ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง SENSING MULTIPLEX BY MICROCOMPUTER

ผู้จัดทำ

1. นาย สมคิด โพธิ์ทอง 32-6224

2. นาย สุรสิทธ์ บรรลือศักดิ์ 32-6233

..... อาจารย์ที่ปรึกษา

(อาจารย์ วิริยะ กรองรัตน์)

ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง SENSING MULTIPLEX BY MICRO COMPUTER

ผู้จัดทำ

1. นาย สมคิด โพธิ์ทอง 32-6224
2. นาย สุรสิทธ์ บรรลือศักดิ์ 32-6233

..... อาจารย์ที่ปรึกษา

(อาจารย์ วิริยะ กรองรัตน์)

SENSING MULTIPLEX BY MICROCOMPUTER

BY

1. MR SOMKID POTHONG 32.6224
2. MR SURASIT BUNLUESAK 32.6233

ADVISOR MR. VIRIYA KONGRATANA

ABSTRACT

This thesis present the design and contruction of analog data aquistion system for measuring analog signal by microcomputer. And use microcomputer to application in receive signal from 16 external input. The interface card built for comfortabler save and accuracy in design HARDWARE.

Use the "C" language for develop SOFTWARE Which - the softwar has been being popular and it has the practical for the component supports. So the stability of HARDWARE and SORTWARE operating are high accuracy. The basic datails has proposed in this thesis.

SENSING MULTIPLEX BY MICRO COMPUTER

ผู้จัดทำ 1. นายสมคิด โพธ์ทอง
2. นายสุรสิทธ์ บรรลือศักดิ์

อาจารย์ที่ปรึกษา
อาจารย์ วิริยะ กองรัตน์

ปีการศึกษา 2533

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เขียนเกี่ยวกับรายละเอียดของการนำไมโครคอมพิวเตอร์เข้ามาประยุกต์ในรับการสัญญาณ อินพุทจากภายนอกถึง 16 จุดโดยผ่านการ์ดอินเตอร์เฟสที่ได้ สร้างขึ้น โดยเน้นและยึดถือเอาความสะดวก ความประหยัด และความแม่นยำเป็นแนวทางในการออกแบบทางด้าน HARD WARE อีกทั้งทางด้าน ซอฟแวร์ก็เลือกเอาภาษา C ซึ่งกำลังจะเป็นที่นิยมในอนาคตมาพัฒนารูปแบบทางด้านซอฟแวร์ ซึ่งเป็นซอฟแวร์ที่มีความคล่องตัวรวดเร็ว และมีสิ่งอำนวยความสะดวกแก่ผู้ใช้สูง ดังนั้นในเสถียรภาพของการทำงานของ HARD WARE และ SOFT WARE จึงนับได้ว่ามีความแม่นยำรวดเร็ว และคล่องตัวอย่างมาก ซึ่งรายละเอียดเบื้องต้นได้เสนอแนะไว้ในปริญญานิพนธ์นี้ด้วย

สารบัญ

บทนำ

ในสภาวะการณ์ในปัจจุบัน ไมโครคอมพิวเตอร์ได้เข้ามามีบทบาททั้งในชีวิตประจำวันและทางด้านอุตสาหกรรมอย่างมากมาก ด้วยความก้าวหน้าทางเทคโนโลยีต่าง ๆ ทางด้านไมโครคอมพิวเตอร์ ทำให้ไมโครคอมพิวเตอร์แทรกซึมเข้าไปในทุก ๆ วงการ อีกทั้งกลไกลทางด้านราคามีทิศทาง และแนวโน้มต่ำลงซึ่งสวนทางกับเทคโนโลยีที่เพิ่มขึ้นตลอดเวลา

ดังนั้นเราจึงควรนำไมโครคอมพิวเตอร์มาใช้งานให้เต็มประสิทธิภาพ ซึ่งนอกเหนือจากการประมวลผล และเก็บข้อมูล โดยการนำไมโครคอมพิวเตอร์มาใช้เป็นตัวควบคุมขบวนการต่าง ๆ เพื่อให้เกิดประโยชน์สูงสุด ซึ่งถือเป็นการพัฒนาตนเองอีกทั้งมีผลช่วยพัฒนาประเทศชาติให้มีความเจริญทางด้านเทคโนโลยีทั้งด้าน HARD WARE และ SOFT WARE ซึ่งเทคโนโลยีทางด้านนี้ยังมีการแพร่หลายน้อยมาก

ดังนั้นคณะผู้จัดทำจึงหวังว่าโครงการนี้คงจะมีประโยชน์สำหรับผู้ที่สนใจทางด้านการนำไมโครคอมพิวเตอร์มาประยุกต์ใช้ในการควบคุมอุปกรณ์ต่าง ๆ แล้วยังส่งผลให้ประเทศไทยเป็นมหาอำนาจทางด้านอิเล็กทรอนิกส์ของโลก

2.1 สัญญาณต่างๆบนสล๊อตของ IBM/PC

ภายใน IBM/PC ได้มีการออกแบบให้สามารถที่จะเพิ่มเติมวงจรอินเตอร์เฟสเข้าไปในภายหลังได้ โดยผ่านทางสล็อตที่อยู่บนเมนบอร์ด (MAIN BOARD) สำหรับสล็อตบนเมนบอร์ดนี้มีจำนวน5 สล็อต (สำหรับใน IBM PC/XT จะมี 8 สล็อต) ซึ่งแต่ละสล็อตจะมีจำนวนขาทั้งสิ้น 62 ขาแบ่งออกเป็น 2 ข้าง ๆละ 31 ขา ส่วนการเรียกตำแหน่งขาของสล็อตเหล่านี้จะขึ้นอยู่กับว่าขานั้นอยู่ข้างไหน(ซ้ายหรือขวา)ของสล็อตโดยขาที่อยู่ทางด้านซ้ายของสล็อตจะเรียกโดยใช้อักษร "B" นำหน้าเลขตำแหน่งของขา เช่นขา B16 ก็คือขาทางด้านซ้ายของสล็อตขาที่ 16 (นับจากทางท้ายของเครื่อง) ส่วนขาที่อยู่ทางด้านขวาของสล็อตจะเรียกโดยใช้อักษร "A" นำหน้าเลขตำแหน่งของขา เช่น ขาA24 ก็คือขาทางด้านขวาของสล็อตขาที่ 24 (นับจากด้านท้ายของเครื่อง) แต่ละขาของสล็อตเหล่านี้จะเชื่อมต่อกับเส้นสัญญาณต่างๆบนเมนบอร์ด ทำให้การสร้างวงจรอินเตอร์เฟสกับ IBM/PC สามารถทำได้โดยสะดวก ซึ่งเส้นสัญญาณที่เชื่อมต่อกับขาของสล็อตเหล่านี้จะประกอบไปด้วย เส้นสัญญาณของ บัสแอดเดรส (ADDRESS BUS), บัสข้อมูล (DATA BUS), บัสควบคุมสำหรับการเขียน/อ่านข้อมูลจากหน่วยความจำ หรือพอร์ท I/O, เส้นสัญญาณสำหรับการขออินเตอร์รัพท์ของวงจรอินเตอร์เฟส, เส้นสัญญาณสำหรับการขอ DMA, สัญญาณฐานเวลา (TIMEING SIGNAL) ต่างๆที่ใช้ในระบบ, เส้นสัญญาณแสดงการรีเฟรสหน่วยความจำและสัญญาณสำหรับตรวจสอบความผิดพลาด (I/O CHCK)

นอกจากเส้นสัญญาณเหล่านี้แล้ว สล็อตบนเมนบอร์ดยังเชื่อมต่อกับแหล่งจ่ายไฟต่างๆ ที่ใช้ในระบบอีกด้วย คือ +5Vdc, -5Vdc, +12Vdc, -12Vdc

รายละเอียดเกี่ยวกับสัญญาณต่างๆ

OSC (Oscillator; ขา B30) :

ขานี้เป็นเอาท์พุทที่เชื่อมต่อกับสัญญาณคล็อกที่มีค่าความถี่สูงสุดบนเมนบอร์ด คือ 14.31818-MH$_z$ ซึ่งมีคาบเวลาประมาณ 70 nanosec.และมี Duty Cycle (ช่วงเวลาใน 1 คาบที่สัญญาณคล็อกมีลอจิกเป็น "1" หารด้วยคาบเวลาทั้งหมด) ประมาณ 50 % สัญญาณคล็อกอื่นๆของระบบเช่น คล็อกที่ป้อนให้กับ 8088 หรือ ชิพซัพพอร์ทต่างๆนั้น จะถูกสร้างขึ้นโดยการหารสัญญาณคล็อกนี้ อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงในการใช้งานสัญญาณ OSC ก็คือ สัญญาณนี้จะไม่ Synchro-nize กับสัญญาณอื่น ๆบนบัสของระบบ ดังนั้นจึงไม่ควรที่จะนำสัญญาณจากขา OSC นี้ไปใช้เป็นสัญญาณคล็อกสำหรับวงจรภายนอกอื่น ๆที่ทำงานร่วมกับระบบ

CLK (clock; ขา B20) :

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งต่อกับสัญญาณคล็อกที่ถูกสร้างขึ้นโดยการหาร 3 ทำให้ได้ความถี่ประมาณ 4.77 MHz (14.31818 MHz/3) หรือมีช่วงเวลาใน 1 คาบ (ช่วงเวลาคาบ 1 ลูก)เท่ากับ 210 nanosec. (1/4.77 MHz) สำหรับค่า Duty - Cycle ของสัญญาณนี้จะมีค่าประมาณ 1/3 คือใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิก "1" เท่ากับ 1/3 ของคาบเวลาทั้งหมดหรือประมาณ 70 nanosec. และช่วงเวลาที่เป็นลอจิก "0" เท่ากับ 2/3 ของคาบเวลาทั้งหมด หรือประมาณ 140 nanosec. สัญญาณนี้ถูกใช้เป็นคล็อกของระบบ

RESET DRV (ขา B2) :

ขาสัญญาณนี้เป็นเอาท์พุท ซึ่งจะแอคทีฟ (ลอจิก "1") ในช่วงที่เราเริ่มจ่ายไฟให้กับระบบและจะยังคงแอคทีฟไปจนกว่าระบบต่าง ๆภายใน IBM/PC จะพร้อมที่จะทำงานได้จากนั้นสัญญาณนี้ก็จะเปลี่ยนกลับเป็นลอจิก "0" นอกจากนี้ในระหว่างการทำงานของ IBM/PC ถ้าระดับแรงดันของแหล่งจ่ายไฟตรง สัญญาณนี้ก็จะถูกทำให้แอคทีฟเช่นกัน โดยทั่วไปแล้วสัญญาณนี้จะถูกนำไปใช้ในการรีเซ็ทวงจรอินเทอร์เฟส หรืออุปกรณ์เหล่านั้นถูกปรับให้อยู่ในสภาวะที่แน่นอน ก่อนที่จะเริ่มต้นการทำงานในระบบ (สภาวะนี้เป็นสภาวะที่เราทราบ และต้องการให้วงจรทำงานในขณะที่ระบบถูกรีเซ็ท)

A0 - A9 (Address Bus; ขา A31 - A12) :

ขาสัญญาณทั้ง 20 ขานี้เป็นเอาท์พุท ซึ่งใช้สำหรับกำหนดแอดเดรสของหน่วยความจำ หรืออุปกรณ์ I/O ที่ 8088 ต้องการติดต่อด้วย โดยที่สัญญาณ A0 จะมีนัยสำคัญต่ำสุด (Least Significant Bit) และ A19 จะมีนัยสำคัญสูงสุด (Most Significant Bit) สำหรับค่าแอดเดรสบนบัสแอดเดรส A0-A19 นี้ จะถูกกำหนดโดย 8088 ในระหว่างขบวนการอ่าน/เขียนข้อมูลลงในหน่วยความจำหรืออุปกรณ์ I/O แต่ในช่วงของขบวนการ DMA นั้น DMA-Controller จะเป็นผู้กำหนดค่าแอดเดรสบนบัสแอดเดรสเอง (ในระหว่างนี้ 8088 จะถูกตัดออกจากระบบ)

จะเห็นได้ว่าจำนวนเส้นแอดเดรสนี้จะมีอยู่ 20 เส้น ซึ่งสามารถที่จะอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte แต่อย่างไรก็ตามจะมีแอดเดรสบางแอด - เดรสที่ถูกใช้งานโดย IBM/PC อยู่ก่อนแล้ว คือแอดเดรสของหน่วยความจำ RAM บนเมนบอร์ดที่ถูกใช้โดยระบบ จำนวน 64 Kbyte (สำหรับ IBM PC/XT จะเป็นจำนวน 256 Kbyte) และแอดเดรสสำหรับหน่วยความจำ ROM อีก 48 Kbyte ซึ่งถูกจัดใน

ช่วงของแอดเดรสบนสุดใน 1 Mbyte คือ OFCOOH จนถึง OFFFFFH (สำหรับ IBM PC/XT จะเป็น 64 Kbyte)

สำหรับการอ้างแอดเดรสของพอร์ท I/O นั้น จะใช้เส้นแอดเดรสเพียง 16 เส้นคือ AO-15 ซึ่งจะทำให้อ้างแอดเดรสของพอร์ทได้ 64K พอร์ท โดยผ่านทางชุดคำสั่ง IN และ OUT ส่วนเส้นแอดเดรสที่เหลือคือ A16-A19 นั้นจะไม่ถูกใช้งาน อย่างไรก็ตาม ภายใน IBM/PC จะใช้เส้นแอดเดรสในการอ้างแอดเดรสของพอร์ทเพียง 10 เส้น คือ จาก AO-A9 และค่าแอดเดรสที่ใช้งานจะต้องอยู่ในช่วง 0200H จนถึง 03FFH - เท่านั้น(ดูรายละเอียดในบทที่ 9 "การจัดแอดเดรสสำหรับหน่วยความจำและพอร์ท I/O")

DO-D7 (Data Bus; ขา A9-A2) :

ขาสัญญาณนี้จะเป็นแบบ Bi-Directional ซึ่งต่อกับบัสข้อมูลของระบบเพื่อทำหน้าที่ในการส่งผ่านข้อมูลระหว่างพอร์ท I/O กับ IBM/PC โดยบิท DO จะมีนัยสำคัญต่ำสุดและบิท D7 จะมีนัยสำคัญสูงสุด

สำหรับในบัสไซเคิลของการเขียนข้อมูลที่สร้างขึ้นโดย 8088 นั้น ข้อมูลจะถูกส่งออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOW (ในกรณีที่ต้องการส่งข้อมูลให้กับพอร์ท) หรือ MEMW (ในกรณีที่ต้องการส่งข้อมูลให้กับหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น) ซึ่งโดยทั่วไปขอบขาขึ้นของสัญญาณ IOW หรือ MEMW นี้จะถูกใช้เพื่อสั่งให้พอร์ท I/O หรือหน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสนั้นรับข้อมูลไปเก็บไว้

สำหรับในบัสไซเคิลของการอ่านข้อมูลที่สร้างขึ้นโดย 8088 นั้น พอร์ท I/O หรือหน่วยความจำที่ถูกอ้างถึงจะต้องส่งข้อมูลออกมาบนบัสข้อมูล ก่อนที่สัญญาณ IOR - (ในกรณีที่ต้องการอ่านข้อมูลจากพอร์ท) หรือ MEMR (ในกรณีที่ต้องการอ่านข้อมูลจากหน่วยความจำ) จะเปลี่ยนจากลอจิก "0" เป็นลอจิก "1" (ขอบขาขึ้น)

ALE (Address Latch Enable; ขา B28) :

ขาสัญญาณนี้เป็นสัญญาณเอาท์พุทที่ 8288 Bus Controller สร้างขึ้นเพื่อใช้สำหรับแสดงการเริ่มต้นของบัสไซเคิล และแสดงให้อุปกรณ์ภายนอกทราบว่าแอดเดรสที่ 8088 ต้องการจะติดต่อด้วยนั้นถูกส่งออกมาบนบัสแอดเดรสแล้ว โดยที่สัญญาณ ALE นี้จะเปลี่ยนจากลอจิก "1" เป็นลอจิก "0" เมื่อค่าแอดเดรสที่ถูกต้องถูกส่งออกมาบนบัสข้อมูลเรียบร้อยแล้ว ดังนั้นขอบขาลงของสัญญาณ ALE นี้จะถูกใช้ในการแลทช์ค่าแอดเดรสจากบัสแอดเดรส/ข้อมูล (Adress/Data Bus; AD0-AD7) ของ 8088 ทำให้

สามารถแยกค่าแอดเดรส (A0-A19) และข้อมูล (A0-A7) ออกจากกันได้ อย่างไรก็ตาม สัญญาณ ALE จะแอคทีฟเฉพาะในบัสไซเคิลที่สร้างขึ้นโดย 8088 เท่านั้น โดยจะไม่แอคทีฟในระหว่างขบวนการ DMA

I/O CHCK (I/O Channel Check; ขา A1) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้ในการแสดงความผิดพลาดเกี่ยวกับพาริตี้ ที่เกิดขึ้นในการทำงานของวงจรอินเทอร์เฟสหรืออุปกรณ์ I/O เมื่อขาสัญญาณนี้ได้รับลอจิก "0" จะทำให้ 8088 ถูกอินเทอร์รัพท์แบบ Non-Maskable (NMI) อย่างไรก็ตามเราสามารถที่จะกำหนดให้วงจรภายในของ IBM/PC ทำการขออินเทอร์รัพท์ (เมื่อได้รับสัญญาณ I/O CHCK) หรือไม่ก็ได้โดยการกำหนดลอจิกของบิทข้อมูลของพอร์ทที่ควบคุมการขออินเทอร์รัพท์แบบ NMI คือบิท D7 ของพอร์ท 00A0H ในกรณีที่บิท D7 ของพอร์ท 00A0H ถูกเซ็ทเป็น "1" ก็จะทำให้วงจรภายนอกขออินเทอร์รัพท์แบบ NMIได้ (Enable) แต่ถ้าบิท D7 ของพอร์ท 00A0H ถูกเซ็ทเป็น "0" ก็จะเป็นการดิสเอเบิล (Disable) การขออินเทอร์รัพท์แบบ NMI ดังนี้

Enable : ใช้คำสั่ง OUT ส่งข้อมูล 80H ไปยังพอร์ท 00A0H

Disable : ใช้คำสั่ง OUT ส่งข้อมูล 00H ไปยังพอร์ท 00A0H

และเนื่องจากยังมีอุปกรณ์อื่นที่สามารถขออินเทอร์รัพท์แบบ NMI ได้อีก ดังนั้นซอฟท์แวร์ที่ใช้งานจะต้องสามารถตรวจสอบว่าการขออินเทอร์รัพท์

I/O CHRDY (I/O Channel Ready; ขา A10) :

ขาสัญญาณนี้เป็นอินพุทที่ใช้เพิ่มช่วงเวลาในบัสไซเคิลในกรณีที่อุปกรณ์ I/O หรือหน่วยความจำที่เกี่ยวข้องกับขบวนการในบัสไซเคิลที่เกิดขึ้นนั้น ไม่สามารถทำงานทันตามช่วงเวลาปกติของบัสไซบัสไซเคิลนั้น ๆ ได้ (ช่วงเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูกหรือ 840 nanosec. ในขณะที่บัไซเคิลที่เกี่ยวกับ I/O จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 5 ลูกหรือ 1.05 usec.)

เมื่ออุปกรณ์ I/O หรือหน่วยความจำต้องการที่จะเพิ่มช่วงเวลาในบัสไซเคิลให้นานขึ้นอีกนั้นจะสามารถทำได้โดยการป้อนลอจิก "0" ให้กับขา I/O CHRDY ในช่วงเวลาที่ I/O หรือ หน่วยความจำที่ถูกกำหนดนั้น ได้รับสัญญาณจากการดีโค้ดแอดเดรสและสัญญาณ MEMR, MEMW, IOR หรือ IOW แอคทีฟ

IRQ2-IRQ7 (Interrupt Request 2 Through 7; ขา B4 และ B25-B21) :

ขาสัญญาณทั้ง 6 นี้เป็นขาอินพุทที่ใช้สำหรับการขออินเทอร์รัพท์จาก 8088 โดย

สัญญาณเหล่านี้จะต่อเข้ากับ 8259A บนเมนบอร์ดโดยตรงโปรแกรมในส่วน BIOS ของ IBM/PC จะทำการโปรแกรม 8259A ให้ IRQ2 มีลำดับความสำคัญสูงสุด (Highest Priority) และ IRQ7 มีลำดับความสำคัญต่ำสุด ในกรณีที่มีการขออินเทอร์รัพท์เกิดขึ้น คือระดับลอจิกที่ขา IRQ ขาใดขาหนึ่งถูกเปลี่ยนจากลอจิก "0"เป็นลอจิก "1"(ขอบขาขึ้น) 8259A ก็จะทำการส่งสัญญาณ INT ให้กับ 8088 เพื่อทำการขออินเทอร์รัพท์สิ่งที่สำคัญในการขออินเทอร์รัพท์โดยผ่านทาง IRQ2-IRQ7 นี้ก็คืออุปกรณ์ที่ทำการขออินเทอร์รัพท์โดยผ่านทาง IRQ ขาใดก็จะต้องรักษาระดับสัญญาณที่ขา IRQ นั้น ให้แอคทีฟ (ลอจิก "1") อยู่จนกว่าจะได้รับสัญญาณ INTA (Interrupt Acknowledge) จาก 8088 เสียก่อน ถ้าไม่เช่นนั้นการขออินเทอร์รัพท์จะถูกยกเลิก และอินเทอร์รัพท์ Level 7 (IRQ7) ก็จะถูกสร้างขึ้นโดยอัตโนมัติ ไม่ว่าการขออินเทอร์รัพท์ที่ถูกยกเลิกนั้น จะเป็นการขออินเทอร์รัพท์ใน Level หรือขาใด

แต่อย่างไรก็ตามสัญญาณ INTA นี้จะไม่ถูกต่อออกมาที่ขาของสล็อตด้วย ดังนั้นโปรแกรมที่ทำการตอบสนองต่อการขออินเทอร์รัพท์ (Interrupt Service Routine) จะต้องทำการริเซ็ทสัญญาณ IRC เอง โดยใช้คำสั่ง OUT ไปยังพอร์ท I/O ที่เกี่ยวข้อง

IOR (I/O Read; ขา B14) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ที่สร้างขึ้นโดย 8288 Bus-Controller เพื่อใช้ในการแสดงว่าบัสไซเคิลที่เกิดขึ้น เป็นบัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล โดยข้อมูลจะต้องถูกส่งออกมาบนบัสข้อมูลก่อนขอบขาขึ้นของสัญญาณ IOR ประมาณ 30 nanosec. เพื่อให้มั่นใจได้ว่า 8088 สามารถรับข้อมูลได้ถูกต้อง สำหรับในขบวนการ DMA 8237A-5 DMA Controller จะทำการสร้างสัญญาณ IOR เอง โดยที่ค่าแอดเดรสที่อยู่บนแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำ -(แทนที่จะเป็นแอดเดรสของพอร์ท I/O) ที่พอร์ท I/O ที่ขอ DMA ต้องการจะนำข้อมูลไปเก็บ การที่พอร์ทใดจะส่งข้อมูลออกมาบนบัสข้อมูลนั้น จะอาศัยสัญญาณ DACK จาก-DMA Controller เป็นตัวกำหนด เช่นกรณีที่สัญญาณ DACK1 แอคทีฟก็จะแสดงว่าพอร์ท I/O ที่จะต้องส่งข้อมูลออกมาบนบัสก็คือพอร์ท I/O ที่ขอ DMA ผ่านทางแชนแนลที่ 1 (DRQ1) เป็นต้น

IOW (I/O Write; ขา B13) :

ขาสัญญาณนี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่งถูกสร้างขึ้นโดย 8288 Bus-

Controller เพื่อใช้แสดงว่าบัสไซเคิลที่เกิดขึ้นนี้เป็นบัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O เพื่อให้พอร์ท I/O ที่มีแอดเดรสตรงกับแอดเดรสบัสแอดเดรสนั้นรับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ อย่างไรก็ตามเนื่องจากในช่วงเวลาที่สัญญาณ IOW นี้แอคทีฟ (ลอจิก "0") นั้นข้อมูลบนบัสข้อมูลอาจจะยังไม่สมบูรณ์ ดังนั้นในการออกแบบจึงควรใช้ขอบขาขึ้นของสัญญาณ IOW แทนขอบขาลงในการทำให้พอร์ท I/O ที่เกี่ยวข้องรับข้อมูลไปเก็บไว้ เพื่อให้ข้อมูลบนบัสข้อมูลสมบูรณ์เสียก่อน สำหรับในขบวนการ DMA นั้น DMA-Controller จะทำการสร้างสัญญาณ IOWเองโดยที่ค่าแอดเดรสที่อยู่บนบัสแอดเดรสจะเป็นค่าแอดเดรสของหน่วยความจำที่พอร์ท I/O ที่ขอ DMA ต้องการจะอ่านข้อมูล

MEMW (Memory Write; ขา B11) :

ขานี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่ง 8288 Bus Controller สร้างขึ้นในระหว่างบัสไซเคิลในการเขียนข้อมูลลงในหน่วยความจำของ 8088 สัญญาณ MEMW นี้จะถูกส่งออกมาเพื่อให้หน่วยความจำที่แอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการรับข้อมูลที่อยู่บนบัสข้อมูลไปเก็บไว้ โดยทั่วไปหน่วยความจำจะรับข้อมูลในช่วงขอบขาขึ้นของสัญญาณ MEMW

สำหรับในระหว่างขบวนการ DMA นั้น 8237A-5 DMA-Controller จะทำการควบคุมบัสต่าง ๆของระบบแทน 8088 และสัญญาณ MEMW จะถูกใช้ในบัสไซเคิลของการเขียนข้อมูลลงในหน่วยความจำ(ข้อมูลถูกส่งจากอุปกรณ์I/Oไปให้กับหน่วยความจำ)

MEMR (Memory Read; ขา B12) :

ขานี้เป็นเอาท์พุทจาก 8288 ซึ่งสัญญาณนี้จะแอคทีฟ (ลอจิก "0") ในระหว่างบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำของ8088 เพื่อให้หน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรสนั้น ทำการส่งข้อมูลออกมาบนบัสข้อมูล โดยหน่วยความจำนั้นจะต้องส่งข้อมูลออกมาในช่วงเวลา 30 nanosec. ก่อนที่สัญญาณ MEMW จะกลับเป็นลอจิก "1" ทั้งนี้ก็เพื่อให้ 8088 ได้รับข้อมูลที่ถูกต้อง

สำหรับในระหว่างขบวนการ DMA นั้น DMA-Controller จะควบคุมบัสต่างๆของระบบแทน 8088 และสัญญาณ MEMR จะถูกใช้ในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำ (ข้อมูลถูกส่งจากหน่วยความจำไปให้กับอุปกรณ์ I/O)

DRQ1-DRQ3 (DMA Request 1-3; ขา B18, B6 และ ขา B16) :

ขาสัญญาณทั้งสามนี้เป็นสัญญาณอินพุทแอคทีฟที่ลอจิก "1" ซึ่งอุปกรณ์ภายนอก

สามารถใช้ในการขอ DMA จากระบบ โดยการป้อนระดับสัญญาณลอจิก "1" ให้กับขา DRQ ขาใดขาหนึ่ง (ขา DRQ ทั้งสามนี้จะต่อเข้ากับ DRQ1-DRQ3 ของ 8237A-5)

เมื่อ 8237A-5 ได้รับสัญญาณนี้แล้วก็จะตรวจสอบว่ามีการขอ DMA ในแชนแนลที่มีลำดับความสำคัญ (Priority) สูงกว่าหรือไม่ ถ้าไม่มีก็จะทำการขอ DMA จาก 8088 และตอบรับการขอ DMA จากอุปกรณ์ภายนอก (สัญญาณ DACK ของแชนแนลที่ขอ DMA จะแอคทีฟ) แต่ถ้ามี 8237A-5 ก็จะทำการขอ DMA ให้กับแชนแนลที่มีลำดับความสำคัญสูงกว่าก่อนแล้วจึงทำการขอ DMA ให้กับแชนแนลที่มีลำดับความสำคัญต่ำกว่า ภายใน ROM-BIOS ของ IBM/PC จะโปรแกรม 8237A-5 ให้ DRQ1 มีลำดับความสำคัญสูงสุดและ DRQ3 มีลำดับความสำคัญต่ำสุด ดังนั้นถ้ามีการขอ DMA ของอุปกรณ์ภายนอกผ่านทางแชนแนลที่ 1 ก่อน จากนั้นเมื่อเสร็จจากขบวนการ DMA ของแชนแนลที่ 1 แล้วจึงจะทำการขอ DMA ให้กับแชนแนลที่ 2

อย่างไรก็ตาม 8237A-5 ยังมีแชนแนลสำหรับการขอ DMA อยู่อีก 1 แชนแนล คือแชนแนลที่ 0 (DRQ0) ซึ่งในความเป็นจริงแล้วแชนแนลนี้จะมีลำดับความสำคัญที่สูงกว่าแชนแนลที่ 1 แต่จะไม่ถูกต่อออกมายังขาของสล็อต เนื่องจาก IBM/PC จะใช้แชนแนลที่ 0 นี้ในการรีเฟรชหน่วยความจำที่เป็น Dynamic RAM

ในการขอ DMA นั้นสัญญาณ DRQ นี้จะต้องแอคทีฟอยู่ในช่วงระยะเวลาหนึ่งเท่านั้น ถ้าสัญญาณแอคทีฟอยู่นานเกินไปจะทำให้เกิดขบวนการ DMA ขึ้นมากกว่า 1 ขบวนการได้ สำหรับวงจรที่ขอ DMA โดยทั่วไปแล้วจะใช้สัญญาณตอบรับการขอ DMAหรือสัญญาณ DACK ของแชนแนลที่ขอ DMA นั้น ในการรีเซ็ทสัญญาณ DRQ เช่นอุปกรณ์ภายนอกที่ขอ DMA ผ่านทางแชนแนลที่ 1(DRQ1) ก็จะคอยตรวจสอบการตอบรับการขอ DMA จากสัญญาณ DACK ของแชนแนลที่ 1 (DACK1) เมื่อได้รับสัญญาณจาก DACK1 แล้วก็จะรีเซ็ทสัญญาณ DRQ1 (เปลี่ยนจากลอจิก "1" เป็น "0")

DACK0-DACK3 (DMA Acknowledge 0-3; ขา B19, B17, B26 และ B15) :

สัญญาณทั้ง 4 นี้เป็นเอาท์พุทแอคทีฟที่ลอจิก "0" ซึ่ง 8237A-5 สร้างขึ้นเพื่อแสดงให้วงจรภายนอกที่ขอ DMA ทราบว่าการขอ DMA นั้นได้รับการตอบสนองแล้ว และ 8237A-5 จะเข้าสู่ขบวนการ DMA เพื่อให้การส่งผ่านข้อมูลระหว่างอุปกรณ์ I/O ที่ขอ DMA กับหน่วยความจำเกิดขึ้นได้โดยตรง (คือไม่ต้องผ่าน 8088) โดยสัญญาณ DACK นี้จะแอคทีฟในแชนแนลใดก็ขึ้นอยู่กับว่าขบวนการ DMA ที่จะเกิดขึ้นนั้นเป็นการตอบสนองต่อการขอ DMA ในแชนแนลใด เช่นถ้าขบวนการ DMA ที่จะเกิดขึ้นนั้นเป็นการตอบสนองต่อการขอ DMA ในแชนแนลที่ 2 (DRQ2) สัญญาณ DACK2 ก็จะแอคทีฟ เป็นต้น

ดังที่ได้กล่าวแล้วว่าสัญญาณ DRQO นั้น จะไม่ถูกต่อออกมายังขาของสล็อต ดังนั้น วงจรอินเทอร์เฟสจึงไม่สามารถจะขอ DMA ผ่านทางแชนแนล 0 ได้แต่สัญญาณ DACKO จะถูกต่อออกมายังสล็อตด้วย (ขา B19), ทั้งนี้ก็เพื่อที่จะแสดงให้วงจรอินเทอร์เฟสต่าง ๆ ทราบว่าขบวนการ DMA ที่เกิดขึ้นในช่วงเวลาที่ DACKO แอคทีฟนั้น เป็นขบวนการที่ใช้สำหรับการรีเฟรชหน่วยความจำที่เป็น Dynamic RAM ซึ่งวงจรอินเทอร์เฟสที่ใช้หน่วยความจำประเภทนี้สามารถจะนำไปใช้ในการรีเฟรช Dynamic RMA ที่อยู่ในวงจรได้ โดยที่การรีเฟรชหน่วยความจำนั้นจะเกิดขึ้นในทุก ๆ 15.12 usec. หรือ ทุก ๆ 72 คล็อกดังนั้นสัญญาณ DACKO นี้ก็จะแอคทีฟในทุกๆ 15.12 usce. ด้วย

AEN (Address Enable; ขา A11)

สัญญาณนี้เป็นเอาท์พุทที่ใช้ในการแสดงว่า บัสไซเคิลที่เกิดขึ้นในช่วงเวลาที่สัญญาณ AEN แอคทีฟ (ลอจิก "1") นั้น เป็นบัสไซเคิลของขบวนการ DMA

สำหรับบนเมนบอร์ดของ IBM/PC นั้น จะใช้สัญญาณนี้ในการดิสเอเบิล - (Disable) 8288 Bus Controller และจะใช้ดิสเอเบิลพอร์ท I/O ต่าง ๆ ที่ไม่เกี่ยวข้องกับขบวนการ DMA ที่เกิดขึ้น ที่จำเป็นต้องทำเช่นนี้ก็เพราะในระหว่างขบวนการ DMA นั้น 8237A-5 จะส่งแอดเดรสของหน่วยความจำออกมาบนบัสแอดเดรส และจะทำให้สัญญาณ IOR หรือ IOW แอคทีฟด้วย ดังนั้นถ้าไม่ทำการดิสเอเบิลพอร์ท I/O ที่ไม่เกี่ยวข้องไว้ ก็อาจจะทำให้พอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสบนบัสแอดเดรส (ซึ่งเป็นแอดเดรสของหน่วยความจำ) นั้น ทำการอ่าน หรือส่งข้อมูลออกมาบนบัสข้อมูลทำให้เกิดความผิดพลาดขึ้นได้

T/C (Terminal Count; ขา B27) ;

สัญญาณนี้ถูกสร้างขึ้นจากการนำเอาสัญญาณเอาท์พุทที่ขา EOP ของ 8273A-5 มากลับลอจิก (โดยการใช้เกท Inverter) ทำให้สัญญาณ T/C นี้แอคทีฟที่ลอจิก "1"

สำหรับสัญญาณนี้จะแอคทีฟเมื่อจำนวนไบท์ในการส่งผ่านข้อมูลของขบวนการDMA ในแชนแนลหนึ่ง ครบตามจำนวนที่กำหนดไว้ โดยทั่วไปแล้วสัญญาณที่จะถูกใช้ในการสิ้นสุดขบวนการ DMA ที่ทำการส่งผ่านข้อมูลเป็นบล็อก เนื่องจากสัญญาณนี้จะแอคทีฟโดยไม่แสดงว่าเป็นสัญญาณของแชนแนลใด ดังนั้นจึงต้องทำการนำสัญญาณ T/C นี้ผ่านเกท Inverter แล้วนำไป OR กับสัญญาณ DACK เพื่อให้สามารถทราบได้ว่าสัญญาณ T/C ที่เกิดขึ้นนั้นเป็นสัญญาณของแชนแนลใด สำหรับในแชนแนลที่ 0 นั้นสัญญาณ T/C จะแอคทีฟในช่วงเวลาที่คงที่คือ ทุก ๆ 990.804 millisec. ซึ่งก็คือช่วงเวลาที่ใช้ในการรีเฟรชหน่วยความจำ

ขนาด 64 Kbyte นั่นเอง

บัสของแหล่งจ่ายไฟของระบบ

+5Vdc (ขา B3 และ B29) :

ขาทั้งสองนี้ต่อกับแหล่งจ่ายไฟ DC+5V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 5 % คืออยู่ในช่วง +4.75 ถึง +5.25 Vdc

+12Vdc (ขา B9) :

ขานี้จะต่อกับแหล่งจ่ายไฟ DC+12V ของระบบโดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 5 % คืออยู่ในช่วง+11.4 ถึง+12.6 Vdc

-5Vdc (ขา B5) :

ขานี้จะต่อกับแหล่งจ่ายไฟ DC -5V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 10 % คืออยู่ในช่วง -5.5 ถึง -4.5 Vdc

-12Vdc (ขา B7) :

ขานี้จะต่อกับแหล่งจ่ายไฟ DC -12V ของระบบ โดยจะมีค่าความเที่ยงตรง (Regulated) $\pm$ 10 % คืออยู่ในช่วง -13.2 ถึง -10.8 Vdc

GND (ขา B1, B10 และ B31) :

ขาทั้งสามนี้จะต่อเข้ากับกราวด์ (Ground) ของระบบ

การจัดสัญญาณบนสล็อตของ IBM PC/XT

สำหรับใน IBM PC/XT นั้นจะสล็อตสำหรับเชื่อมต่อกับวงจรภายนอกได้มากขึ้น คือใน IBM PC/XT จะทำการเพิ่มจำนวนสล็อตบนเมนบอร์ดขึ้นเป็น 8 สล็อต จากเดิมที่มีอยู่เพียง 5 สล็อตบน IBM PC โดยการจัดสัญญาณต่าง ๆในทั้ง 8 สล็อตจะยังคงเหมือนกับใน IBM PC เพียงแต่สัญญาณต่าง ๆที่จะถูกส่งออกมายังขาของสล็อตที่ 8 นั้นจะถูกต่อผ่านวงจรขับกระแส (Buffer) ก่อนและในสล็อตที่ 8 นี้ขา B8 จะถูกใช้งานด้วย โดยจะถูกใช้เป็นขา CARD SLCTD (หรือ CardSelected) ซึ่งขาสัญญาณอินพุทจากวงจรภายนอกที่เสียบอยู่บนสล็อตที่ 8เพื่อให้วงจรบนเมนบอร์ดทราบว่าคาร์ดที่อยู่บนสล็อตนี้ถูกเลือกใช้งานอยู่ ซึ่งจะทำให้ Driver บนเมนบอร์ดทำการอ่านหรือส่งข้อมูลไปยังสล็อตที่ 8

การจัดแอดเดรสสำหรับหน่วยความจำและ I/O

ในบทนี้จะกล่าวถึง การจัดแอดเดรสสำหรับหน่วยความจำและพอร์ท I/O ต่าง ๆ ภายใน IBM/PC ซึ่งจะแสดงถึงแอดเดรสต่าง ๆ ที่ถูกใช้งานโดยพอร์ท I/O (Input/Output Port) และหน่วยความจำ นอกจากนี้จะได้กล่าวถึงเทคนิคการดีโค้ด (Decode) แอดเดรสในแบบต่างๆ ด้วย

การจัดแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

ในหัวข้อนี้จะกล่าวถึงวิธีการอ้างและใช้งานแอดเดรสต่าง ๆ ของพอร์ท I/O ที่ใช้งานอยู่ใน IBM/PC

2.2 การอ้างแอดเดรสของพอร์ท I/O

ในการควบคุมและตรวจสอบสภาวะการทำงาน รวมทั้งการอ่านข้อมูลจากอุปกรณ์ที่เป็นชิพชัพพอร์ทหรือคาร์ดต่าง ๆ ที่ใช้ในระบบของ IBM/PC นั้น จะกระทำโดยผ่านทางพอร์ท I/O ของระบบ ดังนั้นในการที่จะใช้งานหรือควบคุมการทำงานของอุปกณ์เหล่านี้ จึงจำเป็นต้องศึกษาถึงวิธีการควบคุมพอร์ท I/O ต่าง ๆ ของระบบด้วย และเนื่องจากการควบคุม หรือติดต่อกับพอร์ทเหล่านี้ต้องกระทำโดยการอ้างถึงแอดเดรสของพอร์ท I/O เหล่านั้นโดยตรง เราจึงจำเป็นต้องศึกษาถึงหลักการอ้างแอดเดรสของ 8088 ใน - IBM/PC ด้วย

สำหรับแอดเดรสของพอร์ท I/O ต่าง ๆ นั้นจะเป็นแอดเดรสที่ถูกสร้างขึ้นโดย 8088 ซึ่งแอดเดรสเหล่านี้เป็นแอดเดรสที่จัดไว้สำหรับพอร์ท I/O โดยเฉพาะ คือแยกจากแอดเดรสของหน่วยความจำโดยเด็ดขาด ส่วนการส่งข้อมูลให้กับพอร์ทเหล่านี้จะทำได้โดยการใช้คำสั่ง OUT ของ 8088 ส่งข้อมูลนั้นไปยังแอดเดรสของพอร์ทที่ต้องการ และสำหรับการตรวจสอบหรือการอ่านข้อมูลจากพอร์ทก็จะทำได้โดยการใช้คำสั่ง IN ของ8088 อ่านข้อมูลจากแอดเดรสของพอร์ทที่ต้องการเช่นกัน

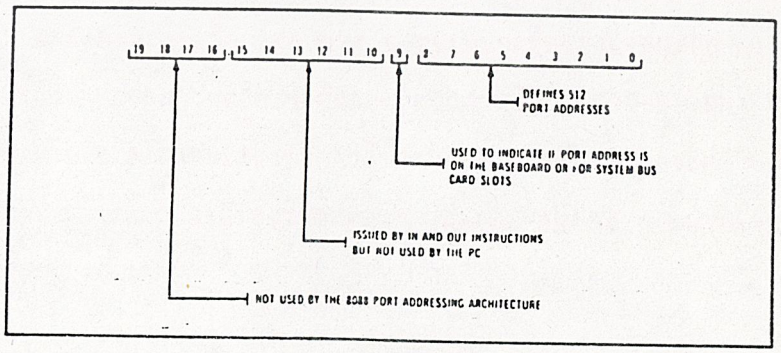
ภายในไมโครโปรเซสเซอร์เบอร์ 8088 นี้จะมีแอดเดรสสำหรับใช้กับพอร์ท I/O อยู่ทั้งสิ้น 65,536 หรือ 64K แอดเดรส(ในขณะที่มีแอดเดรสสำหรับหน่วยความจำอยู่ 1 Mbyte) ซึ่งทำให้การอ้างแอดเดรสของพอร์ท I/O ที่ทำงานร่วมกับ 8088 นั้น ต้องใช้จำนวนเส้นแอดเดรสในบัสแอดเดรสทั้งสิ้น 16 เส้นคือ A0-A15 แต่สำหรับใน IBM/PC นี้ถูกออกแบบมาให้ใช้เส้นแอดเดรสเฉพาะ 10 เส้นล่าง คือ A0-A9 เท่านั้น ดังนั้นในการอ้างถึงแอดเดรสของพอร์ทของอุปกรณ์หรือชิพชัพพอร์ทใด ๆ ที่ใช้ร่วมกับ IBM/PC จึงใช้จำนวนเส้นแอดเดรสเพียง 10 เส้นด้วยโดยเส้นแอดเดรสที่เหลือ คือ A10-A15 นั้นจะไม่ถูกนำไปใช้งาน อย่างไรก็ตามถึงแม้ว่าเส้นแอดเดรส A10-A15 นี้จะไม่ถูกนำไปใช้งาน แต่ค่าแอดเดรสบนเส้นแอดเดรสเหล่านี้ ยังคงเปลี่ยนแปลงตามค่าแอดแดรสของพอร์ทที่กำหนดไว้ในคำสั่ง OUT หรือ IN อยู่ด้วยเพียงแต่ไม่ได้ถูกนำมาดีโค้ดร่วมกับแอดเดรส A0-A9 เท่านั้น ตัวอย่าง เช่นในการใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0010H นั้นจะให้ผลเหมือนกับการส่งข้อมูลไปยังพอร์ทที่ตรงกับแอดเดรส 0410H, 0810, 0C10H, ทั้งนี้เนื่องจากแอดเดรส 6 บิทบนไม่ได้ถูกใช้งานจึงทำให้การเปลี่ยนแปลงค่าแอดเดรสบนเส้นแอดเดรส A0-A15 นั้น ไม่ทำให้เกิดความแตกต่างใด ๆ ขึ้นเนื่องจากใน IBM/PC ได้ใช้งานเส้นแอดเดรสเพียง 10 เส้น (คือ A0-A9) ดังนั้นจึงสามารถที่จะอ้างแอดเดรสของพอร์ทได้สูงสุดเพียง 1024 พอร์ท(จากจำนวน 64K พอร์ท) เท่านั้นนอกจาก

นี้ในกรณีที่เป็นการอ่านข้อมูลจากพอร์ทของ IBM/PC ข้อมูลในบิท A9 จะถูกจัดให้มีหน้าที่ในการแบ่งพอร์ททั้ง 1024 พอร์ทออกเป็น 2 ส่วน (ส่วนละ 512 พอร์ท) อีกด้วย กล่าวคือ ถ้าข้อมูลในบิท A9 เป็น "0" แล้ว เราจะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทของอุปกรณ์ หรือชิพชัพพอร์ทต่าง ๆ ที่อยู่บนเมนบอร์ด (Main Board) ของ IBM/PC เช่น 8253-5, 8237-5 หรือ 8259A เท่านั้น แต่ถ้าข้อมูลในบิท A9 นี้เป็น "1" ก็จะทำการอ่านข้อมูลได้เฉพาะจากพอร์ทที่อยู่บนคาร์ดต่างๆ เท่านั้น

จากที่ได้กล่าวมานั้นจะสรุปได้ว่าพอร์ทบน IBM/PC ทั้ง 1024 พอร์ทถูกแบ่งออกเป็น 2 กลุ่ม โดยที่กลุ่มแรกเป็นกลุ่มของพอร์ทที่อยู่บนเมนบอร์ดและกลุ่มที่สองเป็นกลุ่มที่จัดเตรียมไว้สำหรับพอร์ทที่อยู่บนคาร์ดต่าง ๆ

สำหรับในกรณีของการส่งข้อมูลให้กับพอร์ททั้ง 1024 พอร์ท เราสามารถที่จะเลือกส่งไปยังพอร์ทใด ๆ ใน IBM/PC ได้ ดังนั้นการเลือกแอดเดรสสำหรับพอร์ทที่อยู่บนคาร์ดจึงสามารถทำได้ โดยสะดวกแต่อย่างไรก็ตามสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือถ้าแอดเดรสที่เราเลือกให้กับพอร์ทนี้ตรงกับค่าแอดเดรสเดิมที่มีอยู่แล้ว เมื่อเราทำการส่งข้อมูลให้กับพอร์ทที่อยู่ในตำแหน่งแอดเดรสนี้ ก็จะเท่ากับเป็นการส่งข้อมูลให้กับทั้งพอร์ทที่อยู่บนเมนบอร์ดและพอร์ทที่อยู่บนคาร์ดด้วย ซึ่งในกรณีเช่นนี้อาจจะก่อให้เกิดความผิดพลาดขึ้นได้เช่นกัน ดังนั้นในการกำหนดค่าแอดเดรสให้กับพอร์ทที่ถูกสร้างขึ้นบนคาร์ดต่าง ๆ จึงควรจะใช้ค่าแอดเดรสที่แอดเดรสบิท A9 มีค่าเป็น "1"คือแอดเดรส 0FE00H จนถึง 0FFFFH เท่านั้น(แอดเดรสบิท A10-A15 ไม่ถูกใช้ในการดีโค้ด แต่เพื่อความสะดวกจึงกำหนดให้มีค่าเป็น "1" ในฐานสองทั้งหมด แต่ในการใช้งานจริงอาจเปลี่ยนให้แอดเดรส A10-A15 แต่ละบิทมีค่าเป็น "1" หรือ "0" ก็ได้)

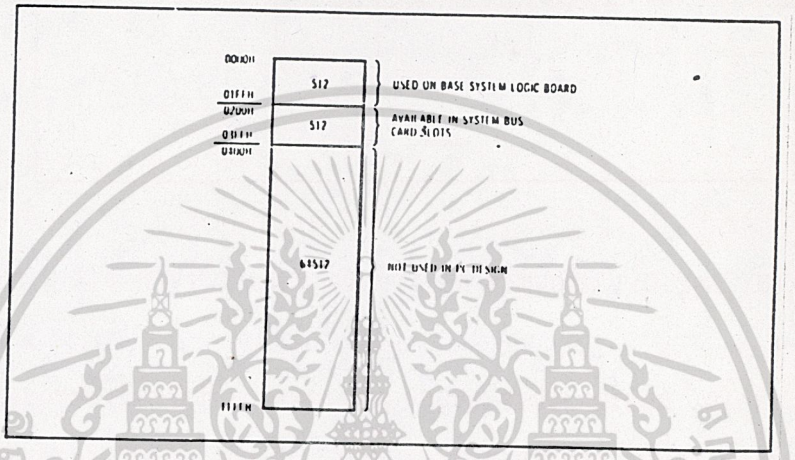
สำหรับรูปที่ 2.2.1 นี้ จะแสดงถึงการใช้งานแอดเดรสบิทต่างๆ ในการอ้างแอดเดรสของพอร์ทใน IBM/PC



