwpalette(void);
//void display(int far *prt,int dimen,int xp,int yp);
void squr(int posx1,int posy1,int posx2,int posy2);

void main()
{
        unsigned int a,b,i,j,k,w,l,status,numline,temp,temp1,numcolumn,line1[30];
        unsigned int even,x1,x2,y1,y2;
        int dy,result,dim;
        long filesize;
        FILE *cp,*dp,*dp1,*dp2;
        char tex1[20],tex2[30],test;
//      char tex3[20] = "c:\\out\\a\\thres1.126";
//      char tex4[20] = "c:\\out\\a\\cut1.128";
//      char tex5[20] = "c:\\out\\a\\cut2.128";
//      char tex6[20] = "c:\\out\\a\\cut.126";

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

        clrscr();

        clearram();

        printf("Enter Graphic file :");
        gets(tex1);
        printf("Enter Program file :");
        gets(tex2);

        status = loadpic(tex1);
        switch(status)
        {
    case  -2 : exit(0);
    case  -1 : exit(0);
    case   0 : printf("Load picture sucess \n");
               break;
        }
        status = loadprg(tex2);
        switch(status)
        {
    case -2 : exit(0);
    case -1 : exit(0);
    case  0 : printf("Load Program sucess \n");
              break;
        }
        cp=fopen("c:\\out\\a\\dat2.lin","w+b");
        if( cp == NULL)
        {
    printf("I can't open write file\n");
    getch();
```

```
    exit(0);
        }

        printf("**********************\n");
        printf("stage1\n");

        printf("Scan of row \n");
        outportb(CARD_PORT,0x04);
        *(cardmem+0x5000) = 1;      /* row  or coloumn */
        *(cardmem+0x5001) = 0;      /* upper y position */
        *(cardmem+0x5002) = 0;      /* upper x position */
        *(cardmem+0x5003) = 127;    /* lower y position */
        *(cardmem+0x5004) = 127;    /* lower x position */
        outportb(CARD_PORT,0x00);   /*  0 0 0 0  0 1 0 1 */

       /* Start cut of row */
        result = spawnl(P_WAIT,"c:\\out\\a\\runrun.exe", NULL);
        if (result == -1)
        {
    printf("Error from spawnl");
    exit(1);
        }

        outportb(CARD_PORT,0x04);
        line1[0] = *(cardmem+0x5030);       /* numline */
        for(i=0;i<line1[0];i++)
        {
    line1[i+1] = *(cardmem+0x5031+i);
        }
//.....................................
/*
        for(i=0;i<line1[0];i++)
        {
              for(j=0;j<128;j++)
              {
                   *(cardmem+128*line1[i+1]+j+offset) = 0;
              }
        }
        dp1=fopen("c:\\out\\a\\cut1.128","w+b");
        if( dp1 == NULL)
        {
    printf("I can't open write file\n");
    getch();
    exit(0);
        }

        for(i=0;i<(128*128);i++)
        {
        test =(char)*(cardmem+i+offset);
        status  = fwrite(&test,sizeof(char),1,dp1);
        if(status != 1)
            {
        printf("I can't write data colum to file \n");
        fclose(dp1);
        getch();
        exit(0);
            }
        }
        fclose(dp1);
        status = ldpic(tex1);
        switch(status)
        {
    case -2 : exit(0);
    case -1 : exit(0);
    case  0 : printf("Load picture sucess \n");
              break;
```

```
*/
//.....................................

        printf("**********************\n");
        printf("stage 2\n");
        printf("Scan Column \n");

        for(i=0;i<(line1[0]/2);i++)
        {
        outportb(CARD_PORT,0x04);
        for(w=0;w<0x3500;w++)
        {
            *(cardmem+0x4500+w) = 0;
        }

       /* Set cut of coloumn */

    outportb(CARD_PORT,0x04);
    *(cardmem+0x5000) = 0;
    *(cardmem+0x5001) = line1[2*i+1];
    *(cardmem+0x5002) = 0;
    *(cardmem+0x5003) = line1[2*i+2];
    *(cardmem+0x5004) = 127;
    outportb(CARD_PORT,0x00);

         /* Begin cut of coloumn */

    result = spawnl(P_WAIT,"c:\\out\\a\\rurun.exe", NULL);
    if (result == -1)
    {
         printf("Error from spawnl");
         exit(1);
    }

    outportb(CARD_PORT,0x04);
    numcolumn = *(cardmem+0x5030);
    even = (unsigned int)numcolumn/2;
    even = even*2;
    if( numcolumn != even )
       numcolumn = even;
    printf("numcolumn %d is %d\n",i,numcolumn);
//  getch();

    status = fwrite(&numcolumn,sizeof(unsigned int),1,cp);
    if(status != 1)
    {
        printf("I can't write data column to file \n ");
        fclose(cp);
        getch();
        exit(0);
    }

    for(j=0;j<numcolumn;j++)
    {
            temp = *(cardmem+0x5031+j);
            status = fwrite(&temp,sizeof(unsigned int),1,cp);
            if(status != 1)
            {
                printf("I can't write data column to file \n ");
                fclose(cp);
                getch();
                exit(0);
            }
    }
         }
         fclose(cp);
```

```
/*
                fseek(cp,0L,SEEK_END);
                file_size=ftell(cp);
//              printf("File size is %d\n ",file_size);
//              getch();
                fseek(cp,0l,SEEK_SET);
                for(w=0;w<(file_size/2);w++)
                {
        status = fread(&temp,sizeof(unsigned int),1,cp);
        if(status != 1)
        {
                printf("w is  %d\n ",w);
                printf("I can't read data column to file \n ");
                fclose(cp);
                getch();
                exit(0);
        }
        *(cardmem+0x6000+w) = (unsigned int)temp;
                }
*/
                outportb(CARD_PORT,0x04);
//.....
/*
                cp=fopen("c:\\out\\a\\dat2.inn","r+b");
                if(cp == NULL)
                {
         printf("I can't open file data\n");
         getch();
         exit(0);
                }

                for(i=0;i<(line1[0]/2);i++)
                {
        status = fread(&temp,sizeof(unsigned int),1,cp);
        if(status != 1)
        {
                printf("I can't read data column to file \n ");
                fclose(cp);
                getch();
                exit(0);
        }
         /* number column each of line */
/*      numcolumn = temp;
//      printf("numcolumn is %d \n",numcolumn);
//      getch();
//      numcolumn = (unsigned int)numcolumn/2;
        dy = line1[2*i+2] - line1[2*i+1] + 1;
        for(j=0;j<numcolumn;j++)
        {
                status = fread(&temp,sizeof(unsigned int),1,cp);
                if(status != 1)
                {
                                printf("I can't read data column to file \n ");
                                fclose(cp);
                                getch();
                                exit(0);
                }
                x1 = (unsigned int)temp;
//              printf("x1 is %d\n",x1);
                for(k=0;k<dy;k++)
                {
                        *(cardmem+(128*(line1[2*i+1]+k))+x1+offset) = 0;
                }
        }
                }
                for(i=0;i<line1[0];i++)
```

```
        {
                for(j=0;j<128;j++)
                {
                 *(cardmem+(128*line1[i+1])+j+offset) = 0;
                }
        }

        dp2=fopen("c:\\out\\a\\cut2.128","w+b");
        if( dp1 == NULL)
        {
printf("I can't open write file\n");
getch();
exit(0);
        }

        for(i=0;i<(128*128);i++)
        {
        test =(char)*(cardmem+i+offset);
        status  = fwrite(&test,sizeof(char),1,dp2);
        if(status != 1)
                {
               printf("I can't write data column to file \n ");
               fclose(dp2);
               getch();
               exit(0);
                }
        }
        fclose(dp2);
        fclose(cp);

        status = ldpic(text);
        switch(status)
        {
case  -2 : exit(0);
case  -1 : exit(0);
case   0 : printf("Load picture sucess \n");
                break;
        }
*/
//.....................................................
        printf("**********************\n");
        printf("stage 3 \n");
        printf("Scan Row\n");

        outportb(CARD_PORT,0x04);

        cp=fopen("c:\\out\\a\\dat2.bin","r+b");
        if(cp == NULL)
        {
 printf("I can't open file data\n");
 getch();
 exit(0);
        }

        for(i=0;i<(line1[0]/2);i++)
        {
outportb(CARD_PORT,0x04);
status = fread(&temp,sizeof(unsigned int),1,cp);
if(status != 1)
{
        printf("I can't read data column to file \n ");
        fclose(cp);
        getch();
        exit(0);
}
```