รูป 2.2.1 การใช้แอดเดรสบิทต่างๆ ในการอ้างแอดเดรสของพอร์ทใน IBM/PC

การใช้งานแอดเดรสสำหรับพอร์ท I/O ใน IBM/PC

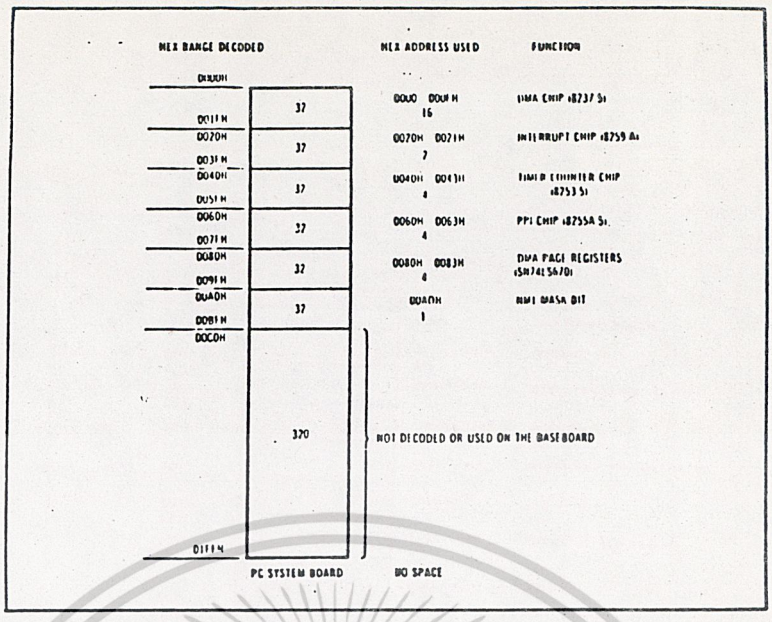
จากที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้นพอร์ท I/O ทั้ง 1024 พอร์ทใน IBM/PC จะถูกแบ่งออกเป็น 2 กลุ่ม ๆ ละ 512 พอร์ท สำหรับในหัวข้อนี้จะกล่าวถึงการใช้งานพอร์ทต่าง ๆเหล่านี้โดยจะแบ่งออกเป็น 2 กลุ่มตามที่ได้อธิบายไว้ในหัวข้อที่ผ่านมาดังนี้



2.2.2 การใช้งานแอดเดรสของพอร์ทบน IBM/PC

1. ในกลุ่มแรกนี้เป็นกลุ่มของพอร์ท I/O ที่อยู่บนเมนบอร์ดของ IBM/PC ซึ่งจะมีแอดเดรสอยู่ในตำแหน่ง 0000H จนถึง 01FFH (ขอให้ระลึกอยู่เสมอว่า A10-A15 นั้นไม่ถูกใช้งาน) หรือ แอดเดรสที่มีบิท A9 เป็น "0" นั่นเอง

สำหรับแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะถูกใช้ในการอ้างแอดเดรสของชิพซัพพอร์ท และ อุปกรณ์ที่เป็น I/O ต่าง ๆ บนเมนบอร์ดของ IBM/PC เช่น แอดเดรส 0000H จนถึง 000FH จะถูกใช้เป็นแอดเดรสสำหรับ 8237-5 DMA Controller เป็นต้น ในรูปที่ 2.1.3 จะแสดงถึงการใช้งานแอดเดรสต่างๆ ตั้งแต่ 0000H จนถึง 01FFH ในการอ้างแอดเดรสของชิพซัพพอร์ท และอุปกรณ์ต่าง ๆ ที่ทำหน้าที่เป็น I/O บนเมนบอร์ดของ IBM/PC

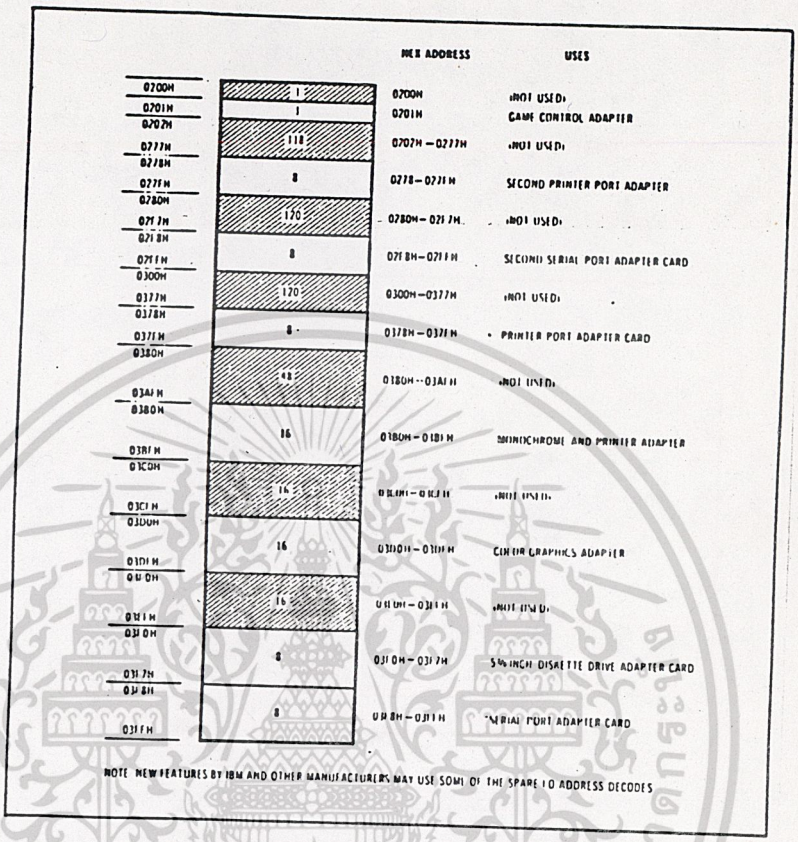


รูปที่ 2.2.3 การใช้งานแอดเดรสต่างๆ สำหรับพอร์ท I/O ของ IBM/PC

จากรูปข้างต้นจะเห็นว่าแอดเดรส 00C0H จนถึงแอดเดรส 01FFH นั้นไม่ได้ถูกใช้งานบนเมนบอร์ดของ IBM/PC ดังนั้นในกรณีนี้เราก็สามารถที่จะใช้งานแอดเดรสต่าง ๆ เหล่านี้ได้ แต่อย่างไรก็ตามแอดเดรสเหล่านี้ยังคงถูกดีโค้ดให้เป็นแอดเดรสที่ใช้ในการอ่านข้อมูลจากพอร์ท I/O บนเมนบอร์ดเท่านั้น ดังนั้นการใช้ค่าแอดเดรส 00C0H -01FFH กับพอร์ท I/O บนคาร์ด หรือวงจรอินเทอร์เฟสที่เราสร้างขึ้นนั้น ต้องเป็นพอร์ทเอาท์พุทเพียงชนิดเดียวเท่านั้น กล่าวคือ จะทำการอ่านข้อมูลจากพอร์ท I/O (ที่ไม่ได้อยู่บนเมนบอร์ด)ที่มีค่าแอดเดรสอยู่ในช่วง 00C0H-01FFH ไม่ได้

2. ในกลุ่มที่สองนี้ จะเป็นกลุ่มของพอร์ท I/O ที่ถูกใช้งานอยู่บนคาร์ดที่ใช้เสียบบนสล็อตต่าง ๆ ของ IBM/PC สำหรับแอดเดรสของพอร์ทเหล่านี้จะเริ่มต้นจากแอดเดรส 0200H จนถึง 03FFH ซึ่งก็คือแอดเดรสที่มีบิท A9 เป็น "1" นั่นเองสำหรับการใช้งานแอดเดรสของพอร์ท I/O ในกลุ่มนี้จะแสดงได้ดังรูป 2.2.4

อย่างไรก็ตามการใช้งานแอดเดรสในกลุ่มนี้อาจจะเปลี่ยนแปลงไปได้ทั้งนี้ขึ้นอยู่กับการใช้งานคาร์ดต่าง ๆ ร่วมกับ IBM/PC โดยคาร์ดที่ถูกออกแบบผลิตขึ้นใหม่นั้นอาจจะใช้ค่าแอดเดรสต่าง ๆ ที่เหลืออยู่นี้ได้ ดังนั้นก่อนที่จะทำการออกแบบวงจรอินเทอร์เฟสที่จำเป็นต้องใช่ค่าแอดเดรสสำหรับพอร์ท I/O จึงควรจะตรวจสอบดูก่อนว่าคาร์ดต่าง ๆที่ใช้อยู่ในระบบของ IBM/PC ที่เราใช้งานอยู่นั้นมีคาร์ดใดบ้าง และ คาร์ดเหล่านั้นใช้งานแอดเดรสใดบ้าง จากนั้นจึงทำการออกแบบวงจรอินเทอร์เฟสโดยเลือกใช้เฉพาะแอดเดรสที่ยังไม่ถูกใช้งาน

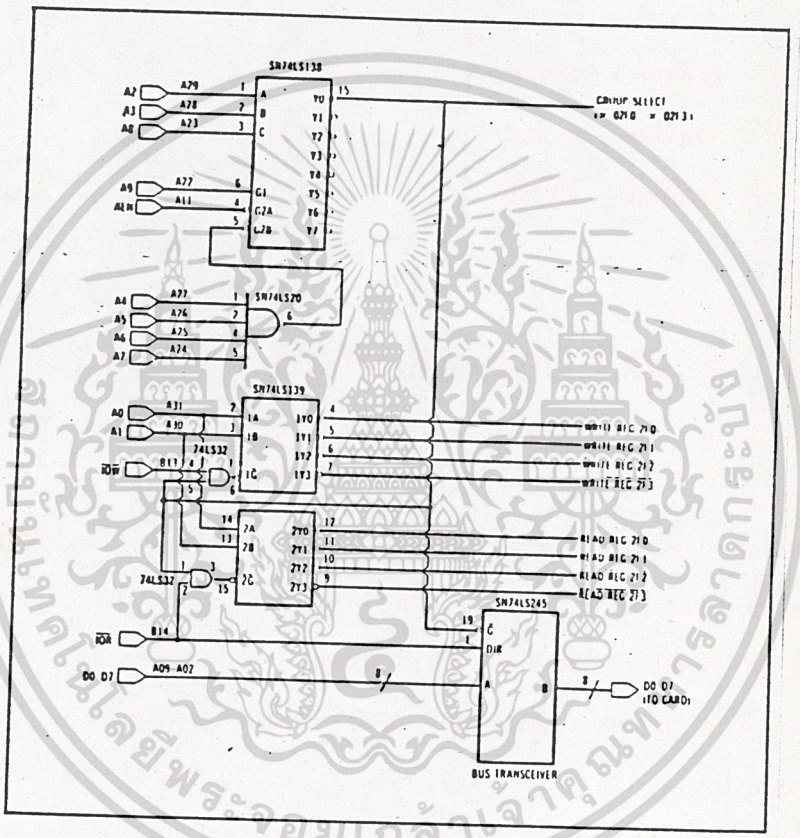


รูปที่ 2.2.4 การใช้งานแอดเดรสสำหรับพอร์ท I/O บนคาร์ดต่างๆ

เทคนิคในการดีโค้ดแอดเดรสสำหรับพอร์ท I/O

ในหัวข้อต่าง ๆ ที่ผ่านมาข้างต้นนั้น ได้กล่าวถึงการอ้างแอดเดรส และการใช้งานแอดเดรสต่าง ๆ ของพอร์ท I/O ใน IBM/PC สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการต่างๆ ที่ใช้ในการดีโค้ดแบบ Fixed

วิธีการดีโค้ดแบบนี้เป็นวิธีที่ง่ายและสะดวกในการดีโค้ดแอดเดรส หรือกลุ่มของแอดเดรสของพอร์ท I/O ซึ่งวิธีนี้เป็นการกำหนดจำนวนของแอดเดรสที่เราต้องการใช้ จากนั้นจึงทำการเลือกบล็อกของแอดเดรสที่ยังไม่ถูกใช้งานโดยคาร์ดหรือวงจรอินเทอร์เฟสอื่น ๆ (บล็อกของแอดเดรสที่เลือกต้องมีจำนวนแอดเดรสเพียงพอกับจำนวนแอดเดรสที่เราต้องการใช้งาน) แล้วจึงออกแบบวงจรที่ทำการดีโค้ดแอดเดรสที่เราต้องการ สำหรับตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสในแบบนี้จะแสดงได้ดังรูป 2.2.5



รูปที่ 2.2.5 ตัวอย่างวงจรดีโค้ดแอดเดรสแบบ Fixed

จากรูปจะเห็นได้ว่า วงจรที่ใช้นี้เป็นวงจรที่สามารถทำการดีโค้ดแอดเดรสได้ 8 กลุ่ม โดยแต่ละกลุ่มจะมีจำนวนแอดเดรส 4 แอดเดรสซึ่งแอดเดรสทั้ง 8 กลุ่มจะแสดงได้ดังตารางข้างล่าง

กลุ่ม	แอดเดรส
0 (Y0)	02F0H - 02F3H
1 (Y1)	02F4H - 02F7H
2 (Y2)	02F8H - 02FBH
3 (Y3)	02FCH - 02FFH
4 (Y4)	03F0H - 03F3H
5 (Y5)	03F4H - 03F7H
6 (Y6)	03F8H - 03FBH
7 (Y7)	03FCH - 03FFH

สำหรับในตัวอย่างนี้จะเลือกใช้การดีโค้ดแอดเดรสในกลุ่ม 0 (เริ่มจากแอดเดรส 02F0H จนถึง 02F3H) คือ ใช้สัญญาณเอาท์พุท(สัญญาณ GROUPSELECT) จากขา Y0 (ขา 15) ของ 74LS138 ไปทำการ OR กับสัญญาณ IOR และ IOW เพื่อสร้างเป็นสัญญาณอินาเบิลวงจรดีโค้ด (74LS139) แอดเดรสอีก 4 แอดเดรส ซึ่งแบ่งเป็น 2 ชุดคือ ชุดที่เป็น WRITE REG ซึ่งจะแอคทีฟ(ลอจิก "0") เมื่อ CPU ต้องการจะส่งข้อมูลให้กับวงจรภายนอก (สัญญาณ IOW แอคทีฟ) และชุดที่เป็น READ REG ซึ่งจะแอคทีฟเมื่อ CPU ต้องการจะอ่านข้อมูลจากวงจรภายนอก(สัญญาณ IOR แอคทีฟ) สัญญาณ WRITE REG และ READ REG นี้โดยทั่วไปจะนำไปเป็นสัญญาณสโตรป(Strobe) ให้กับวงจรภายนอกที่เกี่ยวข้อง เพื่อให้สามารถส่งหรือรับข้อมูลจาก CPU ได้ในช่วงเวลาที่เหมาะสม นอกจากนี้สัญญาณ GROUPSELECT ยังถูกนำไปใช้ในการ อินาเบิลบัฟเฟอร์ 74LS245 ด้วย เพื่อให้ CPU สามารถส่ง หรือรับข้อมูลจากอุปกรณ์ภายนอกได้เมื่อแอดเดรสในกลุ่มนี้ถูกเลือก สำหรับทิศทางของข้อมูลจะถูกควบคุมโดยสัญญาณ IOR ส่วนสัญญาณ AEN จะถูกนำมาใช้ในการดิสเอเบิลวงจรดีโค้ดโดยถ้าสัญญาณ AEN เป็น "1" ซึ่งเป็นช่วงเวลาของขบวนการ DMA นั้น 74LS138 จะถูกดิสเอเบิลทันที ทั้งนี้ก็เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้น เนื่องจากการดีโค้ดแอดเดรสของพอร์ทในระหว่างขบวนการ DMAนั่นเอง(ในระหว่างนี้แอดเดรสบนบัสแอดเดรสจะเป็นแอดเดรสของหน่วยความจำ คือ สัญญาณ MEMW หรือ MEMR จะแอดทีฟ แต่ในขณะเดียวกันสัญญาณ IOR หรือ IOW ก็จะแอดทีฟด้วย ดังนั้นถ้าไม่ดิสเอเบิลวงจรดีโค้ดไว้แล้ว อาจจะทำให้วงจรดีโค้ดคิดว่า แอดเดรสบนบัสแอดเดรสเป็นแอดเดรสของ

พอร์ท I/O ก็ได้)

ในการดีโค้ดแอดเดรสของพอร์ท I/O เราจะต้องคำนึงถึงช่วงเวลาของสัญญาณที่เกิดขึ้นในขบวนการอ่าน หรือเขียนข้อมูลลงบนพอร์ท I/O ดังนี้

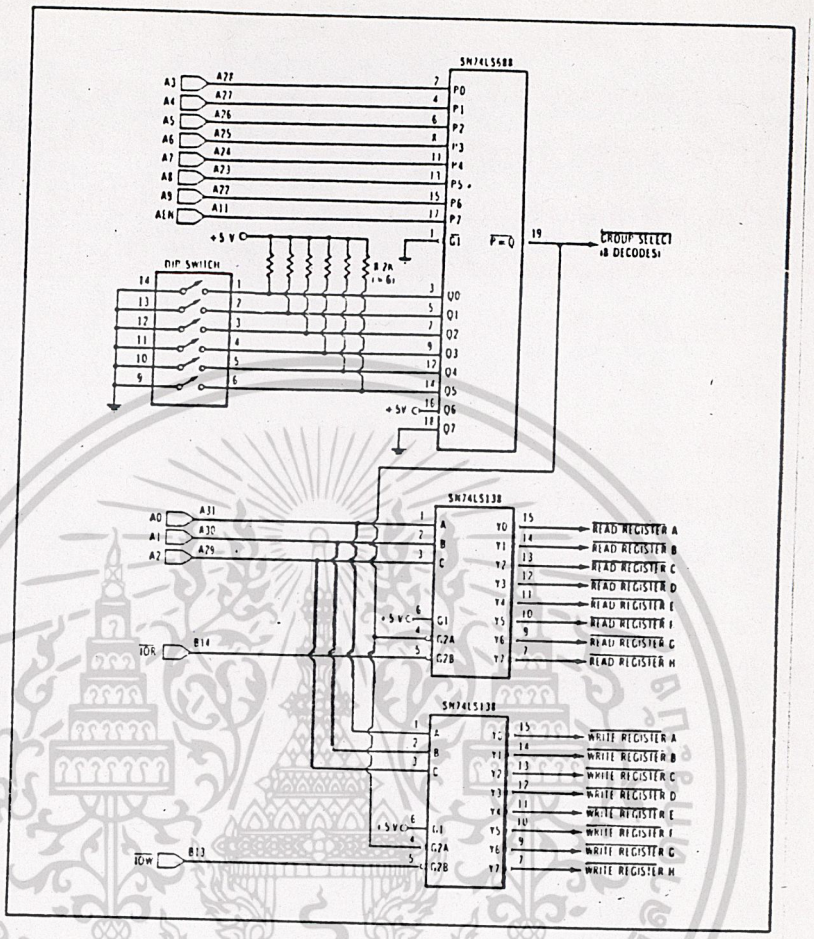
1. ในช่วงเริ่มต้นของบัสไซเคิลที่เกี่ยวกับพอร์ท I/O นั้น ถ้าสัญญาณจากวงจรดีโค้ดมีการหน่วงเวลา (Delay) มากเกินไป อาจจะทำให้สัญญาณดีโค้ดนี้เกิดขึ้นหลังจากที่สัญญาณ IOR หรือ IOW แอดทีฟ และเนื่องจากค่าแอดเดรสบนบัสแอดเดรสนั้นเปลี่ยนแปลงได้ตลอดเวลา ดังนั้นก่อนที่ค่าแอดเดรสที่ถูกต้องจะถูกส่งออกมาบนบัสแอดเดรสนั้น วงจรดีโค้ดจะได้รับค่าแอดเดรสอื่น ๆอยู่ซึ่งถ้าหากวงจรดีโค้ดมีการหน่วงเวลามากเกินไปแล้ว สัญญาณดีโค้ดแอดเดรสที่ไม่ถูกต้องนี้อาจจะถูกหน่วงเวลาจนเกิดขึ้นในช่วงเวลาที่สัญญาณIOR หรือ IOW เกิดขึ้นแล้วก็ได้ ทำให้ข้อมูลบนบัสข้อมูลนั้นถูกส่งไปยังพอร์ทที่ไม่ถูกต้อง สำหรับใน IBM/PC จะถูกออกแบบให้การหน่วงเวลาในวงจรดีโค้ดนั้นมีค่าไม่เกิน 92 nanosec.

2. ในช่วงท้ายของบัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท I/O นั้น ถ้าสัญญาณ IOW มีการหน่วงเวลาออกไป และวงจรดีโค้ดมีความเร็วในการทำงานสูงแล้วอาจจะทำให้ข้อมูลในบัสไซเคิลนี้ถูกส่งให้กับพอร์ท I/O ที่มีแอดเดรสตรงกับค่าแอดเดรสในบัสไซเคิลต่อไปก็ได้ สำหรับใน IBM/PC สัญญาณ IOW จะมีการหน่วงเวลาไปไม่เกิน 200 nanosec.

อย่างไรก็ตามช่วงเวลาที่ต้องสนใจมากอีกช่วงเวลาหนึ่ง ก็คือ ช่วงเวลาระหว่างขอบขาขึ้นของสัญญาณ IOW กับช่วงเวลาที่ข้อมูลที่ถูกต้องถูกส่งออกมาบนบัสข้อมูล ถ้าสัญญาณ IOW ถูกหน่วงเวลาไปเกินกว่า 120 nanosec. แล้ว อาจจะทำให้พอร์ท I/O ได้รับข้อมูลที่ไม่ถูกต้องก็ได้และสำหรับสัญญาณ IOR นั้นถ้ามีการหน่วงเวลาเกิดขึ้นแล้ว ก็จะทำให้ความเร็วในการอ่านข้อมูลถูกลดลง

การดีโค้ดโดยใช้สวิทช์เลือก

การดีโค้ดในแบบ Fixed ที่ได้กล่าวไว้ในหัวข้อที่ผ่านมานั้น มีข้อเสียอยู่บางประการคือแอดเดรสที่เราเลือกใช้งานไว้นั้นอาจจะซ้ำกับแอดเดรสของคาร์ดอื่นที่เรานำมาเพิ่มเข้าไปในระบบในภายหลังก็ได้ ซึ่งกรณีเช่นนี้เราต้องแก้ไขวงจรเพื่อหลีกเลี่ยงไปใช้แอดเดรสอื่นที่ยังว่างอยู่ และไม่ถูกใช้งานโดยคาร์ดที่จะเพิ่มเข้าไปใหม่ซึ่งยุ่งยากและต้องเสียเวลามากขึ้นปัญหาเช่นนี้เราสามารถแก้ไขได้โดยใช้วงจรดีโค้ด ที่สามารถเปลี่ยนแปลงค่าแอดเดรสได้ โดยเพียงแต่เปลี่ยนตำแหน่งของสวิทช์(ในที่นี้คือ DIP Switch) ที่เซ็ทไว้ในวงจรเท่านั้น ดังรูปที่ 2.2.6



รูปที่ 2.2.6 ตัวอย่างวงจรดีโค้ดโดยใช้สวิทช์เลือก

จากรูปเป็นวงจรที่ทำการดีโค้ดกลุ่มแอดเดรสขนาด 8 แอดเดรสซึ่งการเลือกกลุ่มแอดเดรสที่จะทำการดีโค้ดนี้จะทำได้โดยการเซ็ท DIP Switch ที่ขา Q0-Q5 ของ 74LS688

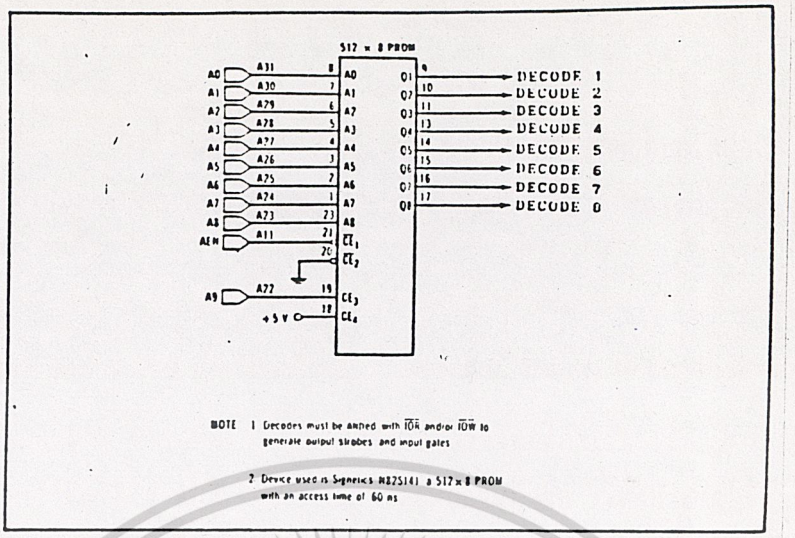
สำหรับหน้าที่ของ 74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุท 2 ชุดที่ถูกส่งเข้ามาทางขา P0-P7 และขา Q0-Q7 ถ้าอินพุททั้ง 2 ชุดนี้เท่ากันแล้วเอาท์พุทที่ขา P=Q จะให้เอาท์พุทเป็นลอจิก "0" จากในวงจรขา P0-P6 ของ 74S688 ต่อกับแอดเดรสบิท A3-A9 ในขณะที่ขา Q0-Q5 ต่อกับความต้านทานที่ทำหน้าที่เป็น Pull Up (รักษาระดับแรงดันให้เป็นลอจิก "1" ไว้ในกรณีที่ไม่มีอินพุทใด ๆ เข้ามา) และขา Q0-Q5 นี้จะต่อกับปลายข้างหนึ่งของ DIP Switch ด้วยส่วนปลายอีกข้างหนึ่งของ DIP Switch

นั้นจะต่อลง Ground (ลอจิก "0") ไว้ ดังนั้นถ้าเราทำการ "ON" DIP Switch ที่ต่อกับขาใดขานั้นก็จะได้รับลิอจิก "0" ในขณะที่ถ้า DIP Switchที่ต่อกับขาใดถูก "OFF" ขานั้นก็จะได้รับลอจิก "1" และเนื่องจากอินพุทที่ขา P0-P5(แอดเดรส A3-A9) ต้องเท่ากับอินพุทที่ขา Q0-Q5 ดังนั้นถ้าเราเปลี่ยนแปลงการเซ็ท DIP Switch เหล่านี้ก็จะทำให้แอดเดรสบิท A3-A5 ซึ่งต่อกับขา P0-P5 นั้นต้องเปลี่ยนแปลงตามไปด้วยจึงจะทำให้เอาท์พุทของ 74LS688 แอคทีฟได้ ทำให้เราสามารถเปลี่ยนแปลงค่าแอดเดรสที่ต้องการจะดีโค้ดได้ง่ายกว่าวิธีการดีโค้ดแบบ Fixed สำหรับขา Q6 นั้นจะต่อกับลอจิก "1" (+5 V) และขา P6 ต่อกับแอดเดรสบิท A9 ในกรณีเช่นนี้จึงเท่ากับเป็นการบังคับให้แอดเดรสที่จะทำการดีโค้ดได้นั้นจะต้องมีแอดเดรสบิท A9 เป็น "1" เท่านั้น ส่วนขา P7 จะต่อกับสัญญาณ AEN โดยมีขา Q7 ต่อกับลอจิก "0" การต่อในลักษณะนี้ก็เพื่อป้องกันไม่ให้ 74LS688 ทำการดีโค้ดในระหว่างขบวนการ DMA นั่นเอง เอาท์พุทจากขา P=Q ของ 74LS688 นี้ จะถูกนำไปใช้ในการอินาเบิล 74LS138 ซึ่งทำหน้าที่ในการดีโค้ดแอดเดรส 8 แอดเดรสของกลุ่มแอดเดรสที่เราเลือก (โดยใช้ DIPSwitch ดังที่ได้กล่าวในตอนต้น)

วงจรในลักษณะนี้เราสามารถจะนำไปใช้เป็นการดีโค้ดในแบบ Fixed ได้โดยการนำเอา DIP Switch ออก จากนั้นถ้าอินพุทใดต้องการลอจิก "0" จึงจะใช้ตัวนำเชื่อมต่อระหว่างขั้วทั้งสองแทนการเซ็ท DIP Switch ให้ "ON" แต่ถ้าอินพุทใดต้องการลอจิก "1" ก็ปล่อยขั้วทั้งสองนั้นไว้

การดีโค้ดโดยใช้ PROM

การดีโค้ดในแบบต่างที่กล่าวมาแล้วนั้น เป็นการดีโค้ดในลักษณะที่แอดเดรสของพอร์ทต่าง ๆอยู่ร่วมกันเป็นกลุ่ม แต่ในบางกรณีพอร์ทที่เราใช้งานนั้นมีแอดเดรสแยกกันอย่างเป็นอิสระ เช่นในการนำเอาหน้าที่การทำงานที่อยู่บนคาร์ดต่าง ๆ มารวมไว้บนคาร์ดเพียงคาร์ดเดียว และมีความจำเป็นต้องคงค่าแอดเดรสของพอร์ทเดิม(ที่อยู่บนคาร์ดเดิม)ไว้ด้วย ทำให้ไม่สามารถใช้การดีโค้ดในแบบต่าง ๆ ที่ผ่านมาได้ เนื่องจากการใช้วิธีการดีโค้ดในแบบที่ผ่านมานั้นจะทำให้ต้องใช้อุปกรณ์ที่ทำการดีโค้ดนั้นมากเกินไป ในกรณีเช่นนี้เราจำเป็นต้องใชการดีโค้ดอีกแบบหนึ่งซึ่งจะได้กล่าวถึงในหัวข้อนี้ คือ การดีโค้ดโดยใช้ PROM (Programmable Read Only Memory) ดังในรูปที่ 2.2.7



รูปที่ 2.2.7 ตัวอย่างวงจรดีโค้ดโดยใช้ PROM

จากรูปข้างต้นเป็นวิธีการง่าย ๆแบบหนึ่งในการดีโค้ดโดยใช้ PROM ซึ่งจะเห็นได้ว่าเราใช้เส้นแอดเดรส A0-A8 ของระบบต่อเข้ากับเส้นแอดเดรส A0-A8 ของ PROM และใช้บัสข้อมูลทั้ง 8 ของ PROM คือ Q1-Q8 เป็นเอาท์พุท สำหรับใช้เป็นสัญญาณดีโค้ดให้กับพอร์ทต่าง ๆ 8 พอร์ทอย่างไรก็ตามสัญญาณดีโค้ดทั้ง 8 เส้น คือ DECODE1-DECODE8 นี้ยังคงต้องนำไป OR กับสัญญาณ IOR หรือ IOW ก่อนที่จะนำไปอีนาเบิลพอร์ทที่มีแอดเดรสตรงกับแอดเดรสที่ป้อนให้กับ PROM นั้น

จากที่ได้กล่าวมานั้นจะเห็นได้ว่าส่วนของวงจรดีโค้ดนั้น จะมี PROM เพียงตัวเดียวเท่านั้นซึ่ง PROM ที่จะนำมาใช้งานนี้จะต้องถูกโปรแกรมมาก่อนแล้ว โดยข้อมูลที่โปรแกรมให้กับแอดเดรสต่าง ๆ ของ PROM นั้น จะต้องสัมพันธ์กับสัญญาณดีโค้ดที่เราต้องการ กล่าวคือ เราจะต้องทราบเสียก่อนว่าแอดเดรสของพอร์ททั้ง 8 ที่เราต้องการจะดีโค้ดนั้นมีแอดเดรสใดบ้าง แล้วจึงกำหนดว่าพอร์ทใดจะใช้สัญญาณดีโค้ดเส้นใด จากนั้นจึงโปรแกรมข้อมูลให้กับ PROM โดยแอดเดรสใดถ้าต้องการให้สัญญาณดีโค้ดแอคทีฟ (ในที่นี้จะกำหนดให้สัญญาณดีโค้ดแอคทีฟที่ลอจิก "0") ก็กำหนดให้ข้อมูลในบิทที่ตรงกับสัญญาณดีโค้ดนั้นเป็น "0" เช่นถ้าเรากำหนดให้แอดเดรสของพอร์ทที่เราต้องการจะดีโค้ดเป็น 0393H และเลือกใช้สัญญาณ DECODE5 เราก็จะต้องทำการโปรแกรมให้แอดเดรส 0193H ของ PROM (เหตุที่แอดเดรสของ PROMเป็น 0193H แทนที่จะเป็นแอดเดรส 0393H เหมือนกับแอดเดรสของพอร์ทก็เพราะแอดเดรสของ PROM มีเพียง 9 บิทคือ A0-A8 เท่านั้นส่วนบิท A9 จะถูกต่อเข้ากับ PROM ในภายหลัง เพื่ออีนาเบิล PROM เมื่อข้อมูลในบิท A9 นี้เป็น "1" เท่านั้น)

มีข้อมูลในบิท Q5(ถ้านับเริ่มจากบิท D0 ก็คือ D4)เป็น "0" ส่วนบิทอื่น ๆ นั้นมีค่าเป็น"1" ทั้งหมด ดังนั้นการโปรแกรมแอดเดรส 0193H ของ PROM จึงต้องโปรแกรมด้วยข้อมูล 0EFH เป็นต้น สำหรับข้อมูลในแอดเดรสอื่น ๆ ที่นอกเหนือจากแอดเดรสทั้ง 8 ที่กำหนดแล้วจะต้องโปรแกรมให้ข้อมูลทุกบิทเป็น "1" ทั้งหมด ซึ่งก็คือ โปรแกรมด้วยข้อมูล 0FFH นั่นเอง

ตัวอย่างเช่น

ถ้าแอดเดรสของพอร์ททั้ง 8 ที่เราต้องการจะดีโค้ดเป็น 024A,02B5,0317, 0361,0382,03A8,03C4 และ 03DB ในฐานสิยหกตามลำดับ โดยกำหนดให้สัญญาณจากการดีโค้ดแอดเดรสเหล่านี้เป็นสัญญาณ DECODE1 จนถึง DECODE8 ตามลำดับแล้ว(เช่นสัญญาณจากการดีโค้ด 024AH ก็คือสัญญาณ DECODE1และสัญญาณจากการดีโค้ด 02B5H คือสัญญาณ DECODE2เป็นต้น) เราจะต้องทำการโปรแกรม PROM ให้มีข้อมูลที่สัมพันธ์กับเอาท์พุทที่เราต้องการ ดังนี้

1. แอดเดรสของพอร์ทเป็นแอดเดรสที่บิท A9 ถูกใช้งานร่วมด้วย โดยในบิทนี้จะต้องมีข้อมูลเป็น "1"ในขณะที่แอดเดรสของ PROM จะมีเพียง 9 บิทคือ A0-A8เท่านั้น เราจึงจัดแอดเดรสของ PROM เมื่อเทียบกับแอดเดรสของพอร์ทดังนี้

แอดเดรสของพอร์ท (บิท A9 ถูกใช้งาน)	แอดเดรสของ PROM (ฉพาะบิท A0-A8)
024A	04A
02B5	0B5
0317	117
0361	161
0382	182
03A8	1A8
03C4	1C4
03D4	1DB

2. ข้อมูลที่จะโปรแกรมให้กับแอดเดรสทั้ง 8 ของ PROM จะต้องสัมพันธ์กับเอาท์พุทที่ต้องการ เช่น ถ้ามีการอ้างถึงแอดเดรสของพอร์ท 02B5H แล้ว PROM จะต้องให้เอาท์พุทที่มีลอจิก "0" ที่ขา Q2 (DECODE2) ส่วนเอาท์พุทที่ขาอื่นต้องเป็น "1" ดังนั้นจึงต้องโปรแกรม ให้แอดเดรส 00B5H ของ PROM มีข้อมูลเป็น 1111 1101 (ฐานสอง) หรือ 0FDH เป็นต้น สำหรับแอดเดรสอื่น ๆ นอกเหนือจากแอดเดรสทั้ง 8 นี้แล้วจะต้องถูกโปรแกรมให้มีข้อมูลเป็น 0FF (ฐานสิบหก) ทั้งหมด ดังนี้

แอดเดรสของ PROM (ฐานสิบหก)	ข้อมูล(ฐานสิบหก)
000--049	0FF
04A	0FE
04B--0B4	0FF
0B5	0FD
0B6--116	0FF
117	0FB
118--160	0FF
161	0F7
162--181	0FF
182	0EF
183--1A7	0FF
1A8	0DF
1A9--1C3	0FF
1C4	0BF
1C5--1DA	0FF
1DB	07F
1DC--1FF	0FF

อย่างไรก็ตามสิ่งสำคัญอีกสิ่งหนึ่งที่จะต้องคำนึงถึงเสมอ เมื่อใช้วิธีการดีโค้ดในแบบนี้ก็คือ PROM ที่ใช้นั้นจะต้องใช้เวลาในการทำงานน้อยกว่า 92 nanosec. ด้วย

การเพิ่มจำนวนแอดเดรสของพอร์ทบน IBM/PC

จากที่ได้กล่าวแล้วว่าภายใน IBM/PC นั้น ได้ออกแบบให้มีการอ้างแอดเดรสของพอร์ท(โดยวงจรอินเทอร์เฟสภายนอก; ไม่รวมพอร์ทที่อ้างถึงโดยระบบบนเมนบอร์ดของ IBM/PC อีก512 พอร์ท) ได้เพียง 512 พอร์ท จากจำนวนที่ 8088 สามารถอ้างได้สูงสุดถึง 64K พอร์ทและในจำนวน 512 พอร์ท ดังกล่าวนี้ได้ถูกใช้งานโดยอุปกรณ์หรือคาร์ดต่างๆ ไปไม่น้อย ซึ่งอาจจะทำให้การออกแบบใช้งานวงจรอินเทอร์เฟสที่จำเป็นต้องใช้แอดเดรสของพอร์ทเป็นจำนวนมากนั้นเกิดปัญหาขึ้นได้ดังนั้นในหัวข้อนี้จะได้กล่าวถึงวิธีการง่ายๆ ในการเพิ่มจำนวนแอดเดรสของพอร์ทต่างๆบน IBM/PC

การใช้เส้นแอดเดรส 6 บิทบน

เราได้ทราบมาแล้วว่า การออกแบบวิธีการดีโค้ดแอดเดรสของพอร์ทต่าง ๆใน IBM/PC นั้นจะใช้เส้นแอดเดรสเพียง 10 บิทคือ A0-A9 เท่านั้นจากจำนวนทั้งสิ้น 16 บิท (A0-A15) จะเห็นได้ว้ามีจำนวนเส้นแอดเดรสที่ไม่ได้ถูกใช้งานอยู่อีกถึง 6 บิทคือ A10-A15 ในหัวข้อนี้จึงจะกล่าวถึงวิธีการที่จะใช้งานเส้นแอดเดรสเหล่านี้ในการอ้างแอดเดรสของพอร์ทบนวงจรอินเทอร์เฟสที่เราออกแบบขึ้น

ในการที่จะใช้งานเส้นแอดเดรสเหล่านี้เราจะต้องเริ่มต้นจากการตรวจสอบว่าแอดเดรสใดบ้างที่ถูกใช้งานโดยคาร์ดหรือวงจรอินเทอร์เฟสอื่นๆ ที่เราเพิ่มเข้าไปในระบบจากนั้นจึงเลือกแอดเดรสที่ไม่ถูกใช้งานโดยวงจร หรือคาร์ดเหล่านี้มาใช้งาน ทั้งนี้ก็เพื่อให้แน่ใจได้ว่าในขณะที่เราใช้งานแอดเดรส 5 บิทบนอยู่นั้นจะไม่มีผลกับแอดเดรสของพอร์ทเดิมที่มีอยู่บนคาร์ดต่าง ๆ

ตัวอย่างเช่น

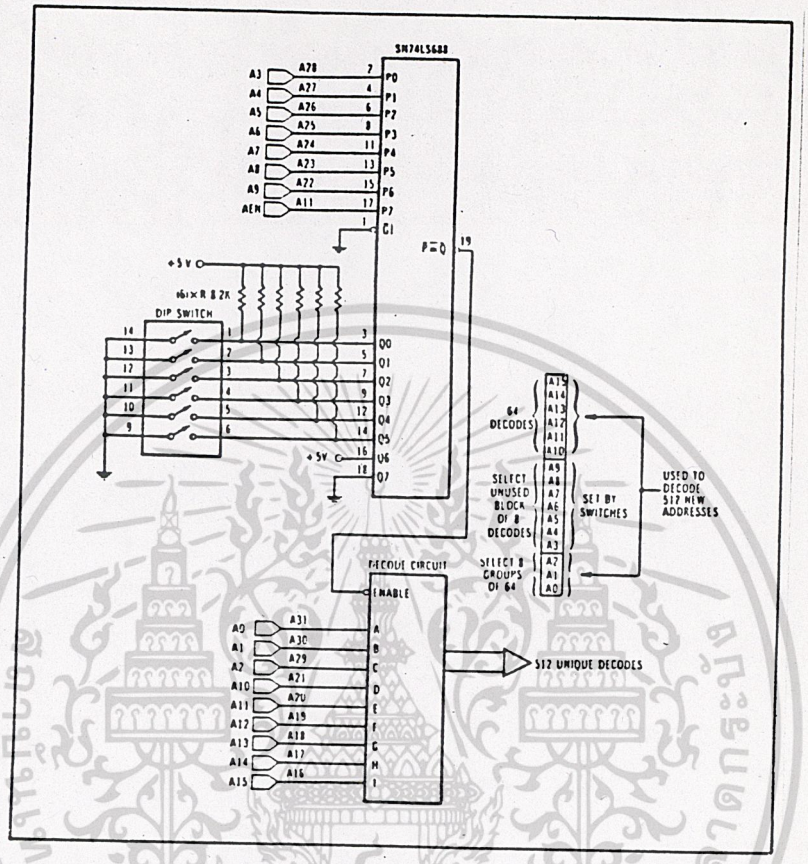
ถ้าในระบบของเรามีการใช้งานคาร์ดเพียง 2 แผ่น คือ Color/Graphics Card ซึ่งใช้แอดเดรสของพอร์ทตั้งแต่ 03D0H และ 5"1/4 Disk Drive Adapter Card ซึ่งใช้แอดเดรสของพอร์ทตั้งแต่ 03F0H-03F7H แล้วเราก็สามารถจะใช้แอดเดรส 6 บิทบนนี้ กับแอดเดรสใดก็ได้ที่แอดเดรส 10 บิทล่างไม่ได้อยู่ในช่วงของ 0300H-03DFH และ 03F0H-03F7H ทั้งนี้ก็เพราะแอดเดรสของพอร์ทเหล่านี้ใช้งานเฉพาะเส้นแอดเดรส 10 บิทล่างเท่านั้น ดังนั้นไม่ว่าแอดเดรส 6 บิทบนที่เหลือจะเปลี่ยนแปลงไปอย่างไรก็ตาม โดยที่แอดเดรส 10 บิทล่างยังคงเดิมอยู่ พอร์ทเหล่านี้ก็จะถูกทำให้แอคทีฟได้เสมอ เช่น แอดเดรสของพอร์ท 0300Hนั้นไม่ว่าเราจะอ้างแอดเดรสบิท A9-A0เป็น 11 0000 0000 ในฐานสองอยู่ พอร์ท ๆนี้ก็จะยังคงถูกทำให้แอคทีฟ คือ สามารถที่จะรับส่งข้อมูลให้กับระ

บบได้อยู่ดี ดังนั้นเราจึงต้องเลือกใช้เฉพาะแอดเดรสของพอร์ทที่แอดเดรส 10 บิทล่างยังไม่ถูกใช้งานโดยวงจรหรือคาร์ดอื่นใดในระบบ

หลังจากได้แอดเดรส 10 บิทล่างนี้แล้ว จึงใช้สัญญาณที่ได้จากการดีโค้ดแอดเดรส 10 บิทล่างนี้ไปอีนาเบิลวงจรดีโค้ดแอดเดรส 6 บิทบนที่เหลือ จากลักษณะเช่นนี้จะเห็นได้ว่าสำหรับแอดเดรส 10 บิทล่างที่เราเลือกไว้ 1 แอดเดรส เมื่อนำมาดีโค้ดร่วมกับแอดเดรส 6 บิทที่เหลือ(A10-A15)แล้วจะทำให้ได้แอดเดรสเพิ่มขึ้นเป็น 64 แอดเดรส เช่นถ้าแอดเดรส 10 บิทล่างที่เราเลือกเป็น 0213H เมื่อนำมาดีโค้ดร่วมกับแอดเดรส 6 บิทบนแล้วจะทำให้ได้แอดเดรสเพิ่มขึ้นเป็น 64 แอดเดรสคือ

แอดเดรส(ฐานสิบหก)	แอดเดรส(ฐานสิบหก)
0213	0000 00 100011
0613	0000 01 100011
0A13	0000 10 100011
0E13	0000 11 100011
1213	0001 00 100011
:	:
:	:
:	:
OFA13	1111 10 100011
OFE13	1111 11 100011

จากตัวอย่างข้างต้นจะเห็นได้ว่า แอดเดรส 10 บิทล่างเพียงแอดเดรสเดียว เมื่อนำมาดีโค้ดร่วมกับแอดเดรส 6 บิทบนก็จะทำให้ได้แอดเดรสเพิ่มขึ้นอีก 63 แอดเดรส (64-1) แอดเดรส ดังนั้นถ้าเราเลือกกลุ่มของแอดเดรส 10 บิทล่างที่มีจำนวน 8 แอดเดรส ก็จะทำให้เราได้แอดเดรสเพิ่มขึ้นอีกเป็นจำนวนถึง 502 แอดเดรส (63x8=502) เมื่อรวมกับแอดเดรสเดิมอีก 8 แอดเดรส ก็จะได้จำนวนแอดเดรสทั้งสิ้นถึง 512 แอดเดรสสำหรับรูปข้างล่างนี้ จะแสดงวงจรที่สามารถใช้ในการดีโค้ดในลักษณะนี้ได้



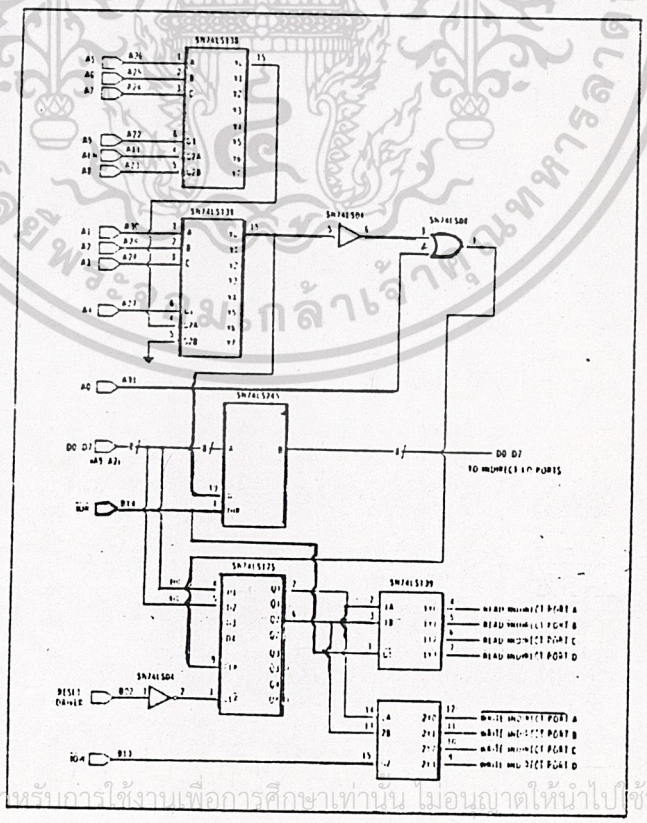
รูปที่ 2.2.8 ตัวอย่างวงจรดีโค้ดแอดเดรสโดยใช้แอดเดรส 6บิทบน

จากรูปข้างต้นนั้นเราจะเลือกแอดเดรสทั้งกลุ่มโดยการเซ็ท DIP Switch ที่ขาอินพุท Q0-Q5 ของ IC SN74LS688 สำหรับเอาท์พุท P=Q ซึ่งเป็นสัญญาณที่แอคทีฟเมื่อค่าแอดเดรสบิท A3-A9 บนบัสแอดเดรสของระบบนั้นตรงกับค่าที่เซ็ทไว้ในอินพุท Q0-Q6 ของ 74LS688 นั้นจะถูกนำไปใช้ในการอีนาเบิลวงจรดีโค้ดแอดเดรส A0-A2 และ A1-A15 ซึ่งจะทำให้การสัญญาณการดีโค้ดเป็นจำนวนถึง 512 เส้นอย่างไรก็ตามการดีโค้ดเหล่านี้ยังจำเป็นต้องนำไปใช้ OR หรือ AND กับสัญญาณ IOR หรือ IOW ของระบบก่อนที่จะนำไปใช้งานจริงด้วย (วงจรนี้มีลักษณะเป็นเพียงบล็อกไดอะแกรมอย่างง่าย ๆเพื่อเป็นแนวทางในการออกแบบวงจรเท่านั้น ในทางปฏิบัติผู้อ่านอาจจะดัดแปลงแก้ไขเพื่อให้เหมะสมกับสภาพของงานได้ในภายหลัง

การใช้วิธีอ้างแอดเดรสแบบ Indirect

วิธีการอ้างแอดเดรสในแบบนี้เป็นวิธีที่นิยมใช้กันโดยทั่วไป ซึ่งหลักการของวิธีการอ้างแอดเดรสในแบบนี้ ก็คือ ใช้ข้อมูลของพอร์ทหนึ่งในการอ้างแอดเดรสของอีกพอร์ทหนึ่ง กล่าวคือเมื่อเราส่งข้อมูลไปยังแอดเดรสของพอร์ทที่เรากำหนดให้ใช้สำหรับการอ้างแอดเดรสแบบ Indirect นี้แล้ว ข้อมูลที่เราส่งไปให้กับพอร์ทนั้น ๆ ก็จะถูกนำไปเข้าวงจรดีโค้ดอีกครั้งหนึ่ง จึงจะได้สัญญาณดีโค้ดเพื่อส่งให้กับพอร์ทต่าง ๆ ที่ต้องใช้การอ้างแอดเดรสแบบ Indirect นี้ดังนั้นในกรณีที่บัสข้อมูลของระบบเป็นขนาด 8 บิท เมื่อใช้วิธีการอ้างแอดเดรสแบบ Indirect นี้ ทำให้เราสามารถขยายแอดเดรสของพอร์ทขึ้นได้เป็นจำนวนถึง 256 แอดเดรส

สำหรับจำนวนพอร์ทที่ใช้ในการอ้างแอดเดรสแบบ Indirect นี้เราจำเป็นต้องใช้ 2 พอร์ท คือพอร์ทที่ถูกกำหนดให้ใช้สำหรับการอ้างแอดเดรสแบบ Indirect ซึ่งข้อมูลที่ส่งให้กับพอร์ทนี้ จะเป็นข้อมูลที่ถูกนำไปใช้ในการดีโค้ด เพื่อให้ได้สัญญาณดีโค้ดแอดเดรส ของพอร์ทที่เป็นแบบ Indirect ส่วนอีกพอร์ทหนึ่งจะเป็นพอร์ทที่ใช้สำหรับรับหรือส่งข้อมูลให้กับพอร์ทที่เป็นแบบ Indirect นี้โดยเฉพาะ สำหรับตัวอย่างวงจรที่ใช้สำหรับการดีโค้ดแอดเดรสแบบ Indirect นี้จะแสดงดังในรูป 2.2.9



รูปที่ 2.2.9 ตัวอย่างวงจรดีโค้ดแบบ Indirect

จากตัวอย่าง วงจรข้างต้นนั้นเป็นวงจรที่ใช้สำหรับการดีโค้ดแอดเดรสของ - พอร์ทที่เป็นแบบ Indirect 4 พอร์ท โดยแอดเดรสของพอร์ทที่ใช้ในการอ้างแอดเดรสแบบ Indirect นี้คือ แอดเดรส 0211H และแอดเดรสของพอร์ทที่ใช้ในการรับและส่งข้อมูลให้กับพอร์ทแบบ Indirect ทั้ง 4 พอร์ทก็คือแอดเดรส 0210H

จากวงจรนั้นเมื่อเราต้องการจะอ้างถึงแอดเดรสของพอร์ทแบบ Indirect นี้ก็จะทำได้โดยการใช้คำสั่ง OUT ส่งข้อมูลให้กับพอร์ท 0211H โดยข้อมูลที่ส่งให้กับพอร์ท 0211H นี้จะขึ้นอยู่กับพอร์ทที่เราต้องการติดต่อด้วย เช่น ถ้าเราต้องการจะติดต่อกับพอร์ทที่ 1 ก็ส่งข้อมูลเป็น 01H ให้กับพอร์ท 0211 H และเนื่องจากในวงจรนี้เราทำการดีโค้ดเพียง 4 พอร์ทดังนั้นข้อมูลจึงมีเพียง 4 ลักษณะคือ 01H, 02H, 03H และ 04Hเมื่อพอร์ท 0211H นี้ได้รับข้อมูลที่เราส่งไปให้แล้วก็จะนำเอาข้อมูลบิท D0 และ D1 ไปทำการดีโค้ดซึ่งจะทำให้ได้สัญญาณดีโค้ดของทั้ง 4 พอร์ทตามที่เราต้องการ ส่วนการอ่านหรือเขียนข้อมูลลงบนพอร์ททั้ง 4 นี้ จะทำได้โดยใช้คำสั่ง IN หรือ OUT รับ หรือ ส่งข้อมูลไปยังพอร์ท 0210H

การอ้างแอดเดรสของพอร์ทโดยใช้วิธี Memory-Mapped

การอ้างแอดเดรสโดยวิธีต่าง ๆ ที่กล่าวถึงข้างต้นนั้น เป็นวิธีการอ้างในแบบที่เรียกว่า I/O Mapped กล่าวคือ เป็นวิธีการอ้างแอดเดรสของพอร์ท I/O โดยตรง คือใช้คำสั่ง IN หรือ OUT ในการรับหรือส่งข้อมูล และใช้สัญญาณ IOR หรือ IOW เป็นสัญญาณควบคุมการรับส่งข้อมูล ซึ่งจะเป็นการแยกออกจากการอ้างแอดเดรสของหน่วยความจำโดยสิ้นเชิง และภายใน IBM/PCก็ใช้วิธีการอ้างแอดเดรสของพอร์ทแบบ I/O Mapped นี้

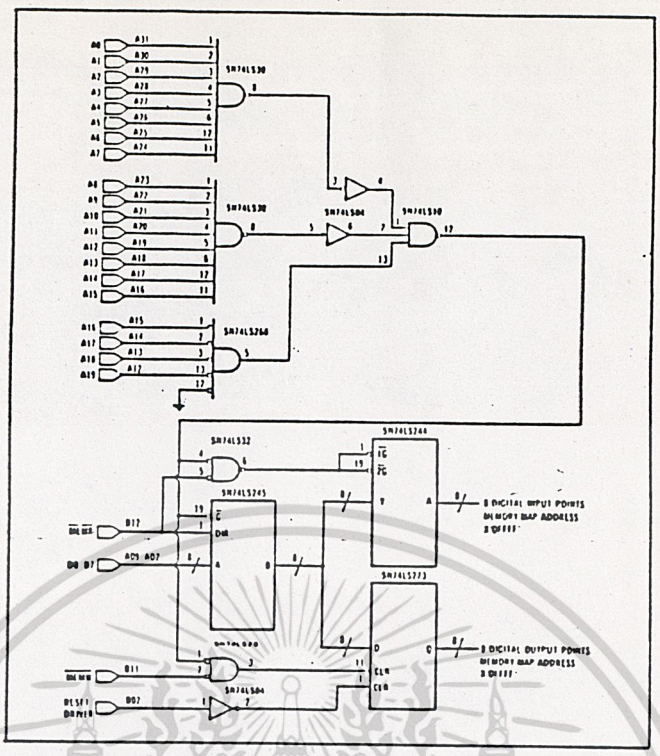
แต่ในหัวข้อนี้เราจำศึกษาถึงการอ้างแอดเดรสของพอร์ท I/O ในอีกแบบหนึ่งซึ่งเป็นแบบที่เรียกว่า Memory-Mapped ในการอ้างแอดเดรสของพอร์ท I/O ในแบบนี้จะเป็นการใช้แอดเดรสของหน่วยความจำเป็นแอดเดรสของพอร์ท I/O คือการอ้างถึงแอดเดรสของพอร์ทเหล่านี้จะใช้วิธีเดียวกับการอ้างแอดเดรสของหน่วยความจำ เช่น ใช้คำสั่ง MOV แทนคำสั่ง IN หรือ OUT และใช้สัญญาณ MEMR และ MEMW ในการควบคุมการรับส่งข้อมูลแทนที่จะใช้ IOR หรือ IOW สำหรับวิธีการอ้างแอดเดรสแบบ Memory-Mapped นี้ถึงแม้จะไม่มีการใช้งานใน IBM/PC ก็ตามแต่ใน CPU บางเบอร์เช่น 6502 นั้นจะใช้ได้เฉพาะวิธีการแบบ Memory-Mapped นี้ ในการอ้างแอดเดรสของพอร์ท I/O เท่านั้น ทั้งนี้เนื่องจากภายใน 6502 ไม่มีการแยกคำสั่งที่ใช้ในการจัดการเกี่ยวกับพอร์ท I/O ไว้โดยเฉพาะ และไม่มีขาสัญญาณที่จัดไว้สำหรับควบคุมพอร์ท I/O โดยเฉพาะด้วย

การใช้วิธีการอ้างแอดเดรสแบบ Memory-Mapped นี้จะช่วยให้เราสามารถ

ขยายจำนวนแอดเดรสของพอร์ท I/O ออกไปได้อีกเป็นจำนวนมาก ทั้งนี้เนื่องจาก CPU 8088 สามารถจะอ้างแอดเดรสสำหรับหน่วยความจำได้เป็นจำนวนถึง 1 Mbyte ในขณะที่อ้างแอดเดรสของพอร์ท I/O ได้เพียง 64K พอร์ทเท่านั้น นอกจากนี้การใช้วิธีการอ้างแอดเดรสแบบ Memory-Mapped นี้ยังทำให้เราแน่ใจได้ว่าแอดเดรสของพอร์ท I/Oที่เรากำหนดขึ้นนั้นจะไม่ตรงกับแอดเดรสของพอร์ทให้กับพอร์ท I/O นั้น สามารถทำได้สะดวกขึ้น สำหรับข้อดีอีกประการหนึ่งของการอ้างแอดเดรสแบบนี้ ก็คือ เราสามารถที่จะนำเอาชุดคำสั่งต่าง ๆที่เกี่ยวกับการจัดการข้อมูลของหน่วยความจำมาใช้ในการเกี่ยวกับข้อมูลของพอร์ทที่มีการอ้างแอดเดรสแบบนี้ได้ ซึ่งจะทำให้การจัดการเกี่ยวกับข้อมูลของพอร์ทเหล่านี้มีประสิทธิภาพสูงขึ้น

สำหรับข้อเสียของการใช้วิธีการอ้างแอดเดรสแบบ Memory-Mapped นี้ มีอยู่ 3 ประการ คือ เนื่องจากการอ้างแอดเดรสในรูปแบบนี้เป็นการอ้างแอดเดรสในรูปแบบเดียวกับหน่วยความจำ ดังนั้นในการดีโค้ดแอดเดรสจึงจำเป็นต้องทำการดีโค้ดทั้ง 20 บิท ซึ่งจะทำให้ต้องใช้วงจรดีโค้ดที่ยุ่งยากกว่าการอ้างแอดเดรสแบบ I/O Mapped และในการที่จะอ้างแอดเดรสของพอร์ทได้นั้น เราอาจจะจำเป็นต้องทำการเปลี่ยนแปลงค่าในรีจิสเตอร์ Segment ก่อน โดยเฉพาะในกรณีที่ค่าเซกเมนต์ของพอร์ทที่เราต้องการอ้างถึงนั้นไม่ตรงกับค่าเซกเมนต์ของโปรแกรมที่ CPU กำลังทำอยู่ ซึ่งจะทำให้ขาดความคล่องตัวในการอ้างถึงแอดเดรสของพอร์ทต่าง ๆเหล่านี้ไปได้ นอกจากนั้นยังมีปัญหาเกี่ยวกับช่วงเวลาของบัสไซเคิลเข้ามาเกี่ยวข้องด้วย กล่าวคือ ช่วงเวลาในบัสไซเคิลที่เกี่ยวกับหน่วยความจำนั้นจะใช้ช่วงเวลานานเท่ากับช่วงเวลาของคล็อก 4 ลูก ในขณะที่บัสไซเคิลที่เกี่ยวกับพอร์ท I/O จะใช้ช่วงเวลานานเท่ากับช่วงเวลาของคล็อกถึง 5 ลูก ดังนั้นจึงทำให้การทำงานของพอร์ท I/O ที่มีการอ้างแอดเดรสแบบ Memory-Mapped นี้มีช่วงเวลาการทำงานที่สั้นกว่าช่วงเวลาการทำงานของพอร์ท I/O ที่มีการอ้างแอดเดรสแบบ I/O Mapped ซึ่งจะทำให้อุปกรณ์ที่ใช้ในการอ้างแอดเดรสแบบ Memory-Mapped ต้องมีความเร็วสูงกว่าอุปกรณ์ที่ใช้ในการอ้างแอดเดรสแบบ I/O Mapped

สำหรับรูปที่ 2.2.10 จะแสดงตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสแบบ - Memory-Mapped ให้กับพอร์ท I/O



รูปที่ 2.2.10 ตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสแบบ Memory-Mapped

การใช้งานหน่วยความจำใน IBM/PC

สำหรับไมโครโปรเซสเซอร์เบอร์ 8088 ซึ่งใช้เป็น CPU ใน IBM/PC นี้จะมีความสามารถในการอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte และใน IBM/PC ก็ถูกออกแบบให้ใช้เนื้อที่ในหน่วยความจำเหล่านี้เพื่อการทำงานในส่วนต่าง ๆของระบบด้วย ดังนั้นในการที่จะศึกษาถึงระบบของ IBM/PC จึงจำเป็นต้องศึกษาถึงการใช้เนื้อที่ในหน่วยความจำของ IBM/PC ด้วย

หน่วยความจำภายใน IBM/PC นั้นจะแบ่งออกเป็น 2 ส่วน คือส่วนที่เป็น ROM ซึ่งอยู่ตำแหน่งแอดเดรส 64K ไบท์บน (0F0000H-0FFFFFH) และส่วนที่เป็น RAM ซึ่งเริ่มต้นจากตำแหน่งแอดเดรสแรกคือ 00000H ขึ้นมา

สำหรับหน่วยความจำส่วนที่เป็น ROM ทั้ง 64 Kbyte จะถูกใช้งานอยู่เพียง 40 Kbyte เท่านั้น โดยภายใน ROM ทั้ง 40 Kbyte นี้จะประกอบด้วยโปรแกรมในส่วนที่เป็น BIOS, Diagnostics, Cassette OpOerating system (โปรแกรมส่วนนี้ใน IBM PC/XT จะไม่มี), Bootstrap Loader และ Interpreter ของภาษา BASIC

นอกจาก ROM 40 Kbyte ที่กล่าวถึงนี้แล้ว เรายังสามารถจะเพิ่ม ROM ให้กับระบบได้อีก 8 Kbyte โดยเสียบเพิ่มเข้าไปในซอกเก็ต (Socket) ที่เตรียมไว้บนเมนบอร์ด สำหรับแอดเดรสของหน่วยความจำในส่วนที่เหลืออยู่อีก 16 Kbyte -

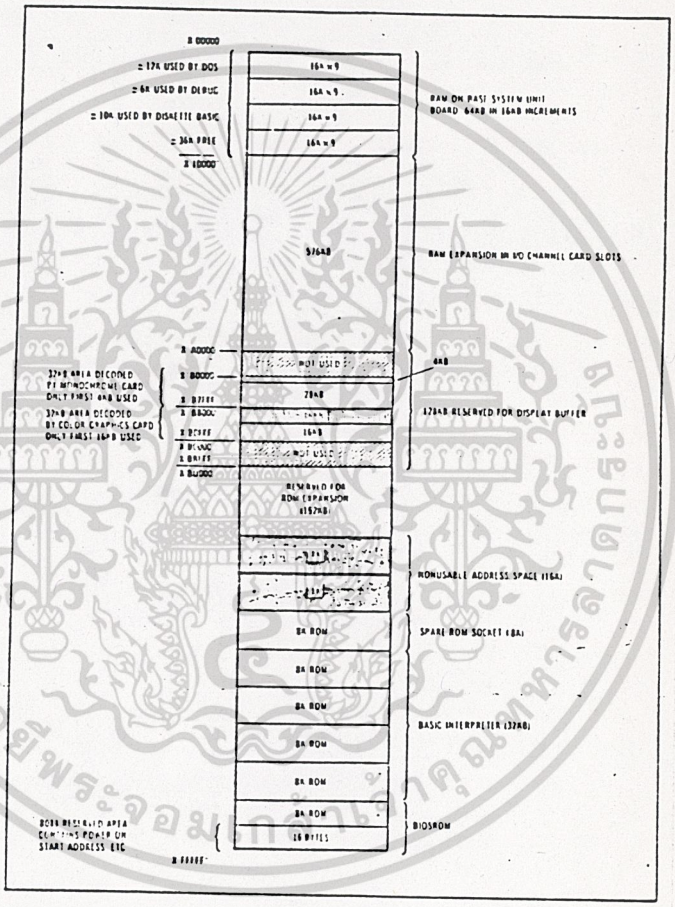
(64K-40K-8K=16K) นั้นเป็นส่วนที่ถูกดีโค้ดไว้แล้ว แต่ไม่ถูกใช้งานใด ๆ ทั้งสิ้น(คือเป็นแอดเดรสส่วนที่สูญเปล่า) ในส่วนของแอดเดรสที่เป็นหน่วยความจำ ROM ทั้ง64 Kbyte นี้ เป็นส่วนที่ถูกดีโค้ดไว้เพื่อใช้งานบนเมนบอร์ดเท่านั้นไม่สามารถนำไปใช้งานบนสล็อตของ IBM/PC ได้

หน่วยความจำในอีกส่วนหนึ่งซึ่งเป็นส่วนที่เป็น ROM นั้นจะเริ่มต้นจากแอดเดรส 00000H โดยแอดเดรสในส่วนของ 64 Kbyte แรก (00000H-0FFFFH: ในกรณีของ IBM/PC/XT จะเป็น 256 Kbyte)จะถูกดีโค้ดบนเมนบอร์ด นอกจากนั้นจะเป็นส่วนของ RAM ที่ถูกเพิ่มเติมขึ้นโดยคาร์ดต่าง ๆ บนสล็อตของ IBM/PC

สำหรับการใช้งานหน่วยความจำ 64 Kbyte แรกนี้จะแบ่งออกเป็นส่วน ๆเช่นเดียวกับในหน่วยความจำส่วนที่เป็น ROM โดยหน่วยความจำ 12 Kbyte แรกจะใช้สำหรับเก็บโปรแกรมส่วนที่เป็น DOS เมื่อมีการบูท DOS เข้ามาในระบบ และเมื่อมีการโหลดโปรแกรม DEBUG เข้ามาในระบบ IBM/PC ก็จะใช้เนื้อที่ในอีก 64ม Kbyteต่อมาในการเก็บโปรแกรม DEBUG นี้ และในกรณีที่มีการโหลดโปรแกรมที่เป็น Interpreter ของ Advance BASIC เข้ามาในระบบด้วย IBM/PC ก็จะใช้เนื้อที่ในหน่วยความจำอีก - 10 Kbyte ในการเก็บส่วนของโปรแกรมนี้ ดังนั้นในส่วนของหน่วยความจำ 64 Kbyte แรกนี้จะเหลือเนื้อที่สำหรับใช้งานอยู่อีก 36 Kbyte อย่างไรก็ตามเนื้อที่ว่างที่เหลืออยู่นี้จะเปลี่ยนแปลงไปตามเวอร์ชัน (Version) ของซอฟท์แวร์ต่าง ๆ ที่ใช้ในระบบ

นอกจากหน่วยความจำทั้ง 2 ส่วนที่กล่าวถึงแล้วยังมีหน่วยความจำในส่วนอื่น ๆ อีกโดยเฉพาะในส่วนที่เกี่ยวกับการแสดงผลออกทางจอภาพหน่วยความจำในส่วนนี้เป็นส่วนที่เก็บข้อมูล ซึ่งใช้ในการแสดงผลออกทางจอภาพ โดยทั่วไปจะเรียกหน่วยความจำนี้ว่าเป็น Display Buffer หน่วยความจำในส่วนนี้จะเป็นหน่วยความจำ RAM ที่ถูกดีโค้ด และใช้งานบนคาร์ดที่ทำหน้าที่ในการแสดงผลออกบนจอภาพ (Display Card) ซึ่งจะมีอยู่ 2 ลักษณะ คือ คาร์ดที่แสดงผลบนจอภาพสี (Color Monitor) หรือที่เรียกโดยทั่วไปว่า Color Graphic Cardซึ่งจะใช้หน่วยความจำขนาด 16 Kbyte เป็นDisplay Buffer (จากที่เตรียมไว้ 32 Kbyte สำหรับ 16 Kbyte ที่เหลือจะไม่ถูกใช้งาน) ส่วนคาร์ดอีกลักษณะหนึ่งจะเป็นคาร์ดที่แสดงผลออกบนจอ Monochrome หรือ ที่เรียกโดยทั่วไปว่า Monochrome display Card ซึ่งจะใช้เนื้อที่ในหน่วยความจำเพื่อใช้เป็น Dissplay-Buffer เพียง 4K เท่านั้น(จากที่เตรียมไว้ถึง 32 Kbyte โดยแอดเดรสของหน่วยความจำที่เหลืออีก 28 Kbyte นั้นจะไม่ถูกใช้งานใด) อย่างไรก็ตามเนื่องจากคาร์ดที่ใช้ในการแสดงผลนี้ โดยทั่วไปจะเลือกใช้เพียงชนิดเดียวเท่านั้น ดังนั้นเมื่อเราเลือกใช้คาร์ดแสดงผลแบบหนึ่งในระบบเราก็จะสามารถใช้แอดเดรสของหน่วยความจำที่เป็น Display -

Buffer ของอีกคาร์ดหนึ่งเพื่อนำไปใช้ในงานอื่น ๆได้นอกจากแอดเดรสของหน่วยความจำที่จัดเตรียมไว้สำหรับใช้เป็น Display Buffer แล้ว ใน IBM/PC ยังกัน (Reserve) เนื้อที่อีกส่วนหนึ่งไว้ด้วย เพื่อเตรียมไว้ให้กับอุปกรณ์หรือคาร์ดอื่น ๆ ที่อาจจะถูกสร้างขึ้นในอนาคต ดังนั้นจึงควรที่จะหลีกเลี่ยงการใช้หน่วยความจำในส่วนนี้ไว้ด้วยทั้งนี้ก็เพื่อให้ระบบของเรานั้นสามารถใช้งานอุปกรณ์ หรือ คาร์ดที่ถูกสร้างขึ้นใหม่เหล่านี้ได้ สำหรับรูปที่ 2.2.11 นี้จะแสดงถึงแผนผังการใช้งานหน่วยความจำต่าง ๆภายในระบบของ IBM/PC



รูปที่ 2.2.11 แผนผังการใช้งานหน่วยความจำใน IBM/PC

การใช้งานหน่วยความจำใน IBM PC/XT

ภายใน IBM PC/XT จะมีการใช้งานหน่วยความจำในลักษณะที่คล้ายกับใน IBM PC เพียงแต่ขนาดของหน่วยความจำที่จัดไว้นั้นจะแตกต่างกัน กล่าวคือหน่วยความจำส่วนที่เป็น ROM นั้นจะมีขนาด 64 Kbyte(จาก 0F0000H-0FFFFFH) ซึ่งใช้สำหรับเก็บ

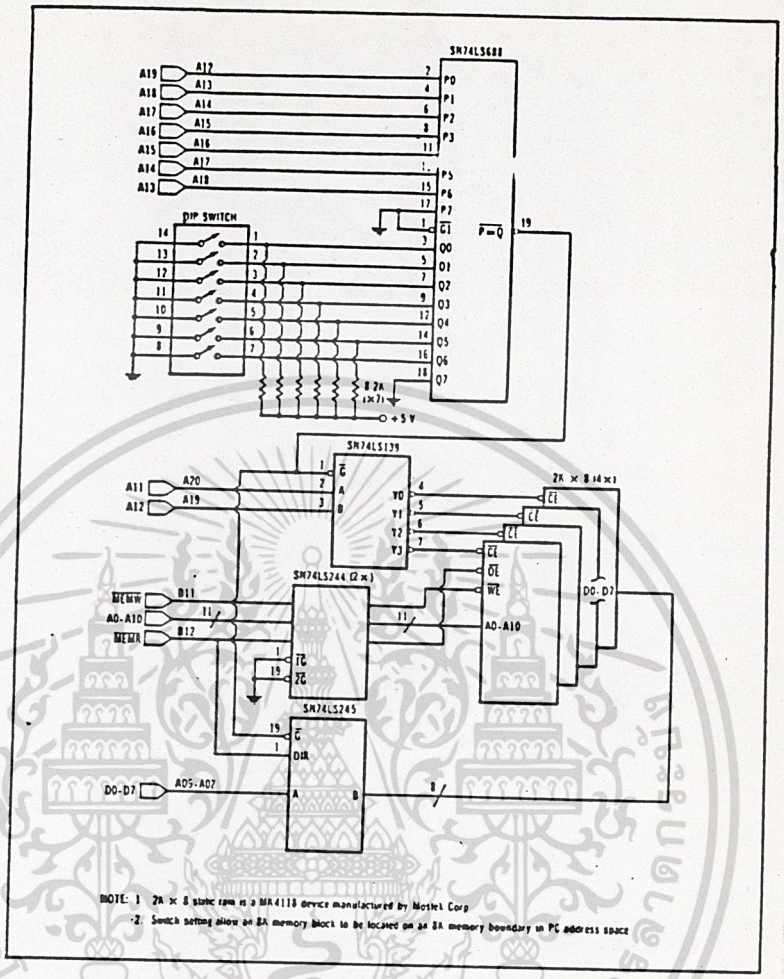
BIOS,Interpreter ของภาษา BASIC ฯลฯ เช่นเดียวกับใน IBM PC นอกจากนี้ยังเตรียมเนื้อที่ในหน่วยความจำอีก 192 KByte(จาก OC0000H-OEFFFFH)ไว้สำหรับหน่วยความจำ ROM ที่จะเพิ่มเติมขึ้นในภายหลังอีกด้วย

สำหรับหน่วยความจำ RAM บนเมนบอร์ดจะถูกเพิ่มขึ้นให้มีขนาดตั้งแต่ 128Kx9 จนถึง 256 Kx9 (บิทที่ 9 ซึ่งเกินมานั้นจะถูกใช้เป็นบิท Parity ของข้อมูลในแต่ละไบท์) โดยแอดเดรสของหน่วยความจำ RAM ในส่วนนี้จะอยู่ในช่วง 00000H-3FFFFH สำหรับในกรณีที่มีความจำเป็นต้องใช้หน่วยความจำ RAM ที่มีขนาดมากกว่า 256 KByte นี้เราก็สามารถจะทำได้โดยการเสียบคาร์ดที่ใช้ในการเพิ่มหน่วยความจำให้กับ IBM PC/XT ลงบนสล็อตบนเมนบอร์ดของ IBM PC/XT ซึ่งหน่วยความจำที่สามารถจะเพิ่มเข้าไปในระบบนี้ จะมีขนาด 384 KByte คือตั้งแต่แอดเดรส 40000H-9FFFFH (ภายในคาร์ดบางรุ่นอาจสามารถเพิ่มจำนวนหน่วยความจำได้มากกว่า 384 KB(Bytes)

นอกจากนี้ยังมีหน่วยความจำ RAM อีกส่วนหนึ่งซึ่งเตรียมไว้สำหรับการแสดงผล คือใช้เป็น Display Buffer อีก 128 KByte จากแอดเดรส 0A0000H-0BFFFFH แต่จะถูกจัดไว้สำหรับการแสดงผลบน Monochrome Card และColor Graphics Card เพียงคาร์ดละ 32KByteเท่านั้น Monochrome Cardจะจัดไว้ตั้งแต่ 0B8000H-0BFFFFH แต่จะถูกใช้งานจริงเพียง 16 KByte คือจาก 0B8000H จนถึง 0BBFFFH เท่านั้น

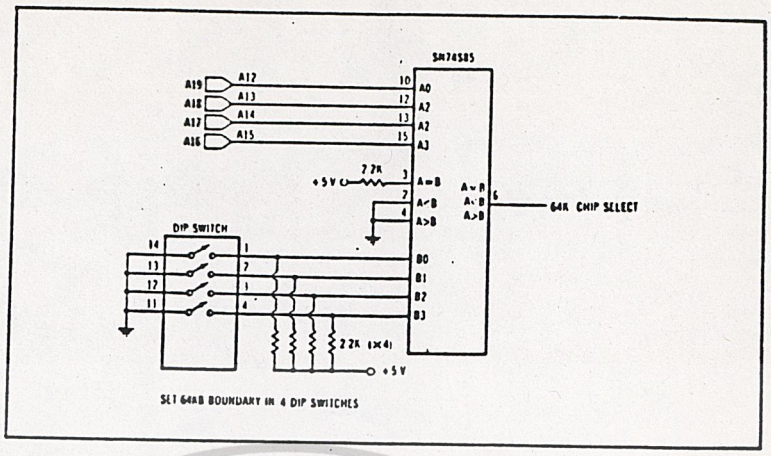
การดีโค้ดแอดเดรสของหน่วยความจำ

การดีโค้ดแอดเดรสของหน่วยความจำนี้จะมีลักษณะที่คล้ายคลึงกับวิธีการดีโค้ดแอดเดรสของพอร์ท I/O มาก อย่างไรก็ตามการดีโค้ดแอดเดรสของหน่วยความจำยังมีข้อแตกต่างจากการดีโค้ดแอดเดรสของพอร์ท I/O อยู่บางประการ กล่าวคือ การดีโค้ดแอดเดรสของหน่วยความจำนี้ต้องทำการดีโค้ดแอดเดรสทั้ง 20 บิท เพื่อให้สามารถอ้างแอดเดรสได้ครบทั้ง 1 Mbyteและการดีโค้ดแอดเดรสของหน่วยความจำไม่ใช้สัญญาณ AEN ในการควบคุมวงจรดีโค้ดเหมือนกับในการดีโค้ดแอดเดรสของพอร์ท I/O สำหรับวงจรที่ใช้ในการดีโค้ดแอดเดรสของหน่วยความจำนั้น เราสามารถจะนำเอาการดีโค้ดแบบ Fixed และ แบบสวิทช์เลือกเข้ามาใช้งานได้เช่นกัน



รูปที่ 2.2.12 ตัวอย่างวงจรที่ใช้ในการดีโค้ดแอดเดรสของหน่วยความจำ

จากรูปที่ 2.2.12 นั้นจะแสดงตัวอย่างวงจรที่ทำการดีโค้ดแอดเดรสของหน่วยความจำ static RAM ขนาด 8 Kbyte ภายในวงจรนี้จะมีวงจรในส่วนที่เป็นบัฟเฟอร์รวมอยู่ด้วย ซึ่งในวงจรจะแสดงถึงการอีนาเบิล และควบคุมทิศทางการส่งผ่านข้อมูลของบัฟเฟอร์เหล่านี้ด้วย อย่างไรก็ตามวงจรดีโค้ดในลักษณะนี้จะใช้ได้ดีสำหรับการดีโค้ดหน่วยความจำที่จำนวนแอดเดรสไม่มากนัก แต่ในกรณีที่จำเป็นต้องทำการดีโค้ดบล็อกของแอดเดรสที่มีจำนวนมากเราอาจจะใช้วงจรในรูป 2.1.13 ช่วยได้โดยใช้ร่วมกับหน่วยความจำประเภท DRAM



รูปที่ 2.2.13 ตัวอย่างวงจรดีโค้ดบล็อกแอดเดรสของหน่วยความจำ

จากวงจรในรูปที่ 2.2.13 เป็นวงจรที่ทำการดีโค้ดบล็อกของแอดเดรส ของหน่วยความจำขนาด 64 Kbyte โดยตำแหน่งแอดเดรสของบล็อกของหน่วยความจำขนาด 64 Kbyteนี้อาจจะกำหนดให้อยู่ในตำแหน่งใดก็ได้โดยการเซ็ทตำแหน่งของ DIP Switch นี้ต่ออยู่กับขา B0-B3 และ 74S85

การรีเฟรชหน่วยความจำที่เป็น DRAM

ในกรณีที่มีความจำเป็นต้องใช้หน่วยความจำที่เป็น DRAM กับระบบนั้นสิ่งหนึ่งที่มีความจำเป็นต้องออกแบบให้ระบบให้ระบบทำอยู่ตลอดเวลาก็ คือ การรีเฟรหน่วยความจำ

สำหรับการรีเฟรชหน่วยความจำนี้จะมีปัญหาอยู่ 2 ประการ คือในระหว่างการรีเฟรชหน่วยความจำนั้น ค่าแอดเดรสที่ใช้ในการรีเฟรชหน่วยความจำจะต้องถูกส่งไปให้กับ DRAM ด้วย และบัสไซเคิลในการรีเฟรชหน่วยความจำจะต้องเกิดขึ้นได้ในทุก ๆช่วงเวลาที่ขบวนการ DMA ในการสร้างบัสไซเคิลในการรีเฟรหน่วยความจำประเภท DRAM นี้

สำหรับใน IBM/PC ได้ออกแบบให้มีการสร้างบัสไซเคิลที่ใช้ในการรีเฟรช-DRAM ขึ้นในทุก ๆช่วงเวลาของคล็อก 72 ลูก หรือในทุก ๆช่วงเวลาประมาณ 15 usec. ซึ่งอุปกรณ์ หรือ วงจรอื่นจะทราบว่าบัสไซเคิลนั้นเป็นบัสไซเคิลที่ใช้ในการรีเฟรช DRAM หรือไม่ ได้โดยการตรวจสอบการแอคทีฟของสัญญาณ DACKO ขอบระบบ สำหรับสัญญาณ DACKO นี้จะถูกต่อออกมายังขาของสล็อตต่าง ๆ ด้วย ซึ่งจะทำให้วงจรอินเทอร์เฟสภายนอกสามารถใช้การรีเฟรชหน่วยความจำที่ระบบสร้างขึ้นนี้ได้ด้วย ทำให้การใช้งาน DRAM เป็นไปได้โดยง่ายยิ่งขึ้น

บัสไซเคิลของการรีเฟรชหน่วยความจำนี้ จะมีลักษณะเป็นบัสไซเคิล ของการอ่านข้อมูลจากหน่วยความจำ และค่าแอดเดรสของหน่วยความจำที่ต้องการจะรีเฟรชนั้นจะถูกส่งออกมาบนบัสแอดเดรสของระบบด้วย

2.3 บัสไซเคิลของระบบ

บัสไซเคิล คือ ขบวนการของสัญญาณที่เกิดขึ้นในช่วงของการส่งผ่านข้อมูลระหว่างหน่วยความจำ, อุปกรณ์อินพุท/เอาท์พุท (Input/Output;I/O) และ ไมโครโปรเซสเซอร์ 8088 ดังนั้นในการออกแบบวงจรอินเทอร์เฟส (Interface) สำหรับเครื่อง IBM/PC ซึ่งไมโครโปรเซสเซอร์เบอร์ 8088 นั้น เราจะต้องเข้าใจถึงบัสไซเคิลของการส่งผ่านข้อมูลในลักษณะต่าง ๆของระบบเสียก่อน

สำหรับบัสไซเคิลของระบบนั้น เราสามารถจะแบ่งออกตามลักษณะของอุปกรณ์ที่สร้างบัสไซเคิลนั้นได้เป็น 2 กลุ่ม คือ

1. บัสไซเคิลที่สร้างขึ้นโดย 8088 (8088 Driver Bus Cycle):สำหรับบัสไซเคิลในกลุ่มนี้จะเป็นบัสไซเคิลที่ 8088 สร้างขึ้นในการส่งผ่านข้อมูลระหว่าง 8088 กับ I/O หรือระหว่าง 8088 กับหน่วยความจำ โดยในบัสไซเคิลเหล่านี้ 8088 จะส่งแอดเดรสของหน่วยความจำหรือพอร์ท (Port) ที่ต้องการจะติดต่อด้วย ออกมาบนแอดเดรสบัส, ส่งหรือรับข้อมูลจากบัสข้อมูล และส่งสัญญาณควบคุมที่จำเป็นออกมาบนบัสควบคุม ซึ่งบัสไซเคิลในกลุ่มนี้จะแบ่งออกเป็น 5 ชนิด คือบัสไซเคิลในการอ่านข้อมูลจากหน่วยความจำ (Memory-Read Bus Cicle), บัสไซเคิลในการเขียนข้อมูลในหน่วยความจำ (Memory-Write Bus Cyl=cle), บัสไซเคิลในการรับข้อมูลจากพอร์ท I/O (I/O Port-Read Bus Cycle), บัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท I/O (I/O Port-Write Bus Cycle) สำหรับบัสไซเคิลในการตอบรับการขออินเทอร์รัพท์นี้จะเป็นบัสไซเคิลที่เกิดเฉพาะบน Main Board ของ IBM/PC เท่านั้น โดยจะไม่ปรากฏบนสล็อต (Slot) ทั้งห้าของ IBM/PC (8 สล็อตในสกรณีของ IBM PC/XT)

2. บัสไซเคิลที่สร้างขึ้นโดย DMA Controller (DMA-Driver Bus Cycle): สำหรับบัสไซเคิลในกลุ่มนี้จะเป็นบัสไซเคิลที่ DMA Controller (ในที่นี้คือ 8237-5) สร้างขึ้นในขบวนการ DMA โดยในระหว่างขบวนการ DMA นี้ 8088 จะทำให้บัสแอดเดรส, บัสข้อมูล และบัสควบคุมบางเส้นของ 8088 มีสถานะเป็น High Impedance ซึ่งจะทำให้ 8237-5 DMA Controller สามารถควบคุมบัสต่าง ๆ เหล่านี้ได้ จากนั้น 8237-5 ก็จะสร้างบัสไซเคิลขึ้นเพื่อให้อุปกรณ์ I/O หรือ 8237-5 ก่อน ซึ่งจะทำให้ประหยัดเวลาและลดความยุ่งยากในการส่งผ่านข้อมูลลงได้) โดยในบัสไซเคิลเหล่านี้ 8237-5 จะทำหน้าที่ในการส่งแอดเดรสและสัญญาณควบคุมที่จำเป็นออกมาบนบัสแอดเดรส และบัสควบคุมแทน 8088 สำหรับบัสไซเคิลในกลุ่มนี้จะแบ่งออกเป็น 2 ชนิด คือ บัสไซเคิลในการอ่านข้อมูลจากอุปกรณ์ I/O เพื่อส่งให้กับหน่วยความจำ

(DMA Memory-Writw Bus Cycle) และบัสไซเคิลในการอ่านข้อมูลจากหน่วยความจำเพื่อส่งให้กับอุปกรณ์ I/O (DMA Memory -Read Bus Cycle)

บัสไซเคิลต่าง ๆในระบบ

ในหัวข้อนี้จะกล่าวถึงรายละเอียดของการเกิดสัญญาณในบัสไซเคิลต่าง ๆในระบบ โดยในการกล่าวถึงแต่ละบัสไซเคิลจะแสดงลักษณะลำดับการเกิดของแต่ละสัญญาณในบัสไซเคิลนั้นด้วย อย่างไรก็ตามรูปเหล่านี้จะแสดงการเกิดของแต่ละสัญญาณเท่านั้น สำหรับค่าเวลาที่แท้จริงนั้นจะแสดงไว้ในภาคผนวกท้ายเล่ม

บัสไซเคิลที่สร้างขึ้นโดย 8088

การทำงานในแต่ละบัวไซเคิลของ 8088 นั้น จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อกที่ป้อนให้กับ 8088 จำนวน 4 ลูกสำหรับใน IBM/PC คล็อกที่ป้อนให้กับ 8088 จะมีความถี่ประมาณ 4.77 MHzหรือ มีคาบเวลาของคล็อก 1ลูกประมาณ 210 nanosec. อย่างไรก็ตามในกรณีที่อุปกรณ์ที่ทำงานร่วมกับ 8088 เช่นหน่วยความจำ หรือพอร์ท I/O นั้น มีความเร็วในการทำงานต่ำไม่สามารถที่จะตอบสนองการทำงานในบัสไซเคิลของ8088 ได้ทัน อุปกรณ์นั้นก็สามารถเพิ่มช่วงเวลาในบัสไซเคิลได้ โดยการป้อนลอจิก "0" ให้กับขา READY ของ 8088 (ดูรายละเอียดเกี่ยวกับสัญญาณ READYได้ในบทที่ 10 "การสร้าง Wait State")

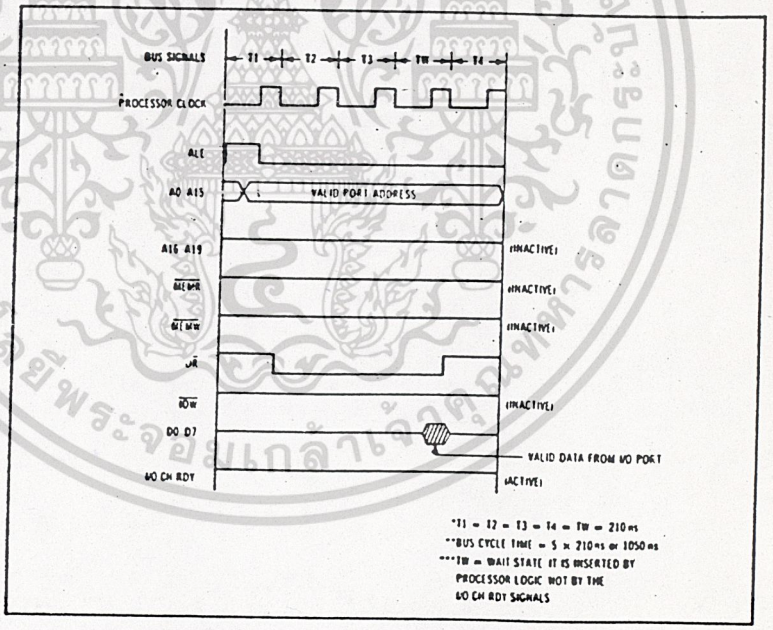
สำหรับอุปกรณ์ที่เป็นหน่วยความจำ เช่น ROM หรือ RAM นั้น โดยทั่วไปจะมีความเร็วในการทำงานสูงพอที่จะตอบสนองต่อการทำงานในบัสไซเคิลของ 8088 ได้ ดังนั้นใน IBM/PC จึงไม่มีการเพิ่มช่วงเวลาของบัสไซเคิลในการอ่าน/เขียนข้อมูลลงในหน่วยความจำ คือ ใน 1 บัสไซเคิลจะยังคงใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูกตามปกติ (840 nanosec.) แต่สำหรับอุปกรณ์ในส่วนที่เกี่ยวกับพอร์ต I/O นั้นมักจะมีความเร็วต่ำดังนั้นใน IBM/PC จึงมีการเพิ่มช่วงเวลาของบัสไซเคิลในการอ่าน/เขียนข้อมูลในพอร์ท I/O ขึ้นโดยจะเพิ่มช่วงเวลาในบัสไซเคิลจากคล็อก 4 ลูก (840 nanosec.) ใน 1 บัสไซเคิลเป็นคล็อก 5 ลูก (1.05 usec) ใน 1 บัสไซเคิล สำหรับคล็อกที่เพิ่มเข้ามานั้นจะเรียกว่า "Tw" และ สภาวะที่ 8088 หยุดรอ เพื่อให้อุปกรณ์ภายนอกรับหรือส่งข้อมูลได้ทันนั้นก็เพื่อ "Wait state"

ในส่วนต่อไปจะกล่าวถึงบัสไซเคิลต่าง ๆที่ถูกสร้างขึ้นโดย 8088 ซึ่งจะประกอบด้วยบัสไซเคิลในการอ่านข้อมูลจากหน่วยความจำ,บัสไซเคิลในการเขียนข้อมูลบนหน่วยความจำ,บัสไซเคิลในการรับข้อมูลจากพอร์ท I/O บัสไซเคิลในการส่งข้อมูลออกทางพอร์ท

I/O และบัสไซเคิลในการตอบรับการขออินเทอร์รัพท์ โดยบัสไซเคิลที่จะกล่าวถึงนี้เป็นบัสไซเคิลที่ปรากฏบนสล็อตทั้ง 5 ของ IBM/PC (หรือ 8 สล็อตใน IBM PC/XT) ซึ่งสัญญาณต่าง ๆ จะแตกต่างจากบัสไซเคิลที่ปรากฏบนบัสของ 8088 อยู่บ้าง เช่น ขาA0-A19 และ D0-A7 ซึ่งแยกกันอยู่แทนที่จะอยู่รวมกันเหมือนบนบัสของ 8088 เป็นต้น อย่างไรก็ตามเนื่องจากบัสไซเคิลในการตอบรับการขออินเทอร์รัพท์นั้นเป็นบัสไซเคิลที่เกิดขึ้นเฉพาะในบัสของระบบเท่านั้น โดยจะไม่ปรากฏบนสล็อตทั้ง 5 ของ IBM/PC ดังนั้นในที่นี้จะไม่กล่าวถึงบัสไซเคิลในการตอบรับการขออินเทอร์รัพท์นี้

บัสไซเคิลในการอ่านข้อมูลจากพอร์ท

ในขณะที่ 8088 เอ็กซีคิวท์ (Execute)ชุดคำสั่ง IN เช่น ax,DATA ซึ่งเป็นชุดคำสั่งที่ทำให้ 8088 ทำการอ่านข้อมูลจากพอร์ที่ในส่วนของโอเปอแรนด์ (Operand) นั้น 8088 จะสร้างบัสไซเคิลในการอ่านข้อมูลจากพอร์ท เพื่อให้พอร์ทที่ถูกกำหนดนั้นส่งข้อมูลออกมาบนบัสข้อมูล สำหรับขบวนการของสัญญาณที่เกิดขึ้นในบัสไซเคิลนี้สามารถแสดงได้ดังรูปที่ 2.3.1



รูปที่ 2.3.1 บัสไซเคิลของการอ่านข้อมูลจากพอร์ท I/O

จากรูป 2.3.1 บัสไซเคิลนี้จะเริ่มต้นในช่วงของคล็อก T1 ซึ่งเป็นช่วงเวลาที่สัญญาณ ALE แอคทีฟ(ลอจิก "1") สัญญาณ ALE นี้จะถูกใช้เพื่อแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าข้อมูลที่อยู่บนบัสแอดเดรสในช่วงขอบขาลงของสัญญาณ ALE นั้น เป็นแอดเดรสของพอร์ที่ 8088 ต้องการจะติดต่อด้วย (ในที่นี้คือ แอดเดรสของพอร์ท

ที่ 8088 ต้องการอ่านข้อมูล)

สำหรับใน 8088 สามารถอ้างแอดเดรสของพอร์ทได้เพียง 64 K พอร์ทเท่านั้นในขณะที่สามารถอ้างแอดเดรสของหน่วยความจำได้ถึง 1 Mbyte ดังนั้นในกรณีของบัสไซเคิลที่เกี่ยวกับการอ้างแอดเดรสของพอร์ท 8088 จะใช้เส้นแอดเดรสเพียง 16 บิท คือ A0-A15 เท่านั้น (ในขณะที่การอ้างแอดเดรสของหน่วยความจำจะใช้เส้นแอดเดรสทั้ง 20 เส้น คือ A0-A19)

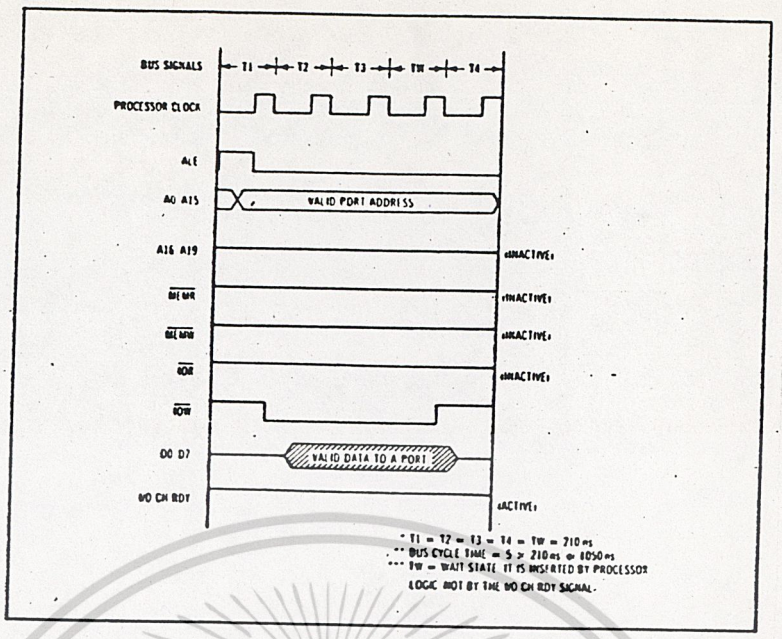
หลังจากนั้นในช่วงของคล็อก T2 สัญญาณ IOR จะแอคทีฟ (ลอจิก "0") ซึ่งเป็นการแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าบัสไซเคิลในการอ่านข้อมูลจากพอร์ท(I/O-Port Read Bus Cyxle) และเป็นการทำให้พอร์ทที่มีแอดเดรสตรงกับค่าแอดเดรสที่อยู่บนบัสแอดเดรสนั้นส่งข้อมูลออกมาบนบัสข้อมูล เมื่อพอร์ทที่ถูกอ้างถึงโดยแอดเดรสบนบัสข้อมูลทำการส่งข้อมูลออกมาบนบัสข้อมูลแล้ว 8088 จะอ่านข้อมูลนั้นในช่วงเริ่มต้นของคล็อก T4 จากนั้นสัญญาณ IOR จะถูกปรับให้เป็นลอจิก "1" และจะสิ้นสุดการทำงานในบัสไซเคิลเมื่อสิ้นสุดช่วงเวลาของคล็อก T4