```
        /* number column each of line */

        numcolumn = temp;
//      printf("****************\n");
//      printf("numcolumn is %d \n",numcolumn);

        numcolumn = (unsigned int)numcolumn/2;
        for(j=0;j<numcolumn;j++)
        {
                status = fread(&temp,sizeof(unsigned int),1,cp);
                if(status != 1)
                {
                        printf("I can't read data column to file \n ");
                        fclose(cp);
                        getch();
                        exit(0);
                }
                x1 = (unsigned int)temp;
//              printf("x1 is %d\n",x1);

                status = fread(&temp,sizeof(unsigned int),1,cp);
                if(status != 1)
                {
                        printf("I can't read data column to file \n ");
                        fclose(cp);
                        getch();
                        exit(0);
                }
                x2 = (unsigned int)temp;
//              printf("x2 is %d\n",x2);

                outportb(CARD_PORT,0x04);
                for(w=0;w<0x3500;w++)
                {
                        *(cardmem+0x4500+w) = 0;
                }

        /* Set cut of row */

                outportb(CARD_PORT,0x04);
                *(cardmem+0x5000) = 1;      /* cut in row each coloumn */
                *(cardmem+0x5001) = line1[(2*j)+1];
                *(cardmem+0x5002) = x1;
                *(cardmem+0x5003) = line1[(2*j)+2];
                *(cardmem+0x5004) = x2;
                outportb(CARD_PORT,0x00);

                /* Begin cut line each in column */

                result = spawnl(P_WAIT, "c:\\out\\a\\runrun.exe", NULL);
                if (result == -1)
                {
                        printf("Error from spawnl");
                        exit(1);
                }

                outportb(CARD_PORT,0x04);
                numline = *(cardmem+0x5030);
                if(numline > 8)
                {
//                      printf(" CPU TWS RUN ERROR !\n ");
//              for(a=0;a<45;a++)
//                      for(b=0;b<15;b++)
                                delay(2);

                    printf("Program success !\n");
```

```
                        getch();
                        exit(0);
                }
//              printf("numline is %d\n",numline);
//              getch();
                if(numline > 2)
                {
                        for(k=0;k<(numline/2);k++)
                        {
                y1 = *(cardmem+0x5031+(2*k));
                y2 = *(cardmem+0x5032+(2*k));
                squr(x1,y1,x2,y2);
                        }
                }
                else
                {
                        y1 = *(cardmem+0x5031);
                        y2 = *(cardmem+0x5032);
                        squr(x1,y1,x2,y2);
                }
        }
                }
//.....
/*
                dp=fopen("c:\\out\\a\\pcut.128","w+b");
                if( dp == NULL)
                {
        printf("I can't open file data\n");
        getch();
        exit(0);
                }

                outportb(CARD_PORT,0x04);
                for(i=0;i<(128*128);i++)
                {
                test =(char)* cardmem+offset);
                status  = fwrite(&test,sizeof(char),1,dp);
                if(status != 1)
                        {
                printf("I can't write data column to file \n");
                fclose(dp);
                getch();
                exit(0);
                        }
                }
                fclose(dp);
*/
//.....

                outportb(CARD_PORT,0x00);       /* 0 0 0 0  0 1 0 1 */
                fclose(cp);
                printf("Program sucess");
                getch();
/*
                ldpic(tex3);
                mode(0x10);
                display(cardmem+offset,128,20,10);
                ldpic(tex3);
                display(cardmem+offset,128,150,10);
                ldpic(tex3);
                display(cardmem+offset,128,20,150);
                ldpic(tex3);
                display(cardmem+offset,128,150,150);
                getch();
                mode(0x3);
*/
```

```
}

void squr(int posx1,int posy1,int posx2,int posy2)
{
        unsigned int dx,dy;
        register int i,w;

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

        dx = posx2 - posx1 +1;
        dy = posy2 - posy1 +1;
/*
    outportb(CARD_PORT,0x04);
    *(cardmem+0x7000) =0x77;
    for(w=0;w<0xfff;w++)
    {
         *(cardmem+0x7001+w) = 44;
    }
*/
    /* Draw line in Row */
        outportb(CARD_PORT,0x04);
        for(i=0;i<dx;i++)
        {
            *(cardmem+(128*posy1)+posx1+i+offset) = 0;
            *(cardmem+(128*posy2)+posx1+i+offset) = 0;
        }

    /* Draw line in Column */
        for(i=0;i<dy;i++)
        {
            *(cardmem+(128*(posy1+i))+posx1+offset) = 0;
            *(cardmem+(128*(posy1+i))+posx2+offset) = 0;
        }
}
*
void mode_int mode_code)
{
    union REGS r;
    r.h.al = mode_code;
    r.h.ah = 0;
    int86(0x10,&r,&r);
}

void putpoint(int x,int y,int color)
{
    union REGS r;
    r.h.bh = 0;
    r.h.ah = 12;
    r.h.al = color;
    r.x.dx = y;
    r.x.cx = x;
    int86(16,&r,&r);
}

void bwpalette(void)
{
    union REGS r;
    int i;
    for(i=0;i<64;i++)
        {
        r.h.ah = 0x10;
        r.h.al = 0x10;
        r.x.bx = i;
        r.h.dh = i;
        r.h.ch = i;
```

```
                r.h.cl = i;
                int86(16,&r,&r);
                }
}

void display(int far *ptr,int dimen,int xp,int yp)
{
        unsigned int i,j,step;
        int mx,mn,tmp;
        long diff;

        mx = mn = *ptr;
        for(i=0;i<dimen;i++)
                for(j=0;j<dimen;j++)
                        {
                        tmp = *(ptr+j+i*dimen);
                        if(tmp>mx)
                                mx = tmp;
                        else if(tmp<mn)
                                mn = tmp;
                        }
        if((diff=mx-mn)!=0)
                {
                step=diff/64;
                if((diff%64)>0)
                        step++;
                }
        else step=1;

        bwpalette();

        for(i=0;i<dimen;i++)
                for(j=0;j<dimen;j++)
                        {
                        tmp = (* ptr+j+i*dimen)-mn)/step;
                        putpoint j+xp,i+yp,tmp);
                        }