จะเห็นได้ว่าโดยปกติแล้วบัสไซเคิลนี้จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อกจำนวน 4 ลูก เช่นเดียวกับบัสไซเคิลในการอ่าน/เขียนข้อมูลลงในหน่วยความจำ แต่ภายใน IBM/PC นั้นจะเพิ่มช่วงเวลา (TW)ในบัสไซเคิลนี้ขึ้นอีก 1 ลูก ทำให้ช่วงเวลาในบัสไซเคิลเพิ่มขึ้นเป็น 1.05 usec. โดย Tw นี้จะถูกเพิ่มเข้าไปในระหว่างช่วงต่อของคล็อก T3 และ T4 เพื่อให้พอร์ท I/O ซึ่งปกติมักจะมีความเร็วในการทำงานต่ำ สามารถที่จะส่งข้อมูลออกมาบนบัสข้อมูลได้ทัน อย่างไรก็ตามในกรณีที่เราต้องการจะเพิ่มช่วงเวลาในบัสไซเคิลนี้ขึ้นอีก (เพิ่มจำนวน Tw) ก็สามารถทำได้โดยการป้อนลอจิก "0" ให้กับขา I/O CHARDY บนสล็อตของ IBM/PC

บัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท

ในขณะที่ 8088 เอ็กซีคิวท์ชุดคำสั่ง OUT เช่น OUT DATA,as ซึ่งเป็นชุดคำสั่งที่ให้ 8088 ทำการเขียนข้อมูลลงบนพอร์ทที่กำหนดในส่วนของโอเปอร์แรนด์นั้น 8088 จะสร้างบัสไซเคิลในการเขียนข้อมูลบนพอร์ท เพื่อให้พอร์ทที่ถูกกำหนดทำการรับข้อมูลที่อยู่บนบัสข้อมูล

สำหรับขบวนการของสัญญาณต่าง ๆในบัสไซเคิลนี้จะแสดงดังรูปที่ 2.3.2



รูปที่ 2.3.2 บัสไซเคิลของการเขียนข้อมูลลงบนพอร์ท I/O

จากรูป 2.3.2 บัสไซเคิลนี้จะเริ่มต้นในช่วงของคล็อก T1 ซึ่งเป็นช่วงเวลาที่สัญญาณ ALE แอคทีฟ (ลอจิก"1") สัญญาณ ALE นี้จะถูกใช้เพื่อแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าข้อมูลที่อยู่บนบัสแอดเดรสในช่วงขอบขาลงของสัญญาณ ALE นั้นเป็นแอดเดรสของพอร์ทที่ 8088 ต้องการจะติดต่อด้วย (ในที่นี้คือแอดเดรสของพอร์ทที่ 8088 ต้องการส่งข้อมูลให้) สำหรับเส้นแอดเดรสที่ใช้ในบัสไซเคิลนี้จะมีจำนวน 16 เส้นเท่ากับในกรณีของบัสไซเคิลในการอ่านข้อมูลจากพอร์ท คือใช้เส้นแอดเดรส A0-A15 ในการอ้างแอดเดรสของพอร์ทนั่นเอง

หลังจากนั้นในช่วงคล็อก T2 สัญญาณ LOW จะแอคทีฟ (ลอจิก "0") ซึ่งเป็นการแสดงให้อุปกรณ์ที่ทำงานร่วมกับ 8088 ทราบว่าบัสไซเคิลนี้เป็นบัสไซเคิลในการเขียนข้อมูลลงบนพอร์ท (I/O-Port Write Bus Cucle) จากนั้น 8088 จะทำการส่งข้อมูลที่ต้องการจะส่งให้กับพอร์ทที่กำหนดนั้นออกมาบนบัสข้อมูล ในช่วงของคล็อก T4 สัญญาณ LOW จะถูกปรับให้เป็นลอจิก "1" และจะสิ้นสุดการทำงานในบัสไซเคิลเมื่อสิ้นสุดช่วงเวลาของคล็อก T4

สำหรับในกรณีของบัสไซเคิลนี้ IBM/PC ก็จะทำการเพิ่ม Tw เข้าไประหว่างช่วงต่อของคล็อก T3 และ T4 เช่นเดียวกับกรณีของบัสไซเคิลในการอ่านข้อมูลจากพอร์ทเพื่อให้พอร์ท I/O สามารถที่จะทำงานในบัสไซเคิลนี้ได้ทัน อย่างไรก็ตามถ้าต้องการจะเพิ่มช่วงเวลาในบัสไซเคิลนี้ขึ้นอีก ก็สามารถจะทำได้โดยการป้อนลอจิก "0" ให้กับขา I/O CHRDY บนสล็อตของ IBM/PC

บัสไซเคิลที่สร้างขึ้นโดย DMA-Controller

สำหรับบัสไซเคิลที่จะกล่าวถึงต่อไปนี้ จะแตกต่างจากบัสไซเคิลที่ได้กล่าวถึงในหัวข้อที่ผ่านมา เนื่องจากบัสไซเคิลที่จะกล่าวถึงนี้เป็นบัสไซเคิลที่ถูกสร้างขึ้นโดย DMA-Controller แทนที่จะถูกสร้างขึ้นโดย 8088 กล่าวคือในช่วงเวลาของบัสไซเคิลเหล่านี้ DMA-Controller จะทำการควบคุมบัสต่าง ๆของระบบแทน 8088 (ในช่วงเวลานี้ 8088 จะทำให้บัสแอดเดรส,ข้อมูล และบัสควบคุมบางเส้นมีสถานะ High Impedance)

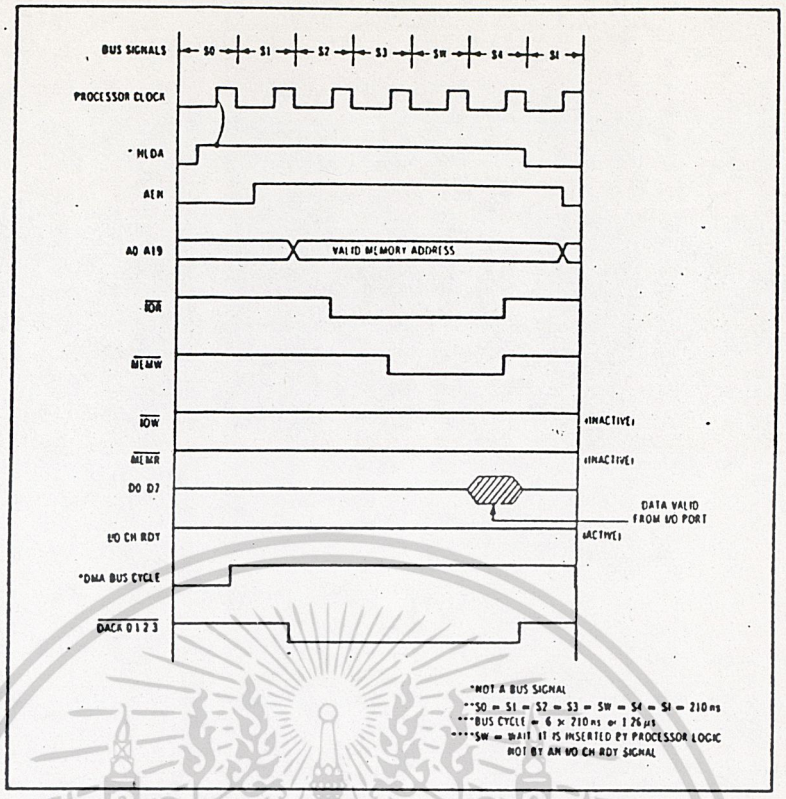
ในหัวข้อนี้จะกล่าวถึงเฉพาะส่วนที่เกี่ยวกับสัญญาณต่าง ๆในบัสไซเคิลในการ - เขียนข้อมูลลงในหน่วยความจำ, บัสไซเคิลในการอ่านข้อมูลจากหน่วยความจำ และบัสไซเคิลในการรีเฟรสหน่วยความจำโดยใช้ขบวนการ DNA เท่านั้น สำหรับการทำงานในขบวนการ DMA และการใช้งาน DMA-Controller นั้นจะกล่าวถึงโดยละเอียดในบทที่ 7

สำหรับการขอใช้ขบวนการ DMA ของพอร์ทหรืออุปกรณ์ภายนอกนั้น สามารถที่จะทำได้โดยการป้อนสัญญาณลอจิก "1" (High) เข้าที่ขา DRQ2 หรือ DRQ3 ที่อยู่บนสล็อตของ IBM/PC ซึ่งหลังจากที่ DMA-Controller ได้รับสัญญาณนี้แล้วก็จะตอบรับโดยการทำให้สัญญาณ AEN แอคทีฟ (ลอจิก"1") จากนั้น DMA Controllerจะทำให้สัญญาณ DACK1, DACK2 หรือ DACK3 แอคทีฟ ซึ่งสัญญาณใดจะแอคทีฟนั้นขึ้นอยู่กับแชนแนลของสัญญาณที่ขอ DMA เช่นกรณีที่อุปกรณ์ภายนอกขอ DMA โดยผ่านทาง DRQ1 DMA-Controller ก็จะตอบรับโดยการทำให้ DACK1 แอคทีฟ เป็นต้น เมื่อสัญญาณ AEN และ DACK นี้แอคทีฟก็จะเป็นการแสดงให้อุปกรณ์ภายนอกที่ทำการขอ DMA ทราบว่าบัสไซเคิลของระบบในขณะนั้นเป็นบัสไซเคิลที่ DMA-Controller สร้างขึ้นเพื่อตอบสนองต่อการขอ DMA จากนั้นอุปกรณ์ภายนอกก็สามารถรับ หรือ ส่งข้อมูลให้กับหน่วยความจำได้โดยขึ้นกับบัสไซเคิลที่ DMA-Controller สร้างขึ้น

บัสไซเคิลในการเขียนข้อมูลลงในหน่วยความจำโดยใช้ DMA

บัสไซเคิลนี้ DMA-Controller จะสร้างขึ้นเพื่อให้อุปกรณ์ภายนอก (พอร์ท) การขอใช้ขบวนการ DMA นั้น สามารถที่จะนำเอาข้อมูลไปเก็บในหน่วยความจำได้โดยตรง (คือไม่ต้องผ่านทาง 8088)

สำหรับขบวนการของสัญญาณที่เกิดขึ้นในบัสไซเคิลนี้ สามารถที่จะแสดงได้ดังรูปที่ 2.3.3



รูปที่ 2.3.3 บัสไซเคิลของขบวนการ DMA ในการเขียนข้อมูลลงในหน่วยความจำ

จากรูป2.3.3 สัญญาณคล็อกที่ป้อนให้กับ DMA-Controller นั้นจะถูกสร้างขึ้นมาจากสัญญาณคล็อกที่ป้อนให้กับ 8088 สำหรับสัญญลักษณ์ที่ใช้แทนคล็อกที่ป้อนให้กับ DMA-Controller จะแตกต่างจากสัญญลักาณ์ที่ใช้กับคล็อก 8088 ทั้งนี้ก็เพื่อให้ทราบว่าบัสไซเคิลที่กล่าวถึงนั้น เป็นบัสไซเคิลที่สร้างขึ้นโดย 8088 หรือ DMA-CXontroller โดยสัญญลักษณ์ "T" จะใช้กับคล็อกในบัสไซเคิลขอ 8088 ส่วนสัญญลักษณ์ "S" จะใช้กับคล็อกในบัสไซเคิลของ DN+MA-Controller

ในขณะที่ DMA-Controller ไม่ได้ทำการตอบสนองต่อการขอ DMA และยังไม่มีการขอ DMA จากอุปกรณ์ภายนอกนั้น DMA-Controller จะทำงานอยู่ในสภาวะ "Idle"(สภาวะนี้จะเป็นช่วงที่แทนด้วยคล็อก S1) ซึ่งในสภาวะนี้ DMA-Controller จะคอยตรวจสอบว่ามีการขอ DMA จากอุปกรณ์ภายนอกหรือไม่ ถ้าหากพบว่ามีการขอ DMA-Controller ก็จะส่งสัญญาณให้กับ 8088 เพื่อขอใช้บัส (ในทางปฏิบัติ IBM/PC จะออกแบบให้สัญญาณขอใช้บัสนี้ถูกส่งให้กับวงจรสร้าง WAIT Stats แทนที่จะส่งให้กับ 8088 โดยตรง; จะกล่าวโดยละเอียดในบทที่ 7 "การจัด DMA ของระบบ" และจะเข้าสู่สภาวะของคล็อก S0 ซึ่งเป็นสภาวะที่ DMA-Controller คอยตรวจสอบสัญญาณตอบรับการขอใช้บัสหรือ HLDA (Hold Acknowledge; สัญญาณนี้จะไม่ปรากฏบนสล็อตของ IBM/PC) จากนั้นเมื่อ DMA-Controller ได้รับสัญญาณ HLDA ก็จะนำเข้าสู่สภาวะ

คล็อก S1 (ในกรณีที่DMA-Controllerยังไม่ได้รับสัญญาณ HLDA ก็ยังคงอยู่ในสภาวะ SO ต่อไปดังนั้นสภาวะ SO นั้นอาจจะมีช่วงเวลามากกว่าช่วงเวลาของคล็อก 1 ลูกก็ได้)

เมื่อเข้สู่สภาวะของคล็อก S1 DMA-Controller ก็จะทำให้สัญญาณ AEN แอคทีฟ(ลอจิก"1") จากนั้น DM<A-Controller ก็จะส่งทำให้สัญญาณ DACK ของแชนแนลที่ขอ DMA นั้นแอคทีฟตาม โดยที่สัญญาณ AEN และ DACK นี้จะแอคทีฟจนสิ้นสุดขบวนการในบัสไซเคิล ในช่วงของคล็อก S2 DMA-Controller และ วงจรที่ทำงานร่วมกับ DMA-Controller ก็จะส่งค่าแอดเดรสของหน่วยความจำที่ต้องการใช้ในการเก็บข้อมูลจากอุปกรณ์ภายนอกที่ขอ DMA ออกมาบนบัสแอดเดรส หลังจากนั้นสัญญาณ IOR ก็จะแอคทีฟ (ลอจิก"0") เพื่อให้อุปกรณ์ภายนอกทำการขอ DMA นั้นส่งข้อมูลออกมาบนบัสข้อมูล จากนั้นสัญญาณ MEMW ก็จะถูกทำให้แอคทีฟตาม เพื่อให้หน่วยความจำที่มีแอดเดรสตรงกับค่าแอดเดรสที่ DMA-Controller ส่งออกมาบนบัสแอดเดรสนั้น ทำการอ่านข้อมูลบนบัสข้อมูลไปเก็บไว้ สำหรับในช่วงเวลาของคล็อก S4 สัญญาณ IOR และ MEMW ถูกปรับให้กับเป็นลอจิก "1" อีกครั้ง

สิ่งสำคัญสิ่งหนึ่งที่ต้องคำสิงถึงในการออกแบบวงจรอินเทอร์แฟส(Interface) ที่ใช้ขบวนการ DMA ก็คือ บัสข้อมูลที่ใช้เชื่อมระหว่างหน่วยความจำบน IBM/PC กับวงจรอินเทอร์เฟสที่เราออกแบบนั้นจะไม่มีบัฟเฟอร์ (Buffer) คั่นอยู่ ดังนั้นวงจรอินเทอร์เฟสที่เราออกแบบจะต้องสามารถแลกช์(Latch)ข้อมูลที่ส่งออกมาบนบัสข้อมูลไว้ได้จนกว่าหน่วยความจำจะทำการอ่านข้อมูลนั้นไปเก็บไว้ได้

จะเห็นได้ว่า ภายในแต่ละบัสไซเคิลนั้นจะใช้เวลาเท่ากับช่วงเวลาของคล็อกประมาณ 5 ลูก(ในที่นี้สมมติให้สภาวะ SO ใช้คีล็อกเพียงลูกเดียว) คือ SO-S4 ดังนั้นใน 1 บัสไซเคิลจะใช้ช่วงเวลาประมาณ 5x210 nanosec.หรือ 1.05 usec. แต่สำหรับใน IBM/PC นั้นจะเพิ่ม Wait State(Sw) ไว้ระหว่างคล็อก S3 และ S4 ด้วย เพื่อให้อุปกรณ์ภายนอกสามารถส่งข้อมูลออกมาได้ทัน ดังนั้นใน 1 บัสไซเคิลจึงประกอบด้วยคล็อกประมาณ 6 ลูก คือ SO-S3, Sw และ S4 หรือ ประมาณม 6x210 nanosec. หรือประมาณ 1.26 usec.(สำหรับการเพิ่มสภาวะ Wait ในบัสไซเคิลเพิ่มสภาวะ Waitในบัสไซเคิลนี้จะกล่าวถึงโดยละเอียดในบทที่ 10)

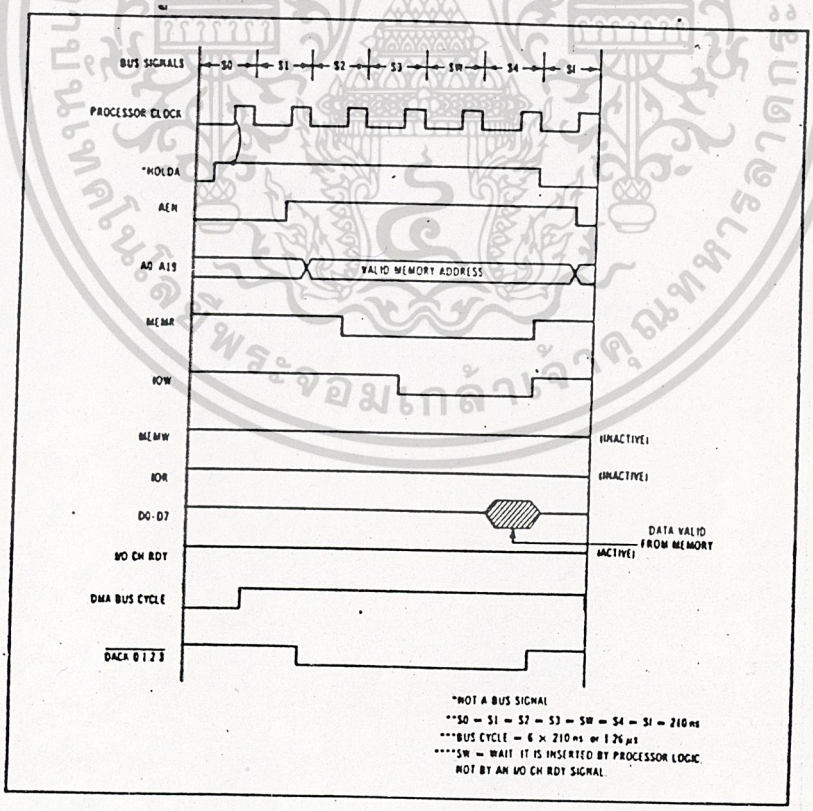
บัสไซเคิลในการอ่านข้อมูลจากหน่วยความจำโดยใช้ DMA

บัสไซเคิลนี้ DMA-Controller จะสร้างขึ้นเพื่อให้อุปกรณ์ภายนอก (พอร์ท) ที่ขอใช้ขบวนการ DMA สามารถที่จะอ่านข้อมูลจากหน่วยความจำได้โดยตรง

หลังจากเริ่มต้นบัสไซเคิลนี้ (สภาวะ "SO") และ 8088 ตอบรับการขอใช้บัส

(HLDA) แล้ว DMA-Controllerก็จะเริ่มเข้าสู่สภาวะของคล็อก S1โดยการทำให้สัญญาณ AEN แอคทีฟ (ลอจิก "1"สัญญาณนี้จะแอคทีฟจนสิ้นสุดบัสไซเคิล) และ DMA-Controller จะส่งแอดเดรสของหน่วยความจำที่อุปกรณ์ภายนอกต้องการอ่านข้อมูลออกมาบนบัสแอดเดรส (ค่าแอดเดรสนี้สามารถกำหนดได้ในระหว่างโปรแกรมสั่งงาน DMA-Controller; ขอให้ดูรายละเอียดในบทที่ 7 "การจัด DMA ของระบบ")จากนั้นสัญญาณ MEMR จะถูกทำให้แอคทีฟ (ลอจิก "0") เพื่อให้หน่วยความจำที่แอดเดรสตรงกับค่าแอดเดรสที่อยู่บนบัสแอดเดรสนั้นส่งข้อมูลออกมบนบัสข้อมูล หลังจากนั้น DMA-Controller จะทำให้สัญญาณ LOW แอคทีฟตาม เพื่อให้อุปกรณ์ภายนอกทำการอ่านข้อมูลที่อยู่บนบัสข้อมูล สำหรับในบัสไซเคิลนี้จะใช้ช่วงเวลาเท่ากับบัสไซเคิลในหัวข้อที่ผ่านมา คือ จะใช้ช่วงเวลาเท่ากับช่วงเวลาของคล็อก 6 ลูก คือ S0-S3, Sw และ S4 (สมมติให้สภาวะ S0 ใช้คล็อกเพียงลูกเดียว) หรือ ใช้เวลาประมาณ 6X210 nanosec. หรือ 1.26 usec.

สำหรับไดอะแกรมเวลาของสญญาณต่าง ๆ ในบัสไซเคิลของการอ่านข้อมูลจากหน่วยความจำนั้นจะแสดงในรูป 2.3.4



รูปที่ 2.3.4 ไดอะแกรมเวลาของขบวนการ DMA ในการอ่านข้อมูลจากหน่วยความจำบัส

บัสไซเคิลในการรีเฟรสหน่วยความจำโดยใช้ DMA

ภายในระบบที่ใช้หน่วยความจำ RAM ที่เป็นหน่วยความจำประเภทไดนามิค RAM(Dynamic Ran)หรือDRAMนั้นจำเป็นต้องใช้วงจรที่ทำหน้าที่ในการรีเฟรช(Refresh) หน่วยความจำในทุกช่วงเวลาหนึ่งภายใน IBM/PC ซึ่งในหน่วยความจำประเภท DRAM ก็จะต้องกทำการรีเฟรชหน่วยความจำเหล่านี้เช่นกัน โดยจะใช้วงจรที่ทำการขอ DMA ผ่านทาง DRQ0 (ภายใน IBM/PCจะใช้ DRQ0 ซึ่งมีลำดับความสำคัญในการขอ DMA สูงสุดในขบวนการรีเฟรชหน่วยความจำส่วนวงจรภายนอกจะขอ DMA ได้โดยผ่านทาง DRQ1-DRQ3 เท่านั้น) ในช่วงเวลาทุก ๆ72 คล็อกหรือทุก ๆ72x210 nanosex.หรือ 15.12 usec. สำหรับบัสไซเคิลที่ใช้ขบวนการรีเฟรชนี้ จะเป็นบัสไซเคิลในการอ่าข้อมูลจากหน่วยความจำโดยใช้ DMA โดยที่ DMA-Controllerจะถูกโปรแกรมให้เพิ่มค่าแอดเดรสของหน่วยความจำที่ส่งออกมาบนบัสแอดเดรสขึ้นหนึ่ง ในทุก ๆบัสไซเคิลที่ใช้ในขบวนการรีเฟรช ซึ่งก็คือทุกบัสไซเคิลที่ตอบสนองต่อการขอ DMA ที่ผ่านทาง DRQ0 นั่นเอง ด้วยวิธีการเช่นนี้ก็จะทำให้หน่วยความจำเป็น DRAM บน IBM/PCนั้น ถูกรีเฟรชได้อย่างต่อเนื่องในช่วงเวลาทุก 72 คล็อก อย่างไรก็ตามเนื่องจากในการรีเฟรชหน่วยความจำนี้เราไม่ต้องการที่จะอ่านข้อมูลใด ๆจากหน่วยความจำ ดังนั้นจึงไม่จำเป็นต้องสร้าง Wait State หรือ Swขึ้นในบัสไซเคิลนี้ทำให้ภายในบัสไซเคิลนี้ใช้เวลาเท่ากับช่วงเวลาของคล็อก 5 ลูก คือ S0-S4 หรือ 14.05 usec. เท่านั้น ดังนั้นจึงสรุปได้ว่าการทำงานของระบบในทุก ๆ 72คล็อกจะต้องใช้ช่วงเวลาของคล็อก 5 ลูกในการรีเฟรชหน่วยความจำซึ่งก็คือต้องใช้เวลาในการรีเฟรชหน่วยความจำประมาณม 7% ของเวลาทำงานทั้งหมด

2.4 การสร้าง Wait State

ในการออกแบบระบบที่เชื่อมต่อกับเครื่องคอมพิวเตอร์นั้น เรามักจะพบปัญหาเกี่ยวกับความเร็วในการทำงานของวงจรอินเทอร์เฟส ซึ่งโดยทั่วไปจะช้ากว่าการทำงานของเครื่องคอมพิวเตอร์ สำหรับใน IBM/PC ก็เช่นกับวงจรอินเทอร์เฟสที่สร้างขึ้นอาจจะทำงานได้ช้าเกินกว่าที่จะตอบสนองต่อบัสไซเคิลต่าง ๆ IBM/PC ได้ ด้วยเหตุนี้ภายใน IBM/PCจึงถูกออกแบบให้มีขาสัญญาณอินพุทเพื่อให้วงจรหรืออุปกรณ์ภายนอกที่ทำงานช้าเกินกว่าจะตอบสนองต่อบัสไซเคิลบน IBM/PC ได้นั้นใช้ในการหน่วงเวลาในบัสไซเคิลให้ช้าลง สำหรับสัญญาณอินพุทดังกล่าวนี้ คือ สัญญาณ "READY" (หรือ I/O CHRDY) ซึ่งจะถูกต่อออกมาบนขาของสล็อตของ IBM/PC ด้วย

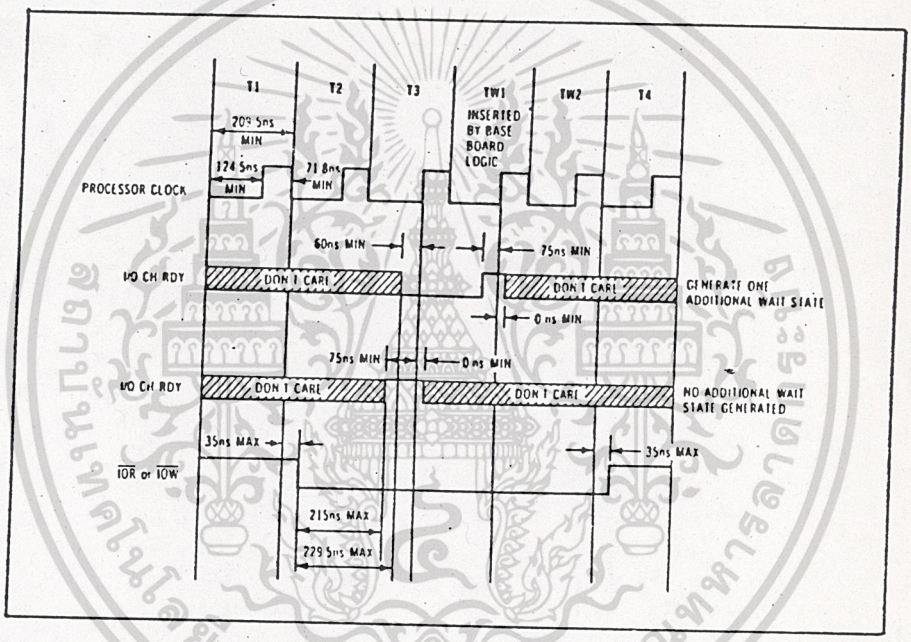
การสร้าง Wait State ในบัสไซเคิลของ 8088

จากที่ได้กล่าวถึงในบทต่าง ๆเหล่านั้น จะเห็นได้ว่าโดยทั่วไปในแต่ละบัสไซเคิลของ 8088 จะใช้ช่วงเวลานานเท่ากับช่วงเวลาของคล็อก 4 ลูก คือ T1, T2 T3 และ T4 แต่สำหรับใน IBM/PC จะทำการเพิ่มจำนวนสล็อตในบางบัสไซเคิลขึ้นอีก 1 ลูก ซึ่งคล็อกที่เพิ่มเข้าไปนี้มีชื่อว่า Tw ทำให้ในบางบัสไซเคิลมีช่วงเวลานานเท่ากับช่วงเวลาของคล็อก 5 ลูกทั้งนี้ก็เพื่อหน่วงเวลาในบัสไซเคิลให้วงจรเทอร์เฟสสามารถทำงานตอบสนองต่อบัสไซเคิลเหล่านี้ได้ทัน สำหรับในหัวข้อนี้จะได้กล่าวถึงวิธีการสร้าง Wait State(ในบทนี้คือ Tw) ในบัสไซเคิลที่เกี่ยวกับการอ่าน/เขียนข้อมูลในหน่วยความจำ และ ในบัสไซเคิลที่เกี่ยวกับการอ่าน/เขียนข้อมูลลงบนพอร์ท I/O ซึ่งมีวิธีการควบคุมสัญญาณ READY (I/O CHRDY) แตกต่างกัน

การสร้าง Wait state ในบัสไซเคิลที่เกี่ยวกับพอร์ท I/O

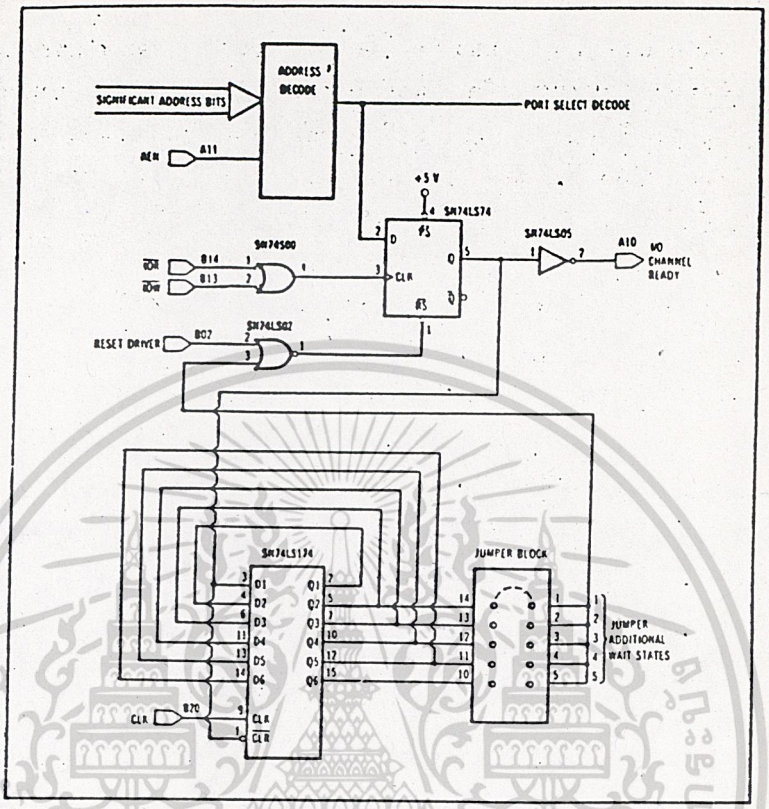
สำหรับในบัสไซเคิลที่เกี่ยวกับพอร์ท I/O นี้ โดยทั่วไปก็จะมีช่วงเวลาเท่ากับเวลาของบัสไซเคิลที่เกี่ยวกับหน่วยความจำ คือ ใช้เวลานานเท่ากับช่วงคล็อก 4 ลูก แต่วงจรภายใน IBM/PC จะทำการเพิ่มช่วงเวลาในบัสไซเคิลที่เกี่ยวกับพอร์ท I/O นี้ขึ้นอีก ซึ่งจะทำให้ช่วงเวลาในบัสไซเคิลนานเท่ากับช่วงเวลาของคล็อก 5 ลูกสำหรับคล็อกลูกที่เพิ่มเข้าไปในบัสไซเคิลนี้ก็คือ Wait State หรือ Tw นั่นเอง ทั้งนี้ก็เพื่อให้วงจรอินเทอร์เฟซหรือพอร์ท I/O ต่าง ๆสามารถตอบสนองต่อการทำงานในบัสไซเคิลนี้ได้ทัน อย่างไรก็ตามวงจรอินเทอร์เฟซที่เราสร้างขึ้น ก็อาจจะยังไม่สามารถทำงานได้เร็วพอที่จะตอบสนองต่อการทำงานในบัสไซเคิลที่ถูกเพิ่ม Wait State เข้าไปนี้ได้ทัน

ดังนั้นใน IBM/PC จึงยังคงยอมให้วงจรภายนอกสามารถขอ Wait State เพิ่มขึ้นได้อีก โดยผ่านทางขา I/O CHRDY เช่นเดียวกับกรณีของบัสไซเคิลที่เกี่ยกับหน่วยความจำ แต่ยังคงมีรายละเอียดปลีกย่อยบางประการในการขอ Wait State จากวงจรบนเมนบอร์ดของ IBM/PC กล่าวคือ ในขณะที่บัสไซเคิลที่เกี่ยวกับหน่วยความจำนั้น วงจรที่ทำการสร้าง Wait State จะทำการตรวจสอบสัญญาณ READY นี้ในช่วงเวลาขอบขาขึ้นของคล็อก T2 แต่ในบัสไซเคิลที่เกี่ยวกับพอร์ท I/O วงจรที่ทำการสร้าง Wait State จะตวจสอบสัญญาณที่ใช้ในการขอ WAit State นั้น จะเหมือนกับบัสไซเคิลที่เกี่ยวกับหน่วยความจำ ดังแสดงในรูปที่ 2.4.1



รูปที่ 2.4.1 ไดอะแกรมแวลาของ Wait State ในบัสไซเคิลของการอ่านหรือเขียนข้อมูลลงบนพอร์ท

จากไดอะแกรมเวลาข้างต้นนั้น ถ้าต้องการเพิ่ม Tw เข้าไปในบัสไซเคิล (จากเดิมที่มี Tw อยู่แล้ว 1 ลูก) เราจะทำให้ระดับลอจิกของสัญญาณ READY เป็น "0" เป็นเวลา 60 nanosec. ก่อนช่วงเวลาขอบขาขึ้นของคล็อก T3 ของบัสไซเคิลที่เกี่ยวกับพอร์ท I/O และ ถ้าไม่ต้องการให้มี Tw ใดๆเกิดขึ้นอีก ก็จะต้องทำให้ระดับลอจิกของสัญญาณ READY เป็น "1" เป็นเวลา 75 nanosec. ก่อนช่วงเวลาขอบขาขึ้นของคล็อก T3 ของบัสไซเคิลที่เกี่ยวกับพอร์ท I/O สำหรับตัวอย่างวงจรที่ใช้ในการขอ Wait State จากวงจรบนเมนบอร์ดนั้นจะแสดงในรูปที่ 2.4.3



รูปที่ 2.4.3 วงจรสร้าง Wait State ในบัสไซเคิลของการอ่านหรือเขียนข้อมูลลงบนพอร์ท

จากตัวอย่างวงจรข้างต้นนั้น จะเป็นวงจรที่สามารถใช้ในการขอให้วงจรบนเมนบอร์ดทำการสร้าง Wait State ในบัสไซเคิลของการอ่าน/เขียนข้อมูลลงบนพอร์ท I/O ได้ตั้งแต่ 1-5 ลูก สำหรับการทำงานของวงจรนี้จะคล้ายกับในวงจรที่ใช้สำหรับการขอ Wait state ในบัสไซเคิลที่เกี่ยวกับหน่วยความจำ กล่าวคือ วงจรในส่วนที่ทำการดีโค้ดแอดเดรสจะทำหน้าที่ในการกำหนดบล็อกแอดเดรสของพอร์ท I/O ที่ต้องการจะให้วงจรบนเมนบอร์ดสร้าง Wait state ขึ้น โดยวงจรในส่วนนี้จะสร้างสัญญาณดีโค้ดส่งให้กับวงจรในส่วนอื่น ๆเมื่อเกิดบัสไซเคิลของการอ่าน/เขียนข้อมูลลงบนพอร์ท I/O ที่อ้างถึงแอดเดรสของพอร์ทที่อยู่ในบล็อกแอดเดรสของพอร์ท I/O ที่กำหนด

หลังจากที่สัญญาณดีโค้ดถูกสร้างขึ้นแล้ว จะถูกส่งไปยังอินพุทที่ขา D(ขา2)ของ 74LS74 และสัญญาณนี้จะถูกส่งไปยังเอาท์พุทที่ขา Q ของ 74LS74 เมื่อสัญญาณ IORหรือ

LOW แอคทีฟ (ลอจิก "0") เอาท์พุทขา Q นั้นจะถูกส่งผ่าน Invweter ไปยังขา I/O CHRDY บนสล็อตของ IBM/PC เพื่อขอให้วงจรบนเมนบอร์ดสร้าง Wait State ขึ้นในบัสไซเคิลนั้น ๆนอกจากนี้เอาท์พุทที่ขา Q ยังถูกนำไปใช้ในวงจร Shift Registerซึ่งสร้างขึ้นจาก 74LS74 เพื่อกำหนด Tw ที่ต้องการเพิ่มเข้าไปในบัสไซเคิล ซึ่งเราสามารถจะกำหนดจำนวน Tw นี้ได้ โดยการเลือกเชื่อมต่อเอาท์พุท Q2,Q3,....., หรือ Q6 กับอินพุทของเกท NOR สำหรับในกรณีของบัสไซเคิลที่เกี่ยวกับพอร์ท I/O นี้ จะเห็นได้ว่าเราไม่สามารถจะใช้เอาท์พุท Q1 ของ 74LS74ได้ ทั้งนี้ก็เพราะ Tw ลูกแรกนั้นถูกสร้างขึ้นโดยวงจรบนเมนบอร์ดของ IBM/PC แล้ว ดังนั้นการเลือกเชื่อมต่อ Q2 กับอินพุทของเกท NOR จะเป็นการสร้าง Tw ขึ้นเพียง 1 ลูก (เมื่อรวมกับ Tw ที่วงจรบนเมนบอร์ดสร้างขึ้นจิงจะเป็น2 ลูก) แต่ถ้าเลือกเชื่อมต่อ Q3,Q4,Q5 กับอินพุทของเกท NOR ก็จะเป็นการเพิ่ม Tw ขึ้นอีก 2,3,4 หรือ 5 ลูกตามลำดับ

จากวงจรข้างต้นเป็นตัวอย่างวงจรที่ทำการขอ Wait state ทั้งในกรณีของบัสไซเคิลที่ทำการอ่านและบัสไซเคิลที่ทำการเขียนข้อมูลลงบนพอร์ท I/O ที่มีแอดเดรสอยู่ในบล็อกแอดเดรสที่เรากำหนดไว้ แต่ถ้าเราต้องการวงจรที่ทำการขอ Wait state เฉพาะในกรณีของบัสไซเคิลที่ทำการอ่านข้อมูลจากพอร์ท I/O เราก็สามารถจะทำได้โดยการนำเอาเกท NAND ที่ขา CLK ของ 74S74 ออกและนำเอาเฉพาะสัญญาณ IOR เพียงสัญญาณเดียวต่อผ่าน Inverter เข้ากับขา CLK ของ 74S74 แทน และถ้าต้องการวงจรที่ทำการขอ Wait state เฉพาะในกรณีของบัสไซเคิลที่ทำการเขียนข้อมูลลงบนพอร์ท I/O เราก็สามารถจะทำการได้โดยการนำเอาเกท NAMD ที่ขา CLK ของ 74S74 ออก และนำเอาเฉพาะสัญญาณ LOW เพียงสัญญาณเดียวต่อผ่าน Inverter เข้ากับขา CLK ของ 74LS74 แทน

2.5 สถาปัตยกรรมของ 8088

ไมโครโปรเซสเซอร์เบอร์ 8088 เป็นไมโครโปรเซสเซอร์ที่มีการทำงานคล้ายกับไมโครโปรเซสเซอร์เบอร์ 8086 คือเป็นไมโครโปรเซสเซอร์ที่มีการประมวลผลภายในแบบ 16 บิทเหมือนกัน แต่ต่างกันที่บัสข้อมูลที่ใช้เชื่อมต่อกับวงจรภายนอก คือ 8086 จะมีขนาด 16 บิท ส่วน 8088 จะมีขนาด 8 บิท ด้วยเหตุนี้ทำให้ 8086 สามารถทำงานได้เร็วกว่า 8088 (ที่ความถี่คล็อกเดียวกัน) เพราะ 8086 สามารถทำงานได้เร็วกว่า 8088 ต้องอ่านถึง 2 ครั้งจึงจะได้ข้อมูลครบทั้ง 16 บิท

สำหรับบัสแอดเดรสของ 8088 จะมีขนาด 20 บิท คือ A0-A19 ซึ่งจะทำให้ 8088 สามารถอ้างแอดเดรสของหน่วยความจำโดยตรงได้ถึง 1Mbyte แต่สำหรับการอ้างแอดเดรสของพอร์ท I/O 8088 จะใช้แอดเดรสเพียง 16 บิท ซึ่งทำให้ 8088 สามารถอ้างแอดเดรสของพอร์ท I/O ได้เพียง 64K พอร์ทเท่านั้น และเนื่องจากการที่ 8088 สามารถอ้างแอดเดรสขนาด 20 บิทในการอ้างแอดเดรสของหน่วยความจำ ซึ่งทำให้ไม่สะดวกและเปลืองเนื้อที่ในการเขียนโปรแกรม ดังนั้น 8088 จึงทำการแบ่งแอดเดรสนี้ออกเป็นแอดเดรสเซกเมนต์ (Segment Address) และ แอดเดรสออฟเซ็ท (Offest Address) ซึ่งจะช่วยลดความยุ่งยากต่าง ๆลงได้ (จะกล่าวถึงโดยละเอียดในหัวข้อ "การแบ่งแอดเดรสของ 8088")

โหมดการทำงานของ 8088 จะแบ่งออกเป็น 2 โหมด คือโหมด Minimum และโหมด Maximum ซึ่งในโหมด Minimum นั้น 8088จะจัดเตรียมสัญญาณควบคุมต่าง ๆ ที่จำเป็นจะต้องใช้เช่น M/IO WR RD ฯลฯ ไว้เรียบร้อยแล้ว ในขณะที่โหมด Maximum นั้นเราจะต้องใช้ 8288 Bus Controller เข้าช่วยในการถอดรหัสของสัญญาณควบคุมต่าง ๆเหล่านี้ด้วย แต่ในโหมด Maximum จะมีข้อดีกว่าในโหมด Maximum อยู่คือ สามารถที่จะใช้งาน 8088 ร่วมกับ Coprocessor อื่น ๆเช่น 8087 Maath-Coprocessor ได้

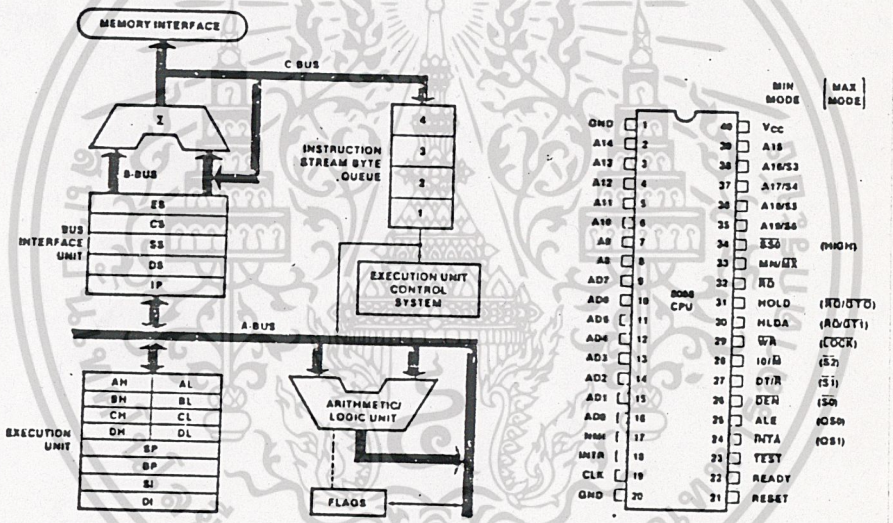
การจัดอินเทอร์รัพของ 8088 จะมีลักษณะคล้ายกับไมโครโปรเซสเซอร์เบอร์อื่น ๆ คือ แบ่งออกเป็น Non- Maskable Interrupt และ Maskable Interrupt ส่วนในกรณีที่จำเป็นต้องใช้การอินเทอร์รัพนี้มากกว่า 1 ตัวแล้ว ก็สามารถจะใช้ชิพซัพพอร์ทเบอร์ 8259A Interrupt Comtroller เข้าช่วยในการจัดอินเทอร์รัพท์ของระบบได้

นอกจากนี้ 8088 ยังมีฟังก์ชั่นการทำงานอื่น ๆอีกเช่น การจัด DMA สำหรับอุปกรณ์อื่น ๆ ซึ่งเราสามารถใช้ชิพซัพพอร์ทเบอร์ 8237A-5 ช่วยในการทำขบวนการ

DMAได้ การ Wait เพื่อรออุปกรณ์อื่น ๆที่ไม่สามารถทำงานตอบสนองต่อบัสไซเคิลของ 8088 ได้ทันในภายหลัง ส่วนในบทนี้จะกล่าวถึงแต่เฉพาะส่วนที่เกี่ยวกับ 8088 โดยตรงเท่านั้น

การจัดเรียงและหน้าที่ของขาต่าง ๆ ใน8088

ในการที่จะศึกษาถึงการใช้งานไมโครโปรเซสเซอร์เบอร์ 8088 หรือการเชื่อมต่ออุปกรณ์ต่าง ๆกับระบบคอมพิวเตอร์ที่ใช้ไมโครโปรเซสเซอร์เบอร์นี้นั้น เราจำเป็นจะต้องศึกษาถึงหน้าที่ของขาต่าง ๆใน 8088 เสียก่อน ซึ่งในรูป 2.5.1 จะแสดงถึงการจัดเรียงขาและหน้าที่ของขาต่าง ๆใน 8088



รูปที่ 2.5.1 การจัดเรียงและหน้าที่ของขาต่าง ๆใน 8088

จากรูปที่ 2.5.1จะเห็นว่า 8088 นั้นมีโหมดการทำงานอยู่ 2 โหมด คือโหมด Minimum (MN) และโหมด Maximum (MX) ด้วยเหตุนี้จึงทำให้ขาบางขาของ 8088มีหน้าที่การทำงานมากกว่า 1หน้าที่เช่น ขา 25 ของ8088 จะทำหน้าที่เป็นขา ALE ในโหมด Minimum และเป็นขา QS0 ในโหมด Maximum เป็นต้น ซึ่งการที่จะกำหนดให้ 8088 ทำงานในโหมดใดนั้น เราสามารถจะกำหนดได้ โดย

1. ป้อนลอจิก "1" ให้กับขา MN/MX (ขา 33) ในกรณีที่ต้องการให้ 8088 ทำงานในโหมด Minimum

2. ป้อนลอจิก "0" ให้กับขา MN/MX ในกรณีที่ต้องการให้ 8088 ทำงานในโหมด Maximum

สำหรับการทำงานในโหมด Minimum นั้น หน้าที่ขอขาต่าง ๆ ใน 8088 จะเหมือนกับใน 8085 ซึ่งจะทำให้เราสามารถใช้ชิพซัพพอร์ท (Chip Support) ต่าง ๆ ของ 8085 ได้ทันที และข้อดีอีกประการหนึ่งของการทำงานในโหมดนี้ของ 8088 ก็คือ เราไม่จำเป็นจะต้องใช้ IC ที่ทำหน้าที่ในการควบคุมบัสหรือ Bus Controller (ในที่นี้คือ 8288) ร่วมกับ 8088 เนื่องจากในโหมดนี้ 8088จะสร้างสัญญาณที่ใช้ในการติดต่อกับวงจรภายนอกไว้ครบถ้วนอยู่แล้ว อย่างไรก็ตามถึงแม้การใช้งาน 8088 ในโหมด Minimum จะสามารถทำได้ง่ายและสะดวกกว่าในโหมด Maximum ก็ตาม แต่เนื่องจากในโหมด Minimum นี้ไม่มีบัสในส่วนที่แสดงสภานะของ Queue Lock และส่วนที่เป็น Request/Grant ดังนั้นจึงทำให้ไม่สามารถใช้งานร่วมกับ Co-Processor เช่น 8087 ซึ่งเป็น Co-Processor ที่ช่วยในการคำนวณทางคณิตศาสตร์ของ 8088 ได้

ส่วนในโหมด Maximum นั้น สัญญาณควบคุมบางสัญญาณ เช่น IOW IOR หรือ ALE จะถูกเข้ารหัสไว้ ทั้งนี้ก็เพื่อให้สามารถเพิ่มหน้าที่ให้กับขาต่าง ๆของ 8088 ได้อีก ซึ่งหน้าที่ที่ทำการเพิ่มให้กับขาของ 8088 นั้น จะทำให้เราสามารถใช้งาน Co-Processor ร่วมกับ 8088 ได้ ดังนั้นภายในโหมดนี้จึงจำเป็นต้องใช้ 8288 Bus Controller เข้าช่วยในการสร้าง (ถอดรหัส) สัญญาณต่าง ๆ เหล่านี้ด้วย