        getch();
}
*/
```

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>

typedef struct tagBITMAPFILEHEADER
{
        int bfType;
        long int bfSize;                /*246838*/
        int bfReserved1;
        int bfReserved2;
        long int bfoffBits;             /*1078*/
} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER
{
        long int biSize;                /*40*/
        long int biWidth;               /*640*/
        long int biHeight;              /*384*/
        int      biPlanes;
        int      biBitCount;

        long int biCompression;

        long int biSizeImage;           /* 640 * 384 = 245760 */
        long int biXPelsPerMeter;
        long int biYPelsPerMeter;
        long int ClrUsed;
} BITMAPINFOHEADER;

typedef struct tagRGBQUAD
{
        unsigned char rgbBlue;
        unsigned char rgbGreen;
        unsigned char rgbRed;
        unsigned char rgbReserved;
} RGBQUAD;

void mode(int mode_code);
void putpoint(int x,int y,int color);
void bwpalette(void);
void display(unsigned char huge *ptr,int xp,int yp);

void main(void)
{
        BITMAPFILEHEADER *BmFile;
        BITMAPINFOHEADER *BmInfoHeader;
        RGBQUAD *Rgb;
        FILE *fp;
        char tex[10];
        unsigned long int bfsize,size;
        unsigned char huge *pic;
        unsigned char temp;
        unsigned  int d,k,i,a,j,b,l,status;
        unsigned  int offset,wide,high;

        clrscr();
   printf("Enter your file name :");
   gets(tex);
   fp = fopen(tex,"r+b");
   if(fp==NULL)
```

```
    {
      printf("I can't open your file :");
      exit(0);
    }
    BmFile = (BITMAPFILEHEADER *)calloc(1,sizeof(BITMAPFILEHEADER));
    BmInfoHeader = (BITMAPINFOHEADER *)calloc(1,sizeof(BITMAPINFOHEADER));

    fread(BmFile,sizeof(BITMAPFILEHEADER),1,fp);
    bfsize = BmFile->bfSize;
    offset = BmFile->bfoffBits;

    fread(BmInfoHeader,sizeof(BITMAPINFOHEADER),1,fp);
    size = BmInfoHeader->biSizeImage;
    wide = BmInfoHeader->biWidth;
    high = BmInfoHeader->biHeight;

    fread(Rgb,sizeof(RGBQUAD),1,fp);


       k=ftell(fp);

    pic = (unsigned char huge *)farcalloc(bfsize,1);
    if(pic==NULL)
       {
       printf("Can not load");
       fclose(fp);
       exit(1);
       }
   fseek(fp,1120L,SEEK_SET);
    d=ftell(fp);

    for(i=169;i>0;i--)
     for(j=400;j>0;j--)
     {

          status = fread(&temp,sizeof(char),1,fp);
          if(status == 0)
            {
              printf("  fp %lo : ",ftell(fp));
              printf("Read file error  1 \n");
               fclose(fp);getch();
              exit(1);
            }

          *(pic+i*400-j)=(unsigned char)temp ;
     }


    fclose(fp);
    mode(0x13);
    display(pic,0,0);
    getch();
    mode(0x3);
    exit(0);


}


      /*SET MODE_CODE*/
void mode(int mode_code)
{     union REGS r;
      r.h.al = mode_code;
```

```
    r.h.ah = 0;
    int86(0x10,&r,&r);  /*INT 10H (VIDEO)*/
}
      /*SEND TO X,Y POSITION*/
void putpoint(int x,int y,int color)
{
    union REGS r;
    r.h.bh = 0;         /* PAGE 0 */
    r.h.ah = 12;        /* FUNCTION WRITE PIXEL */
    r.h.al = color;     /* PIXEL VALUE */
    r.x.dx = y;         /* ROW */
    r.x.cx = x;         /* COLUMN */
    int86(0x10,&r,&r);
}
      /*DEFINE COLOR(X,Y)*/
void bwpalette(void)
{
    union REGS r;
    int i;
    for(i=0;i<64;i++)
       {
       r.h.ah = 0x10;
       r.h.al = 0x10;
       r.x.bx = i;      /* DAC REG NUMBER */
       r.h.dh = i;      /* RED INTEN VALUE */
       r.h.ch = i;      /* GREEN INTEN VALUE */
       r.h.cl = i;      /* BLUE INTEN VALUE */
       int86(0x10,&r,&r);
       }
}
void display(unsigned char huge *ptr,int xp,int yp)
{
   unsigned i,j,tmp;

   bwpalette();

   for(i=0;i<159;i++)
        for(j=0;j<400;j++)
           {
           tmp =  (*(ptr+(i*400+j)))/4;
           putpoint(j+xp,i+yp,tmp);
           }
           getch();

}
```

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>


void mode(int mode_code);
void putpoint(int x,int y,int color);
void bwpalette(void);
void display(unsigned char huge *ptr,int xp,int yp);

void main(void)
{
        FILE *fp;
        char tex[10];
        unsigned long int bfsize;
        unsigned char huge *pic;
        unsigned char temp;
        unsigned  int d,k,i,a,j,b,l,c,e,status;
        unsigned  int coline[10][128],row[1][128],column[7][1][128],line[10] ;
        unsigned  int kl[7].offset,wide,high;
        unsigned  int aline[1][128];

        clrscr();

   printf("Enter your file name :");
   gets(tex);
   fp = fopen(tex,"r+b");
   if(fp==NULL)
   {
     printf("I can't open your file :");
     exit(0);
   }


   bfsize = 128*128;

   pic = (unsigned char huge *)farcalloc(bfsize,1);
   if(pic==NULL)
      {
      printf("Can not load");
      fclose(fp);
      exit(1);
      }
  fseek(fp,0L,SEEK_SET);


   for(i=0;i<128;i++)
    for(j=0;j<128;j++)
    {

         status = fread(&temp,sizeof(char),1,fp);
         if(status == 0)
           {
             printf(" fp %ld  :",ftell(fp));
             printf("Read file error  1 \n");
              fclose(fp);getch();
             exit(1);
           }
         *(pic+i*128+j)=(unsigned char)temp ;
    }
   mode(0x13);
```

```
display(pic,20,10);
mode(0x3);


      for(j=0;j<128;j++)
        for(i=0;i<2;i++)
        {   row[i][j] = 0x00 ; } ;


      for(j=0;j<128;j++)
        for(i=0;i<128;i++)

       {
       /*  CLIP ROW  */
       /*  ROW[1][128] => 1   : COLOR
                          128 : NO. ROW        */

            temp = *(pic+j*128+i);

               if(temp==0x00)
               row[0][j]++;
               else
               if(temp==0x0f)
               row[1][j]++;
        };


       d=0;l=1;
loop:   for(j=d+1;j<128;j++)
       {
          if( row[0][j] > 0 )              /*BLACK*/
           {
             for(i=d;i<128;i++)
               {
                 if( row[0][j] == 0 )      /*WHITE*/
                   {
                     d=j;
                     line[l]   = j;
                     line[l+1] = j;l=l+2;
                     goto loop;

                   };
               };
           };
       };




        /*CLIP COLUMN*/
        /*COLUMN [7][i][400]  =>   3   :  FIRST LINE OF COLUMN   0,1,2
                                   1   :  COLOR
                                   400 :  NO. COLUMN
                                                                 */


        /*CLEAR COLUMN*/
                                    /* b => NO.line
                                       j => color.
                                       i => NO.column
                                                                 */
```

```
  for(j=0;j<128;j++)
    for(i=0;i<2;i++)
      row[i][j] = 0x00;


for(i=0;i<128;j++)
 for(b=1;b<1;b=b+2)
   for(j=0;j<2;j++)
      column[b][j][i] = 0x00;


/*COLLECT DATA*/

 for(b=1;b<1;b=b+2)
   for(i=0;i<=128;i++)
    for(j=line[b];j<=line[b+1];j++)
       {

         temp = *(pic+j*128+i) ;

          if(temp==0x00)
            {
              column[b][0][i]++;
            }
          else
          if(temp==0xff)
            {
              column[b][1][i]++;
            } ;

       };


 /*CLIP*/          /*COLINE[10][128] => 10  : NO.LINE
                                      128 : NO.COLUMN */

   d=0;
   for(b=1;b<1;b=b+2)
     {
        kl[b]=1;
loop1:  for(j=d+1;j<=128;j++)
       {

          if(j==128)
            d=1;

         if(column[b][0][j] > 0 )              /*BLACK*/
           {
             for(i=j+1;i<=128;i++)
              {
                 if(i==128)
                 {
                   d=0;
                 }else
                 if(i==128 && b==5)
                 {
                   goto loop2;
                 }else
                 if(column[b][0][i] == 0 )      /*WHITE*/
                 {
                   d=i;
                   coline[b][kl[b]]   = j ;
                   coline[b][kl[b]+1] = i ;
                   kl[b]=kl[b]+2 ;

                   goto loop1 ;
```

```
                    };
                  };
                };
              };
            };

loop2:      for(i=0;i<128;i++)
          for(b=1;b<l;b=b+2)
            for(j=0;j<2;j++)
               column[b][j][i] = 0x00;



          /*DISPLAY ROW*/

        for(b=1;b<l;b++)
          for(i=1;i<=128;i++)
            for(j=0;j<128;j++)
             {
                if( i == line[b] )
                {
                   if(b%2 == 1)
                     *(pic+(i-1)*128+j) = 0x00 ;
                   else
                   if(b%2 == 0)
                     *(pic+(i+1)*128+j) = 0x00 ;

                }
             }
        mode(0x13);
        display(pic,20,10);
        mode(0x3);




                /*DISPLAY column*/

        for(b=1;b<=6;b=b+2)
          for(i=1;i<=128;i++)
            for(a=1;a<=l[b];a++)
              for(j=1;j<=128;j++)
               {
                if( (i == line[b]) && (j == coline[b][a]) )
                  {
                   for(k=line[b];k<=line[b+1];k++)

                    {
                      if(a%2 == 1)
                        *(pic+k*128+coline[b][a]-1) = 0x00 ;
                      else
                      if(a%2 == 0)
                        *(pic+k*128+coline[b][a]+1) = 0x00 ;

                    }

                  }
               }
```

```
mode(0x13);  /*GRAPHIC*/
display(pic,20,10);
mode(0x3);   /*TEXT*/



/* CLIP ROW AGAIN */


    for(j=0;j<128;j++)
      for(i=0;i<2;i++)
        aline[i][j] = 0x00 ;

  for(b=1;b<7;b=b+2)
    for(a=1;a<kl[b];a=a+2)

    {
          /*  CLIP ROW  */
          /*  ROW[1][126] => 1   : COLOR
                             128 : NO. ROW        */


      for(j=line[b];j<line[b+1];j++)
        for(i=coline[b][a];i<coline[b][a+1];i++)
          {
              temp = *(pic+j*128+i);

              if(temp==0x00)
              aline[0][j]++;
              else
              if(temp==0xff)
              aline[1][j]++;

          }

        d=line[b]+1;
      for(j=d+1;j<line[b+1];j++)
        {
           if( aline[0][j] >0 )              /*BLACK*/
            {
              for(i=j-1;i<line[b+1];i++)
                {
                   if(i==line[b+1]-2)
                   goto loop3;
                   else
                   if( aline[0][i]   == 0 )      /*WHITE*/
                     {
                       for(c=i+1;c<line[b+1];c++)

                       {
                         if(aline[0][c] >0)    /*BLACK*/
                           {
                             for(e=coline[b][a];e<=coline[b][a+1];e++)
                             *(pic+(j)*128+e) = 0x00 ;
                             goto loop3;
                           }

                       }
                     }
                }
            }
        }
    }
```

```
loop3 :         for(j=0;j<128;j++)
                  for(i=0;i<2;i++)
                    aline[i][j] = 0x00 ;



        }


                for(j=0;j<128;j++)
                  for(i=0;i<2;i++)
                    aline[i][j] = 0x00 ;

                for(j=0;j<=10;j++)
                 for(i=0;i<=128;i++)
                    coline[10][128] = 0x00;

   fclose(fp);
   mode(0x13);  /*GRAPHIC*/
   display(pic,20,10);
   mode(0x3);   /*TEXT*/
   farfree(pic);
   exit(0);

  }

       /*SET MODE_CODE*/
void mode(int mode_code)
{    union REGS r;
     r.h.al = mode_code;
     r.h.ah = 0;
     int86(0x10,&r,&r);  /*INT 10H (VIDEO)*
 }
       /*SEND TO X,Y POSITION*
 void putpoint(int x,int y,int color
 {
     union REGS r;
     r.h.bh = 0;         /* PAGE 0 */
     r.h.ah = 12;        /* FUNCTION WRITE PIXEL */
     r.h.al = color;     /* PIXEL VALUE */
     r.x.dx = y;         /* ROW */
     r.x.cx = x;         /* COLUMN */
     int86(0x10,&r,&r);
 }
       /*DEFINE COLOR(X,Y)*/
 void bwpalette(void)
 {
     union REGS r;
     int i;
     for(i=0;i<64;i++)
        {
        r.h.ah = 0x10;
        r.h.al = 0x10;
        r.x.bx = i;      /* DAC REG NUMBER */
        r.h.dh = i;      /* RED INTEN VALUE */
        r.h.ch = i;      /* GREEN INTEN VALUE */
        r.h.cl = i;      /* BLUE INTEN VALUE */
        int86(0x10,&r,&r);
        }
 }
 void display(unsigned char huge *ptr,int xp,int yp)
 {
    unsigned i,j,tmp;
    bwpalette();
```

```
   for(i=0;i<128;i++)
        for(j=0;j<128;j++)
           {
           tmp =  (*(ptr+(i*128+j)))/4;
           putpoint(j+xp,i+yp,tmp);
           }
           getch();