สำหรับการใช้งาน 8088 ใน IBM/PC นั้น จะทำการต่อขา MN/MX ลงกราวด์ไว้ ซึ่งจะเป็นการทำให้ 8088 ทำงานในโหมด Maximum ดังนั้นในระบบของ IBM/PC จึงสามารถที่จะใช้ Co-Processor เข้าช่วยในการทำงานได้ในส่วนต่อไปจะกล่าวถึงหน้าที่ต่าง ๆของ 8088

บัสแอดเดรสและบัสข้อมูล

AD0-Ad7(Address/Data Bus; ขา 16-9):

เส้นสัญญาณทั้ง 8 เส้นนี้ จะถูกใช้สำหรับส่งแอดเดรสหรือข้อมูลให้กับหน่วยความจำหรือวงจร I/O (Input /Ootput) ที่ทำงานร่วมกับ 8088 ซึ่งสัญญาณทั้ง 8 เส้นนี้ใช้หลักการของมัลติเพล็กซ์ กล่าวคือในช่วงเวลาของคล็อกลูกแรกในบัสไซเคิล (บัสไซเคิล คือ ช่วงของการส่งผ่านข้อมูลหนึ่ง ๆของ 8088 ; ดูรายละเอียดเกี่ยวกับไซเคิลต่าง ๆในบทที่ 5) 8088 จะใช้ขา AD0-AD7 นี้เป็นบัสแอดเดรสโดยจะส่งแอดเดรส A0-A7 ออกมาที่ขา AD0-AD7 ส่วนในช่วงเวลา (ในบัสไซเคิลเดียวกัน) ที่ถัดจากคล็อกลูกแรกของบัสไซเคิล 8088 ก็จะใช้ขา AD0-AD7 นี้เป็นบัสข้อมูล Data Bus) เพื่อทำ

การส่งผ่านข้อมูลต่าง ๆระหว่างหน่วยความจำหรืออุปกรณ์ I/O กับ 8088

สัญญาณที่ขา AD0-AD7 นี้จะกลายเป็น High Impedance ในขณะที่ท8088 ตอบรับการขออินเทอร์รัพท์ (Interrupt Acknorledge) หรือในขณะที่ 8088 ตอบรับการใช้บัส (Hold Acknowledge) จากภายนอก

A8-A15 (Address Bus; ขา 8-2):

เส้นสัญญาณทั้ง 8 เส้นนี้จะถูกใช้สำหรับส่งแอดเดรสให้กับหน่วยความจำ หรือวงจร I/O ซึ่ง 8088 จะส่งแอดเดรส A8-A15) ของหน่วยความจำหรือ I/O ที่ต้องการจะติดต่อด้วย ออกมาในช่วงของคล็อกลูกที่หนึ่งในแต่ละบัสไซเคิล และจะรักษาสถานะของแอดเดรสนี้ไว้จนสิ้นสุดบัสไซเคิลนั้น

A 16/S3 (Address/Statis;ขา 37)

ในช่วงเวลาของคล็อกลูกที่หนึ่งในแต่ละบัสไซเคิล 8088 จะส่งแอดเดรส A17 ออกมาที่ขานี้ แต่ถ้าคำสั่งที่ 8088 กำลังทำงานอยู่นั้น เป็นคำสั่งที่เกี่ยวกับ I/O (คือ ถ้า8088 ต้องการจะติดต่อกับพอร์ท I/O เช่น คำสั่ง IN หรือ OUT 8088 ก็จะปรับให้สถานะของขานี้ (ในช่วงเวลาของคล็อกลูกที่หนึ่ง) ให้เป็นลอจิก "0" ส่วนในช่วงเวลาที่ถัดจากคล็อกลูกแรกของบัสไซเคิลจนสิ้นสุดช่วงเวลาของบัสไซเคิลนั้น 8088 จะใช้เส้นสัญญาณนี้ร่วมกับ A17/S4 เพื่อแสดงว่ารีจิสเตอร์เซกเมนต์ (Segment Reguster)ใดที่ถูกใช้งานอยู่ในบัสไซเคิลนั้น

A17/S4 (Address/Status; ขา 37):

ในช่วงเวลาของคล็อกลูกหนึ่งแต่ละไซเคิล 8088 จะส่งแอดเดรส A17 ออกมาที่ขานี้ แต่ถ้าคำสั่งที่ 8088 กำลังทำงานอยู่นั้น เป็นคำสั่งที่เกี่ยวกับ I/O 8088 ก็จะปรับให้สถานะของขานี้เป็นลอจิก "0"ส่วนในช่วงเวลาที่ถัดจากคล็อกลูกแรกของบัสไซเคิลจนสิ้นสุดช่วงเวลาของบัสไซเคิลนั้น 8088 จะใช้เส้นสัญญาณนี้ร่วมกับ A17/S4 เพื่อแสดงว่ารีจิสเตอร์เซกเมนต์ (Segment Register)ใดที่ถูกใช้งานอยู่ในบัสไซเคิลนั้น

A17/S4	A16/S3	Meaning
0	0	Extre Segment
0	1	Stack segment
1	0	Code segment or nosegment
1	1	Data segment

สำหรับสัญญาณที่แสดงเซกเมสต์ที่ 8088 ใช้งานอยู่นี้ จะช่วยให้เราสามารถขยายจำนวนหน่วยความจำ สำหรับแต่ละรีจิสเตอร์เซกเมนต์ให้เป็น 1 Mbyte ได้ ซึ่งจะเป็นผลให้จำนวนของหน่วยความจำในระบบเพิ่มขึ้นเป็น 4 Mbyte

A18/S5(Address Status;ขา 36):

ในช่วงเวลาของคล็อกลูกที่หนึ่งในแต่ละบัสไซเคิล 8088 จะส่งแอดเดรส A18 ออกมาที่ขานี้ แต่ถ้าคำสั่งที่ 8088 กำลังทำงานอยู่นั้น เป็นคำสั่งที่เกี่ยวกับ I/O 8088 ก็จะปรับในสถานะของขานี้เป็นลอจิก "0" ส่วนในช่วงเวลาที่ถัดจากคล็อกลูกแรกของบัสไซเคิลจนสิ้นสุดช่วงเวลาของบัสไซเคิลนั้น 8088 จะใช้เส้นสัญญาณนี้ในการแสดงสถานะของแฟล็ก I (Interrupt Enable Flag)

A18/S5(Address Status;ขา 36):

ในช่วงเวลาของคล็อกลูกที่หนึ่งในแต่ละบัสไซเคิล 8088 จะส่งแอดเดรส A19 ออกมาที่ขานี้ แต่ถ้าคำสั่งที่ 8088 กำลังทำงานอยู่นั้นเป็นคำสั่งที่เกี่ยวกับ I/O 8088 ก็จะปรับให้สถานะของขานี้เป็นลอจิก "0" ส่วนในช่วงเวลาที่ถัดจากลูกแรกของบัสไซเคิลจนสิ้นสุดช่วงเวลาของบัสไซเคิลนั้น ถ้า 8088 เป็นผู้ที่ควบคุม (ใช้งาน)บัสต่าง ๆ อยู่ก็จะให้เอาท์พุทที่ขาเป็นลอจิก "0" แต่ถ้าเป็นช่วงที่ 8088 ทำการตอบรับการขอให้บัส(Hold Acknowledge)8088 ก็จะปรับให้เอาท์พุทที่ขานี้อยู่ในสภาวะ Hight - Impedance

บัสควบคุมและแสดงสถานะ

สำหรับหน้าที่ต่าง ๆที่อยู่ในบัสควบคุมและแสดงสถานะนั้น จะแบ่งออกเป็น 2 กลุ่ม คือ กลุ่มที่ทำหน้าที่การทำงานไม่ขึ้นกับสถานะของลอจิกที่ขา MN/MX และกลุ่มที่หน้าที่การทำงานขึ้นอยู่กับสภาวะของลอจิกที่ขา MN/MX

ในส่วนแรกนี้จะกล่าวถึงกลุ่มที่หน้าที่การทำงานที่ไม่ขึ้นกับสถานะของลอจิกที่ขา MN/MX ของ 8088 ซึ่งจะประกอบไปด้วย:

Rd(Read;ขา 32) :

สัญญาณที่ขานี้จะเป็นลอจิก "0" ในช่วงเวลาที่ 8088 อ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์ I/O โดยที่สถานะของขา S2-IO/M (ขา 28) จะแสดงว่า 8088ต้องการอ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์ I/O

READY (ขา 22) :

เป็นอินพุทที่หน่วยความจำหรืออุปกรณ์ I/O ส่งให้กับ 8088 เพื่อแสดงให้ 8088 ทราบว่าหน่วยความจำหรืออุปกรณ์นั้น พร้อมที่จะส่งหรือรับข้อมูลจาก 8088 แล้วหรือไม่ ในกรณีที่อินพุทที่ขานี้ไดรับลอจิก "0" 8088 จะเข้าสู่สภาวะ "Wait" ซึ่งจะทำให้ 8088 รอจนกว่าหน่วยความจำหรืออุปกรณ์ I/O พร้อมที่จะส่งหรือรับข้อมูลจาก 8088 (คือ 8088 จะรอจนกว่าสถานะที่ขา READY กลับเป็นลอจิก "1")

การใช้งานขานี้ของ 8088 จะมีประโยชน์มากในกรณีที่หน่วยความจำหรือI/O ทำงานได้ช้ากว่า 8088 สำหรับในระบบของ IBM/PC นั้นสัญญาณที่ป้อนเข้าที่ขา READY นี้จะถูกนำมาจากเอาท์พุทของ 8284 Clock Generator ซึ่งจะใช้ในการเพิ่ม Wait State 1 ลูก (TW) เข้าไปในบัสไซเคิลเกี่ยวกับพอร์ท I/O ทุกบัวไซเคิล, เพิ่ม Wait State 1 ลูก เข้าไปในบัสไซเคิลที่เกี่ยวกับ DMA และใช้ในการสร้าง Wait State เมื่อมีอุปกรณ์ภายนอกอื่น ๆต้องการ (ดูรายละเอียดในบทที่ 5,7 และ 10)

TEST(ขา23) :

ขานี้เป็นอินพุทที่ 8088 จะใช้เฉพาะที่อยู่ในระหว่างการทำงานในคำสั่ง Wait เท่านั้น โดยเมื่อ 8088 ทำงานในคำสั่ง Wait 8088 จะหยุดรอจนกว่าขา TEST นี้จะได้รับลอจิก "0" จากภายนอก สำหรับในระบบของ IBM/PC นั้น ขานี้จะถูกต่อเข้ากับขา Socket ของ 8087 (ขา BUSY; ขา 23)

INTR (Interrupt Request;ขา 18) :

ขานี้เป็นอินพุทที่วงจรหรืออุปกรณ์ภายนอกใช้เพื่ออินเทอร์รัพท์ 8088 ซึ่ง 8088 จะทำการตรวจสอบระดับของสัญญาณที่ขานี้ ในช่วงเวลาของคล็อกลูกสุดท้ายในการทำงานในแต่ละคำสั่ง ในกรณีที่ 8088 ตรวจพบว่าระดับลอจิกที่ขา INTR นี้เป็น "1" และ แฟล็ก I (Interrupt Enable Flag) เป็น "1" 8088 ก็จะเข้าสู่การทำงานในส่วนของการตอบสนองการขออินเทอร์รัพท์และจะกระโดดไปทำงานในโปรแกรมตอบสนองการขออินเทอร์รัพท์ (Interrupt Service Routine) แต่ถ้าแฟล็ก I เป็น "0" หรือระดับสัญญาณที่ขา INTR นี้ไม่เป็น "1" ในช่วงเวลานี้ 8088 ก็จะเข้าสู่การทำงานในคำสั่งที่อยู่ถัดจากคำสั่งที่ 8088 ทำอยู่

NMI(Non-Maskable Interrupt;ขา 17) :

ขานี้เป็นอินพุทของ 8088 ที่อุปกรณ์ภายนอกใช้ เพื่อขออินเทอร์รัพท์แบบ

Non-Maskable (NMI) จาก 8088 ซึ่งการอินเทอร์รัพท์แบบ NMI นี้ 8088 จะทำการตอบรับในทุกกรณี (ในขณะที่เราสามารถจะสั่งให้ 8088 ไม่ยอมรับ (Disable) การขออินเทอร์รัพท์แบบ Maskable (โดยป้อนลอจิก "1" ให้กับขา INTR) ได้ โดยการเคลื่อนแฟล็ก I ให้เป็น "0" (ใช้คำสั่ง CLI)) ในกรณีที่อินพุทที่ขา NMI ของ 8088 ถูกเปลี่ยนจากลอจิก "0" เป็นลอจิก "1"(ขอบขาขึ้น) 8088 ก็จะทำการเอ็กซีคิวท์คำสั่งที่กำลังทำอยู่นั้นจนเสร็จ จากนั้นจึงกระโดดไปทำงานในโปรแกรมตอบสนองการขออินเทอร์รัพท์ ในตำแหน่งแอดเดรส 00008H

RESET (ขา 21) :

เป็นขาอินพุทที่ใช้สำหรับเซ็ท 8088 โดยทั่วไปขา RESET นั้นจะต่อจากขาเอาท์พุท RESET (ขา 10) ของ 8284 Clock Generator ซึ่งในกรณีที่ต้องการขาจะรีเซ็ท 8088 ก็จะทำได้โดยการป้อนลอจิก "0" ให้กับขา RES (ขา 9) ของ 8284 เป็นเวลานานเท่ากับช่วงเวลาของคล็อก 4 ลูก เช่นถ้าใช้คล็อก 4.77 MHz ก็จะต้องป้อนลอจิก "0" ให้กับขา RES ของ 8284เป็นเวลานาน= 1/4.77MHz X 4 หรือเท่ากับ 210 X 4 nanosec. 8nv 840 nanosec. เป็นต้น แต่สำหรับกรณีที่เริ่มจ่ายไฟให้กับระบบนั้น จะต้องป้อนลอจิก "0" เป็นเวลานานอย่างน้อย 50 usec.

เมื่อ 8284 ได้รับสัญญาณลอจิก "0" ที่ขา RES นี้แล้วก็จะทำการสร้างสัญญาณ RESET (ลอจิก "1")ให้ซิงก์ (Synchronize) กับคล็อกของระบบและส่งให้กับ 8088 โดยผ่านทางขา RESET (ขา 10) ของ 8284 หลังจากที่ 8088 ถูกรีเซ็ทแล้วก็จะทำงานตามขั้นตอนดังนี้:

1. 8088 จะเคลียร์ค่าของรีจิสเตอร์แฟล็กให้เป็น 0000H ซึ่งจะมีผลทำให้ 8088 ทำการดีสเอเบิล (Disable) การขออินเทอร์รัพท์แบบ Maskable และเข้าการทำงานในโหมด Single Step

2. 8088 จะเคลียร์ค่าของรีจิสเตอร์ DS,SS,ES และ PC ให้เป็น 0000H

3. 8088 จะเซ็ทค่าของรีจิสเตอร์ CS ให้เป็น OFFFFH

จากการทำงานตามขั้นตอนข้างต้นนั้น จะทำให้ 8088 กระโดดไปทำงานในตำแหน่งแอดเดรสที่ OFFFFOH

สำหรับในส่วนต่อไปจะกล่าถึง กลุ่มที่หน้าที่การทำงานขึ้นกับสถานะของลอจิกที่ขา MN/MX ซึ่งจะแสดงดังตาราง :

ตำแหน่งขา	หน้าที่การทำงาน	
	โหมด Maximum	โหมด Minimum
26	S0	DEN
27	S1	DT/R
28	S2	IO/M
31	RQ/GT0	HOLD
30	RQ/GT1	HLDA
25	QS0	ALE
24	QS1	INTA
29	LOCK	WR
34	--	SS0

S0;DEN (Status;Enable;ขา26) :

* โหมด Maximum ขานี้จะถูกใช้เป็นขา S0 ในการแสดงสถานะการทำงานของ 8088 ซึ่งจะกล่าวถึงในการอธิบายถึงหน้าที่การทำงานของขา S2;IO/M

* โหมด Minimum ขานี้จะถูกใช้งานเป็น DEN ซึ่ง 8088 จะทำให้ขา DEN นี้แอคทีฟ (Active; ในที่นี้เป็นลอจิก "0") เมื่อ 8088 พร้อมที่จะรับข้อมูลในขบวนการอ่านข้อมูล (Read Cycle) หรือเมื่อ 8088 พร้อมที่จะส่งข้อมูลออกมาบนบัสข้อมูลในขบวนการเขียนข้อมูล (Write Cycle) โดยทั่วไปเอาท์พุทที่ขานี้จะถูกใช้งานร่วมกับขา DT/R (ขา 27) ในการควบคุมบัฟเฟอร์ 8286/8287 (หรือ 74LS245) โดยที่ขานี้จะต่อเข้ากับขา OE ของ 8286/8287 โดยตรงเพื่อใช้ในการอินาเบิลบัฟเฟอร์ 8286/8287 (หรือ 74LS245)

S1;DT/R(Status;Data Transmit/Receive; ขา27):

* โหมด Maximum ขานี้จะถูกใช้งานเป็น DT/R ซึ่งจะใช้เพื่อบอกให้อุปกรณ์ภายนอกทราบว่า 8088 ต้องการที่จะรับข้อมูลเข้าหรือส่งข้อมูลออกไปบนบัสข้อมูล

(คือใช้แสดงทิศทางของการส่งผ่านข้อมูลระหว่าง 8088 กับวงจรภายนอกนั่นเอง) โดยปกติแล้วเอาท์พุทที่ขานี้จะเป็นลอจิก "1" เมื่อ 8088 ต้องการที่จะรับข้อมูลภายนอกจึงจะทำให้เอาท์พุทที่ขานี้เป็นเป็น "0" โดยทั่วไปเอาท์พุทที่ขานี้จะถูกใช้งานร่วมกับขา DEN (ขา 26) ในการควบคุมบัฟเฟอร์ 8286/8287 โดยตรงเพื่อควบคุมทิศทางการส่งผ่านข้อมูลของ 8286/8287

S2;IO/M(Status;I/O Or Memory Access; ขา28) :

* โหมด Maximum ขานี้จะถูกใช้เป็น S2 นี้จะถูกใช้ร่วมกับ So (ขา 26) และ S1 (ขา 27) ในการแสดงสถานะการทำงานของ 8088 ดังตารางข้างล่างนี้

S2	S1	S0	สถานะการทำงาน
0	0	0	Interrup Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Instruction Fecth
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	Inactive

โดยทั่วไปแล้วขา So, S1 และ S2 นี้จะถูกต่อกับ 8288 Bus controller โดยตรง เพื่อสร้างสัญญาณบางสัญญาณที่ไม่มีในโหมด Maximum เช่น สัญญาณ Den (สำหรับสัญญาณ DEN ที่ 8288 สร้างขึ้นนี้ จะแอคทีฟที่ลอจิก " 1" ซึ่งกลับกับสัญญาณ DEN ที่ 8088 สร้างขึ้นในโหมด Minimum ดังนั้นในการใช้งานหรือนำไปป้อนให้กับขา OE ของ 8286/8287 (หรือขา 19 ของ 74LS245) จะต้องกลับลอจิกของสัญญาณนี้เสียก่อน)ม DT/R ฯลฯ

* โหมด Minimum ขานี้จะถูกใช้งานเป็น IO/M โดยขานี้จะให้เอาท์พุทเป็นลอจิก "1" ในระหว่างการติดต่อกับอุปกรณ์ I/O และจะได้เอาท์พุทเป็นลอจิก "0" ในระหว่างการติดต่อกับหน่วยความจำ

QS0;ALE Queue Status; Address Latch Enable;ขา25) :

โหมด Maximum ขานี้จะถูกใช้เป็น QS0 ซึ่งจะใช้งานร่วมกับ QS1 (ขา 24) เพื่อแสดงสถานะของชุดคำสั่งของ 8088 (Instruction Queue; เป็นส่วนที่ใช้สำหรับเก็บคำสั่งและโอเปอแรนด์ต่าง ๆที่ 8088 ที่จะต้องใช้ในการเอ็กซีคิวท์ต่อไป สำหรับ Instruction Queue ใน 8088 จะมีขนาด 4 ไบท์)

* โหมด Minimum ขานี้จะถูกใช้งานเป็น ALE เพื่อแสดงให้วงจรภายนอกทราบว่าข้อมูลที่อยู่บนบัสแอดเดรสในขณะนั้นเป็นค่าของแอดเดรสที่ 8088 ต้องการจะติดต่อด้วย โดย8088 จะสร้างพัลส์ (ลอจิก"1")ออกมาที่ขา ALE นี้ เมื่อ 8088 ทำการส่งแอดเดรสที่ต้องการจะติดต่อด้วยออกมาเป็นบัสแอดเดรสแล้ว สำหรับในโหมด Maximum ของ 8088ซึ่งไม่มีขา ALE นั้น เราสามารถที่จะใช้ 8288 Bus Controller ช่วยในการสร้างสัญญาณ ALE นี้ขึ้นได้

AS1;INTA(Queue Status; Interrupt Acknowledgelขา QSS0(ขา24)

* โหมด Mixiมum ขานี้จะถูกใช้เป็น QS0 ซึ่งจะใช้งานร่วมกับขา QS0 (ขา25)เพื่อแสดงสถานะของส่วนเก็บชุดคำสั่งของ 8088 ดังตารางข้างล่างนี้

QS0	QS1	สถานะของส่วนเก็บชุดคำสั่ง
0	0	No Operation
0	1	ไบท์แรกของคำสั่งกำลังถูกเอ็กซีคิวท์
1	0	ส่วนเก็บชุดคำสั่งว่าง
1	1	ไบท์ต่อไปของคำสั่งกำลังถูกอ่านจากส่วนเก็บชุดคำสั่ง

โดยเมื่อ 8088 ทำการใด ๆกับส่วนเก็บชุดคำสั่ง 8088 ก็จะแสดงสถานะของส่วนเก็บชุดคำสั่งออกมาบน QS0 และ QS1ในช่วงเวลาของคล็อกลูกต่อไป

* โหมด Minimum ขานี้จะถูกใช้งานเป็น INTA โดยที่ 8088 จะทำให้เอาท์พุทที่ขานี้เป็น "0" ในระหว่างขบวนการตอบสนอง ของการอินเทอร์รัพท์ของอุปกรณ์ภายนอก

สำหรับในโหมด Maximum ของ 8088 ซึ่งไม่มีขา INTA เราสามารถจะ

ใช้ 8288 ช่วยในการสร้างสัญญาณ INTA นี้ได้

RQ/GT0;HOLD(Request/Grant0;Hold;ขา31 :

* โหมด Maximum ขานี้จะถูกใช้เป็น RQ/GT0 สำหรับขานี้ของ 8088 จะถูกใช้เพื่อให้อุปกรณ์ภายนอกหรืออุปกรณ์ที่ควบคุมบัส (Bus Master; เช่น 8237A-5 DMA-Controller) อื่น ๆสามารถที่จะควบคุมและใช้งานบัสต่าง ๆในระบบแทน CPU (8088) ได้ โดยเมื่ออุปกรณ์ที่ควบคุมบัสแทน 8088ทำการส่งลอจิก "0" ให้กับขา RQ /GT0 นี้ 8088 ก็จะเข้าสู่สภาวะ HOLD โดยจะปรับให้บัสแอดเดรส บัสข้อมูล และบัสควบคุมบางเส้นมีสถานะเป็นHigh Impedance หลังจากที่ 8088 เข้าสู่สภาวะ HOLD แล้วก็จะส่งเอาท์พุท(ลอจิก "0") ออกทางขา RQ/GT0 นี้เพื่อบอกให้อุปกรณ์ที่ควบคุมบัสแทน 8088 ทราบ จากนั้นอุปกรณ์ที่ควบคุมบัสแทน 8088 ก็สามารถจะใช้งานบัสแอดเดรส, บัสข้อมูล และ บัสควบคุมของระบบได้และหลังจากที่อุปกรณ์ที่ควบคุมบัสแทน 8088 นำขบวนการต่าง ๆ เสร็จเรียบร้อยแล้ว ก็จะต้องส่งพัลส์ (ลอจิก "0") กลับไปให้กับ 8088 อีกครั้ง เพื่อให้ 8088 ออกจากสภาวะ HOLD และเข้าส่การทำงานตามปกติต่อไป

* โหมด Minimum ขานี้จะถูกใช้งานเป็น HOLD โดยถ้ามีสัญญาณลอจิก "1" เข้ามาที่ขานี้ 8088 ก็จะเข้าสู่สภาวะ HOLD และจะส่งเอาท์พุท (ลอจิก "1") ออกมาที่ขา HLDA (ขา 30) เพื่อให้อุปกรณ์ภายนอกทราบว่า 8088 เข้าสู่สภาวะ HOLD แล้ว

RQ/GT1;HLDA(Request/Grant1;Holdacknowledgw;ขา30) :

* โหมด Maximum สำหรับในโหมดนี้จะมีลักษณะการใช้งานเหมือนกับขา RQ/GT0 แต่มีข้อมแตกต่างกันตรงที่ขา RQ/GT1

* โหมด Minimum ขานี้จะถูกใช้งานเป็น HLDA ดังที่ได้กล่าวไว้ในโหมด Minimum ของขา RQ/GT0; HOLD

LOCK;WR(Lock;Write Signal;ขา 29)

* โหมด Maximum ขานี้จะถูใช้งานเป็น LOCK โดยเมื่อ 8088 พบคำสั่ง LOCK (คำสั่งนี้เป็น Perfix) 8088 ก็จะทำให้เอาท์พุทที่ขานี้มีลอจิกเป็น "0" และจะคงสภาวะนี้ไว้จนกระทั่ง 8088 เสร็จจากการเอ็กซีคิวท์คำสั่งที่อยู่ต่อจากคำสั่ง LOCK ทั้งนี้ก็เพื่อทำให้ วงจรภายนอกทราบว่า 8088 ยังไม่พร้อมที่จะให้อุปกรณ์อื่น ๆเข้ามาทำ

การควบคุมบัสในช่วงนี้ ดังนั้นในกรณีที่เราไม่ต้องการให้มีการขอใช้บัสเกิดขึ้นในระหว่างการเอ็กซีคิวท์คำสั่งสั่งใด ก็เพียงแต่ใช้คำสั่ง LOCK นำหน้าคำสั่งนั้นเท่านั้น

* โหมด Minimum ขานี้จะถูกใช้เป็นขา WR ซึ่ง 8088 จะทำให้สัญญาณที่ขานี้แอคทีฟ(ลอจิก "0") ในช่วงของคล็อก T2, T3 และ Tw (ในกรณีที่มีการสร้าง Wait State ขึ้นในบัวไซเคิล;ดูรายละเอียดในบทที่ 10) ในบัสไซเคิลของการเขียนข้อมูลลงในหน่วยความจำหรือบนพอร์ท I/O

บัสจ่ายกำลังและฐานเวลา

บัสในส่วนนี้จะใช้สำหรับจ่ายไฟให้กับ 8088 และป้อนคล็อกให้กับ 8088 เพื่อควบคุมการทำงานของ 8088

CLK(Clock;ขา 19)

สัญญาณคล็อกที่ป้อนให้กับขา CLK นี้ จะถูกใช้เพื่อควบคุมจังหวะการทำงานของส่วนต่าง ๆภายใน 8088 สัมพันธ์กัน โดยปกติแล้วสัญญาณคล็อกที่ป้อนให้กับขา CLK จะถูกนำมาจากเอาท์พุทของ 8284 Clock Generator

Vcc(ขา 40) :

เป็นขาที่ใช้สำหรับจ่ายไฟให้กับ 8088 โดยแรงดันที่ป้อนให้กับขา Vcc ของ 8088 จะต้องมีค่าเท่ากับ +5V +10% คืออยู่ในช่วงระหว่าง +4.5 V โดยเทียบกับแรงดันที่ขา GND

GND(Ground; ขา 1และ ขา 20) :

สำหรับใน 8088 นี้จะมีขา GND อยู่ 2 ขาคือ ขา 1และ 20 ซึ่งขาทั้งสองจะถูกต่อถึงกัน (ภายใน 8088) โดยขานี้จพถูกต่อกับกราวด์ของระบบ

นอกจากการศึกษาถึงการจัดเรียงและหน้าที่ของขาต่าง ๆของ 8088 แล้วเรายังจำเป็นที่จะต้องศึกษาการอ้างแอดเดรสในลักษณะต่าง ๆของ 8088 ด้วย ดังนั้นสำหรับในหัวข้อต่อไปจะกล่าวถึงการบ่งแอดเดรสของ 8088

การบ่งแอดเดรสของ 8088

สำหรับในไมโครโปรเซสเซอร์เบอร์ 8088 นี้ จะมีเส้นสัญญาณที่ใช้เป็นบัสแอดเดรสจำนวน 20 เส้น คือจาก A0-A19 ดังนั้นจึงทำให้ 8088 สามารถที่จะใช้งาน

หน่วยความจำได้ถึง 1024 Kbyte หรือ 1Mbyte ในขณะที่ไมโครโปรเซสเซอร์ขนาด 5 บิททั่วไป เช่น Z80 นั้นสามารถใช้งานหน่วยความจำได้ถึง 1Mbyte นี้ ทำให้การเขียนโปรแกรมมีความยุ่งยากและสิ้นเปลืองเนื้อที่ในหน่วยความจำมากขึ้น เนื่องจากจำนวนหลักของเลขฐานสิบหกที่ใช้ในการอ้างแอดเดรสจะเป็น 5 หลัก (จาก 00000H จนถึง OFFFFFH) ในขณะที่ไมโครโปรเซสเซอร์ที่ใช้หน่วยความจำได้ 64Kbyte จะใช้เพียง 4 หลัก (จาก 0000H จนถึง OFFFFH) นอกจากนี้ยังทำให้เสียเวลาในการทำงานของ 8088 อีกด้วย เนื่องจากการอ่านแอดเดรสจากหน่วยความจำจะต้องอ่านถึง 3 ครั้ง (ครั้งละ 8 บิท) ในขณะที่ไมโครโปรเซสเซอร์ 8 บิทจะทำการอ่านเพียง 2 ครั้งเท่านั้น ดังนั้นเพื่อความสะดวกในการเขียนโปรแกรม 8088 จะบ่งแอดเดรส โดยการแบ่งแอดเดรสทั้ง 20 บิทออกเป็นเซกเมนต์ (segment Address) และออฟเซท (Offest Address) ดังนี้

สำหรับแอดเดรสทั้งที่เป็นเซกเมนต์และออฟเซ็ทนั้น จะเป็นแอดเดรสขนาด 16 บิท ทั้งคู่ ส่วนค่าแอดเดรสที่แท้จริงนั้นจะได้จากการนำเอาค่าแอดเดรสที่เป็นเซกเมสต์และออฟเซ็ทมารวมกัน โดยก่อนที่จะรวมแอดเดรสทั้งสองเข้าด้วยกัน 8088 จะเติมเลข "0"(ฐานสอง) จำนวน 4 บิทเข้าไปที่ท้ายของแอดเดรสที่เป็นเซ็กเมนต์แล้วจึงจะนำมาบวงกับแอดเดรสที่เป็นออฟเซ็ท (ในทางปฏิบัติก็คือการเลื่อน หรือ Shift ข้อมูลที่เป็นแอดเดรสเซกเมสต์ไปทางซ้าย 4 บิทนั่นเอง) ดังนั้น

```
    XXXX XXXX XXXX XXXX 0000   : แอดเดรสเซกเมนต์
+   0000 yyyy yyyy yyyy yyyy   : แอดเดรสออฟเซ็ท
    XXXZ ZZZZ ZZZZ ZZZZ YYYY   : แอดเดรสที่แท้จริง
```

ในที่นี้ X, Y และ Z เป็นเลขฐานสอง

จากที่กล่าวมานั้น จะเห็นได้ว่าถ้าเราคงค่าแอดเดรสเซ็กเมนต์ไว้ และเปลี่ยนแปลงค่าแอดเดรสออฟเซ็ท จะทำให้ได้ค่าแอดเดรสต่าง ๆถึง 65,536 แอดเดรส (64 K แอดเดรส) เช่นถ้ากำหนดแอดเดรสเซ็กเมนต์เป็น 0A320H จนถึง 1A31FH ดังนั้นถ้าเราต้องการค่าแอดเดรสที่อยู่ในช่วง 64K แอดเดรสของเดรส - เซ็กเมนต์ที่กำหนดไว้ก่อนหน้าแล้ว เราก็ทำการโหมดเฉพาะค่าแอดเดรสออฟเซ็ทค่าใหม่ที่รวมกับแอดเดรสเซกเมนต์แล้วได้ค่าแอดเดรสที่ต้องการให้กับ 8088 เท่านั้น โดยไม่จำเป็นต้องเปลี่ยนแปลงค่าแอดเดรสเซกเมนต์แต่อย่างใด เช่นถ้าค่าแอดเดรสเซกเมนต์ที่กำหนดไว้เดิม คือ 0A32H และแอดเดรสที่เราต้องการคือ 10430H ซึ่งอยู่ในช่วง 64K

แอดเดรสของแอดเดรสเซกเมนต์เดิม ดังนั้นในกรณีนี้เราก็เพียงแต่ทำการโหลดเฉพาะค่าแอดเดรสออฟเซ็ทให้กับ 8088 เท่านั้นซึ่งในที่นี้คือ ค่า 6110H

ด้วยวิธีการเช่นนี้จะทำให้การบ่งแอดเดรสที่ต้องการ สามารถทำได้โดยการโหลดค่าแอดเดรสเพียง 2 ครั้ง (ครั้งละ 8 บิท) เท่านั้น ทำให้ลดเวลาในการทำงานของ 8088ลงได้

แต่อย่างไรก็ต้องการค่าแอดเดรสที่อยู่นอกช่วง 64K แอดเดรสของแอดเดรสเซ็กเมลต์ที่กำหนดไว้เดิม เราก็จะต้องทำการดโหลดทั้งค่าแอดเดรสเซ็กเมนต์และแอดเดรสออฟเซ็ทให้กับ 8088 (ในกรณีนี้เราจะต้องทำการโหลดทั้งค่าแอดเดรสให้กับ 8088 ถึง 4 ไบท์) เช่นถ้าแอดเดรสเซ็กเมลต์เดิมเป็น 0A32H แต่แอดเดรสที่เราต้องการคือ 4B93AH ซึ่งอยู่นอกช่วง 64K แอดดรสของเซกเมนต์ 0A32H นี้(แอดเดรสสูงสุดของเซกเมนต์นี้คือ 1A31FH) เราจะต้องทำการโหลดค่าแอดเดรสเซกเมนต์ใหม่ให้กับ 8088 โดยค่าแอดเดรสเซกเมนต์ที่โหลดให้กับ 8088 นี้อาจเป็นค่าใดก็ได้ในช่วง 3B94H-4B93H ขึ้นอยู่กับความเหมาะสมในการเขียนโปรแกรมและการทำงานของระบบ ส่วนค่าแอดเดรสออฟเซ็ทนั้นจะขึ้นกับค่าแอดเดรสเซ็กเมลต์ที่โหลดให้กับ 8088 ซึ่งกรณีนี้จะเป็นแอดเดรส OFFFAH, OFFEAH, OFFDAH,...., 001AH ตามลำดับ คือ

ถ้าแอดเดรสเซกเมนต์เป็น 3B94H แอดเดรสออฟเซ็ทจะเป็น OFFFAH
ถ้าแอดเดรสเซกเมนต์เป็น 3B95H แอดเดรสออฟเซ็ทจะเป็น OFFEAH
: : :
: : :
ถ้าแอดเดรสเซกเมนต์เป็น 4B93H แอดเดรสออฟเซ็ทจะเป็น 000AH

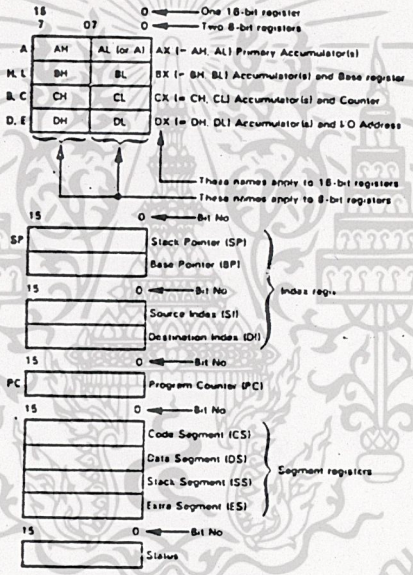
สำหรับค่าแอดเดรสเซกเมนต์และแอดเดรสออฟเซ็ทนี้ จะถูกเก็บไว้ในรีจิสเตอร์ภายใน 8088 (จะกล่าวถึงในหัวข้อ "รีจิสเตอร์ Segment") ซึ่งการโหลดค่าแอดเดรสเซกเมนต์และออฟเซ็ทให้กับ 8088 ก็จะทำได้ โดยการใช้คำสั่ง MOV โหลดค่าแอดเดรสให้กับรีจิสเตอร์เหล่านี้ หรือโดยการใช้คำสั่งอื่น ๆที่มีผลกับข้อมูลในรีจิสเตอร์เหล่านี้

รีจิสเตอร์ของ 8088

สำหรับรีจิสเตอร์ของ 8088 จะประกอบด้วยรีจิสเตอร์ที่ใช้งานด้านต่าง ๆ

จำนวน 14 ตัว ซึ่งสามารถที่จะแบ่งออกเป็นกลุ่มตามลักษณะการใช้งานได้ดังนี้

- รีจิสเตอร์ 16 บิทที่ใช้งานทั่วไป 4 รีจิสเตอร์
- รีจิสเตอร์ 16 บิทที่ใช้งานเป็นตัวชี้ (Pointer Register) 2 รีจิสเตอร์
- รีจิสเตอร์ 16 บิทที่ใช้เป็นอินเด็กซ์ (Index Regis) 2 รีจิสเตอร์
- รีจิสเตอร์ที่ใช้เป็น Program หรือ PC (หนังสือบางเล่มเรียกเป็น IP; Instruction Pointer)
- รีจิสเตอร์ Segment 4 รีจิสเตอร์
- รีจิสเตอร์แฟล็ก (Flag Register) 1 รีจิสเตอร์



รูปที่ 2.5.2 ไดอะแกรมของรีจิสเตอร์ใน 8088

รีจิสเตอร์ที่ใช้งานทั่วไป

รีจิสเตอร์ในกลุ่มนี้จะแบ่งออกเป็นรีจิสเตอร์ขนาด 16 บิทจำนวน 4 รีจิสเตอร์ ซึ่งในแต่ละรีจิสเตอร์นั้น เราสามารถที่จะแบ่งออกเพื่อใช้งานเป็นรีจิสเตอร์ขนาด 8 บิท 2 ตัวได้ โดยรีจิสเตอร์ในส่วนที่เป็น 8 บิทบน (บิท 8-15) ของรีจิสเตอร์ 16 บิทนั้นจะเรียกเป็น "ไบท์สูง" (High Byte) และรีจิสเตอร์ในส่วนที่เป็น 8 บิทล่าง - (บิท 0-7) ของรีจิสเตอร์ 16 บิทนั้น เราจะเรียกเป็น "ไบท์ต่ำ" (Low Byte) ซึ่งมีขนาด 16บิท จะสามารถแบ่งออกเป็นรีจิสเตอร์ขนาด 8 บิท 2 รีจิสเตอร์คือ AH -

High; บิท 8-15)และ AL (A Low; บิท 0-7) สำหรับรีจิสเตอร์ต่าง ๆในกลุ่มนี้ จะแสดงดังไดอะแกรมในรูปที่ 2.5.3

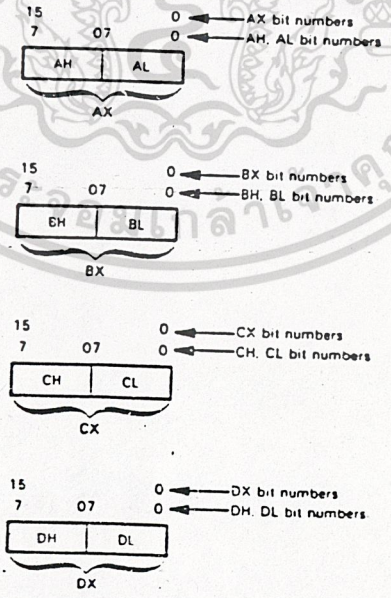
สำหรับการใช้งานรีจิสเตอร์ในกลุ่มนี้นั้น นอกจากจะใช้เป็นที่เก็บข้อมูลและประมวลผลทั่วไปแล้ว ยังใช้เป็นโอเปอร์แรนด์ (Operand) ในชุดคำสั่งต่าง ๆอีกด้วย ซึ่งลักษณะการใช้งานเป็นโอเปอร์แรนด์ในชุดคำสั่งต่าง ๆนั้น จะเป็นดังนี้:

รีจิสเตอร์ AX (Accumulator) :

การนำเอารีจิสเตอร์ AX (ในบางครั้งจะเรียกเป็น Accumulator)นี้ ไปใช้งานเป็นโอเปอร์แรนด์ในชุดคำสั่งของ 8088 นั้น จะมี 2 ลักษณะคือ

1. ในชุดคำสั่งที่เกี่ยวกับ I/O จะต้องใช้รีจิสเตอร์ AX เป็นโอเปอร์แรนด์เท่านั้น

2. ในชุดคำสั่งบางคำสั่งที่ใช้ข้อมูลจากหน่วยความจำโดยตรง(Immediate Data)นั้น ถ้าใช้รีจิสเตอร์ AX เป็นโอเปอร์แรนด์จะทำให้ประหยัดจำนวนไบท์ที่ใช้ในการเขียนโปรแกรมลงได้ เช่น คำสั่ง CMP,ac data ซึ่งใช้รีจิสเตอร์AX (16 บิท) หรือ AL(8 บิท)



รูปที่ 2.5.3 ไดอะแกรมของรีจิสเตอร์ที่ใช้งานทั่วไปใน 8088

เป็นโอเปอร์แรนด์นั้น จะใช้จำนวนไบท์ของชุดคำสั่งนี้เพียง 2(ในกรณีข้อมูล 8 บิท) หรือ 3 ไบท์(ในกรณีข้อมูล 16 บิท) เท่านั้น ในขณะที่ CMP mem/reg,data จะใช้จำนวนไบท์ของชุดคำสั่งนี้ถึง 3 ไบท์ (ในกรณีข้อมูล 8 บิท)หรือ 4 ไบท์(ในกรณีข้อมูล 16 บิท) เป็นต้น

นอกจากนี้ในชุดคำสั่งที่เกี่ยวกับสตริงก์ (String) และการประมวลผลทางคณิตศาสตร์บางคำสั่งจะต้องใช้รีจิสเตอร์นี้เท่านั้น

สำหรับรีจิสเตอร์ AL (8บิทล่างคือบิท 0-7 ของรีจิสเตอร์ AX)นี้ จะมีลักษณะเหมือนกับรีจิสเตอร์ A ของ 8088

รีจิสเตอร์ BX (Base Register) :

รีจิสเตอร์นี้จะถูกนำไปใช้ในการคำนวณหาตำแหน่งของหน่วยความจำที่ 8088 ต้องการจะติดต่อด้วย โดยจะใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์

รีจิสเตอร์ BX ของ 8088 นี้จะมีลักษณะการใช้งานเหมือนกับรีจิสเตอร์ HL ของ 8088 โดยที่รีจิสเตอร์ BH (8 บิทบนของรีจิสเตอร์ BX) จะเหมือนกับรีจิสเตอร์ H และรีจิสเตอร์ BL (8 บิทล่างของรีจิสเตอร์ BX) จะเหมือนกับรีจิสเตอร์ L ของ 8088

รีจิสเตอร์ CX (Counter Register) :

ในกรณีที่เราใช้คำสั่งเกี่ยวกับ Loop เช่นคำสั่ง Loop, LOOPNZ ฯลฯ หรือคำสั่งที่เกี่ยวกับสตริงก์ เช่น MOVS นั้น 8088จะทำการลดค่าในรีจิสเตอร์ CX แลง 1 หลังจากทำการเอ็กซีคิวท์ชุดคำสั่งใน Loop หรือทำงานตามชุดคำพสั่งสตริงก์นั้น 1 ครั้ง และจะวนกลับไปทำงานใน Loop หรือคำสั่งสตริงก์นั้นต่อ จนกว่าค่าของรีจิสเตอร์ CX และแฟล็กที่เกี่ยวข้องจะไม่เป็นไปตามเงื่อนไขของคำสั่ง Loop หรือ สตริงก์ เช่นคำสั่ง LOOPNZ นั้น 8088 จะกลับไปทำงานใน Loop จนกว่าค่าของรีจิสเตอร์ CX จะลดลงจนเป็น 0 หรือแฟล็ก Z (Zero Flag) จะถูกเซ็ทเป็น "1" ซึ่งจะเห็นได้ว่าจำนวนครั้งในการรวน Loop หรือในการทำงานตามคำสั่งสตริงก์ของ 8088 นั้น จะขึ้นอยู่กับค่าของรีจิสเตอร์ CX นี่นเอง

สำหรับรีจิสเตอน์ CX ของ 8088 นั้นจะเทียบได้กับรีจิสเตอร์ BX ของ 8088 โดยที่รีจิสเตอร์ CH (8 บิทบนของรีจิสเตอร์ CX) ของ 8088 นั้นจะเทียบได้กับรีจิสเตอร์ B ของ 8088 และรีจิสเตอร์ CL (8 บิทล่างของรีจิสเตอร์ CX) ของ 8088 นั้น จะเทียบได้กับรีจิสเตอร์ C ของ 8088

รีจิสเตอร์ DX (Data Register) :

รีจิสเตอร์นี้จะถูกใช้เป็นรีจิสเตอร์สำหรับเก็บข้อมูลเพื่อการประมวลผลทั่วไป และจะถูกใช้เป็นโอเปอร์แรนด์สำหรับคำสั่งที่เกี่ยวกับ I/O ซึ่งไม่สามารถใช้รีจิสเตอร์อื่นแทนได้คือคำสั่ง IN ac, DX และ OUT DX,ac

สำหรับรีจิสเตอร์ DX ของ 8088 นี้จะเทียบได้กับรีจิสเตอร์ DE ของ 8088 โดยที่รีจิสเตอร์ DH (8 บิทบนของรีจิสเตอร์ DX) ของ 8088 จะเทียบได้กับรีจิสเตอร์ D ของ 8088 และรีจิสเตอร์ DL (8 บิทล่างของรีจิสเตอร์ DX) ของ 8088 จะเทียบได้กับรีจิสเตอร์ E ของ 8088

รีจิสเตอร์ที่ใช้งานเป็นตัวชี้ (Pointer Register) :

สำหรับรีจิสเตอร์ในกลุ่มนี้ จะถูกใช้ในกรณีที่ต้องการจะประมวลผลข้อมูลที่เก็บอยู่ในสแต็ก (Stack) และสามารถที่จะนำไปใช้เป็นโอเปอร์แรนด์ในชุดคำสั่งที่เกี่ยวกับการประมวลผลข้อมูลทางคณิตศาสตร์ หรือลอจิกของข้อมูล 16 บิทได้อีกด้วย โดยรีจิสเตอร์ในกลุ่มนี้จะประกอบด้วย:

รีจิสเตอร์ SP (Stack Pointer Register)

รีจิสเตอร์นี้จะถูกใช้ในคำสั่ง PUSH และ POP เพื่อคำนวณหาตำแหน่งของหน่วยความจำส่วนที่เป็นสแต็ก โดยการใช้งานรีจิสเตอร์นี้ 8088 จะใช้ข้อมูลในรีจิสเตอร์ SS (Stack Segment) เป็นแอดเดรสเซกเมนต์เสมอ โดยค่าของรีจิสเตอร์ SP จะถูกเพิ่มขึ้น 2 เมื่อ8088 เอ็กซีคิวท์คำสั่ง POP และจะถูกลดลง 2 เมื่อ 8088 เอ็กซีคิวท์คำสั่ง PUSH

รีจิสเตอร์ BP (Base Pointer Regoster) :

โดยทั่วไปรีจิสเตอร์ BP นี้จะถูกใช้ในการคำนวณหาตำแหน่งของหน่วยความจำในสแต็ก ซึ่งการใช้รีจิสเตอร์ BP จะช่วยให้เราสามารถนำข้อมูลไปเก็บหรืออ่านจากหน่วยความจำในส่วนที่เป็นสแต็กในตำแหน่งใดก็ได้ ในขณะที่ทำการใช้รีจิสเตอร์ SP (ในคำสั่ง PUSH และ POP)นั้น จะสามารถนำข้อมูลไปเก็บหรืออ่านจากหน่วยความจำได้เฉพาะในหน่วยความจำ 2 ไบท์ที่อยู่บนสุดของสแต็กเท่านั้น โดยทั่วไปข้อมูลในรีจิสเตอร์นี้จะถูกนำไปใช้ในการคำนวณหาตำแหน่งแอดเดรสที่แท้จริง โดยใช้ข้อมูลในรีจิสเตอร์ SS (Stack Segment) เป็นแอดเดรสเซกเมนต์