}
```

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>


void main(void)
{
        FILE *fp,*fp1;
        char tex[10];
        unsigned long int offset,high,wide,size,bisize;
        unsigned char huge *pic;
        unsigned char temp,temp1;
        unsigned long int accu,i,j,status,status1,status2;
        unsigned long int  threshold;


        clrscr();
        printf("Enter your file name to histogram : ");
        gets(tex);

        fp = fopen(tex,"r+b");
        if(fp==NULL)
        {
                printf("I can't open your file.");
                getch();
                exit(0);
                fclose(fp);
        }


        bisize = 128*128;
        offset = 0;

        /*FIND ARRAY[i][j]*/
        pic = (unsigned char huge *)farcalloc(bisize,1);
        if(pic==NULL)
          {
            printf("I Can't reserve pic");
            fclose(fp);getch();
            exit(1);
          }


        for(i=0;i<bisize;i++)
         {
            status = fread(&temp,sizeof(char),1,fp);
              if(status==NULL)
                {
                  printf(" Read file too pic error :");
                  getch();fclose(fp);
                  exit(1);
                }
            *(pic+i) = (unsigned char)temp;
         } ;
```

```
  /*CLIPPER PICTURE AT THRESHOLD VALUE */

threshold=28;

 fp1 = fopen("a:\\threshol.128","wb");
   if(fp1==NULL)
     {
       printf("I can't write new file");
       getch();
       exit(0);
       fclose(fp1);
     }



for(i=0;i<bisize;i++)
   {

      temp1 = *(pic+i);

      if(temp1 < threshold)
         temp1 = 0xff;

      else
      if(temp1 > threshold)
         temp1 = 0x00;


      status1 = fwrite(&temp1,sizeof(char),1,fp1);
      if(status1==NULL)
        {
          printf("Write file pic1 error  i=%d ",i);
          getch();fclose(fp1);
          exit(1);
        };

   };



fclose(fp);
fclose(fp1);
farfree(pic);
getch();
getch();


}
```

```
#include <graphics.h>
#include <stdio.h>
#include <alloc.h>
#include <math.h>
#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <float.h>
#include <time.h>

typedef struct{
              float
              real,imag;
              } COMPLEX;

void cpxdplay(COMPLEX huge *ptr,int d,int x,int y);
void dplay(unsigned long huge *ptr,int d,int x,int y,unsigned long m);
void setpllt(void);
void putpoint(int x,int y,unsigned int color);
void mode(int mode_code);
int log2(int x);
void fft(COMPLEX huge *x,int m,int g);
void ifft(COMPLEX huge *x,int m,int g);
unsigned long absolt(COMPLEX a);
COMPLEX cmplx(float x,float y);
void transp(COMPLEX huge *x,COMPLEX huge *y,int d);

static COMPLEX *w;
static COMPLEX *w1;
main()
{

        char huge *fp,*fp1;
        COMPLEX huge *array1,*array2,*temp1,*temp2;
        unsigned char huge *image;
        unsigned long *data1,*data2;
        unsigned long size;
        int i,j,k,fx,d;
        unsigned long status1,status,x,m>,mx;
        char te>[30];
        int dims;
        int lg2dm;
        COMPLEX tp;
        struct  time begin,stop;
        time_t first,second;
        char string,temp1;

        clrscr();

        if((fp=fopen("c:\\lek\\out.128","rb"))==NULL)
        {
               printf("Can't open file");
               getch();
               fclose(fp);
               exit(2);
        }

        fseek(fp,0L,SEEK_END);
        size=ftell(fp);
        fseek(fp,0L,SEEK_SET);
        if((fp1=fopen("a:\\filter.128","wb"))==NULL)
        {
```

```
        printf("Can't open file to write :");
        getch();
        fclose(fp);
        exit(2);
}


for(j=0;j<2;j++)
{
      if(j==0)
      {
          if((image=(unsigned char huge *) farmalloc(size/2))==NULL)
           {
             printf("1 Out of memory");
             getch();
             exit(2);
           }
          if((ami1=(COMPLEX huge *)farcalloc(size/2,sizeof(COMPLEX)))==NULL)
            {
             printf("2 Out of memory");
             getch();
             exit(2);
            }


           status = fread(image,sizeof(char),size/2,fp);
           if(status != size/2)
             {
              printf("Read file %d error\n",i);
              getch();
              exit(2);
             }
           for(i=0;i<size/2;i++)
           {
             *(ami1+i) = cmplx(*(image+i),0);
           }

           farfree(image);


      }else

     if(j==1)
     {

          if((image=(unsigned char huge *) farmalloc(size/2))==NULL)
           {
             printf("1 Out of memory");
             getch();
             exit(2);
           }
          if((ami2=(COMPLEX huge *)farcalloc(size/2,sizeof(COMPLEX)))==NULL)
            {
             printf("2 Out of memory");
             getch();
             exit(2);
            }


           status = fread(image,sizeof(char),size/2,fp);
           if(status != size/2)
             {
              printf("Read file %d error\n",i);
              getch();
              exit(2);
```

```
        ami1 = bami1;
        ami2 = bami2;

        transp(ami1,ami2,dims);

        for(i=0;i<size/2;i++)
        {
            (*(ami1+i)).real = (*(ami1+i)).real*pow(1.00,(double)i);
            (*(ami1+i)).imag = (*(ami1+i)).imag*pow(1.00,(double)i);
        }

        for(i=0;i<size/2;i++)
        {
            (*(ami2+i)).real = (*(ami2+i)).real*pow(1.00,(double)i);
            (*(ami2+i)).imag = (*(ami2+i)).imag*pow(1.00,(double)i);
        }


        bami1 = ami1;
        for(i=0;i<dims/2;i++)
        {
            ami1 = bami1+(i*dims);
            fft(ami1,lg2dm,2);

        }
        bami2 = ami2;
        for(i=0;i<dims/2;i++)
        {
            ami2 = bami2+(i*dims);
            fft(ami2,lg2dm,2);


        }


        ami1 = bami1;
        ami2 = bami2;

        free(w);


    /*  HIGH PASS FILTER แบบ fx,d  */

        d=3;

        for(i=0;i<=d;i++)
         {
          fx =  sqrt( (d*d) - (i*i) );

           for(j=0;j<=fx;j++)
            {
              (*(ami1+j+i*128)).real = 0.00 ;
              (*(ami1+j+i*128)).imag = 0.00 ;

              (*(ami1+(128-j)+i*128)).real = 0.00 ;
              (*(ami1+(128-j)+i*128)).imag = 0.00 ;

            }
         }

        for(i=0;i<=d;i++)
```

```
{
 fx =  sqrt( (d*d) - (i*i) );

   for(j=0;j<=fx;j++)
    {
       (*(ami2+j+ ( (127-i) *128) )).real = 0.00 ;
       (*(ami2+j+ ( (127-i) *128) )).imag = 0.00 ;

       (*(ami2+(128-j) + ( (127-i) *128) )).real = 0.00 ;
       (*(ami2+(128-j) + ( (127-i)*128 ) )).imag = 0.00 ;

    }


}
```

```
printf("doing ifft 2 dimension , please wait");

gettime(&begin);
printf("\n\nThe current time is: %2d:%02d:%02d.%02d\n",
begin.ti_hour, begin.ti_min, begin.ti_sec, begin.ti_hund);
first = time(NULL);
printf("The number of seconds since January 1, 1970 is %ld",first);


for(i=0;i<size/2;i++)
{
     (*(ami1+i)).real = (*(ami1+i)).real*pow(1.00,(double)i);
     (*(ami1+i)).imag = (*(ami1+i)).imag*pow(1.00,(double)i);
}

for(i=0;i<size/2;i++)
{
     (*(ami2+i)).real = (*(ami2+i)).real*pow(1.00,(double)i);
     (*(ami2+i)).imag = (*(ami2+i)).imag*pow(1.00,(double)i);
}

bami1 = ami1;
for(i=0;i<dims/2;i++)
{

     ami1 = bami1+(i*dims);
     ifft(ami1,lg2dm,1);

}

bami2 = ami2;
for(i=0;i<dims/2;i++)
{


     ami2 = bami2+(i*dims);

     ifft(ami2,lg2dm,1);
}
ami1 = bami1;
ami2 = bami2;
```

```
transp(ami1,ami2,dims);


for(i=0;i<size/2;i++)
{