รีจิสเตอร์ที่ใช้เป็นอินเด็กซ์ (Index Register)

รีจิสเตอร์ในกลุ่มนี้จะประกอบไปด้วยรีจิสเตอร์ 2 รีจิสเตอร์ คือ รีจิสเตอร์ SI และรีจิสเตอร์ DI โดยรีจิสเตอร์ในกลุ่มนี้จะถูกใช้ในการคำนวณหาแอดเดรสของหน่วยความจำในระหว่างการเอ็กคิวท์คำสั่งที่เกี่ยวกับสตริงก์ของ 8088 และ สามารถจะนำไปใช้เป็นโอเปอร์แรนด์ของชุดคำสั่งที่ทำการประมวลผลข้อมูล 16 บิทของ 8088 - ได้อีกด้วย

รีจิสเตอร์ Segment (Segment Register)

รีจิสเตอร์ในกลุ่มนี้เป็นรีจิสเตอร์ที่ใช้ในการเก็บค่าแอดเดรสเซกเมนต์เพื่อใช้ร่วมกับแอดเดรสออฟเซทในการคำนวณหาค่าแอดเดรสที่แท้จริง ดังที่ได้กล่าวไว้ในหัวข้อ " การบ่งแอดเดรสของ 8088"

สำหรับรีจิสเตอร์ในกลุ่มนี้จะเป็นรีจิสเตอร์ขนาด 16 บิทจำนวน 4 รีจิสเตอร์ คือรีจิสเตอร์ CS, DS, SSและ ES ซึ่งการที่ข้อมูลในรีจิสเตอร์ใดจะถูกนำไปใช้เป็นแอดเดรสเซกเมนต์นั้น จะขึ้นกับลักษณะคำสั่งที่ 8088 ทำการเอ็กซีคิวท์อยู่ (จะกล่าวถึงในภายหลัง)ส่วนรีจิสเตอร์ที่ใช้เป็นแอดเดรสออฟเซ็ทจะเปลี่ยนไปตามชุดคำสั่งและรีจิสเตอร์เซกเมสต์ที่ใช้ดังนี้:

รีจิสเตอร์ CS (Code Segment Register)

ข้อมูลในรีจิสเตอร์ CS จะถูกใช้เป็นแอดเดรสเซกเมสต์ในการเฟทช์ (Fetch) คำสั่งจากหน่วยความจำ เพื่อคำนวณหาแอดเดรสของชุดคำสั่งที่ 8088 จะทำการเฟทช์เข้ามาเพื่อเอ็กซีคิวท์ โดยรีจิสเตอร์ที่จะใช้เป็นรีจิสเตอร์สำหรับเก็บค่าแอดเดรสออฟเซ็ท คือรีจิสเตอร์ PC (Program counter) หรือ IP; Instruction Pointer)

รีจิสเตอร์ DS (Data Segment Register) :

ข้อมูลในรีจิสเตอร์ DS จะถูก 8088 ใช้เป็นแอดเดรสเซกเมนต์ในการคำนวณแอดเดรส เมื่อมีการอ้างถึงข้อมูลในหน่วยความจำโดยชุดคำสั่งต่าง ๆ

อย่างไรก็ตามมีข้อยกเว้นอยู่ 3ประการที่ 8088 จะใช้ข้อมูลในรีจิสเตอร์อื่น ๆ เป็นแอดเดรสเซกเมสต์แทนข้อมูลในรีจิสเตอร์ DS เมื่อมีการอ้างถึงข้อมูลในหน่วยความจำ ดังนี้ :

1. เมื่อมีการอ้างถึงแอดเดรสในหน่วยความจำที่เป็นสแต็ก 8088 จะใช้ข้อ

มูลในรีจิสเตอร์ SS (Stack Segment Register) เป็นแอดเดรสเซ็กเมนต์ โดยจะใช้ข้อมูลในรีจิสเตอร์ SP (Stack Pointer Register) เป็นแอดเดรสออฟเซ็ท

2. เมื่อมีการอ้างถึงแอดเดรสในหน่วยความจำ โดยใช้ข้อมูลในรีจิสเตอร์ BP (Base Pointer Register) เป็นแอดเดรสออฟเซ็ท 8088 จะใช้ข้อมูลในรีจิสเตอร์ SS เป็นแอดเดรสเซกเมนต์

3. เมื่อมีการอ้างถึงแอดเดรสในหน่วยความจำของการทำงานในชุดคำสั่งที่เป็นสตริงก์โดยใช้ข้อมูลในรีจิสเตอร์ DI (Destination Index Register) เป็นแอดเดรสออฟเซ็ท 8088 จะใช้ข้อมูลในรีจิสเตอร์ ES (Extra Segment Register) เป็นแอดเดรสเซกเมนต์

รีจิสเตอร์ SS (Stack Segment Register) :

ข้อมูลในรีจิสเตอร์ SS จะถูกใช้เป็นแอดเดรสเซกเมสต์ เมื่อมีการอ้างถึงแอดเดรสของหน่วยความจำโดยใช้ข้อมูลในรีจิสเตอร์ SP (Stack Pointer Regoster) หรือรีจิสเตอร์ BP (Base Pointer Register) เป็นแอดเดรสออฟเซ็ท ด้วยเหตุนี้จะทำให้ข้อมูลในรีจิสเตอร์ SS นี้ถูกนำมาใช้เป็นแอดเดรสเซกเมสต์ในชุดคำสั่งที่มีการอ้างถึงหน่วยความจำส่วนที่เป็นสแต็ก เช่นคำสั่ง PUSH, POP, CALL, RET, INT เป็นต้น

รีจิสเตอร์ ES (Extra Segment Register) :

ข้อมูลภายในรีจิสเตอร์ ES จะถูกใช้เป็นแดเดรสเซกเมสต์เมื่อมีการใช้ชุดคำสั่งที่เกี่ยวกับสตริงก์ โดยใช้ข้อมูลในรีจิสเตอร์ DI เป็นแอดเดรสออฟเซ็ ทในการอ้างถึงแอดเดรสของหน่วยความจำ

จากที่กล่าวมาข้างต้นนั้นเป็นกรณีโดยทั่วไปในการใช้งานรีจิสเตอร์ Segment ต่าง ๆแต่อย่างไรก็ตามในการทำงานตามคำสั่งชุดต่าง ๆของ 8088 ซึ่งได้กำหนดรีจิสเตอร์ Segment ที่จะใช้งานในชุดคำสั่งต่าง ๆไว้แล้วนั้น ในบางครั้งอาจจะเกิดความไม่สะดวกในการใช้รีจิสเตอร์ Segment ที่ถูกกำหนดไว้นั้นก็ได้ ดังนั้นภายใน 8088 จึงได้เตรียมคำสั่งที่ใช้สำหรับเปลี่ยนรีจิสเตอร์ Segment ที่จะถูกใช้งานในคำสั่งต่อไปไว้ด้วย ซึ่งคำสั่งนี้ก็คือ คำสั่ง

SEG segreg : โดยที่ segreg คือรีจิสเตอร์ ES, CS, SS, หรือ DS

ตัวอย่างเช่น ถ้าเราใช้คำสั่ง MOV AX,[BX] เพียงคำสั่งเดียว 8088 ก็จะ

ใช้ข้อมูลมรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ เมื่อมีการอ้างถึงแอดเดรสของหน่วยความจำในระหว่างการเอ็กซีคิวท์คำสั่งนี้ (ในที่นี้จะใช้ข้อมูลในรีจิสเตอร์ BX เป็นแอดเดรสออฟเซ็ท)

แต่ถ้าเราใช้คำสั่ง SEG segreg นำหน้าคำสั่งนี้ เช่น

```
SEG ES
MOV AX,[BX]
```

ในกรณีเช่นนี้เมื่อ 8088 ทำการเอ็กซีคิวท์คำสั่ง MOV AX, [BX] ซึ่งอยู่หลังคำสั่ง SEG ES นั้น 8088 จะใช้ข้อมูลในรีจิสเตอร์ ES เป็นแอดเดรสเซกเมนต์แทนข้อมูลในรีจิสเตอร์ DS ดังนั้นถ้าสมมติให้ :

- ข้อมูลในรีจิสเตอร์ DS เป็น 2300H
- ข้อมูลในรีจิสเตอร์ ES เป็น 2700H
- ข้อมูลในรีจิสเตอร์ BX เป็น 000AH
- ข้อมูลในหน่วยความจำแอดเดรส 2300AH และ 2300BH เป็น OFEH และ OFOH ตามลำดับ
- ข้อมูลในหน่วยความจำแอดเดรส 2700AH และ 2700BH เป็น 30H และ 41H ตามลำดับ

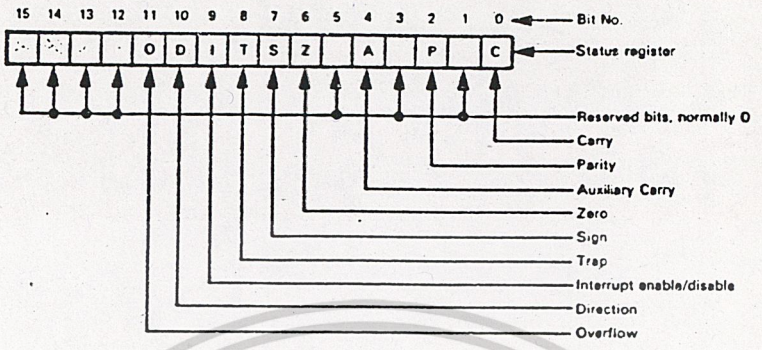
ในกรณีแรกซึ่งใช้คำสั่ง MOV AX,[BX] โดยตรง เมื่อ 8088ทำการเอ็กซีคิวท์ทำสั่งนี้เสร็จแล้ว จะทำให้ข้อมูลในรีจิสเตอร์ AX เป็น OFOFEH (ใช้ข้อมูลในรีจจิสเตอร์DS เป็นแอดเดรสเซกเมนต์ โดยใช้ข้อมูลในรีจิสเตอร์ BX เป็นแอดเดรสออฟเซ็ท ทำให้ได้ค่าแอดเดรสเป็น 2300AH)

ส่วนในกรณีที่สองซึ่งใช้คำสั่ง SEG ES นำหน้าคำสั่ง MOV AX,[[[BX] นั้น เมื่อ 8088 ทำการเอ็กซรคิวท์คำสั่งทั้งสองเสร็จแล้ว จะทำให้ข้อมูลในรีจิสเตอร์ AX เป็น 4130H(ใช้ข้อมูลในรีจิสเตอร์ ES เป็นแอดเดรสเซกเมสต์ โดยใช้ข้อมูลในรีจิสเตอร์ BX เป็นแอดเดรสออฟเซ็ท ทำให้ได้ค่าแอดเดรสเป็น 2700AH) สำหรับการอ้างแอดเดรสในแบบต่าง ๆนั้น จะกล่าวถึงในภายหลัง

รีจิสเตอร์แฟล็ก (Flage Register)

รีจิสเตอร์นี้เป็นรีจิสเตอร์ขนาด 16 บิท ซึ่งใช้สำหรับแสดงสถานะและควบคุมการทำงานของ 8088 โดยแต่ละบิทของรีจิสเตอร์แฟล็ก จะมีหน้าที่ในการแสดงสภานะ

และควบคุมการทำงานของ 8088 แตกต่างกันไป ดังแสดงในรูปข้างล่างนี้ :



รูปที่ 2.5.4 รีจิสเตอร์แฟล็กของ 8088

แฟล็ก C (Carry Flag; บิท 0) :

แฟล็กนี้จะถูกเซ็ท (ลอจิก 1) ในกรณีที่ผลของการกระทำทางคณิตศาสตร์นั้นมีการทดหรือยืมบิทให้กับบิทสูงสุด (บิท 15 ในกรณีของข้อมูล 16 บิท หรือบิท 7 ในกรณีของข้อมูล 8บิท) สำหรับสถานะของแฟล็กนี้เราสามารถเปลี่ยนแปลงได้โดยใช้คำสั่งที่เกี่ยวกับการเลื่อน (Shift) หรือหมุน (Rotate) บิทหรือ คำสั่งที่เกี่ยวกับการเซ็ทหรือเคลียร์แฟล็กโดยตรง คือ CLC, CMC และ STC ก็ได้

แฟล็ก O (Overflow Flag; [bm 11)

แฟล็กนี้จะเป็นผลจากการ XOR ของตัวทดที่เข้าและออกจากบิทสูงสุดของการกระทำทางคณิตศาสตร์ (คือ ในกรณีของข้อมูล 16 บิทคือผลจากการ XOR ของตัวทดจากบิท 14 และตัวทดจากบิท 15 ส่วนในกรณีของข้อมูล 8 บิทก็คือ ผลจากการ XOR ของตัวทดจากบิท 6 และตัวทดจากบิท 7)

แฟล็ก S (Sign Flag; บิท 7) :

แฟล็กนี้จะมีสถานะ (ลอจิก) เหมือนกับบิทสูงสุดของผลการกระทำทางคณิตศาสตร์ ซึ่งแฟล็กนี้จะถูกใช้ในการคำนวณเลขฐานสองแบบมีเครื่องหมาย โดยผลที่ได้นั้นจะมีค่าเป็นบวก ถ้าแฟล็หนี้มีลอจิกเป็น "0" และถ้าแฟล็กนี้มีลอจิกเป็น "1" ผลที่ได้จะมีค่าเป็นลบ

แฟล็ก A (Auxiliary Carry Flag; บิท 4) :

แฟล็กนี้จะถูกเซ็ทในกรณีที่ผลของการกระทำทางคณิตศาสตร์นั้น มีการยืมหรือทดจากบิท 3 (ในกรณีของข้อมูลขนาด 8 บิท)

แฟล็ก P (Parity Flag; บิท 6) :

แฟล็กนี้จะถูกใช้ในการแสดงว่าการประมวลผลข้อมูลนั้น ให้ผลลัพธ์ที่มีจำนวนบิทที่เป็น "1" (ใน 8 บิทล่าง; บิท 0-7) เป็นจำนวนคู่หรือคี่ โดยถ้าจำนวนบิทเป็น "1"ของผลลัพธ์นั้นเป็นจำนวนคี่ แฟล็กนี้จะถูเคลียร์เป็น "0" และถ้าจำนวนบิทที่เป็น "1" ของผลลัพธ์นั้นจะเป็นคู่ แฟล็กนี้จะถูกเซ็ทเป็น "1"

แฟล็ก Z(Zero Flag; บิท 6)

แฟล็กนี้จะถูกเซ็ท (ลอจิก "1") ในกรณีที่การหระมวลผลข้อมูลนั้นให้ผลลัพธ์เป็นศูนย์และขะถูกเคลียร์ (ลอจิก "0") ในกรณีที่การประมวลผลข้อมูลนั้นให้ผลลัพธ์ไม่เท่ากับศูนย์

แฟล็ก D(Direction Flag; บิท 10) :

แฟล็กนี้จะถูกใช้ในการทำงานในชุดคำสั่งสตริงก์ โดยทำหน้าที่กำหนดว่าระหว่างการทำงานในชุดคำสั่งสตริงก์นั้น 8088 จะทำการลด (Autodecrement) หรือเพิ่ม(Autoincrement) ค่าของรีจิสเตอร์ Index (รีจิสเตอร์ DI และ SI)

- ในกรณีที่แฟล็ก D นี้ถูกเซ็ทเป็น "1" 8088 จะทำการลดค่าของรีจิสเตอร์ DI และSI ลงโดยอันโนมัติ (ในระหว่างการทำงานในคำสั่งสตริงก์) ซึ่งในกรณีนี้การทำงานในคำสั่งสตริงก์จะเริ่มจากหน่วยความจำที่มีแอดเดรสสูงไปยังแอดเดรสที่ต่ำกว่า เช่นเริ่มจากแอดเดรส 30000H ไปยังแแอดเดรส 303FFH เป็นต้น

- ในกรณีที่แฟล็ก D นี้ถูกเซ็ทเป็น "0" 8088 จะทำการเพิ่มค่าของรีจิสเตอร์ DI และ SI ขึ้นโดยอัตโนมัติ (ในระหว่างการทำงานในคำสั่งสตริงก์) ซึ่งในกรณีนี้การทำงานในคำสั่งสตริงก์ก็จะเริ่มจากหน่วยความจำที่มีแอดเดรสต่ำไปยังแอดเดรสสูงกว่า เช่น เริ่มจากแอดเดรส 20000H ไปยังแอดเดรส 207FFFH เป็นต้น

แฟล็ก I (Interrupt Flag; บิท 9) :

สถานะของแฟล็กนี้ จะถูกใช้ในการควบคุมการยอมรับการขออินเทอร์รัพท์แบบ Maskable ของ 8088 โดยถ้าแฟล็กนี้ถูกเซ็ทเป็น "1" 8088 จะไม่ยอมรับ(Disable)

การขออินเทอร์รัพท์แบบ Maskable ที่อุปกรณ์ภายนอกส่งเข้ามา ในขณะที่ถ้าแฟล็กนี้ถูกเคลียร์เป็น "0" 8088 จะยอมรับ (Enable) การขออินเทอร์รัพท์แบบ Maskable ที่อุปกรณ์ภายนอกส่งเข้ามา

อย่างไรก็ตามแฟล็ก I นี้จะควบคุมได้เฉพาะการขออินเทอร์รัพท์แบบ Maskable เท่านั้น ถ้าเป็นการขออินเทอร์รัพท์แบบ Non-Maskable แล้ว 8088 จะต้องตอบรับในทุกกรณี (ไม่สามารถ Disable การขออินเทอร์รัพท์โดยทางซอฟท์แวร์ได้)

แฟล็ก T (Trap Flag; บิท 8) :

แฟล็กนี้จะถูกใช้ในกรณีที่ต้องการตรวจสอบ,แก้ไขโปรแกรมหรือระบบ เมื่อแฟล็ก T ถูกเซ็ทเป็น "1" 8088 จะเข้าสู่การทำงานในโหมด Single Step โดยในโหมดนี้ 8088 จะสร้าง Interrupt 1 ขึ้นทุกครั้งหลังจากทำการเอ็กซีคิวท์คำสั่งเสร็จ 1 คำสั่ง

การอ้างแอดเดรสในโหมดต่าง ๆของ 8088

สำหรับโหมดการอ้างแอดเดรสของ 8088 นั้น เราสามารถที่จะแบ่งออกได้เป็น 6 กลุ่ม ดังนี้ :

1. การอ้างแอดเดรสแบบ Immediate

การอ้างแอดเดรสแบบนี้จะใช้โอเปอร์แรนด์ที่ตามหลัง OP CODE เป็นข้อมูลโดยตรงเช่นคำสั่ง

```
ADD AX,4050H
```

เมื่อ 8088 พบคำสั่งนี้ก็จะทำการบวกค่า 4050 (ฐานสิบหก) กับค่าที่อยู่ในรีจิสเตอร์ AX และนำผลลัพธ์ที่ได้ไปเก็บในรีจิสเตอร์ AX

2. การอ้างแอดเดรสแบบ Direct (ข้อมูลในรีจิสเตอร์ DS +ค่าขจัด)

การอ้างแอดเดรสแบบนี้ 8088 จะทำการคำนวณแอดเดรสของข้อมูลโดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ และใช้ข้อมูล (ค่าขจัด; Displacement ขนาด 2 ไบท์หรือ 16 บิท)ที่เป็นโอเปอแรนด์ของคำสั่งนั้นเป็นแอดเดรสออฟเซ็ท เช่นคำสั่ง

```
ADD AX, [4050H]
```

ถ้ารีจิสเตอร์ DS มีข้อมูล 3000H 8088 ก็จะทำการรวมค่าของข้อมูลในรีจิสเตอร์ AX กับค่าของข้อมูลในแอดเดรส 34050H และเก็บผลลัพธ์นั้นไว้ที่รีจิสเตอร์ AX

3. การอ้างแอดเดรสแบบ Direct, Index (ข้อมูลในรีจิสเตอร์ DS + ค่าขจัด)

การอ้างแอดเดรสแบบนี้ 8088 จะทำการคำนวณแอดเดรสของข้อมูลโดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ ส่วนแอดเดรสออฟเซ็ทจะได้จากการนำเอาข้อมูลในรีจิสเตอร์ DI หรือ SI (ขึ้นอยู่กับการกำหนดในคำสั่ง) มาบวกกับข้อมูลที่เป็นค่าขจัดในโอเปอแรนด์คำสั่ง โดยค่าขจัดนี้อาจจะกำหนดเป็นข้อมูล 8 บิทหรือ 16 บิทก็ได้ สำหรับในกรณีที่กำหนดเป็นข้อมูล 8 บิทนั้น ในการนำเอาค่าขจัดนี้บวกกับข้อมูลในรีจิสเตอร์ zdi หรือ SI ซึ่งเป็นข้อมูล 16 บิท 8088 จะทำการขยายค่าขจัดให้เป็น 16 บิทโดยทุกบิทของไบท์สูง (High Order Byte; บิท 8-15) จะมีสถานะ (ลอจิก) เหมือนกับสถานะของบิทสูงสุด(บิท 7) ของค่าขจัดนั้น เช่น

ค่าขจัดที่กำหนด (8 บิท) : 1011 0110

หลังจากการขยายเป็น 16 บิท : 1111 1111 1011 0110

หรือ

ค่าขจัดที่กำหนด (8 บิท) : 0000 0000 0100 1011

เช่นคำสั่ง

```
ADD AX,[SI+05H]
```

ถ้าข้อมูลในรีจิสเตอร์ DS เป็น 3000H และรีจิสเตอร์ SI เป็น 4150H 8088 จะทำการคำนวณแอดเดรสของหน่วยความจำได้

```
 30000H
+ 4150H
 34155H
```

ดังนั้นเมื่อ 8088 พบคำสั่งนี้ก็จะทำการบวกค่าในรีจิเตอร์ AX กับข้อมูลในหน่วยความจำที่อยู่ในแอดเดรส 34155H และเก็บผลลัพธ์ที่ได้ไว้ในรีจิสเตอร์ AX

4. การอ้างแอดเดรสแบบ Implied(ข้อมูลในรีจิสเตอร์ DS+SI หรือ DI)

การอ้างแอดเดรสแบบนี้ 8088 จะทำการคำนวณแอดเดรสของข้อมูลโดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ และใช้ข้อมูลในรีจิสเตอร์ SIหรือ DI (ตามที่ได้กำหนดไว้ในคำสั่ง) เป็นแอดเดรสออฟเซ็ท เช่นคำสั่ง

ADD AX,[SI]

ถ้าข้อมูลในรีจิสเตอร์ DS เป็น 3000H และรีจิสเตอร์ SI เป็น 4150H 8088 จำทำการคำนวณแอดเดรสหน่วยความจำได้เป็น

30000H
4150H
34150H

ดังนั้นเมื่อ 8088 พบคำสั่งนี้ก็จะทำการบวกค่าในรีจิสเตอร์ AX กับข้อมูลในหน่วยความจำที่อยู่ในแอดเดรน 34150H และเก็บผลลัพธ์ที่ได้ไว้ในรีจิสเตอร์ AX

5. การอ้างแอดเดรสแบบ Base Relative

การอ้างแอดเดรสในลักษณะนี้จะมีรูปแบบการใช้งานอยู่ 2 ชนิดคือ

5.1 การอ้างแอดเดรนในหน่วยความจำ โดยใช้รีจิสเตอร์ BX ร่วมกับรีจิสเตอร์ DS

5.2 การอ้างแอดเดรสในหน่วยความจำส่วนที่เป็นสแต็ก โดยใช้รีจิสเตอร์ BP ร่วมกับรีจิสเตอร์ SS

สำหรับการอ้างแอดเดรสในหน่วยความจำนั้น จะมีลักษณะเหมือนกับการอ้างแอดเดรสแบบต่าง ๆที่กล่าวมาแล้วในข้อ 1-4 เพียงแต่ทำการเพิ่มรีจิสเตอร์ BX เข้าไปในการคำนวณตำแหน่งแอดเดรสเท่านั้น ดังนี้ :

- Base Relative Direct (ข้อมูลในรีจิสเตอร์ DS + BX)

8088 จะคำนวณตำแหน่งแอดเดรสของข้อมูลโดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซ็กเมนต์ และใช้ข้อมูลในรีจิสเตอร์ BX เป็นแอดเดรสออฟเซท

- Base Relative Direct (ข้อมูลในรีจิสเตอร์ DS + BX + ค่าขจัด)

8088 จะคำนวณตำแหน่งแอดเดรสของข้อมูล โดยข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ ส่วนแอดเดรสออฟเซ็ทจะได้จากการนำเอาข้อมูลในรีจิสเตอร์ BX บวกับค่าขจัด

- Base Relative Direct, Index (ข้อมูลในรีจิสเตอร์ DS +BX + SI หรือ DI + ค่าขจัด

8088 จะคำนวณตำแหน่งแอดเดรสของข้อมูล โดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนคชต์ ส่วนแอดเดรสออฟเซ็ทจะได้จากกการนำเอาข้อมูลในรีจิสเตอร์ BX บวกกับข้อมูลในรีจิสเตอร์ SI หรือ DI และค่าขจัด

- Base Relative Implied (ข้อมูลในรีจิสเตอร์ DS+BX+ SI หรือ DI)

8088 จะคำนวณตำแหน่งแอดเดรสของข้อมูล โดยใช้ข้อมูลในรีจิสเตอร์ DS เป็นแอดเดรสเซกเมนต์ ส่วนแอดเดรสออฟเซทจะได้จากการนำเอาข้อมูลในรีจีสเตอร์ BX บวกกับข้อมูลในรีจิสเตอร์ SI หรือ DI

ส่วนการอ้างแอดเดรสในหน่วยความจำส่วนที่เป็นสแต็กนั้น จะเป็นการช่วยให้เราสามารถที่จะประมวลข้อมูลในตำแหน่งใดของหน่วยความจำส่วนที่เป็นสแต็กก็ได้ โดยการอ้างแอดเดรสแบบนี้ก็จะมีลักษณะเหมือนกับ การอ้างแอดเดรสในแบบต่าง ๆที่กล่าวมาแล้วในข้อ 1-4 เพียงแต่แอดเดรสเซกเมสต์ที่ใช้จะเป็นข้อมูลจากรีจิสเตอร์ SS แทนรีจิสเตอร์ DS และเพิ่มรีจีสเตอร์ BP เข้าไปด้วย

สำหรับในตารางข้างล่างนี้จะแสดงรูปแบบต่าง ๆในการอ้างแอดเดรสของ 8088

Displacement Only	
Base or Index Only	(BX,BP,SI,DI)
Displacement + Base or Index	(BX,BP,SI,DI)
Base + Index	BP + DI, BX + SI
	BP + SI, BX + DI
Displacement + Base + Index	BP + DI + DISP
	BX + SI + DISP
	BP + SI + DISP
	BX + DI + DISP

(Courtesy Intel Corp.)

คำสั่งของ 8088

สำหรับในที่นี่จะไม่กล่าวถึงรายละเอียดในคำสั่งต่าง ๆของ 8088 แต่จะแสดงเฉพาะชุดคำสั่งที่มีอยู่ใน 8088 เท่านั้น

2.6 TIMER

8253-5 Programmable Interval Timer

8253-5 เป็นชิพชัพพอร์ทในการสร้างฐานเวลาต่าง ๆให้กับระบบ และภายใน 8253-5 จะประกอบด้วยแชนแนลที่ทำงานเป็นอิสระต่อกันอยู่ 3 แชนเนล ซึ่งการที่แชนแนลทั้ง 3 ของ 8253-5 จะทำงานตามที่เราต้องการได้นั้น เราจะต้องทำการโปรแกรมแชนแนลเหล่านี้เสียก่อน

สำหรับการโปรแกรม 8253-5 นั้นจะทำได้ในลักษณะเดียวกับชิพชัพพอร์ทอื่น ๆ ที่ได้กล่าวถึงมาแล้วในบทที่ผ่านมา คือ ในคำสั่ง OUT ส่งข้อมูล (คำสั่ง) ให้กับรีจิสเตอร์ต่าง ๆของ 8053-5 ซึ่งรีจิสเตอร์ของ 8253-5 จะแบ่งออกได้เป็น 2 ประเภทคือ รีจิสเตอร์ Mode Control และ รีจิสเตอร์ Counter ดังนั้นก่อนที่จะศึกษาถึงวิธีการโปรแกรม 8253-5 จึงจำเป็นต้องทำความเข้าใจกับหน้าที่ของบิทต่าง ๆภายในรีจิสเตอร์ทั้ง 2 นี้เสียก่อน

รีจิสเตอร์ Counter

ภายใน 8253-5 จะมีรีจิสเตอร์ Counter ซึ่งเป็นรีจิสตอร์ขนาด 16 บิท อยู่ 3 ตัว โดยข้อมูลในรีจิสเอตร์แต่ละตัวจะควบคุมช่วงเวลาของเอาท์พุทในแต่ละแชนแนล (รีจิสเตอร์ Counter ทั้ง 34 นี้จะทำงานเป็นอิสระต่อกัน)

เมื่อ 8253-5 เริ่มต้นการทำงานในแชนแนลใดแล้ว ข้อมูลในรีจิสเตอร์ Counter ของแชนแนลนั้นจะถูกลดค่าลง 1,2, หรือ 3(ขึ้นอยู่กับโหมดการทำงาน ซึ่งจะกล่าวถึงในภายหลัง) เมื่อแชนแนนนั้นได้รับคล็อกแต่ละลูก ข้อมูลในรีจิสเตอร์ Counter จะถูกลดลงจนมีค่าเป็น "0" (Terminal Counter) จากนั้นจึงจะเกิดการเปลี่ยนแปลงทางด้านเอาท์พุทและการทำงานของแชนแนนนั้น (ขอดูรายละเอียดในคำอธิบายเกี่ยวกับการทำงานในแต่ละโหมด) สำหรับการลดค่าของรีจิสเตอร์ Counter นี้ อาจจะลดค่าลงแบบ BCD หรือ Binary ก็ได้ ขึ้นอยู่กับการเซ็ทข้อมูลในบิท DO ของรีจิสเตอร์ Mode Countrol

เนื่องจากรีจิสเตอร์ Counter เป็นรีจิสเตอร์ขนาด 16 บิท แต่บัสข้อมูลของ 8253-5มีขนาดเพียง 8 บิท ดังนั้นการเขียนหรืออ่านข้อมูลจากรีจิสเตอร์ Counter จึงต้องทำครั้งละ 8 บิทด้วย โดยข้อมูลในรีจิสเตอร์ Counter จะถูกแบ่งออกเป็น 2 ส่วน คือ ส่วน 8 บิทบน (Most Significant Bit; D15-D8)และ 8 บิทล่าง(Least Significant Bit; D7-DO) ส่วนการที่จะกำหนดว่าข้อมูลที่ส่งให้

กับรีจิสเตอร์ Counter นี้เป็นข้อมูลสำหรับ 8 บิทบนหรือ 8 บิทล่างนั้น เราจะกำหนดได้โดยการเซ็ทค่าในบิท D5 และ D4 ของรีจิสเตอร์ Mode Control นอกจากนี้เรายังสามารถใช้ข้อมูลในบิททั้งสองนี้ในการกำหนดว่าการโปรแกรมรีจิสเตอร์ Counter นี้จะทำการโปรแกรมเฉพาะไบท์ใดไบท์หนึ่ง (เฉพาะ 8 บิทบนหรือ 8 บิทล่าง) หรือโปรแกรมทั้ง 8 บิทบนและ 8 บิทล่างได้อีกด้วย (ดูรายละเอียดใน "รีจิสเตอร์ Mode Control")

อีกสิ่งหนึ่งที่จะต้องคำนึงถึงก็คือ ในการโปรแกรมรีจิสเตอร์ Counter นั้น รีจิสเตอร์ Counter จะรับข้อมูลที่เราส่งไปให้ก็ต่อเมื่อสัญญาณที่ขา CLK ถูกเปลี่ยนจากลอจิก "0" เป็น "1"(ขอบขาขึ้น) และเปลี่ยนจากลอจิก "1" กลับเป็นลอจิก "0" (ขอบขาลง) อีกครั้งเสียก่อน (จะกล่าวถึงการโปรแกรมรีจิสเตอร์นี้ในภายหลัง)

สำหรับใน IBM/PC นั้นโปรแกรมในส่วนนี้ BIOS จะทำหน้าที่นี้เองโดยอัตโนมัติส่วนวิธีในการรีจิสเตอร์ Mode Control นี้จะทำได้โดยใช้คำสั่ง OUT ส่งข้อมูลไปยังพอร์ทของ 5253-5 ที่มีแอดเดรส A0 และ A1 เป็น "1" ทั้งคู่ (สำหรับการโปรแกรม 8253-5 จะกล่าวโดยละเอียดในภายหลัง) ซึ่งใน IBM /PC จะตรงกับพอร์ท 0043H สำหรับหน้าที่ของบิทต่าง ๆในรีจิสเตอร์ Mode Control นั้น จะแสดงได้ดังนี้

บิท	D7	D6	D5	d4	d3	d2	d1	d0
หน้าที่	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

บิท D7-D6 : ข้อมูลในบิททั้งสองนี้จะใช้สำหรับเลือกแชนแนนที่ต้องการ ซึ่งจะแสดงได้ดังนี้

(SC1,SC0): (เนื่องจาก 8253-5 มีเพียง 3 แชนแนน ดังนั้นกรณีที่ข้อมูลในบิท SC1 และ SC0 เป็น "1" ทั้งคู่จึงไม่มีผลในการเลือกแชนแนนใด ๆ)

SCI	SCO	แชนแนนที่ถูกเลือก
0	0	0
0	1	1
1	0	2
1	1	-

บิท d5-d4 : ข้อมูลในการกำหนดไบท์ (8 บิทบนหรือ 8 บิทล่าง) ของข้อมูลที่ต้องการจะอ่านจากรีจิสเตอร์ Counter ซึ่งเป็นรีจิสเตอร์ขนาด 16 บิท และใช้ในการกำหนดว่าโปรแกรมรีจิสเตอร์ Counter นี้จะทำการโปรแกรมเฉพาะที่ไบท์ใดไบท์หนึ่ง หรือโปรแกรมทั้ง 2ไบท์ นอกจากนี้ยังใช้ในการแลกท์ช(Latch ค่าในรีจิสเตอร์ Counter ได้อีกด้วย (ดูรายละเอียดในหัวข้อ "การโปรแกรมและอ่านข้อมูลจากรีจิสเตอร์ของ8253-5)" สำหรับผลจากการเซ็ทข้อมูลภายในบิท D5 และ D4 นั้นจะต้องแสดวได้ดังนี้

RL1	RL0	
0	0	ทำการแลกท์ช่ค่าในรีจิสเตอร์-เคาน์เตอร์
0	1	อ่าน/เขียนเฉพาะข้อมูลใน 8 บิทล่าง(Least-significant Bit)
1	0	อ่าน/เขียนเฉพาะข้อมูลเฉพาะใน 8 บิทบน (Most-Significant Bit)
1	1	อ่าน/เขียนเฉพาะข้อมูลทั้ง 16 บิท โดยเริ่มจาก 8 บิทล่างก่อนจากนั้นจึงอ่าน/เขียนข้อมูลใน 8 บิทบน

บิท D3-D1 : ข้อมูลในบิทนี้สำหรับเลือกโหมดการทำงานให้กับแชนแนนที่เรากำหนดในบิท (M2-M0 : SC1 และ SC0

สำหรับโหมดการทำงานของแชนแนนต่าง ๆใน 8253-5จะมี 6 โหมดดังนี้

M2	M1	M0	โหมดการทำงาน
0	0	0	โหมด 0:Interrupt On Terminal Count
0	0	1	โหมด 1:Programmable One-Shot
0	1	0	โหมด 2:Rate Generator
0	1	1	โหมด 3:Square Wavw Generator
0	0	0	หมด 4:Software Triggered Strobe
1	0	1	หมด 5:Hardware Triggered Strobe

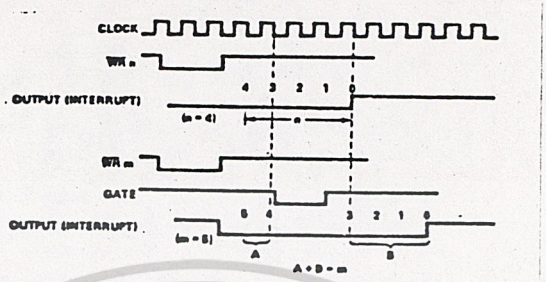
โหมด 0: Interrupt On Terminal Count

หลังจากที่แชนแนนใดถูกโปรแกรมให้ทำงานในโหมด 0 แล้ว (ข้อมูลในบิท D3-D1 ของรีจิสเตอร์ Mode Countrol ถูกเซ็ทเป็น "000") เอาท์พุทของแชนแนนนั้นก็จะมีระดับลอจิกเป็น "0" จากนั้นเมื่อค่า Counter ถูกส่งให้กับรีจิสเตอร์ Counter ของแชนแนนนั้นแล้ว แชนแนนนั้นก็จะเริ่มต้นนับจำนวนคล็อกที่ได้รับเข้ามา ในช่วงนี้เอาท์พุทของแชนแนนนั้นยังคงเป็นลอจิก "0" อยู่ซึ่งในการนับจำนวนคล็อกที่รับเข้ามานั้นข้อมูลในรีจิสเตอร์ Counter แล้ว (ข้อมูลในรีจิสเตอร์ Counter ถูกลดลงจนเป็น "0") เอาท์พุทของแชนแนนนั้นก็จะเปลี่ยนเป็นลอจิก "1" และจะคงสภาพลอจิก "1" นี้อยู่จนกว่าจะมีการส่งข้อมูลให้กับรีจิสเตอร์ Mode Control หรือรสีจิสเตอร์ Counter ใหม่

ถ้ามีการโหลดค่า Counter ใหม่ให้กับแชนแนนที่อยู่ในระหว่างการนับจำนวนคล็อก(ในโหมด 0)อยู่ จะเกิดผลกับการทำงานของแชนแนนนั้น ดังนี้

1. เมื่อมีการโหลดไบท์แรกของ Counter ค่าใหม่ให้กับแชนแนนที่อยู่ในระหว่างการนับจำนวนคล็อกอยู่ แชนแนนนั้นจะหยุดการนับ

2. เมื่อการโหลดไบท์ที่สองของ Counter ค่าใหม่ให้กับแชนแนนที่อยู่ระหว่างการานับจำนวนคล็อกอยู่ แชนแนนนั้นจะเริ่มต้นการนับใหม่ โดยใช้ Counter ค่าใหม่ที่ได้รับ



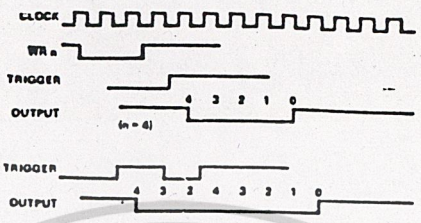
รูปที่ 2.6.1 ไดอะแกรมเวลาการทำงานของ 8253-5 ในโหมด 0

โหมด 1: Programmable One Shot

เมื่อแชนแนลใดของ 8253-5 ถูกโปรแกรมให้ทำงานในโหมดนี้ และค่า Counter ถูกโหมดให้กับรีจิสเตอร์ Counter แล้ว เอาท์พุทของแชนแนลนั้นจะถูกปรับให้เป็นลอจิก "1"จากนั้นเมื่อสัญญาอินพุทที่ขา GATE ของแชนแนลนั้นถูกเปลี่ยนจากลอจิก "0" เป็น "1"(ขอบขาขึ้น)สัญญาที่เอาท์พุทก็จะเปลี่ยนระดับเป็นลอจิก "0" และ จะเริ่มต้นนับจำนวนคล็อกที่ได้รับเข้ามาทางขา CLK ของแชนแนลนั้น เมื่อนับได้ครบตามจำนวนที่ได้กำหนดไว้ในรีจิสเตอร์ Counter แล้ว สัญญาณที่เอาท์พุทจะเปลี่ยนกลับเป็นลอจิก "1" อีกครั้ง ถ้ามีการโหลด Counter ค่าใหม่ที่อยู่ให้กับแชนแนลที่อยู่ระหว่างการนับจำนวนคล็อก (ในโหมด 1) อยู่นั้น ค่า Counter ค่าใหม่นี้จะยังไม่มีผลใด ๆกับการนับของแชนแนนนั้น จนกว่าแชนแนลนั้นจะเสร็จจากการนับจำนวนคล็อกตามค่า Counter ที่เซ็ทไว้เดิมเสียก่อน

เมื่อสัญญาที่ขา GATE เปลี่ยนจากลอจิก "0" เป็ฯ "1" อีกครั้ง 8253-5 ก็จะเริ่มต้นการนับใหม่ โดยใช้ค่า Counter นั้น เราสามารถจะทำการอ่านออกมาได้ตลอดเวลา โดยไม่กระทบกับการนับของแชนแนลที่กำลังนับอยู่เลย

สำหรับอินพุทที่เข้ามาทริก(Trig) ทางขา GATE ของแชนแนลที่ทำงานในโหมด 1 นั้นจะมีลักษณะเป็น Retriggerable ด้วย ดังแสดงในไดอะแกรมเวลา (Timing Diagram) ข้างล่างนี้



รูปที่ 2.6.2 ไดอะแกรมเวลาการทำงานในโหมด 1 ของ 8253-5

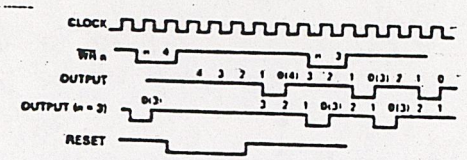
โหมด 2: Rate Generator

การทำงานในโหมดนี้จะเป็นลักษณะหาความถี่ของคล็อกที่ได้รับ ด้วยค่าที่โหลดให้กับรีจิสเตอร์ Counter กล่าวคือสัญญาณที่เอาท์พุทของแชนแนลที่ทำงานในโหมดนี้จะมีระดับลอจิกเป็น "1" และ "0" สลับกันไป โดยที่ความกว้างของช่วงเวลาที่ลอจิก "1" จะเท่ากับช่วงเวลาของคล็อกจำนวน N-1 ลูก (ค่า N ในที่นี้จะแทนค่า counter ที่โหมดให้กับรีจิสเตอร์ Counter) และความกว้างของช่วงเวลาที่ลอจิก "0" จะเท่ากับช่วงเวลาของคล็อกจำนวน 1 ลูก ดังนั้นคาบเวลาของเอาท์พุทจากแชนแนลที่ทำงานในโหมดนี้จึงเท่ากับช่วงเวลาของคล็อก N ลูก ซึ่งก็คือความถี่ของเอาท์พุทจากแชนแนลนี้ จะเท่ากับความถี่ที่ได้รับหารด้วย N นั่นเอง

ตัวอย่างเช่น ถ้าสัญญาณคล็อกที่ป้อนให้กับแชนแนลที่ทำงานในโหมด 2 มีความถี่ 1.19318 MHz หรือมีคาบเวลา (ช่วงเวลาของคล็อก 1ลูก)ประมาณ 838 nanosec. และ เราทำการโหลดค่า 04H ให้กับรีจิสเตอร์ Counter ของแชนแนลนี้ เอาท์พุทของแชนแนลนี้ เอาท์พุทของแชนแนลนี้ก็จะมีค่าความถี่ประมาณ 298.3 KHz หรือมีคาบแวลาประมาณ 3.352 usec. โดยใน 1 คาบจะมีช่วงเวลาที่เป็นลอจิ "1" เท่ากับช่วงเวลาของคล็อก 3 ลูก หรือเท่ากับ 2.514 usec. และช่วงเวลาที่เป็นลอจิก "0" เท่ากับช่วงเวลาของคล็อก 1 ลูก หรือประมาณ 838 nanosec.

ในระหว่างคาบเวลาของเอาท์พุทจากแชนแนลที่ทำงานในโปมด 2 ถ้าเราทำการโหลด Counter ค่าใหม่ให้กับรีจิสเตอร์ Counter ของแชนแนลนั้น ค่า counter ที่โหลดเข้าไปใหม่ จะมีผลต่อคาบเวลาต่อไปเท่านั้น โดยจะไม่มีผลต่อคาบ

เวลาของเอาท์พุทที่แชนแนลนี้กำลังสร้างอยู่



รูปที่ 2.6.3 ไดอะแกรมเวลาการทำงานในโหมด 2 ของ 8253-5

จากไดอะแกรมข้างต้นนั้น ในขั้นแรกเราทำการโหลดค่า 04H ให้กับรีจิสเตอร์ Counter ดังนั้นเอาท์พุทของแชนแนลนี้จึงมีคาบเวลาเท่ากับ ช่วงเวลาของคล็อก 4 ลูก ต่อมาจึงโหลด Counter ค่าใหม่ คือ 03H ให้กับรีจิสเตอร์ counter โดยทำการโหลดในระหว่างคาบเวลาของเอาท์พุทของแชนแนลนี้ ซึ่งจะเห็นได้ว่าเมื่อทำการโหลด Counter ค่าใหม่เสร็จแล้วคาบแวลาของ เอาท์พุทนี้ยังคงไม่เปลี่ยนแปลงคือ มีช่วงเวลาเท่ากับช่วงเวลาของคล็อก 4 ลูกอยู่แต่ในคาบเวลาถัดไปจะมีช่วงเวลาที่เปลี่ยนไป คือ จะมีช่วงเวลาเท่ากับคล็อกเพียง 3 ลูก (ช่วงเวลาที่เป็นลอจิก"1" เท่ากับช่วงเวลาของคล็อก 2 ลูก และช่วงเวลาที่เป็น "0" เท่ากับช่วงเวลาของคล็อก 1 ลูก) ตามที่ได้โหลดเข้าไปใหม่

สำหรับระดับของสัญญาณอินพุทที่ขา GATE ของแชนแนลที่ทำงานในโหมด 2 นี้ จะมีหน้าที่ในการควบคุมการทำงานของแชนแนลนี้ด้วย โดยถ้าสัญญาณอินพุทที่ขา GATE นี้มีลอจิกเป็น "0"(Low) เอาท์พุทของแชนแนลที่ทำงานในโหมด 2 จะเป็นลอจิก "1"ตลอดเวลา และแชนแนลนั้นจะเริ่มต้นการทำงาน(เริ่มการนับ) เมื่อระดับลอจิกของสัญญาณอินพุทที่ขา GATE นี้เปลี่ยนเป็น "1" ดังนั้นจึงอาจจะกล่าวได้ว่าหน้าที่ของขา GATE นี้ก็คือ ใช้ในการอีนาเบิลหรือดิสเอเบิลการทำงานของแชนแนลนี้นั่นเอง และด้วยเหตุนี้เราจึงสามารถจะใช้ขา GATE นี้ในการ Synchronize การทำงานของแชนแนลที่ทำงานในโหมด 2 กับสัญญาณภายนอกได้

อีกสิ่งหนึ่งที่จะกล่าวถึงก็คือ หลังจากที่เราเซ็ทการทำงานในโหมด 2 นี้ให้กับแชนแนลใดแล้ว แชนแนลนั้นก็จะไม่เริ่มต้นการทำงาน (เอาท์พุตจะยังเป็นลอจิก "1" อยู่ตลอดเวลา) จนกว่าเราจะโหลดค่า Counter ให้กับรีจิสเตอร์ Counter ของแชนแนลนั้นเสียก่อน (ถึงแม้ว่าสัญญาณอินพุทที่ขา GATE จะมีระดับลอจิกเป็น "1"

แล้วก็ตาม) ด้วยเหตุนี้เราจึงสามารถจะ Synchronize การทำงานของแชนแนลนี้กับซอฟท์แวร์ที่เราเขียนขึ้นได้

โหมด 3: Square Wave Rate Generator

การทำงานในโหมดนี้จะมีลักษณะคล้ายกับโหมด 2 คือเป็นการหารความถี่ของสัญญาณคล็อกที่ป้อนให้กับแชนแนลที่ทำงานในโหมดนี้ด้วย N (ค่า Counter ที่โหลดให้กับรีจิสเตอร์ Counter แต่จุดที่แตกต่างกันก็คือ ช่วงเวลาที่เอาท์พุทเป็นลอจิก "1" และลอจิก"0" ใน 1 คาบจะเท่ากัน (หรือเกือบเท่ากัน) ในขณะที่โหมด 2 นั้น ช่วงเวลาที่เอาท์พุทเป็นลอจิก "0" จะนานเท่ากับช่วงเวลาของคล็อกเพียง 1 ลูกเท่านั้นไม่ว่าค่า Counter จะมีค่าเปลี่ยนแปลงไปอย่างไรก็ตามการทำงานของแชนแนลที่ทำงานในโหมด 3 จะแบ่งออกได้เป็น 2 ลักษณะตามค่าของ Counter ที่โหลดให้กับรีจิสเตอร์ Counter (ในที่นี้จะแทนค่า Counter ด้วยตัวแปร N) ดังนั้น

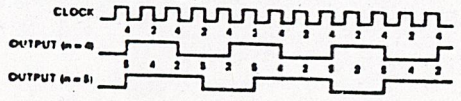
1. **ถ้า Nเป็นเลขคู่:** เอาท์พุทจะมีช่วงเวลาที่เป็นลอจิก"1"นานเท่ากับช่วงเวลาของคล็อก N/2 ลูก และจะเป็นลอจิก "0" อยู่นานเท่ากับช่วงเวลาของคล็อก N/2 ลูกเช่นกัน ซึ่งก็คือใน 1 คาบจะมีช่วงเวลานานเท่ากับช่วงเวลาของคล็อก N ลูก โดยมีค่า Cycle (ช่วงเวลาที่เป็นลอจิก "1"หารด้วยช่วงเวลาใน 1 คาบ) เท่ากับ 50% นั่นเอง

สำหรับการทำงานของแชนแนลที่อยู่ในโหมด 3 และค่า N ที่โหลดให้กับรีจิสเตอร์ Counter เป็นเลขคู่ จะอธิบายโดยย่อได้ดังนี้คือ หลังจากเริ่มการทำงานเอาท์พุทของแชนแนลนี้เป็นลอจิก "1" จากนั้นทุกครั้งที่คล็อกที่ป้อนให้กับแชนแนลนี้เปลี่ยนระดับลอจิกจาก "1" เป็น "0" (ขอบขาลง) ค่าในรีจิสเตอร์ Counter จะถูกลดค่าลง 2 ซึ่งการลดสค่าในรีจิสเตอร์ Counter นี้จะดำเนินไปจนกระทั่งรีจิสเตอร์ Counter มีข้อมูลเป็น "0" (Terminal Count) ในช่วงนี้เอาท์พุทจะเปลี่ยนเป็นลอจิก "0" และรีจิสเตอร์ Counter จะถูกเปลี่ยนข้อมูลเป็นค่า Counter เริ่มต้นที่เราส่งให้ก่อนที่จะเริ่มทำงาน(ในที่นี้คือค่า N) โดยอัตโนมัติ จากนั้นเมื่อสัญญาณคล็อกเปลี่ยนจากลอจิก "1" เป็น "0"(ขอบขาลง) ค่าในรีจิสเตอร์ Counter ก็จะถูกลดลง 2 อีกจนกระทั่งค่าในรีจิสเตอร์ Counter เป็น "0" เอาท์พุทจึงกลับเป็นลอจิก "1" อีกครั้งค่า N ก็จะถูกโหลดให้กับรีจิสเตอร์ Counter อีกโดยอัตโมัติ จากนั้นจึงทำงานตามขั้นตอนที่ได้กล่าวมานั้นต่อไป

2. **ถ้า N เป็นเลขคี่** : ในกรณีนี้ช่วงเวลา 1 คาบของเอาท์พุทจะมีช่วงเวลาที่เป็นลอจิก "1" อยู่นานเท่ากับช่วงเวลาของคล็อก(N+1)/2 ลูก

สำหรับการทำงานที่แชนแนลของแชนแนลที่ถูกโปรแกรมให้ทำงานในโหมด 3 และค่า N ที่โหลดให้กับรีจิสเตอร์ Counter เป็นเลขคี่นั้น จะมีดังนี้คือ ขณะที่เริ่มต้นการทำงานในโหมด 3 เอาท์พุทจะมีระดับลอจิกเป็น "1"และเมื่อได้รับ คล็อกลูกแรกข้อมูลในรีจิสเตอร์ counter จะถูกลดลง "1" จากนั้นเมื่อแชนแนลนี้ได้รับคล็อกลูกต่อไป ข้อมูลในรีจิสเตอร์ Counter ก็จะถูกลดลงครั้งละ"2" จนกระทั่งข้อมูลในรีจิสเตอร์ Counter เป็น "0" ในช่วงนี้เอาท์พุทจะถูกเปลี่ยนระดับลอจิกเป็น "0" และข้อมูลในรีจิสเตอร์ Counter จะถูดโหลดด้วยค่าเริ่มต้น ในที่นี้ค่า N)โดยอัตโนมัติ จากนั้นเมื่อได้รับคล็อกลูกแรก (หลังจากที่รีจิสเตอร Counterถูกโหลดด้วยค่าเริ่มต้น) ข้อมูลในรีจิสเตอร์ Counter จะถูกลดลง "3" และเมื่อแชนแนลนี้ได้รับคล็อกลูกต่อไป ข้อมูลในรีจิสเตอร์ Counter ก็จะถูกลดลงครั้งละ "2" จนกระทั่งค่าในรีจิสเตอร์ Counter เป็น "0" แชนแนลนี้ก็จะเริ่มต้นการทำการทำงานตามขั้นตอนแรกใหม่ต่อไป(เอาท์พุทเปลี่ยนกลับลอจิก "1" และโหลดค่า N ให้กับรีจิสเตอร์ Counter) ด้วยวิธีการนี้จึงทำให้เอาท์พุทที่ได้จากแชนแนลที่ทำงานในโหมด 3 เป็นสัญญาณคล็อกที่มีความถี่เท่ากับ ความถี่ของสัญญาณคล็อกที่ป้อนให้กับแชนแนลนั้นหารด้วย N โดยมีช่วงเวลาที่เป็นลอจิก "1"เท่ากับช่วงเวลาของสัญญาณคล็อก(N+1)/2 ลูก และช่วงเวลาที่เป็นลอจิก "0" เท่ากับช่วงเวลาของคล็อก (N-1)/2 ลูก ดังที่ได้กล่าวมาแล้ว

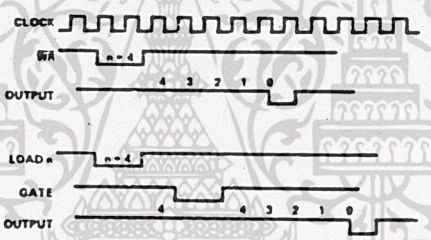
ตัวอย่างเช่น ถ้าสัญญาณคล็อกที่ป้อนให้กับแชนแนลที่ทำงานในโหมด 3 มีความถี่ 1.19318 MHz หรือมีคาบเวลา (ช่วงเวลาของคล็อก 1 ลูก) ประมาณ 838 nanosec. ถ้าเราทำการโหมดค่า 04H (เลขคู่) ให้กับรีจิสเตอร์ Counter ก็จะทำให้เอาท์พุทที่ได้จากแชนแนลนี้มีช่วงเวลาที่เป็นลอจิก "1" และ "0" เท่ากันคือ เท่ากับช่วงเวลาของคล็อก 4/2=2 ลูก หรือ ประมาณ 838*2 = 1.676 usec. แต่ถ้าเราทำการโหลดค่า 05H(เลขคี่) ให้กับรีจิสเตอร์ Counter แล้ว ก็จะทำให้เอาท์พุทที่ได้จากแชนแนลนี้มีช่วงเวลาที่เป็นลอจิก "1" เท่ากับช่วงเวลาของคล็อก (5+1)/2 = 3 ลูก หรือประมาณ 838*3 = 2.514 usec. และช่วงเวลาที่เป็นลอจิก "0" เท่ากับช่วงเวลาของคล็อก (5-1)/2 = 2 ลูก หรือประมาณ 838*2 = 1.676 usec. สำหรับไดอะแกรมข้างล่างนี้ จะแสดง Timing Diagram ในกรณีของตัวอย่างที่ได้กล่าวถึงในทั้งสองกรณี



รูปที่ 2.6.4 ไดอะแกรมเวลาการทำงานในโหมด 3 ของ 8253-5

โหมด 4: Software Triggered Strobe

หลังจากที่เราทำการเซ็ทโหมด 4 โหมดให้กับแชนแนลใดแล้ว แชนแนลนั้นจะเริ่มต้นการทำงานก็ต่อเมื่อเราโหลดค่า Counterให้กับรีจิสเตอร์ Counter แล้วเท่านั้น โดยเมื่อเราโหลดข้อมูลให้กับรีจิสเตอร์ Counter แล้ว เอาท์พุทของแชนแนลนั้นก็จะมีระดับลอจิกเป็น "1" และจะเริ่มต้นนับจำนวนคล็อกที่ได้รับเข้ามาทางขา CLK เมื่อครบตามจำนวนที่ได้กำหนดในรีจิสเตอร์ Counter แล้ว เอาท์พุทจะเปลี่ยนระดับลอจิกเป็น "0" โดยช่วงเวลาที่เอาท์พุทเป็นลอจิก "0" นี้จะนานเท่ากับช่วงเวลาของคล็อก 1 ลูก จากนั้นแชนแนลนั้นจะยุติการทำงานและเอาท์พุทก็จะกลับเป็นลอจิก "1" อีกครั้ง ดังแสดงในไดอะแกรมข้างล่างนี้

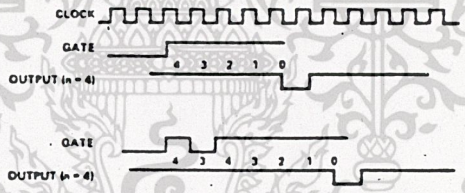


รูปที่ 2.6.5 ไดอะแกรมเวลาการทำงานในโหมด 4 ของ 8253-5

ในระหว่างที่แชนแนลที่ถูกเซ็ทให้ทำงานในโหมด 4 นี้กำลังทำงาน(นับจำนวนคล็อก) อยู่นั้นถ้าเราทำการโหลด Counter ค่าใหม่ให้กับรีจิสเตอร์ Counter ค่า Counter ที่โหลดเข้าไปใหม่นี้จะยังไม่มีผลต่อการทำงานของแชนแนลนี้(ช่วงเวลาก่อนที่เอาท์พุทจะเปลี่ยนเป็นลอจิก "0") จนกว่าแชนแนลนี้จะเสร็จจากการทำงาน (นับจำนวนคล็อก) ที่กำลังทำอยู่นี้เสียก่อน (คือหลังจากที่เอาท์พุทเปลี่ยนเป็นลอจิก "0" แล้วซึ่งจะเป็นการเริ่มต้นการทำงานใหม่ของแชนแนลนี้) และจากไดอะแกรมข้างต้นนั้น จะเห็นว่าในระหว่างที่มีการนับจำนวนคล็อกอยู่นั้น ถ้าสัญญาณ GATE ถูกเปลี่ยนเป็นลอจิก "0" จะทำให้การนับจำนวนคล็อกนั้นสิ้นสุดลง และ จะเริ่มต้นใหม่อีกครั้งหลังจากที่สัญญาณที่ขา GATE กลับเป็นลอจิก "1"

โหมด 5: Hardware Triggered Strobe

สำหรับแชนแนลที่ถูกโปรแกรมให้ทำงานโหมดนี้ จะเริ่มต้นการทำงานก็ต่อเมื่อสัญญาณที่ขา GATE ของแชนแนลนั้นถูกเปลี่ยนจากลอจิก "0" เป็นลอจิก "1"(ขอบขาขึ้น)เท่านั้น และเช่นเดียวกับในโหมด 4 คือเมื่อเริ่มทำงานเอาท์พุทของแชนแนลนี้จะมีระดับเป็นลอจิก "1" และเมื่อนับจำนวนคล็อกได้ครบตามจำนวนที่กำหนดในรีจิสเตอร์ Counter แล้ว เอาท์พุทจึงจะเปลี่ยนเป็นลอจิก "0" โดยช่วงเวลาที่เป็นลอจิก "0" นี้จะนานเท่ากับช่วงเวลาคล็อก 1 ลูก จากนั้นเอาท์พุทจึงกลับเป็น "1" อีกครั้ง สำหรับสัญญาณอินพุทที่ขา GATE นี้จะมีลักษณะเป็น Retriggerable กล่าวคือ แชนแนลที่ทำงานในโหมด 5 นี้จะเริ่มต้นการนับใหม่ทุกครั้งที่ระดับลอจิกที่ขา GATE ถูกเปลี่ยนจากลอจิก "0" เป็น "1" (ขอบขาขึ้น) และเอาท์พุทจะเป็น "0" ก็ต่อเมื่อแชนแนลนั้นเป็นจำนวนนับจำนวนคล็อกได้ครบตามจำนวนที่กำหนดในรีจิสเตอร์ Counter เท่านั้น สำหรับไดอะแกรมข้างล่างนี้จะแสดงให้เห็นถึงไดอะแกรมเวลาของการทำงานในโหมด 5 ที่ได้กล่าวถึงข้างต้น



รูปที่ 2.6.6 ไดอะแกรมเวลาการทำงานในโหมด 5 ของ 8253-5

บิท DO : ข้อมูลในบิทนี้จะให้สำหรับกำหนดว่าการลดค่าของข้อมูลในรีจิสเตอร์ Counter
BCD) : นั้นจะทำการลดในลักษณะของ BCD หรือ Binary โดยถ้าข้อมูลในบิทนี้ถูกเซ็ทเป็น "1" ค่าของข้อมูลในรีจิสเตอร์ Counter จะถูกลดลงในลักษณะของ BCD(Binary Code Decimal) และถ้าข้อมูลในบิทนี้ถูกเซ็ทเป็น "0" ค่าของข้อมูลในรีจิสเตอร์ Counter ก็จะถูกลดลงในลักษณะของ Binary

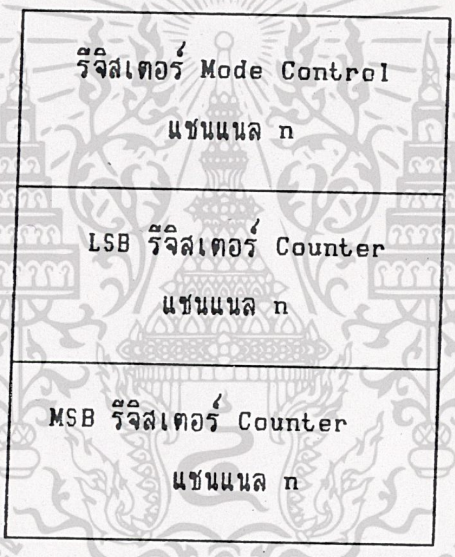
การโปรมแกรมและอ่านข้อมูลจากรีจิสเตอร์ 8253-5

ในหัวข้อนี้จะกล่าวถึงวิธีการโปรแกรมรีจิสเตอร์ Mode Control และรีจิส

เตอร์ Counter ซึ่งใช้สำหรับควบคุมการทำงานของแชนแนลทั้ง 3 ของ 8253-5 รวมถึงการอ่านค่า Counter จากรีจิสเตอร์ Counter ในระหว่างที่แชนแนลของ 8253-5 กำลังทำงานอยู่ด้วย

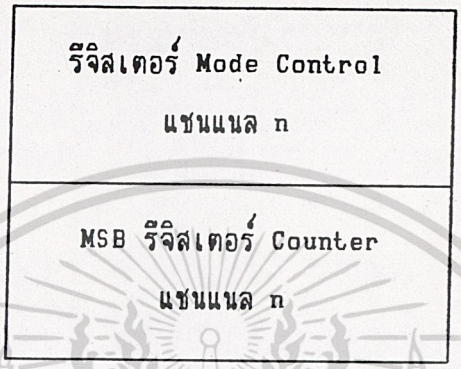
การอ่านโปรแกรมรีจิสเตอร์ 8253-5

ในการโปรแกรมแชนแนลทั้ง 3 ของ 8253-5 เพื่อกำหนดโหมดการทำงาน, จำนวนไบท์ของข้อมูลที่ต้องการจะส่งให้กับรีจิสเตอร์ Counter, ช่วงเวลาของเอาท์พุท และรูปแบบการลดค่าของข้อมูลในรีจิสเตอร์ Counter (BCD หรือ Binary) ของแต่ละแชนแนลเราจะต้องทำการโปรแกรมรีจิสเตอร์ของแชนแนลนั้น 2 รีจิสเตอร์ คือ รีจิสเตอร์ Mode Control และรีจิสเตอร์ Counter เสียก่อน



จากรูปข้างต้นนั้น จะแสดงลักษณะโดยทั่วไปของการโปรแกรมรีจิสเตอร์ Mode Countrol และรีจิสเตอร์ Counter ของแต่ละแชนแนลซึ่งจะเห็นได้ว่าการโปรแกรมแต่ละเชนแนลของ 8253-5 นั้นจะเริ่มจากการโปรแกรมรีจิสเตอร์ Mode Controlก่อน จากนั้นจึงทำการโปรแกรมรีจิสเตอร์ Counter ในภายหลัง สำหรับจำนวนไบท์(1 หรือ 2 ไบท์) หรือตำแหน่งของไบท์ที่ต้องส่งให้กับรีจิสเตอร์ Counter นั้นจะกำหนดโดยบิท RL0 และ RL1(บิท D4 และ D5)ของรีจิสเตอร์ Mode Control ดังที่ได้กล่าวถึงในหัวข้อที่ผ่านมา จากรูปข้างต้นนั้นจะแสดงกรณีที่บิท RL0 และ RL1 ของรีจิสเตอร์ Mode Control เป็น "1" ทั้งคู่ซึ่งทำให้เราต้องส่งข้อมูลให้กับรีจิสเตอร์ Couter 2 ไบท์ คือทั้งไบท์ที่เป็น 8 บิทบน (Most Significant Byte:MSB) และไบท์ที่เป็น 8 บิทล่าง(Least Significant Byte:LSB) ของรีจิสเตอร์

Counter โดยเริ่มจาก LSB ก่อนแล้วตามด้วย MSB แต่ถ้าข้อมูลในบิท RL0 และ RL1 เป็น "01" และ "10" แล้ว ลักษณะของการโปรแกรมจะเปลี่ยนไป ดังนี้ (ในทั้ง 2 กรณีหลังนี้ จะกำหนดให้มีการส่งข้อมูลให้กับรีจิสเตอร์ Counter เพียงไบท์เดียว(8บิท) คือ เฉพาะ MSB หรือ LSB เท่านั้น)



RL0=0 ; RL1=1

RL0=1 ; RL1=0

ในการโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนลทั้ง 3 ของ 8253-5 นั้น เราไม่จำเป็นต้องโปรแกรมเรียงตามลำดับ กล่าวคือ ไม่จำเป็นต้องโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนลที่ 0 ก่อน จากนั้นจึงโปรแกรมรีจิสเตอร์แชนแนลที่ 1 แล้วจึงตามด้วยแชนแนลที่ 2 แต่เราอาจจะโปรแกรมแชนแนลใดก่อนก็ได้ เช่น เราอาจจะโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนลที่ 2 ก่อนแชนแนลที่ 1 ก็ได้ (การจะเลือกโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนลใดนั้น เราสามารถกำหนดได้โดยการเซ็ทบิท SC0 และ SC1 ของรีจิสเตอร์ Mode Control ให้ตรง

กับแชนแนลที่ต้องการ)

หลังจากการโปรแกรมรีจิสเตอร์ Mode Control ในแชนแนลใดแล้ว ถ้าเราต้องการโปรแกรมรีจิสเตอร์ countrer ของแชนแนลนั้นก็สามารถจะทำได้ หรืออาจจะทำการโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนลอื่น ๆจนครบทั้ง 3 แชนแนลก่อนก็ได้ สำหรับลำดับของการโปรแกรมรีจิสเตอร์ Counter นั้น จะมีลักษณะเดียวกับการโปรแกรมรีจิสเตอร์ Mode Control คือเราจะเลือกโปรแกรมรีจิสเตอร์ Counter ของแชนแนลใดก่อนก็ได้ โดยไม่ขึ้นกับลำดับของการโปรแกรมรีจิสเตอร์ Mode Control เช่นเราอาจจะโปรแกรมรีจิสเตอร์ Mode Control ของแชนแนล 1 ก่อนแชนแนล 2 แต่โปรแกรมรีจิสเตอร์ Counter ของแชนแนล 2 ก่อนแชนแนล 1 ก็ได้

อย่างไรก็ตามการโปรแกรมรีจิสเตอร์ Mode Control และรีจิสเตอร์ counter นั้นยังมีข้อแตกต่างกันอยู่บ้าง แต่ก่อนที่จะกล่าวถึงข้อแตกต่างนี้จะอธิบายถึงการจัดแอดเดรสของ 8253-5 เสียก่อน สำหรับ 8253-5 นี้จะมีขาที่ใช้สำหรับต่อกับขาแอดเดรสของระบบอยู่ 2 ขา คือขา A1 และ A0(ขา 19 และ 20; ดูรายละเอียดการจัดเรียงขาในภาคผนวกท้ายเล่ม) ซึ่งจะทำให้ 8253-5 ใช้แอดเดรสทั้งสิ้น 4 แอดเดรส คือ แอดเดรสที่มีบิท A1 และ A0 เป็น "00","01", "10"และ "11" ตามลำดับ สำหรับใน IBM/PC นั้นจะจัดแอดเดรสไว้สำหรับ 8253-5 เป็น 0040H(บิท A1 เป็น "0"; บิท A0 เป็น"0"),0041H(บิท A1 เป็น "0"; บิท A0 เป็น"1"), 0042H(บิท A1 เป็น "1"; บิท A0 เป็น"0") และ 0043H(บิท A1 เป็น "1") แอดเดรสทั้ง 4 ของ 8253-5 นี้จะถูกใช้ในการโปรแกรมรีจิสเตอร์ Mode Control และ รีจิสเตอร์ Counter ซึ่งจะทำได้โดยการใช้คำสั่ง OUT ส่งข้อมูลไปยังแอดเดรสต่าง ๆเหล่านี้ ในตารางข้างล่างจะแสดงถึงการใช้งานแอดเดรสทั้ง 4 ในการโปรแกรมรีจิสเตอร์ Mode Control และ รีจิสเตอร์ Counter

แอดเดรส(ฐาน 16)	รีจิสเตอร์ที่ถูกโปรแกรม
0040	รีจิสเตอร์ Counter แชนแนล 0
0041	รีจิสเตอร์ Counter แชนแนล 1
0042	รีจิสเตอร์ Counter แชนแนล 2
0043	รีจิสเตอร์ Mode Control

จากตารางข้างต้นนั้น เราจะเห็นข้อแตกต่างในการโปรแกรมรีจิสเตอร์ Mode Control และรีจิสเตอร์ Couhter ได้อย่างชัดเจน กล่าวคือ การโปรแกรมรีจิสเตอร์ Mode Control ของทั้ง 3 แชนแนลจะทำโดยการส่งข้อมูลผ่านไปยังแอดเดรสของ 8253-5 เพียงแอดเดรสเดียว คือ แอดเดรสที่มีบิท A1 และ A0 เป็น "1"ทั้งคู่ (ใน IBM/PC คือแอดเดรส 0043H) และทำการเลือกแชนแนลโดยเซ็ทข้อมูลในบิท SC1 และ SC0 ให้ตรงกับแชนแนลที่ต้องการ แต่การโปรแกรมรีจิสเตอร์ Counter ของทั้ง 3 แชนแนลจะทำได้โดยการส่งข้อมูลไปยังแอดเดรสสำหรับรีจิสเตอร์เหล่านี้โดยตรง คือ แอดเดรสที่มีบิท A1 และ A0 เป็น "00"(ใน IBM/PC คือแอดเดรส 0040H) สำหรับแชนแนล 0, "01"(ใน IBM/PC คือแอดเดรส 0041H) สำหรับแชนแนล 1, "10"(ใน IBM/PC คือ แอดเดรส 0042H) สำหรับแชนแนน 2

สิ่งหนึ่งที่ต้องคำนึงถึงอยู่เสมอในการโปรแกรมรีจิสเตอร์ Counter ก็คือ ลำดับและจำนวนไบท์ของข้อมูลที่จะต้องส่งให้กับรีจิสเตอร์ Counter ซึ่งจะขึ้นอยู่กับการเซ็ทค่าของบิท RL1 และ RL0 ดังที่ได้กล่าวไว้แล้วในตอนต้นของบทนี้

สำหรับตารางข้างล่างนี้จะแสดงตัวอย่างลำดับการโปรแกรมรีจิสเตอร์ Mode Control และ รีจิสเตอร์ Counter โดยในตัวอย่างนี้จะทำการโปรแกรมรีจเตอร์ Mode Control ของแชนแนล 0,1 และ 2 ตามลำดับ จากนั้นจึงโปรแกรมรีจิสเตอร์ Counter ของแชนแนล 1,2 และ 0 ตามลำดับ โดยแต่ละแชนแนลนั้นจะโปรแกรมให้รีจิสเตอร์ Counter รับข้อมูลที่แตกต่างกัน คือ ในแชนแนลที่ 1 จะโปรแกรมให้รีจสเตอร์ Counter รับข้อมูล 2 ไบท์ (เซ็ทข้อมูลในบิท RL1 และ RL0 ของรีจิสเตอร์ Mode Control ของแชนแนล 1เป็น "1"ในฐานสอง) ดังนั้นจึงต้องทำการโปรแกรมรีจิสเตอร์ Counter 2 ครั้ง โดยในครั้งแรกทำการส่งข้อมูลที่ต้องการใช้เป็นข้อมูลสำหรับ 8 บิทล่าง(D7-D0;LSB) ของรีจิสเตอร์ Counter ของแชนแนลมรา 1 และต่อมาจึงทำการส่งข้อมูลที่ต้องการใช้เป็นข้อมูลสำหรับ 8 บิทบน(D15-D8;MSB) ของรีจิสเตอร์ Counter ของแชนแนลที่ 1

สำหรับในแชนแนลที่ 2 จะถูกโปรแกรมให้รีจิสเตอร์ Counter รับข้อมูลเพียงไบท์เดียว คือรับเฉพาะข้อมูลในส่วนที่จะใช้เป็นข้อมูลสำหรับ 8 บิทล่างของรีจิสเตอร์ Counter ของแชนแนลที่ 2 (ข้อมูลในบิท RL1 และ RL0 ในรีจิสเตอร์ Mode Ceontrol ของแชนแนลที่ 2 ถูกเซ็ทเป็น "01" ในฐานสอง) และในแชนแนลที่ 0 จะถูกโปรแกรมให้รีจิสเตอร์ Counter รับข้อมูลเพียงไบท์เดียว คือ รับเฉพาะข้อมูลในส่วนที่จะใช้เป็นข้อมูลสำหรับ 8 บิทบนของรีจิสเตอร์ Counter ของแชนแนลที่ 0 (ข้อมูลในบิท RL1 และ RL0 ในรีจิสเตอร์ Mode Control ของแชนแนลที่ 0 ถูกเซ็ทเป็น

"10"ในฐานสอง

ลำดับการโปรแกรม	A1	A0	แอดเดรสใน IBM/PC	ข้อมูล(D7-D0)
รีจิสเตอร์ Mode Conrrol				
แชนแนลที่ 0	1	1	0043	0010xxxx
รีจิสเตอร์ Mode Control				
แชนแนลที่ 1	1	1	0043	0111xxxx
รีจิสเตอร์ Mode Control				
แชนแนลที่ 2	1	1	0043	1001xxxx
LSB รีจิสเตอร์-เคาน์เตอร์				
แชนแนลที่ 1	0	1	0041	xxxxxxxx
MSB รีจิสเตอร์-เคาน์เตอร์				
แชนแนลที่ 1	0	1	0041	xxxxxxxx
LSB รีจิสเตอร์-เคาน์เตอร์				
แชนแนลที่ 2	1	0	0042	xxxxxxxx
MSB รีจิสเตอร์-เคาน์เตอร์				
แชนแนลที่ 0	0	0	0040	xxxxxxxx

การอ่านข้อมูลจากรีจิสเตอร์ Counter ของ 8253-5

ในระหว่างการทำงานของแต่ละแชนแนลนั้น เราอาจจะมีความจำเป็นต้องอ่านข้อมูล (ค่า Counter) ของแชนแนลที่กำลังทำงานอยู่ด้วย ซึ่งการอ่านข้อมูลในระหว่างนี้อาจจะทำให้เกิดความผิดพลาดของข้อมูลที่ 8253-5 ส่งออกมาได้ ถ้าหากว่าข้อมูลในรีจิสเตอร์ Counter กำลังถูกลดค่าอยู่ ดังนั้นเราจึงจำเป็นต้องใช้วิธีการบางอย่างเข้าช่วย เพื่อป้องกันปัญหาที่จะเกิดขึ้นได้ ซึ่งสามารถจะแบ่งออกได้เป็น 2 วิธี ดังนี้

1. หยุดการนับของแชนแนลที่ต้องการจะอ่านข้อมูลนั้นไว้ชั่วคราว โดยการควบคุมสัญญาณที่ป้อนให้กับขา GATE ของแชนแนลนั้น (ทำให้เป็นลอจิก "0") หรือโดยการควบคุมวงจรภายนอกไม่ให้ทำการส่งสัญญาณคล็อกให้กับแชนแนลนั้นได้ จากนั้น

เราก็สามารถจะทำการอ่านข้อมูลจากรีจิสเตอร์ Counter ได้ สำหรับวิธีการอ่านข้อมูลจากรีจิสเตอร์ Counter นั้นเราจะทำได้โดยการใช้คำสั่ง IN อ่านข้อมูลจากแอดเดรสของ 8253-5 ซึ่งมีอยู่ 4 แอดเดรสดังที่ได้กล่าวมาแล้ว โดยแอดเดรสทั้ง 4 นี้จะใช้ในการอ่านรีจิสเตอร์ Counter ของแชนแนลต่าง ๆดังตาราง

บิท A1	บิท A2	แอดเดรสใน IBM/PC	รีจิสเตอร์ที่ถูกอ่าน
0	0	0040	รีจิสเตอร์-เคาน์เตอร์ แชนแนล 0
0	0	0041	รีจิสเตอร์-เคาน์เตอร์ แชนแนล 1
1	0	0042	รีจิสเตอร์-เคาน์เตอร์ แชนแนล 2
0	1	0043	-------

ในกรณีที่แชนแนลที่เราต้องการจะอ่านค่าจากรีจิสเตอร์ Counter นั้น ถูกโปรแกรมให้การอ่านข้อมูลจากรีจิสเตอร์ Counter ต้องอ่านทีละ 2 ไบท์แล้ว(ข้อมูลในบิท RL1 และ RL0 ของรีจิสเตอร์ Mode Control ถูกเซ็ทเป็น "1"ทั้งคู่) เราจะต้องทำการอ่านข้อมูลจากรีจิสเตอร์ Counter ของแชนแนลนี้ให้ครบทั้ง 2 ไบท์ คือ จะต้องทำการอ่าน 2 ครั้ง โดยการอ่านครั้งแรกเป็นการอ่านไบท์ที่เป็น 8 บิทล่าง(D7-D0; LSB) และการอ่านครั้งที่สองเป็นการอ่านไบท์ที่เป็น 8 บิทบน(D15-D8; MSB) ของรีจิสเตอร์ Counter (การอ่านทั้งสองครั้งทำได้โดยการใช้คำสั่ง IN อ่านข้อมูลจากแอดเดรสของ 8253-5 ที่ตรงกับรีจิสเตอร์ Counter ของแชนแนลที่ต้องการ) ซึ่งในกรณีนี้ถ้าเรายังอ่านข้อมูลไม่ครบทั้ง 2 ไบท์แล้ว เราจะไม่สามารถใช้คำสั่ง OUT ในการโปรแกรมรีจิสเตอร์ Counter ของแนลนี้ได้

2. ในกรณีที่เราต้องการจะอ่านข้อมูลจากรีจิสเตอร์ Counter โดยไม่จำเป็นต้องหยุดการทำงานของแชนแนลนั้นไว้ ก็จะทำได้โดยการแลทช์ค่าของข้อมูลจากรีจิสเตอร์ Counter ไว้ก่อนที่จะทำการอ่านข้อมูลจากรีจิสเตอร์ Counter นั้น สำหรับการแลทช์ค่า Counter นี้จะทำได้โดยการใช้คำสั่ง OUT ส่งข้อมูลที่มีบิท D5 และ บิท D4 (บิท RL1และ RL0 เป็น"00" ในฐานสอง) ให้กับรีจิสเตอร์ Mode Control ของแชนแนลที่ต้องการ (การกำหนดแชนแนลจะทำได้โดยกำหนดทางบิท SC1และ SC0 ดังที่ได้กล่าวในตอนต้น)ซึ่งจะทำให้ค่า Counter ในรีจิสเตอร์Counter ถูกแลทช์เก็บเอา

ไว้ จากนั้นเมื่อเราทำการอ่านค่า Counter จากรีจิสเตอร์ Counter นี้ 8253-5 ก็จะส่งข้อมูลที่แลทช์ไว้ออกมายังบัสข้อมูลของระบบ

จะเห็นได้ว่าการใช้วิธีหลังนี้จะทำให้เราสามารถอ่านข้อมูลจากรีจิสเตอร์ Counter ได้โดยไม่จำเป็นต้องหยุดการทำงานของแชนแนลนั้นไว้ก่อน สำหรับวิธีในการอ่านข้อมูลจากรีจิสเตอร์ Counter นั้นจะเหมือนกับในกรณีแรกทุกประการ

สำหรับไดอะแกรมข้างล่างนี้จะแสดงการจัดเรียงบิทของรีจิสเตอร์ Mode Control ในกรณีที่ใช้สำหรับการแลทช์ค่า Counter ของรีจิสเตอร์ Counter (การโปรแกรมรีจิสเตอร์ Mode Control ในลักษณะนี้จะไม่มีผลกับโหมดการทำงานที่ได้เซ็ทไว้ คือ บิท D3 D0 จะไม่มีผลหรือหน้าที่ใด ๆ)

A0,A1 = 11

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	x	x	x	x

SC1,SC0 - specily counter to be latched.

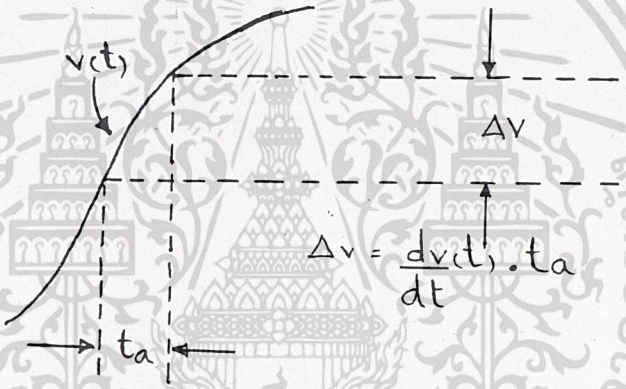
D5.D4 - counter latching operation

X - don't care.

2.7 ทษฎีการ Sempling

ในการแปลงสัญญาณอนาลอกเป็นระหัสดิจิตอลนั้น ADC จะต้องใช้เวลาช่วงหนึ่งในการจัดการ ซึ่งช่วงเวลาดังกล่าวนั้นขึ้นอยู่กับหลายแฟคเตอร์ เช่น ความละเอียดของการเปลี่ยนสัญญาณ เทคนิคของการเปลี่ยนแปลงสัญญาณ และความเร็วในการทำงานของอุปกรณ์ร่วมอื่น ๆความเร็วของการแปลงสัญญาณนี้จำเป็นสำหรับการประยุกต์ใช้งานเฉพาะอย่างและความแม่นยำที่ต้องการ

Aperture time : ช่วงเวลาในการแปลงสัญญาณบางครั้งอาจเรียกว่า Aperturetime ซึ่งคำว่า Aperture time โดยทั่วไปหมายถึงช่วงเวลาที่เกิดความไม่แน่นอนในการวัด และผลก็คือเกิด error ต่อค่าที่วัดได้



รูปที่ 2.7.1 แสดง error จากการวัดใน Aperture time

ในรูปที่ 2.7.1 สัญญาณอนาลอก V(t) มีอัตราการเปลี่ยน dv/dt ในช่วง Aperturetime t_a ดังนั้นช่วงการเปลี่ยแปลงแปลงอนาลอกจะเท่ากัย V โดย

$$V = t_a \frac{dv(t)}{dt}$$

ดังนั้นหากเวลาที่ ADC ใช้เปลี่ยนสัญญาณในเวลา t นี้ระหัสดิจิตอลที่ได้อาจจะตรงกับขนาดของสัญญาณอนาลอกค่าใดค่าหนึ่งในช่วงนี้ และส่วนอื่น ๆที่เหลือคือ error ที่เกิดขึ้นซึ่งแน่นอนบางครั้งเป็นไปได้ที่ระหัสดิจิตอลจะตรงกับค่าอนาลอกที่ถูกต้อง

ตัวอย่างในกรณีสัญญาณอินพุทเป็นรูปซายน์ อัตราการเปลี่ยนแปลงบนรูปคลื่นจะเกิดที่สุดตรงบริเวณจุดตัดแกนเวลารอบ ๆจุดศูนย์โวลท์(Zero Crossing)และ Aperture time error คือ

$$V = t_a \frac{d}{} (A\sin wt)\ t = 0$$

$$= t_a\ AW$$

และ error รวมคิดจากอัตราส่วนของขนาดเต็มสเกล คือ

$$= \frac{V}{2A} = f\ t_a$$

ดังนั้นหากต้องการเปลี่ยนสัญญาณเป็นรูปซายน์ความถี่ 1 กิโลเฮิร์ทให้เป็นสัญญาณดิจิตอล 10 บิท ซึ่งยอมให้ erreo ไม่เกินกว่า resolution (จะกล่าวถึงภายหลัง) คือ $1/2^{10}$ LSB หรือ 0.001 ดังนั้นเวลา Aperture time จะต้องอยู่ในช่วง

$$t_a = \frac{}{f} = 0.001 = 310*10^{-9} \text{ sec}$$

$$*10^{3}$$

จะเห็นว่าแม้ว่าสัญญาณ 1 กิโลเฮิร์ท จะไม่ใช้ความถี่สูงก็จริง แต่ ADC ที่ต้องใช้เวลาในการเปลี่ยนใน 230 นาโนวินาทีเป็นระหัส 10 บิทนั้นไม่ใช้หาได้ง่าย ๆซึ่งวิธีอื่นที่ไม่จำเป็นต้องใช้ ADC ความเร็วสูงเช่นนั้นคือ การใช้ Sample and hold ซึ่ง Sample and hold ที่มี Aperture time น้อย ๆนั้นทำได้ง่ายและราคาถูกกว่า

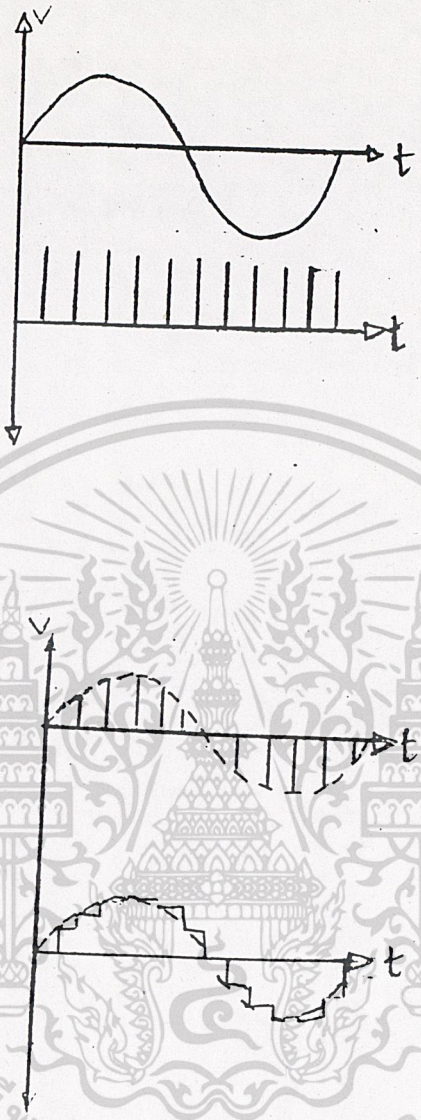
2.3 Sample and hold และ Aperture error

วงจร Sample and hold จะทำการสุ่มสัญญาณอินพุท และนำสัญญาณที่สุ่มนั้นมาเก็บหรือ hold ไว้ในเวลาหนึ่งได้ ซึ่งส่วนใหญ่จะใช้การประจุแรงดันนั้นในตัวเก็บประจุที่รั่วไหลดังนั้นในเมื่อแรงดันอินพุทสามารถคงอยู่ได้นานพอดังนั้น Aperture time ของ Sample & hold คือเวลาตั้งแต่เริ่มสุ่มสัญญาณจนเก็บประจุค่าแรงดันจนถึงค่าที่สุ่ม ซึ่งสำหรับ Sample and hold แล้ว Aperture time ขึ้นอยู่กับแบนด์วิดธ์ และ Switching time ของอุปกรณ์แอคทีฟ(จะกล่าวภายหลัง) ที่ใช้ในวงจรซึ่งหาและสร้างได้ง่ายและราคาถูกกว่าการสร้าง ADC ความเร็วสูง

ระบบ Sample ข้อมูลและทฤฎีการ Sample

ในระบบการ Sampling สัญญาณอนาลอกจะถูกสุ่มเป็นระยะ ๆคงที่ตามรูปที่ 3 กลุ่มขของสัญญาณสุ่มจะแทนวิทย์ที่ทำงานด้วยความเร็วสูง ซึ่งจะทำการตัดต่อสัญ

ญาณอนาลอกในช่วงเวลาอันสั้น



รูปที่ 2.7.2 การ Sampling สัญญาณ

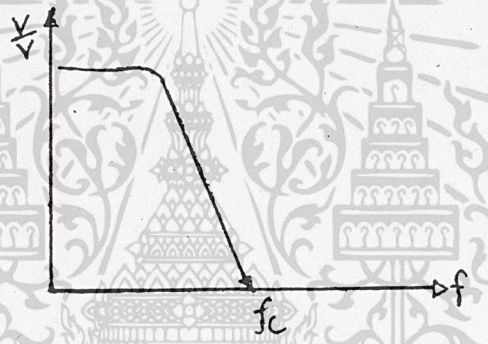
ผลของการสุ่มสัญญาณด้วยความเร็วจะเสมือนกับการคูณขบวนสัญญาณพัลส์แบบ ๆกับสัญญาณอนาลอก ซึ่งจะได้เป็นสัญญาณที่มอดดูเลทระหว่างขบวนพัลส์กับสัญญาณอนาลอกดังแสดงในรูปโดยสัญญาณอนาลอกจะขี่มาบนขบวนพัลส์ ถ้าหากเอาสวิทช์และตัวเก็บประจุแทนสวิทช์แล้วสัญญาณอนาลอกที่ถูกสุ่มจะถูก hold ไว้ในตัวเก็บประจุจนกว่าสัญญาณค่าใหม่ถูกสุ่มเข้ามาซึ่งลักษณะของเอาท์พุทที่ได้แสดงในรูป ง.