    (*(ami1+i)).real = (*(ami1+i)).real*pow(1.00,(double)i);
    (*(ami1+i)).imag = (*(ami1+i)).imag*pow(1.00,(double)i);
}

for(i=0;i<size/2;i++)
{
    (*(ami2+i)).real = (*(ami2+i)).real*pow(1.00,(double)i);
    (*(ami2+i)).imag = (*(ami2+i)).imag*pow(1.00,(double)i);
}


bami1 = ami1;
for(i=0;i<dims/2;i++)
{
    ami1 = bami1+(i*dims);
    ifft(ami1,lg2dm,2);
}

bami2 = ami2;
for(i=0;i<dims/2;i++)
{
    ami2 = bami2+(i*dims);
    ifft(ami2,lg2dm,2);
}


ami1 = bami1;
ami2 = bami2;


free(mx);

if((dabs1=(unsigned long *)farcalloc(size/2,sizeof(unsigned long)))==NULL)
{
    printf("4 Out of memory");
    getch();
    exit(2);
}

if((dabs2=(unsigned long *)farcalloc(size/2,sizeof(unsigned long)))==NULL)
{
    printf("4 Out of memory");
    getch();
    exit(2);
}


for(i=0;i<size 2;i++)
    *(dabs1+i) = absolt(*(ami1+i));
for(i=0;i<size/2;i++)
    *(dabs2+i) = absolt(*(ami2+i));


mx = 0;
for(i=0;i<size/2;i++)
    mx = max(*(dabs1+i),mx);
for(i=0;i<size/2;i++)
    mx = max(*(dabs2+i),mx);
```

```
mxx = 12000;
for(i=0;i<size/2;i++)
{
      if((*(dabs1+i)) > mxx)
        *(dabs1+i) = mxx;
}
for(i=0;i<size/2;i++)
{
      if((*(dabs2+i)) > mxx)
        *(dabs2+i) = mxx;
}


/* WRITE TO FILE  */



for(i=0;i<size/2;i++)
{
  temp1   =   (*(dabs1+i)*63)/mxx ;
  status1 =   fwrite(&temp1,sizeof(char),1,fp1);
   if(status1 == NULL)
    {
    printf("write  to file error");
    getch();
    exit(0);

    }

}

for(i=0;i<size/2;i++)
{
  temp1   =   (*(dabs2+i)*63)/mxx ;
  status1 =   fwrite(&temp1,sizeof(char),1,fp1);
   if(status1 == NULL)
    {
    printf("write  to file error");
    getch();
    exit(0);

    }

}



gettime(&stop);
printf("\nThe current time is: %2d:%02d:%02d.%02d\n",
stop.ti_hour, stop.ti_min, stop.ti_sec, stop.ti_hund);
second = time(NULL);
printf("The number of seconds since January 1, 1970 is %ld",second);
printf("\n\nThe difference is: %f seconds\n",difftime(second,first));
getch();


mode(0x13);
setpltt();
dplay(dabs1,dims,75,35,mxx);
dplay(dabs2,dims,75,35+dims/2,mxx);
getch();
mode(3);
```

```
            }
            for(i=0;i<size/2;i++)
            {
                *(ami2+i) = cmplx(*(image+i),0);
            }

             farfree(image);

        }

}


dims = (int)sqrt(size);
printf("\n dims = %u \n",dims);
lg2dm = log2(dims);
printf("\n lg2dm = %u \n",lg2dm);
getch();



  fclose(fp);

  mode(0x13);
  setpllti();
  cpxdplay(ami1,dims,75,35);
  cpxdplay(ami2,dims,75,35+dims/2);
  getch();
  mode(3);

  printf("doing fft 2 dimension , please wait");

  gettime(&begin);
  printf("\n\nThe current time is: %2d:%02d:%02d.%02d\n",
  begin.ti_hour, begin.ti_min, begin.ti_sec, begin.ti_hund);
  first = time(NULL);
  printf("The number of seconds since January 1, 1970 is %ld \n",first);



  for(i=0;i<size/2;i++)
  {
       (*(ami1+i)).real = (*(ami1+i)).real*pow(1.00,(double)i);
       (*(ami1+i)).imag = (*(ami1+i)).imag*pow(1.00,(double)i);
  }

  for(i=0;i<size/2;i++)
  {
       (*(ami2+i)).real = (*(ami2+i)).real*pow(1.00,(double)i);
       (*(ami2+i)).imag = (*(ami2+i)).imag*pow(1.00,(double)i);
  }


  bami1 = ami1;
  for(i=0;i<dims/2;i++)
  {
       ami1 = bami1+(i*dims);
       fft(ami1,lg2dm,1);
  }
  bami2 = ami2;
  for(i=0;i<dims/2;i++)
  {
       ami2 = bami2+(i*dims);
       fft(ami2,lg2dm,1);
  }
```

```
        farfree(ami1);
        farfree(ami2);
        farfree(dabs1);
        farfree(dabs2);
        exit(2);

}


/*****************************************************************/

void dplay(unsigned long huge *ptr,int d,int x,int y,unsigned long m)
{
        int i,j;
        unsigned int color;

        for(j=0;j<d/2;j++)
        for(i=0;i<d;i++)
         {
              color = (int)(((*(ptr+(j*d)+i))*63)/m);
              putpoint(i+x,j+y,color);
         }
}


void cpxdplay(COMPLEX huge *ptr,int d,int x,int y)
{
        COMPLEX col;
        int i,j;
        unsigned int color;

        for(j=0;j<d/2;j++)
        for(i=0;i<d;i++)
        {
                col = *(ptr+(j*d)+i);
                color = absolt(col)>>2;
                putpoint(i+x,j+y,color);
        }
}

void mode(int mode_code)
{
        union REGS r;
        r.h.al = mode_code;
        r.h.ah = 0;
        int86(0x10,&r,&r);
}

void putpoint(int x,int y,unsigned int color)
{
        union REGS r;
        r.h.bh = 0;
        r.h.ah = 12;

        r.h.al = color;
        r.x.dx = y;
        r.x.cx = x;
        int86(16,&r,&r);
}
```



```
/* set rgb palette */
void setpllt(void)
{
```

```
        union REGS r;
        int i;
        for(i=0;i<64;i++)
        {
        r.h.ah = 0x10;
        r.h.al = 0x10;
        r.x.bx = i;       /* CLUT-register number */
        r.h.dh = i;       /* red value to set */
        r.h.ch = i;       /* green value to set */
        r.h.cl = i;       /* blue value to set */
        int86(16,&r,&r);
        }
}


int log2(int x)
{
     unsigned int mask,i;


     if(x == 0) return(-1);
     x--;

     for(mask = 1 , i = 0 ; ; mask *= 2 , i++)
     {
       if(x == 0) return(i);
       x = x & (~mask);
     }
}


void fft(COMPLEX huge *x,int m,int g)
{

    static int mstore = 0;
    static int n = 1;
    COMPLEX u,temp,tm;
    COMPLEX *xi,*xip,*xj,*wptr;
    int i,j,k,l,le,windex;
    double arg,w_real,w_imag,wrecur_real,wrecur_imag,wtemp_real;
    float scale;


    if(m != mstore)
    {
        if(mstore != 0) free(w);
        mstore = m;
        if(m == 0) return;

        n = 1 << m;
        le = n/2;

        w = (COMPLEX *) calloc(le-1,sizeof(COMPLEX));
        if(!w)
        {
            printf("\nUnable to allocate complex W array\n");
            exit(1);
        }

        arg = 4.0*atan(1.0)/le;
        wrecur_real = w_real = cos(arg);
        wrecur_imag = w_imag = -sin(arg);
        xj = w;
        for (j = 1 ; j < le ; j++)
        {
            xj->real = (float)wrecur_real;
```

```
            xj->imag = (float)wrecur_imag;
            xj++;
            wtemp_real = wrecur_real*w_real - wrecur_imag*w_imag;
            wrecur_imag = wrecur_real*w_imag + wrecur_imag*w_real;
            wrecur_real = wtemp_real;
        }
    }


    le = n;
    windex = 1;
    for (l = 0 ; l < m ; l++)
    {
        le = le/2;

        for(i = 0 ; i < n ; i = i + 2*le)
        {
            xi = x + i;
            xip = xi + le;
            temp.real = (xi->real + xip->real);
            temp.imag = (xi->imag + xip->imag);
            xip->real = xi->real - xip->real;
            xip->imag = xi->imag - xip->imag;
            *xi = temp;

        }

        wptr = w + windex - 1;
        for (j = 1 ; j < le ; j++)
        {
            u = *wptr;
            for (i = j ; i < n ; i = i + 2*le)
            {
                xi = x + i;
                xip = xi + le;
                temp.real = (xi->real + xip->real);
                temp.imag = (xi->imag + xip->imag);
                tm.real = xi->real - xip->real;
                tm.imag = xi->imag - xip->imag;
                xip->real = tm.real*u.real - tm.imag*u.imag;
                xip->imag = tm.real*u.imag + tm.imag*u.real;
                *xi = temp;
            }
            wptr = wptr + windex;
        }
        windex = 2*windex;
    }


    j = 0;
    for (i = 1 ; i < (n-1) ; i++)
    {
        k = n/2;
        while(k <= j)
        {
            j = j - k;
            k = k/2;
        }
        j = j + k;
        if (i < j)
        {
            xi = x + i;
            xj = x + j;
            temp = *xj;
            *xj = *xi;
            *xi = temp;
```

```
        }
    }


}

unsigned long absolt(COMPLEX a)
{
     struct complex z;
     unsigned long val;


     z.x = (double)a.real;
     z.y = (double)a.imag;
     val = (unsigned long)cabs(z);
     return(val);

}

COMPLEX cmplx(float x,float y)
{

     COMPLEX  z;

     z.real = x;
     z.imag = y;
     return(z);
}

void transc(COMPLEX huge *x,COMPLEX huge *y,int d)
{
       int i,j,k;
       COMPLEX temp;

       k = 0;
       for(j=0;j<d/2;j++)
       {
           for(i=k;i<d/2;i++)
           {
                temp = *(x+i+(j*d));
                *(x+i+(j*d)) = *(x+j+(i*d));
                *(x+j+(i*d)) = temp;
           }
           k++;
       }
       for(j=0;j<d/2;j++)
       {
           for(i=0;i<d/2;i++)
           {
                temp = *(x+i+d/2+(j*d));
                *(y+i+d/2+(j*d)) = *(y+j+(i*d));
                *(y+j+(i*d)) = temp;
           }
       }
       k = 0;
       for(j=0;j<d/2;j++)
       {
           for(i=k;i<d/2;i++)
           {
                temp = *(y+i+d/2+(j*d));
                *(y+i+d/2+(j*d)) = *(y+j+(i*d)+d/2);
                *(y+j+(i*d)+d/2) = temp;
           }
       k++;
```

```
    }

}


void ifft(COMPLEX huge *x,int m,int g)
{

    static int mistore = 0;
    static int ni = 1;
    COMPLEX u,temp,tm;
    COMPLEX *xi,*xip,*xj,*wptr;
    int i,j,k,l,le,windex;
    double arg,w_real,w_imag,wrecur_real,wrecur_imag,wtemp_real;
    float scale;



    if(m != mistore)
    {
        if(mistore != 0) free(wi);
        mistore = m;
        if(m == 0) return;

        ni = 1 << m;
        le = ni/2;

        wi = (COMPLEX *) calloc(le-1,sizeof(COMPLEX));
        if(!wi)
        {
            printf("\nUnable to allocate complex W array\n");
            exit(1);
        }



        arg = 4.0*atan(1.0)/le;
        wrecur_real = w_real = cos(arg);
        wrecur_imag = w_imag = sin(arg);
        xj = wi;
        for (j = 1 ; j < le ; j++)
        {
            xj->real = (float)wrecur_real;
            xj->imag = (float)wrecur_imag;
            xj++;

            wtemp_real = wrecur_real*w_real - wrecur_imag*w_imag;
            wrecur_imag = wrecur_real*w_imag + wrecur_imag*w_real;
            wrecur_real = wtemp_real;
        }
    }


    le = ni;
    windex = 1;
    for (l = 0 ; l < m ; l++)
    {
        le = le/2;

        for(i = 0 ; i < ni ; i = i + 2*le)

        {
            xi = x + i;
            xip = xi + le;
            temp.real = (xi->real + xip->real);
```

```
        temp.imag = (xi->imag + xip->imag);
        xip->real = xi->real - xip->real;
        xip->imag = xi->imag - xip->imag;
        *xi = temp;

    }

    wptr = wi + windex - 1;
    for (j = 1 ; j < le ; j++)
    {
        u = *wptr;
        for (i = j ; i < ni ; i = i + 2*le)
        {
            xi = x + i;
            xip = xi + le;
            temp.real = (xi->real + xip->real);
            temp.imag = (xi->imag + xip->imag);
            tm.real = xi->real - xip->real;
            tm.imag = xi->imag - xip->imag;
            xip->real = tm.real*u.real - tm.imag*u.imag;
            xip->imag = tm.real*u.imag + tm.imag*u.real;
            *xi = temp;
        }
        wptr = wptr + windex;
    }
    windex = 2*windex;
}


j = 0;
for (i = 1 ; i < (ni-1) ; i++)
{
    k = ni/2;
    while(k <= j)
    {
        j = j - k;
        k = k/2;
    }
    j = j + k;
    if (i < j)
    {
        xi = x + i;
        xj = x + j;
        temp = *xj;
        *xj = *xi;
        *xi = temp;

    }
}

if(ig==2)
{
  scale = (float)(1.0/(200));
  for(i=0;i<ni;i++)
  {
      x->real = scale*x->real;
      x->imag = scale*x->imag;
      x++;

  }
}
```

```
                    /*PROGRAM  PROJECT "DSP",TERM1*/
/*SELECT PROG.MEM,DATA.MEM OR PROCESS*/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>


void mode(int mode_code);
void putpoint(int x,int y,int color);
void bwpalette(void);
void display(unsigned char huge *ptr,int dimen,int xp,int yp);
void main()
{
   FILE *fp;
   long i,j,status,file_size;
   char tex[10];
   unsigned off_set,dim;
   unsigned char temp;
   unsigned char huge *pic;
   clrscr();
   printf("Enter cata file to monitor :\n");
   gets(tex);
   if((fp=fopen(tex,"rb"))==NULL)
     {
        printf("Can not open file");
        getch();
        fclose(fp);
        exit(1);
     }
   fseek(fp,0l,SEEK_END);
   file_size=ftell(fp) ;

   dim=(unsigned)sqrt((double)file_size);
   if(file_size>0x90000)
     {
     printf("File too big ,can not load file :\n");
     fclose(fp);
     exit(1);

     }
   pic = (unsigned char huge *)farcalloc(file_size,1);
   if(pic==NULL)
      {
      printf("Can not load");
      fclose(fp);
      exit(1);
      }

   fseek(fp,0l,SEEK_SET);
   for(i=0;i<file_size;i++)
     {
         status = fread(&temp,sizeof(char),1,fp);
         if(status == 0)
         {
         printf("Read file error \n");
         fclose(fp);getch();
         exit(1);
         }
   *(pic+i)=(unsigned)temp;
```

```
    }

   fclose(fp);
   printf("dimension = %u\n",dim);
   mode(0x13);  /*GRAPHIC*/
   display(pic,dim,20,10);
   getch();
   mode(0x3);   /*TEXT*/
   farfree(pic);
   getch();
}
      /*SET MODE_CODE*/
void mode(int mode_code)
{    union REGS r;
     r.h.al = mode_code;
     r.h.ah = 0;
     int86(0x10,&r,&r);  /*INT 10H (VIDEO)*/
 }
       /*SEND TO X,Y POSITION*/
 void putpoint(int x,int y,int color)
 {
     union REGS r;
     r.h.bh = 0;        /* PAGE 0 */
     r.h.ah = 12;       /* FUNCTION WRITE PIXEL */
     r.h.al = color;    /* PIXEL VALUE */
     r.x.dx = y;        /* ROW */
     r.x.cx = x;        /* COLUMN */
     int86(0x10,&r,&r);
 }
       /*DEFINE COLOR(X,Y)*/
 void bwpalette(void)
 {
     union REGS r;
     int i;
     for(i=0;i<64;i++)
        {
        r.h.ah = 0x10;
        r.h.al = 0x10;
        r.x.bx = i;      /* DAC REG NUMBER */
        r.h.dh = i;      /* RED INTEN VALUE */
        r.h.ch = i;      /* GREEN INTEN VALUE */
        r.h.cl = i;      /* BLUE INTEN VALUE */
        int86(0x10,&r,&r);
        }
 }
 void display(unsigned char huge *ptr,int dimen,int xp,int yp)
{
    unsigned int i,j,tmp,temp,threshol;
    bwpalette();


    threshol = 28;

    for(i=0;i<dimen;i++)
         for(j=0;j<dimen;j++)
            {

            temp =  (*(ptr+j+i*dimen));

            if(temp<threshol)
            {
             temp = 0xff;
            }
            else
```

```
            if(temp>threshol)
            {
             temp = 0x00;
            }
            temp = temp/4 ;

            putpoint(j+xp,i+yp,temp);
            }
            getch();

}
```

```
/* HISTROGRAM */

#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>


void Initialize_Graphics_Mode(void);
void main(void)
{
        FILE *fp,*fp1;
        char tex[10],new[10];
        unsigned long int offset,high,wide,size,bisize;
        unsigned char huge *pic,*pic1;
        unsigned char temp,temp1;
        unsigned long int accu,i,j,status,status1,status2;
        unsigned long int array[255];
        unsigned long int  mxx,x1,x2,max1,max2,min,threshold;
        int maxX,maxY;


        clrscr();
        printf("Enter your file name to histogram : ");
        gets(tex);


        fp = fopen(tex,"r+b");
        if(fp==NULL)
        {
                printf("I can't open your file.");
                getch();
                exit(0);
                fclose(fp);
        }


        bisize = 128*128;
        offset = 0;

        /*FIND ARRAY[i]*/
        pic = (unsigned char huge *)farcalloc(bisize,1);
        if(pic==NULL)
          {
            printf("I Can't reserve pic");
            fclose(fp);getch();
            exit(1);
           } ;



        for(j=0;j<256;j++)          /*CLEAR ARRAY[i]*/

        {   array[j]=00 ; } ;



        for(i=0;i<bisize;i++)
         {
            status = fread(&temp,sizeof(char),1,fp);
            if(status==NULL)
                {
                  printf(" Read file too pic error :");
                  getch();fclose(fp);
```

```
            exit(1);
          }
        *(pic+i) = (unsigned char)temp;

        array[temp]++;


      } ;

    mxx = 0;
    for(i=0;i<256;i++)
    {
       if(array[i] > mxx)
        {
          mxx = array[i]  ;
        }

    }
    for(i=0;i<256;i++)
    {
       array[i] = (array[i]*400)/mxx ;

    }


  /*GRAPHICS TO DISPLAY HISTOGRAM*/

    Initialize_Graphics_Mode();


       maxX=getmaxx() ; maxY=getmaxy();


    bar(0,0,maxX,maxY);
    setfillstyle(0,RED);
    setcolor 12);
    line(200,10,200,470);
    line(200,470,500,470);

    for(i=0;i<280;i++)
     {
        i+=10;
        line(200+i,470+3,200+i,470-3);

     } ;
     for(j=0;j<440;j++)
     {
        j+=10;
        line(200+3,470-j,200-3,470-j);
     };


     for(i=0;i<255;i++)
     {
       line(i+200,470-array[i]-19,i+200+1,470-array[i+1]-19);
     } ;    getch();



  fclose(fp);
  fclose(fp1);
  farfree(pic1);
  farfree(pic);
  closegraph();
```

```
}
void Initialize_Graphics_Mode(void)
{
   int gdriver = DETECT,gmode,errorcode;

   initgraph(&gdriver,&gmode,"c:\\tc\\graphic");
   errorcode = graphresult();

   if(errorcode != grOk)
   {
      printf("Graphics error : %s\n",grapherrormsg(errorcode));
      printf("Press any key to halt  ");
      getch();
      exit(1);
   }
}
```

```
#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <process.h>
#include <graphics.h>
#include <io.h>

typedef struct tagBITMAPFILEHEADER
{
        int bfType;
        long int bfSize;
        int bfReserved1;
        int bfReserved2;
        long int bfoffBits;
} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER
{
        long int biSize;
        long int biWidth;
        long int biHeight;
        int      biPlanes;
        int      biBitCount;
        long int biCompression;
        long int biSizeImage;
        long int biXPelsPerMeter;
        long int biYPelsPerMeter;
        long int ClrUsed;
} BITMAPINFOHEADER;

typedef struct tagRGBQUAD
{
        unsigned char rgbBlue;
        unsigned char rgbGreen;
        unsigned char rgbRed;
        unsigned char rgbReserved;
} RGBQUAD;

void mode(int mode_code);
void putpoint(int x,int y,int color);
void bwpalette(void);
void display(unsigned char huge *ptr,unsigned long int wide,
            unsigned long int highnew,int xp,int yp);

void main(void)
{
        BITMAPFILEHEADER *BmFile;
        BITMAPINFOHEADER *BminfoHeader;
        FILE *fp,*cp;
        char tex[10];
        unsigned long int offset,bisize,size,high,wide,highnew;
        unsigned char huge *pic;
        unsigned char temp1,temp2;
        unsigned int i,j,status1,status2;

        clrscr();
        printf("Enter your FILE-BMP name : ");
        gets(tex);

        fp = fopen(tex,"r+b");
        if(fp==NULL)
        {
                printf("I can't open your file 1.");
                getch();
```

```
        fclose(fp);
        exit(0);
}

cp = fopen("c:\\lek1\\nut.128","w+b");
if(cp==NULL)
{
        printf("I can't open your file 2.");
        getch();
        fclose(cp);
        exit(0);
}

BmFile = (BITMAPFILEHEADER *) calloc(1,sizeof(BITMAPFILEHEADER));
BmInfoHeader = (BITMAPINFOHEADER *) calloc(1,sizeof(BITMAPINFOHEADER));

fread(BmFile,sizeof(BITMAPFILEHEADER),1,fp);
fread(BmInfoHeader,sizeof(BITMAPINFOHEADER),1,fp);

bisize = BmFile->bfSize;
offset = BmFile->bfoffBits;
high = BmInfoHeader->biHeight;
wide = BmInfoHeader->biWidth;
size = BmInfoHeader->biSizeImage;
highnew = high;

printf("Bisize is      :%ld \n",bisize);

fseek(fp,offset,SEEK_SET);

printf("Pointer now  :%d\n",ftell(fp));
printf("Image size   :%ld \n",wide*high);
printf("High is      :%d \n",high);
printf("Wide is      :%d \n",wide);

printf("Use image size :%ld \n",highnew*wide);
printf("Newhigh is     :%d \n",highnew);
printf("Newwide is     :%d \n",wide);


pic = (unsigned char huge *)farcalloc(highnew*wide,1);
if(pic==NULL)
    {
     printf("I Can't reserve picture");
     farfree(pic);
     fclose(fp);
     fclose(cp);
     free(BmFile);
     free(BmInfoHeader);
     getch();
     exit(1);
    };



for(i=highnew;i>0;i--)
 for(j=wide;j>0;j--)
 {


       status1 = fread(&temp1,sizeof(char),1,fp);
       if(status1 == 0)
       {
        printf("I can read  %ld \n :",ftell(fp));
        printf("But I can't read continue this file.\n");
        fclose(fp);
```

```
                fclose(cp);
                farfree(pic);
                free(BmFile);
                free(BmInfoHeader);
                getch();
                exit(1);
              }

              *(pic+(i*wide)-j) = temp1;
          }

      for(i=0;i<highnew;i++)
       for(j=0;j<wide;j++)
       {
        if( (i>15) && (j>109) && (i<144) && (j<238) )
         {
            temp2 = *(pic+j+(i*wide));
            status2 = fwrite(&temp2,sizeof(unsigned char),1,cp);
            if(status2 != 1)
            {
                 printf("I can't write this file.\n");
                 fclose(fp);
                 fclose(cp);
                 farfree(pic);
                 free(BmFile);
                 free(BmInfoHeader);
                 getch();
                 exit(1);
            }
         }
       }

       farfree(pic);
       fclose(fp);
       fclose(cp);
       free BmFile);
       free(BmInfoHeader);
       getch();


       mode(0x13);
       display(pic,wide,highnew,0.0);
       getch();
       mode(0x3);


}

void mode(int mode_code)
{
     union REGS r;
     r.h.al = mode_code;
     r.h.ah = 0;
     int86(0x10,&r,&r);  /*INT 10H (VIDEO)*/
}
       /*SEND TO X,Y POSITION*/
void putpoint(int x,int y,int color)
{
     union REGS r;
     r.h.bh = 0;       /* PAGE 0 */
     r.h.ah = 12;      /* FUNCTION WRITE PIXEL */
     r.h.al = color;   /* PIXEL VALUE */
     r.x.dx = y;       /* ROW */
     r.x.cx = x;       /* COLUMN */
     int86(0x10,&r,&r);
}
```

```
      /*DEFINE COLOR(X,Y)*/
void bwpalette(void)
{
     union REGS r;
     int i;
     for(i=0;i<64;i++)
        {
        r.h.ah = 0x10;
        r.h.al = 0x10;
        r.x.bx = i;       /* DAC REG NUMBER */
        r.h.dh = i;       /* RED INTEN VALUE */
        r.h.ch = i;       /* GREEN INTEN VALUE */
        r.h.cl = i;       /* BLUE INTEN VALUE */
        int86(0x10,&r,&r);
        }
}

void display(unsigned char huge *ptr,unsigned long int wide,
            unsigned long int highnew,int xp,int yp)
{
    unsigned i,j,tmp;

    bwpalette();

    for(i=0;i<highnew;i++)
         for(j=0;j<wide;j++)
            {
            tmp =  (*(ptr+(i*wide+j)))/4;
            putpoint(j+xp,i+yp,tmp);
            }
            getch();

}
```