มีปัญหาที่ว่าการสุ่มสัญญาณนั้นควรมีขนาดเท่าใดที่จะไม่ทำให้ข้อมูลสูญเสียไปเมื่อสัญญาณนั้นถูกเปลี่ยนกลับมาเป็นเช่นเดิม คำตอบก็คือ ขึ้นอยู่กับความถี่ของสัญ

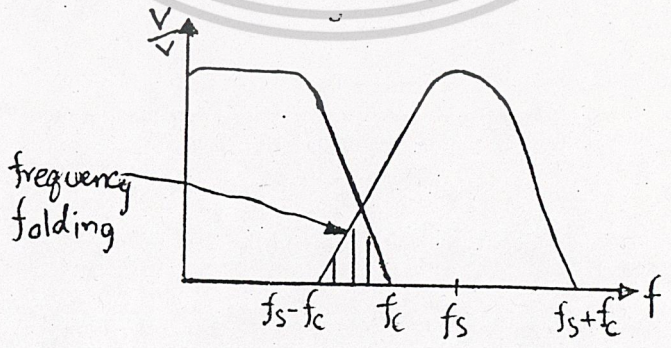
ญาณอนาลอกและทฤษฎีของการสุ่ม กล่าวไว้ว่า "ถ้าสัญญาณต่เนื่องซึ่งมีความถี่และฮาร์โมนิคไม่เกิน f_c และสัญญสณดังกล่าวจะสามารถเปลี่ยนกลับมาได้อย่างเดิมโดยไม่สูญเสียรายละเอียดหรือผิดเพี้ยนไป ถ้าอัตราการสุ่มไม่น้อยกว่า $2f_c$ ต่อวินาที"

2.4 Ferquency folding and Aliasing

จากทษฎีของการสุ่มสามารถอธิบายด้วยลักษณะสเปคตรัมของสัญญาณในรูปที่2.7 รูป 2.7.3 (ก) แสดงให้เห็นสเปคตรัมของสัญญาณที่ถูกสุ่มซึ่งเบนด์วิดธ์ไม่เกิน f_c ในขณะที่สัญญาณนี้จะถูกสุ่มด้วยความถี่ f_s ขบวนการมอดูเลชั่นจะทำให้แถบสเปคตรัมของสัญญาณสุ่มขยายกว้างออกจาก f_s เป็น $2f_s$ $3f_s$ ..... ได้เป็นดังรูป 2.7.3 ข



รูป 2.7.3 ก. แสงดสเปคตัมของสัญญาณอนาลอกที่จะถูกสุ่ม

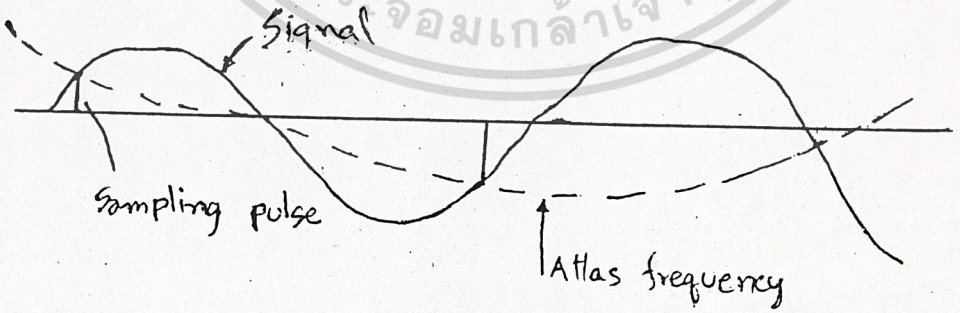


รูป 2.7.3 ข. หลังจากการสุ่มเกิด Frequency folding

ถ้าความถี่ของสัญญาณสุ่ม f_s ไม่สูงพอหลังจากการสุ่มสเปคตัมบางส่วนของ f_s จะหาซ้อนกลับสเปคตัมของสัญญาณซึ่งเรียกว่า Frequency Folding หากเป็นเช่นนี้ก็จะทำให้เกิดความเพี้ยนแก่สัญญาณอนาลอกจากการซ้อนกันของสเปคตัม เมื่อสัญญาณถูกเปลี่ยนกลับให้อยู่ในรูปเดิม และถ้าเลื่อนความถี่ของการสุ่มให้สูงขึ้นจนกว่าจนโอกาสการซ้อนของสเปคตัมหมดไป ($f_s - f_c = f_c$) และการเปลี่ยนกลับของสัญญาณหลังจากถูกสุ่มก็ยังเหมือนเดิมได้

จากที่กล่าวมาแสดงถึงการสนับสนุนของทฤษฎีการสุ่มที่ว่าให้ $f_s > 2f_c$ นั้นคือการกำจัดการการซ้อนกันของสเปคตัมได้สองวิธีคือ หนึ่งด้วยการใช้อัตราการสุ่มที่สูงพอและสองการทำการฟิล-เตอร์ความถี่ของสัญญาณอนาลอกก่อนการสุ่ม เพื่อให้แบนด์วิดธ์ไม่เกินไปกว่า Fs/2 ในทางปฏิบัติแล้วจะยังคงเกิด Frequency Folding ได้เสมอจากส่วนฮาร์โมนิคส์ของสัญญาณรวมทั้งสเปคตัมของสัญญาณรบกวนที่ยังคงอยู่ แม้ว่าทำการฟิวเตอร์ก่อนหน้ามาแล้วก็ตามการกำลังจัดซ้อนกันของสเปคตัมดังนี้วิธีที่ได้ผลคือพยายามให้การสุ่มสัญญาณเป็นไปอย่างรวดเร็วมากที่สุด ซึ่งปกติจะสูงกว่าความถี่ต่ำสุดตามทฤษฎีSampling คือ $2f_c$ เสมอ

ผลของการใช้อัตราการสุ่มที่ไม่เหมาะสม อีกประการหนึ่งที่เกิดสัญญาณรูปซายน์แสดงในรูปที่ 2.7.4 เรียกว่า Alias Frequency ซึ่งเกิดกับสัญญาณที่เปลี่ยนกลับมาเช่นเดิมหลังจากถูกสุ่มแล้ว

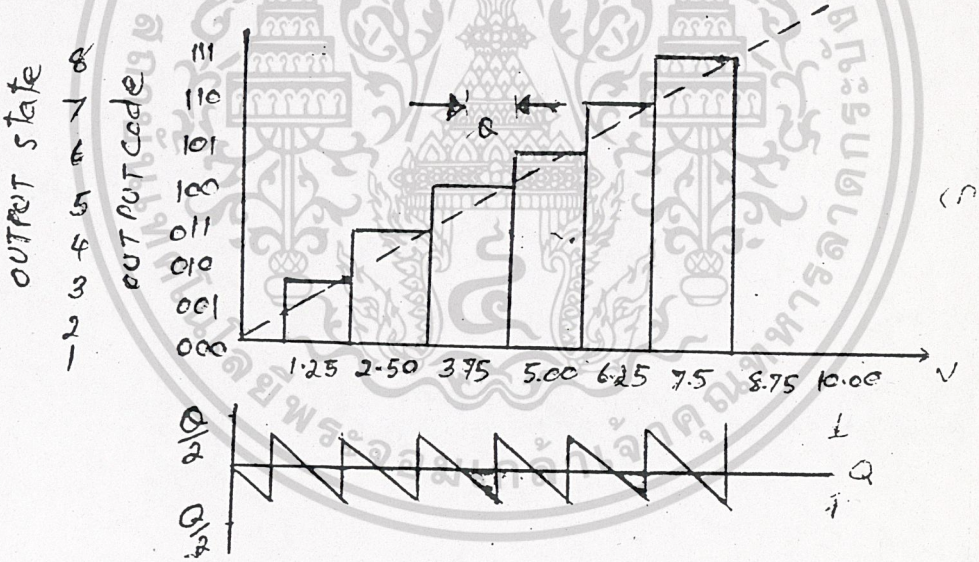


รูปที่ 2.7.4 การเกิด alias frequency จากการสุ่มความถี่ต่ำกว่า 2 เท่าของความถี่สัญญาณอนาลอก

ในกรณีการสุ่มนี้ในอัตราที่ต่ำกว่าสองเท่าต่อรอบสัญญาณอินพุทจะทำให้ได้สัญญาณชายด์ความถี่ต่ำ ๆแสดงเป็นจุดไข่ปลาในรูปที่ 2.7.4 ซึ่งจะเห็นว่าความถี่ alias อาจจะแตกต่างจากความถี่เดิมไปมาก

2.5 Quantizing theory

Quantizing เป็นขบวนการที่เปลี่ยนแปลงสัญญาณอนาลอกเป็นสัญญาณที่ไม่ต่อเนื่อง(discreste signal) ซึ่งจากนี้ก็จะผ่านขบวนการ Coding จัดให้สัญญาณที่ไม่ต่อเนื่องนั้นอยู่ในรูปที่ง่ายต่อการประมวลทำความเข้าใจ และเป็นสัดส่วนสัมพันธ์กับสัญญาณอนาลอก เช่น อยู่ในรูปของระหัสไบนารี เป็นต้นหากนำเอาขนาดของสัญญาณอยสลอกและสัญญาณดิจิตอลที่สัมพันธ์จากการ Quantize และencode และมาเขียนกราฟก็จะได้กราฟแสดง Quantize transfer function ดังรูปที่ 2.7.5



รูปที่ 2.7.5 transferfuction ของ Quantize 3 บิท ตามทฤษฎี

ในรูปกราฟแสดงให้เห็นถึงความสัมพันธ์ระหว่างสัญญาณอนาลอกที่ขนาดอยู่ระหว่าง 0 ถึง+10 โวลท์ Quantize และ encode เป็นระหัสไบนารี 3 บิทได้ 8 ระดับจาก 000 ถึง 111 ในยระบบไบนารี ระหัสดิจิตอลแต่ละค่าจาแทนขนาดของสัญญาณอนาลอกแต่ละค่าที่เป็นสัดส่วนหักค่าเต็มสเกลโดยค่าสูงสุดของระหัสดิจิตอลคือทุกบิทเป็น1จะ

เท่ากับสัญญาณอนาลอกเต็มสเกลคูณด้วย $(1-2^{-n})$ โดย n เป็ตจำนวนบิทของระหัสดิจิตอลและระหัสดิจิตอลแต่ละบิทเป็น 1 จะเท่ากับขนาดเต็มสเกลของอนาลอกคูณกับค่า Weithting ของระหัสชนิดนั้นหารด้วย 2^n ตัวอย่างเช่น ค่าเต็มสเกลของอนาลอกเป็น 10 โวลท์ ระหัส 1001 จะแทนขนาดสัญญาณอนาลอกอินพุท

$$V_{input} = \frac{R}{2^n} \{(1*2^3) + (0*2^2) + (1*2^1) + (1*2^0)\}$$

$$= \frac{10}{2^4} * \{(1*2^3) + (0*2^2) + (1*2^1) + (1*2^0)\}$$

จุดสำคัญที่เกี่ยวกับกราฟ transferfuction ในรูปที่ 6 อันแรกได้แก่ resolution ของ quantizer ซึ่งกำหนดได้จากจำนวนบิทของระหัสดิจิตอล หรือจากกราฟ คือขนาดกว้างของ step ทางแกนอนาลอกว่าเป็นสัดส่วนเท่าใดระหว่างเต็มสเกลกับค่า 2^n

จำนวนสถานะของเอาท์พุทกำหนดได้จากจำนวนบิทคือเท่ากับ 2 สถานะ ตัวอย่างกรณี ADC 8 บิท Quantizer จะให้เอาท์พุท 256 สถานะ และ 12 บิทให้ 4096 สถานะต่อค่าเต็มสเแลของอนาลอกในไดอะแกรมแสดงทรานสเฟอร์ฟังชั่น จะเห็นจุดแบ่งระดับ (decision point หรือ Teshold level) สัญญาณอนาลอกจะมีจำนวน 2^n-1 จุดที่อยู่ที่ 0.625, 1.875, 3.125, 4.375, 5.625, 6.875 และ 8.125 โวลท์ ระหว่างจุดดังกล่าวเป็นสัญญาณอนาลอกซึ่งแปลงเป็นระหัสดิจิตอล 1 สถานะ ดังนั้นค่าเหล่านี้จะต้องปรับให้ถูกตองมากที่สุดเพื่อแปลงขนาดของอนาลอกให้ตรงกับค่าที่ทำการ Quantized

แรงดันที่ 1.25, 2.50, 3.75, 5.0, 6.25, 7.2และ 8.75 โวลท์ เป็นจุดกึ่งกลางในช่วงของสัญญาณอนาลอกที่แสดง 1 สถานะเอาท์พุทดิจิตอลฟังก์ชั่นที่เป็นลักษณะที่เป็นขั้นบันไดนี้สามารถประมาณเป็นเส้นตรงโดยการโยงเส้นตรง ระหว่างจุดเริ่มและจุดปลาย ณ จุดกึ่งกลางของระหัสดิจิตอลสถานะสุดท้าย สังเกตุว่า ในทางทฤษฎีแล้วเส้นตรงนี้จะต้องผ่านจุดกึ่งกลางของทุกระดับดิจิตอล

2.6 Quantizer Resolution and Error

ในแต่ละสถานะของสัญญาณดิจิตอลเอาท์พุทจะแทนขนาดของสัญญาณอนาลอกค่า

ใดค่าหนึ่งในช่วงเล็ก ๆระหว่างจุดแบ่งระดับจุด เรียกว่าช่วงเล็ก ๆนี้ว่าเป็นขนาดหนึ่ง Analog Quantization หรือหนึ่งควันตัม (Quantum) หรือ 1LSB(least significant bit) ของการแปลงสัญญาณ ตัวอย่างในรูปที่ 2.7.5 ก ควันตัม คือ 1.25 โวลท์ ค่านี้ได้จากการคำนวณจาก

$$Q = \frac{FSR}{2^n}$$

FSR คือ ช่วงเต็มสเกลของแรงดันอนาลอก (Full Scall Range)

N คือ จำนวนบิทของระหัสดิจิตอล

จากสมการจะเห็นว่าหากจำนวนบิทมากขนาดของ Quantizing ก็จะลดลงถ้าให้สัญญาณอินพุทของ Quantizer กวาดไปตลอดช่วงของสัญญาณอนาลอกก็จะเห็นช่วงของผลต่างของอนาลอกอินพุทและ ดิจิตอลเอาท์เป็นช่วงซึ่งพล็อตได้เป็นรูปฟันเลื่อยดังรูป 2.7.5 (ข) เรียกว่า Quantizing error ซึ่ง error นั้นก็คือ 1 ช่วงสัญญาณอนาลอกแปลงเป็นระดับดิจิตอล 1 สถานะ ดังกล่าวมาแล้วนั่นเอง

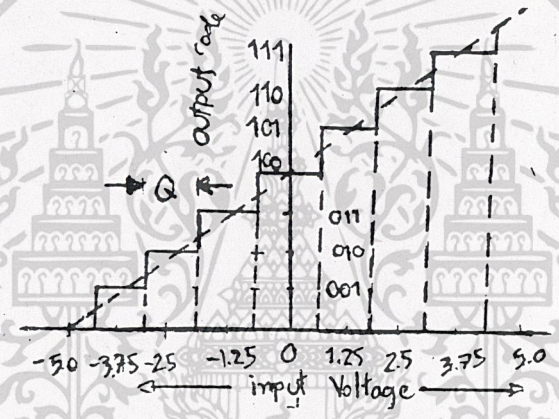
error นี้เป็นธรรมชาติของ Quantizing ซึ่งจัดทำการแก้ไขไม่ได้นอกจากการเพิ่มจำนวนบิทของ Quantizer ให้มากขึ้น และเอาท์พุท error จะอยู่ระหว่าง 0/Q/2 error อาจจะเป็นศูนย์เมื่อสัญญาณอนาลอกที่ค่าที่จุดกึ่งกลางของควันตัมพอดี ลักษณะฟังชั่นของ error จะสามารถคิดเป็นสัญญาณรบกวนทางอินพุท ซึ่งมีค่าเป็น QV_{p-p} และเฉลี่ยเป็นศูนย์ค่าrms เป็น Q/2 3 ซึ่งจะได้จากการวิเคราะห์รูปคลื่นฟันเลื่อย

2.6 ระหัสตัวเลขสำหรับการเปลี่ยนข้อมูล

ระหัสตัวเลขที่นิยมนำมาใช้ในระบบเปลี่ยนข้อมูลได้แก่ระหัสไบนารีหรือเรียกว่า Straight binary โดยที่ระหัสไบนารีสถานะสูงสุดจะแทนสัญญาณอนาลอก FSR $(1-2^n)$ โวลท์ ตัวอย่าง เช่นหากสัญญาณอนาลอกเต็มสเกล (FRS) เท่ากับ 20 โวลท์ สำหรับข้อมูลขนาด 12 บิทดังนั้นระหัส 1111 1111 1111 จะแทนสัญญาณอนาลอกขนาด $20\ (1-2^n) = 19.9951171$ โวลท์ นอกจากระหัสไบนารีธรรมดาดังกล่าวยังมีการใช้ระบบไบนารีแบบอื่น ๆในระบบแปลงสัญญาณได้แก่ ออฟเซ็ทไบนารี, two's complement, BCD ซึ่งแต่ละระบบมีข้อดีและความหมายเหมาะสมต่างกัน

ตัวอย่างเช่น ระบบ BCD เหมาะสำหรับการแสดงตัวเป็นตัวเลขหน้าปัทม์ หรือต่อเข้ากับดิจิตอลมิเตอร์ ระหัส two's complement เหมาะสำหรับการคำนวณลอจิกทางคณิตศาสตร์และระบบออฟเซ็ทไบนารี เหมาะสำหรับการแปลงสัญญาณอินพุทที่มีทั้งช่วงบวกและช่วงลบ ในรูปที่ 2.7.6 แสดง Transfer fimctopn ของ ADC 3 บิทที่ใช้ระหัส off set binary

นอกจากมาตราฐานของการใช้ระหัสตัวเลขแล้ว ยังมีมาตราฐานของการเลือกช่วงของขนาดแรงดันอินพุท สำหรับ ADC คือ หากเป็นสัญญาณช่วงบวกหรือลบอย่างเดียวจะใช้ 0-5 โวลท์หรือ 0-10 โวลท์ แต่ถ้าเป็นช่วงลบจะใช้ -2.5 โวลท์,และ -10 โวลท์เป็นมาตราฐาน



รูปที่ 2.7.6

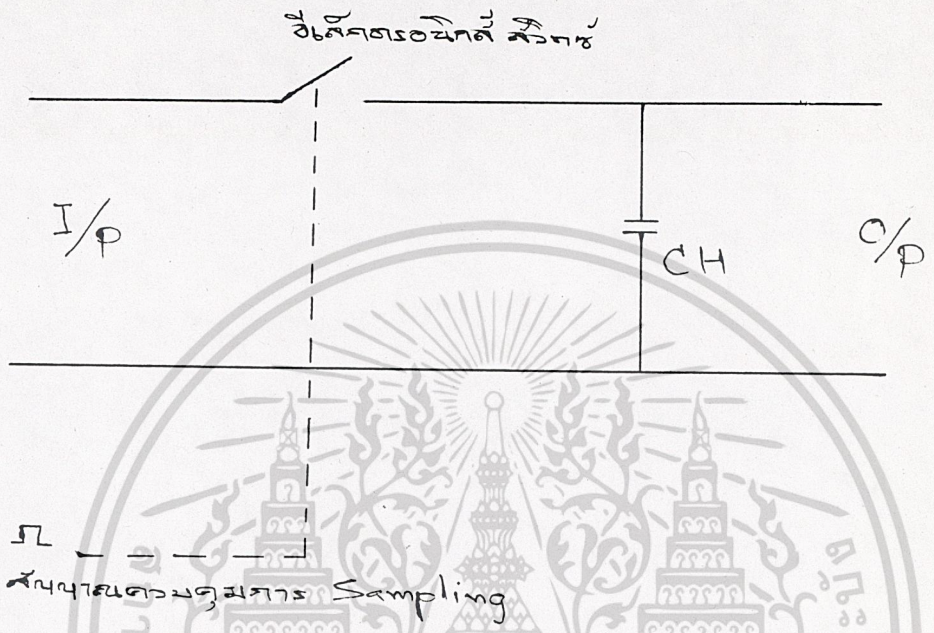
2.7 วงจร Sample-Hold (S/H)

ในตอนต้นได้ว่าถึงจุดมุ่งหมายในการใช้วงจร S/H ถึง ADC และต่อจากนี้จะได้กล่าวถึงรายละเอียดของวงจร S/H บางวงจรในปัจจุบัน

มิได้มีใช้เฉพาะกับ ACD เท่านั้น แต่ก็ยังใช้กันทั่ว ๆไปในระบบ data-distribution, sampling acope, DVM, reconstuction filter และอนาลอกคอมพิวเตอร์เป็นต้น

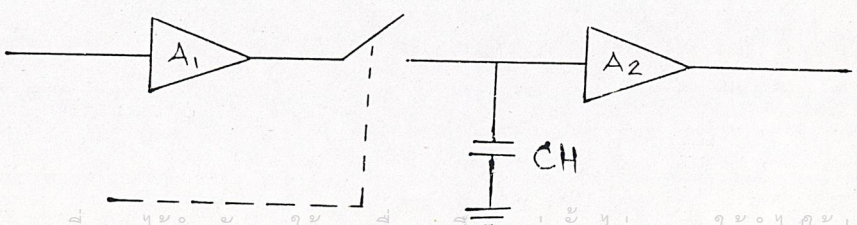
วงจร sample-hold โดยพื้นฐานแล้วเป็นอุปกรณ์หรือวงจรเก็บแรงดัน (Voltage memory) ซึ่งใช้อุปกรณ์ร่วมสำคัญคือ ตัวเก็บประจุ ในรูปที่ 2.7.7 ก แสดงพื้นฐาน Sample-hold อิเล็กทรอนิกส์สวิทช์จะต่อสัญญาณแรงดันเข้ากับตัวเก็บประจุซึ่งสวิทช์ควบคุมจาก Sampling pulse ช่วงเวลาในการตัดสวิทช์และเวลาในการประจุ

แรงดันจนถึงค่าที่ sample มานั้นเรียกว่า Aperture-time ของ S/H จากลักษณะการทำงานดังกล่าววงจร S/H จะมีจุดต่อสัญญาณเข้าออก 3 จุดด้วยกันคือ สัญญาณอนาลอกอินพุท สัญญาณ sample และเอาท์พุท

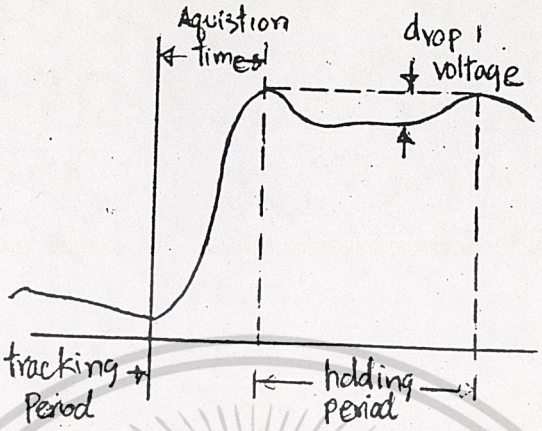


รูปที่ 2.7.6 ก แสดงพื้นฐานของ S/H

รูปที่ 2.7.6 ข แสดงวงจรที่ใกล้เคียงกับวงจรที่ใช้ในทางปฏิบัติ โดยเพิ่มเติมบัฟเฟอร์แอมป์ปริไฟร์เข้าทางส่วนอินพุทและเอาท์พุทของ S/H พื้นฐานแอมปริไฟร์เออร์ทางด้านอินพุท ช่วยทำให้วงจรอินพุทพีแดนซ์สูงสะดวกต่อการใช้งานและสามารถเพิ่มกระแสเพื่อทำการประจุ c ได้เร็วขึ้น ส่วนทางเอาท์พุทช่วยทำให้เอาท์พุทพีแดนซ์สามารถขับ ADC ได้ง่าย มีจุดสำคัญที่ต้องพิจารณา คือ ในส่วนของแอมปริไฟร์ตัวเหล่านี้ กติแล้วจำเป็นต้องเป็นแอมปริไฟร์ที่ใช้กระแสอินพุทต่ำทั้งนี้เพื่อให้ดึงกระแสจากตัวเก็บประจุในช่วง hold สัญญาณน้อยที่สุดมิฉะนั้นจะเกิด Droop แก่แรงดันที่hold ไว้ดังแสดงในรูปที่ 2.7.7



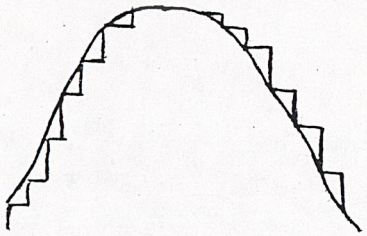
รูปที่ 2.7.6 ข



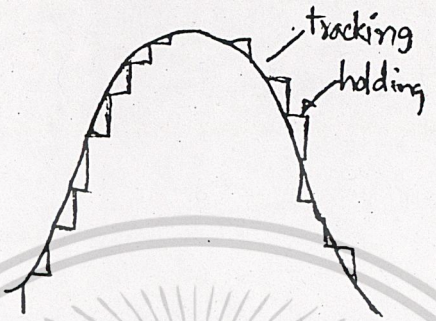
รูปที่ 2.7.7 แสดงรูปคลื่นเอาท์พุทของ S/H

ปกติแล้วมักใช้แอมปริไฟร์เออร์ที่มี FET เป็นอินพุทเพราะการไบอัสด้วยแรงดันทำให้กระแสอินพุทต่ำด้วย

วงจร S/H ใช้ระบบ data Acquistion นิยมใช้สอบแบบคือ Sample-hold และ Track-hold วงจร Sample-hold จะใช้วิธีหรือสุ่มสัญญาณอย่างรวดเร็วแล้วเข้าสู่ holding period ซึ่งหมายความว่าสวิทช์ควบคุมจะต้องต่อช่วงเวลาอันสั้นต่อเนื่องและส่วน track-hold จะต้องต่อสวิทช์ Sample ช้ากว่าแต่ขณะตัดสัญญาณออกวงจรจะ track ตามสัญญาณอินพุทจดกว่าจะมีการ Sample สัญญาณใหม่ ลักษณะของเอาท์พุทของ Sample-hold และ track-hold แสดงในรูปที่ 2.7.8



ก) เอาท์พุทจาก Sample-hold

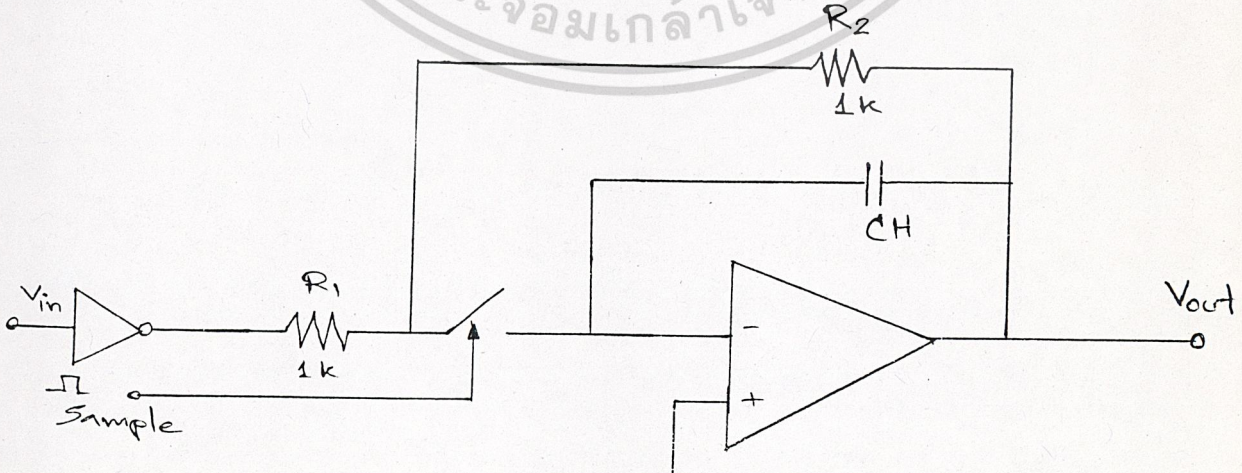


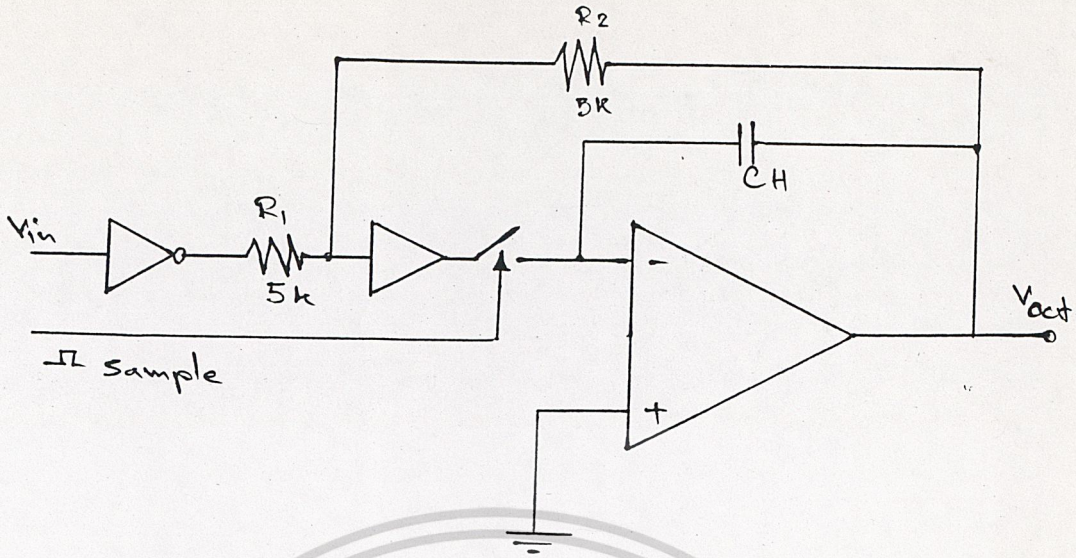
ข) เอาท์พุทจาก track-hold

รูปที่ 2.7.8

นอกจาก Sample-hold ทั้งสองแบบแล้วยังมีแบบอื่น ๆ ที่มีการนำกล่าว ณ ที่นี้ การจัดวงจร Sample-hold มีการจัดวงจรได้หลายลักษณะ ซึ่งอาจนำไอซี หรือทรานซิสเตอร์มาประกอบเป็นวงจร ตลอดจนการสร้างวงจรทั้งหมดของ Sample-hold ลงบนชิบไอซีเดียวกัน เช่น เบอร์ LF 398

ก) inverting close loop circuit



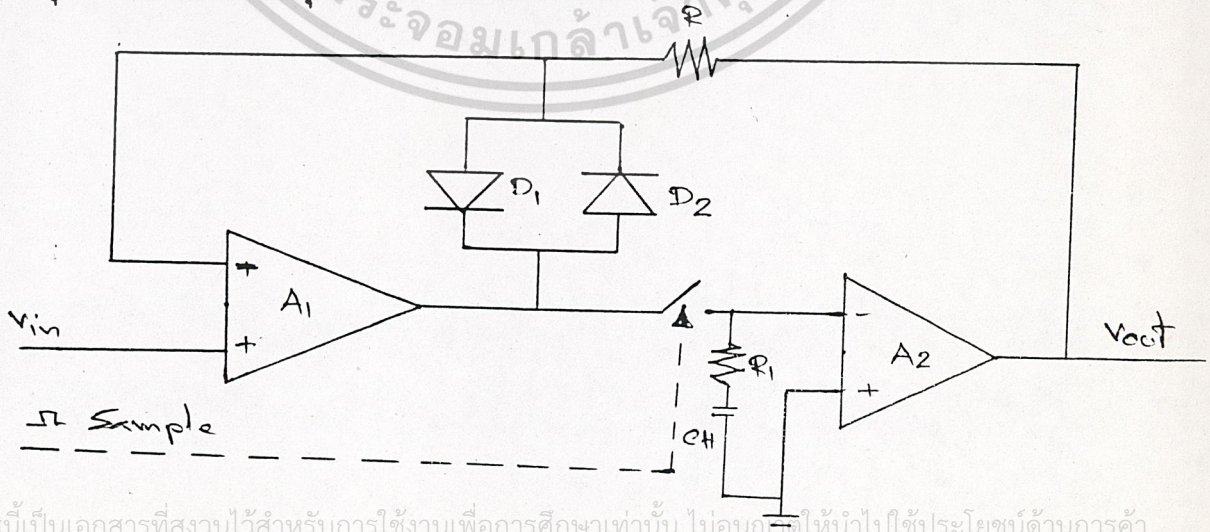


รูปที่ 2.7.9 วงจร Non-inverting closed loop cricuit

ในวงจรนี้ C จะประจุด้วยอัตรา RC ซึ่งสามารถเพิ่มความเร็วได้โดยใช้ Current boot Amplifier อยู่ในรูปป้อนกลับดังรูป ข. โดยแอมปริไฟร์เออร์นี้มีอัตราขยายเท่ากับ 1

ข) Closed loop Non inverting

แสดงในรูปที่ 2.10 ในวงจรนี้ A1 จะทำหน้าที่เป็นบัฟเฟอร์และ error Amplifier ในตัวซึ่งจะทำหน้าที่เปรียบเทียบแรงดันเอาท์พุทกับแรงดันอินพุทแล้วจะประจุ C จน error เท่ากับศูนย์ A2 ในวงจรนี้จะมีอินพุทอิมพีแดนซ์สูง และการป้อนกลับใน A1 ด้วยไดโอดทำให้ A1 ไม่ต้องเป็น op-Amp ที่คุณภาพดีนัก ตัวต้านทาน R จะแยกอินพุทของ A1 และเอาท์พุทของ A2 ออกจากกันในช่วง hold-mode



รูปที่ 2.7.10 Closed loop Non-inverting

ข้อดีของวงจรนี้คือ ทำงานได้รวดเร็วและแม่นยำ ความเร็วในการประจุขึ้นอยู่กับความเร็วของ A1 และความสามารถในการจ่ายกระแสของมัน ไดโอด สองตัวจะทำหน้าที่ Clamp สัญญาณเอาท์พุทไปที่อินพุท อินเวอร์ติ้งของ A1 เพื่อยังคงให้วงจรมีเสถียรภาพดีเมื่อสวิทช์ Sample เปิดวงจร วงจรลักษณะนี้เป็นพื้นฐานของไอซีเบอร์ LF 398

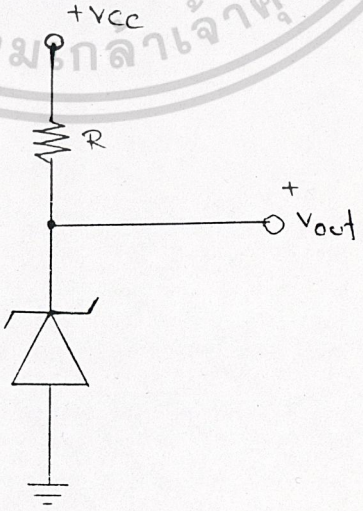
2.8 วงจรแรงดันอ้างอิง(Voltage refference circuit)

วงจรกระแสแรงดันอ้างอิงนั้นเป็นวงจรที่สำคัญวงจรหนึ่งในระบบ data Aquisition เนื่องจากเป็นส่วนสำคัญในการกำหนดคุณภาพของวงจร DAC ทั่วแบบที่เป็นวงจรอิสระหรือวงจรที่อยู่ในวงจร ADC บางแบบในวงจร ADC หรือ DAC บางเบอร์ จะมีวงจรแรงดันอ้างอิงในตัวละบางแบบจะต้องต่อเพิ่มเติมเข้าไป ในตอนนี้จะได้กล่าวถึงแรงดันอ้างอิง

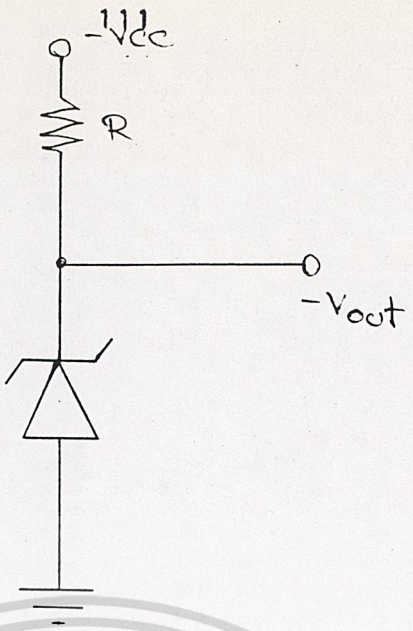
1) Basic viktage refferebce

อุปกรณ์ที่โดยนิยมใช้เป็นแหล่งกำเนิดแรงดันอ้างอิงได้แก่ ซีเนอร์ไดโอดซึ่งเมื่อให้รีเวิร์ดไบอัสจนเกิดการเบรคดาวน์ แรงดันคล่อมซีเนอร์จะคงที่เท่ากับแรงดันเบรคดาวน์ R ที่ต่อกับอนุกรมกับซีเนอร์จะทำหน้าที่จ่ายกระแสไบอัสแก่ซีเนอร์ให้เบรคดาวน์ และจำกัดกระแสรีเวิร์ดไม่ให้ไหลมากจนเป็นอันตรายแก่ซีเนอร์

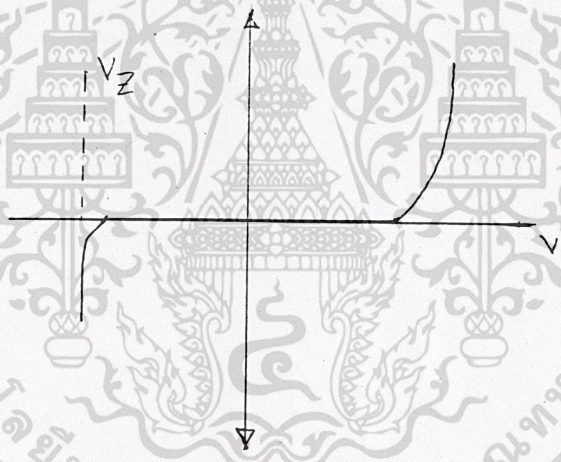
ข้อเสียของวงจรนี้คือ แรงดันมักเปลี่ยนตามอุณหภูมิได้ง่ายหรือเรียกว่ามี ส. ป.ส. ทางอุณหภูมิสูง และจ่ายกระแสได้จำนวนจำกัดจึงใช้จำนวนจำกัดจึงมักใช้วงจรนี้กับ ADC ที่ไม่ต้องการคุณภาพมากนัก



ก. แรงดันอ้างอิงบวก



ข. แรงดันอ้างอิงลบ

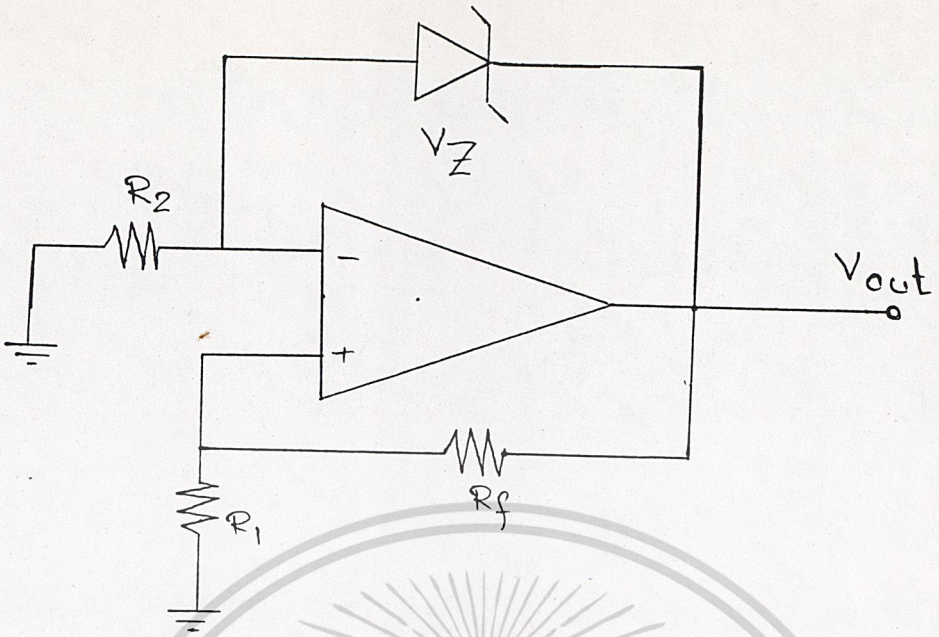


ค. กราฟคุณสมบัติของซีเนอร์

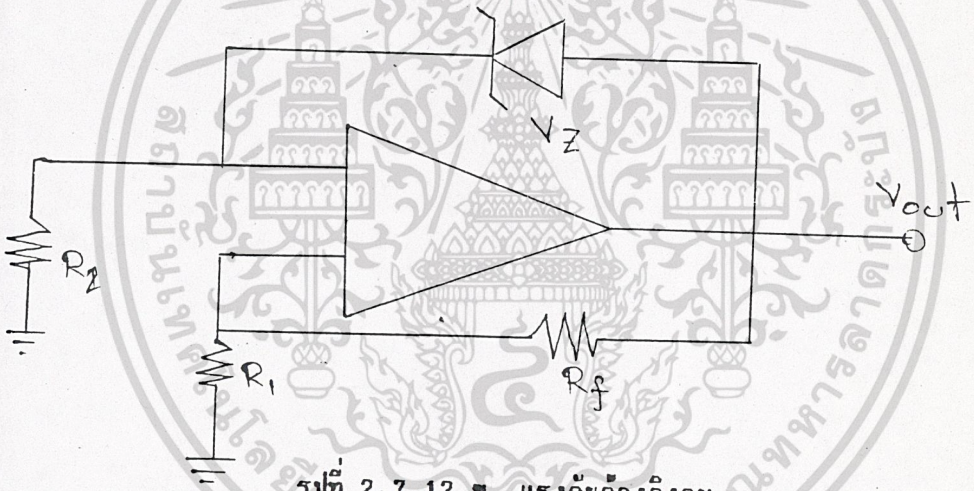
รูปที่ 2.7.11

2) Precision voltage refference

แรงดันอ้างอิงที่คุณภาพดีกว่าจะใช้ออปแอมป์ร่วมกับซีเนอร์ ซึ่งนอกจากจะใช้แรงดันที่คงที่มากกว่าแล้ว ยังสามารถปรับแรงดันเอาท์พุทได้มากหรือน้อยกว่าแรงดันซีเนอร์ไดโอดได้ด้วย ลักษณะการจัดวงจรต่างแสดงได้ดังรูปที่ 2.7.12



รูปที่ 2.7.12 ก. แรงดันอ้างอิงบวก



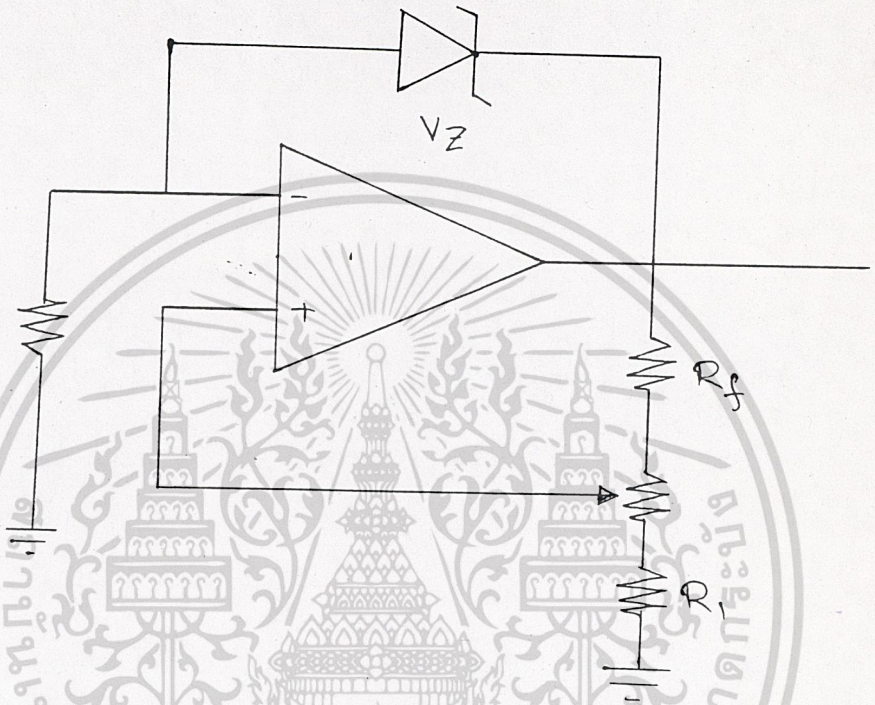
รูปที่ 2.7.12 ข. แรงดันอ้างอิงลบ

ตามปกติแล้วแรงดันอ้างอิงที่ใช้ซีเนอร์ไดโอดจะให้คุณภาพดีก็ต่อเมื่อ กระแสที่จ่ายให้ซีเนอร์คงที่ตลอดเวลา และทุกช่วงของอุณหภูมิในวงจรรูปที่ 13 ออฟแอมป์จะทำหน้าที่จ่ายกระแสคงที่และมี ส.ป.ส. อุณหภูมิต่ำกว่ากระแสที่ผ่านซีเนอร์ขึ้นอยู่กับการเลือกค่า R_1 R_2 และ R_f และ V_o กำหนดจาก R_1, R_2 และ V_z การออกแบบต้องเลือกซีเนอร์ซึ่งรู้ค่า I_z และ V_z และ V_z ทำการเลือกค่า R_1 และหาค่า R_1 จากสมการ V_o

$$\frac{V_o - V_R}{R_f} \qquad \frac{R_1 + R_f}{}$$

$$I_x = \frac{V_o R_1}{R_2(R_1+R_f)} - \frac{V_o R_1}{R_f}$$

$$V_1 = \frac{V_o R_1}{R_1+R_f} - I_x R_2$$



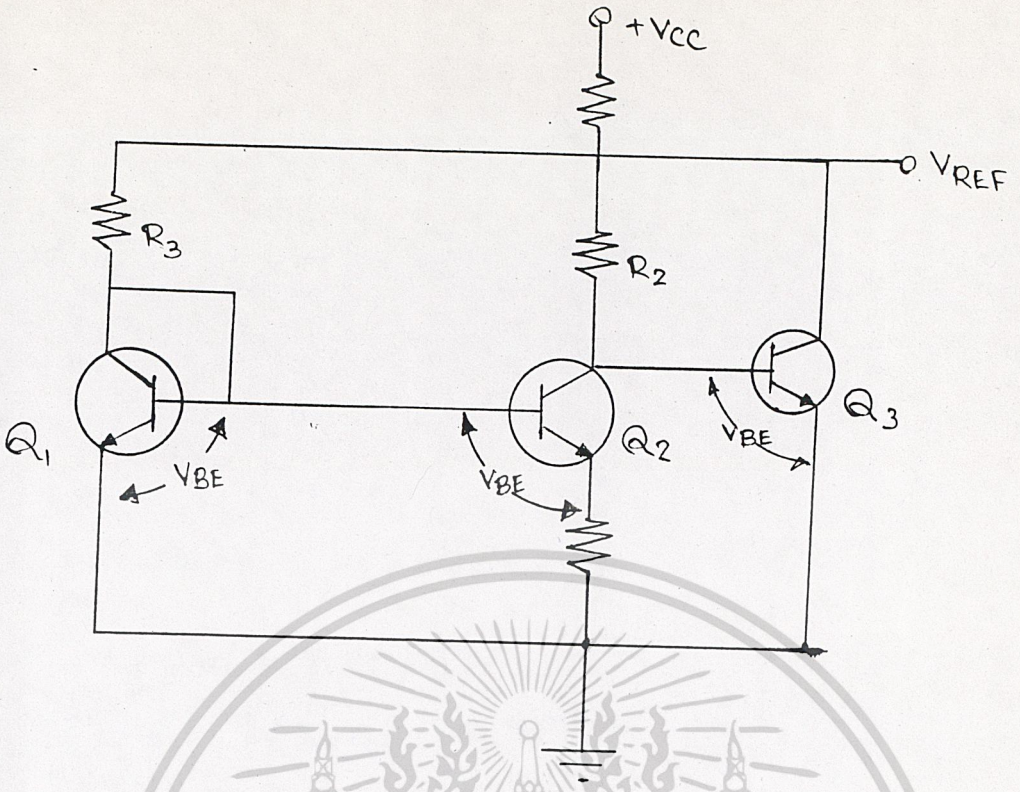
รูปที่ 2.7.12 แรงดันอ้างอิงปรับค่าได้

3) Bandgab voltage refference

Bandgab voltage refference ได้รับการออกแบบเพื่อการแก้ไขทาง ส.ป.ส. ทางอุณหภูมิโดยใช้ผลต่างของแรงดันเบส-อิมิเตอร์ของทรานซิสเตอร์สองตัวที่ทำงานที่กระแสต่างกัน
โดย

$$V_{ref} = V_{beq3} + I_2 R_2$$

$$V_{ref} = V_{beq3} + T_1 K (R_2/R_1) \ln (I_1/I_2)$$



รูปที่ 2.7.13 วงจร Band gab voltage refference พื้นฐาน

แรงดันอ้างอิงแบบแบนด์แกป ได้ถูกสร้างโดยใช้พื้นฐานในรูป 15 และมีจำหน่ายในตัวถังคล้ายทรานซิสเตอร์ เช่น เบอร์ LM 336 ถึงสามารถปรับขนาดของ V_{out} ได้

2.8 ภาษา ซี

ซี เป็นภาษาคอมพิวเตอร์ที่มีการพัฒนาขึ้นใช้งานเพื่อให้เป็นภาษามาตราฐานที่ไม่ขึ้นกับโปรแกรมจัดระบบงาน หรือขึ้นกับฮาร์ดแวร์ จึงทำให้ซีเป็นภาษาคอมพิวเตอร์ตามอุดมการณ์ของนักคอมพิวเตอร์

ซี เป็นภาษาคอมพิวเตอร์ที่อาศัยหลักการทางวิธีการโปรแกรมแบบสมัยใหม่ที่เรียกว่า โปรแกรมโครงสร้าง การออกแบบระบบซอฟต์แวร์จึงมีรูปแบบการออกแบบที่ง่ายเป็นโมดูล และสามารถนำไปใช้ได้ง่าย ผู้พัฒนาโปรแกรมจึงรู้สึกว่าการใช้ภาษาซีเป็นเรื่องที่เหมาะสมในงานพัฒนา

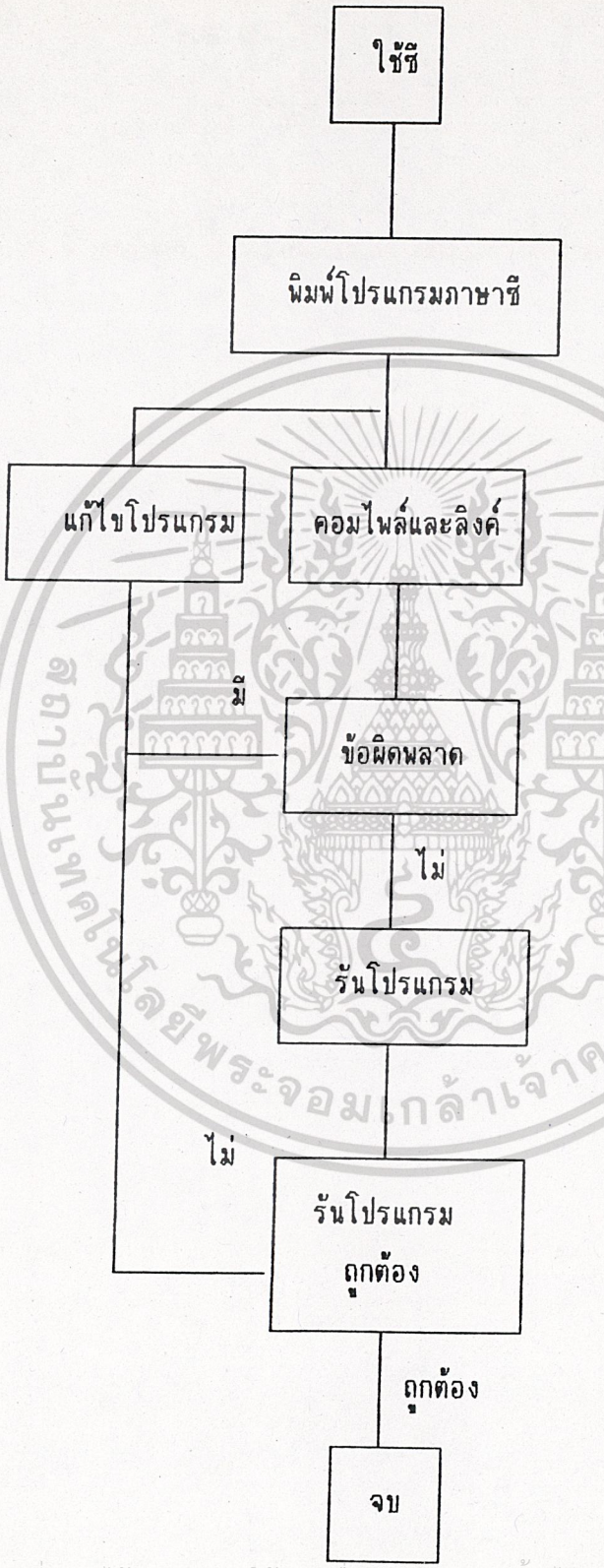
ในการพัฒนาโปรแกรมจัดระบบงาน หรือตัวแปรภาษา เรามักใช้ภาษาระดับต่ำเช่น แอสเซมบลี้ ภาษาแอสเซมบลี้ให้ข้อดีในแง่ความเร็วของภาษาทำงาน แต่ก็มีความยุ่งยากในการเขียนโปรแกรม การแก้ไขดัดแปลงและการค้นหาที่ผิด แต่หากจะใช้ภาษาระดับสูงอื่นก็จะมีความล่าช้าในการการทำงาน ทำให้ยากต่อการใช้เป็นโปรแกรมจัดระบบงานได้ แต่เนื่องจากภาษาซีเป็นภาษาที่มีโครงร่างแบบภาษาระดับสูงและสามารถใช้งาน หรือทำงานได้รวดเร็วเหมือนภาษาระดับต่ำ ภาษาซีจึงเป็โปรแกรมที่น่าสนใจในการพัฒนา

ในการคอมไพล์ด้วย ซี คอมไพเลอร์นั้นปรากฏว่าซีให้ประสิทธิภาพที่เหนือกว่าภาษาขั้นสูงอื่น ๆคือ ภาษาซีให้ออบเจคโค้ดสั้นกว่า ภาษาซีให้ความเร็วในการทำงานเร็วกว่า ภาษาซีจึงเป็นภาษาที่มีประสิทธิภาพในการทำงานได้ดีกว่าหลายภาษาในระดับสูง

ภาษาซีมีความคล่องตัวที่จะได้รับการประยุกต์เข้ากับวานต่าง ๆได้อย่างดี เราสามารถนำภาษาซีมาใช้ในงานพัฒนาโปรแกรมต่าง ๆได้ เช่นโปรแกรมเวิร์ดโปรเซสซิ่ง โปรแกรมสเปรดชีต โปรแกรมสำเร็จรูปอื่น ๆปัจจุบันบริษัทผู้พัฒนาโปรแกรมสำเร็จรูปมักใช้ซีเป็นเครื่องมือในการพัฒนาโปรแกรม

จากรูปโครงที่ทันสมัยของภาษาซี ภาษาซีจึงเป็นภาษาที่เหมาะสมสำหรับผู้สนใจที่จะศึกษาหลักการทางด้านวิธีการโปรแกรมเป็นอย่างดี การศึกษาภาษาซี จะทำให้ผู้สนใจหลักการทางด้านวิธีการเขียนโปรแกรมที่ถูกต้อง เป็นการฝึกนิสัยในเรื่องการเขียนโปรแกรมและก้าวหน้าต่อไปในอนาคตได้เป็นอย่างดี

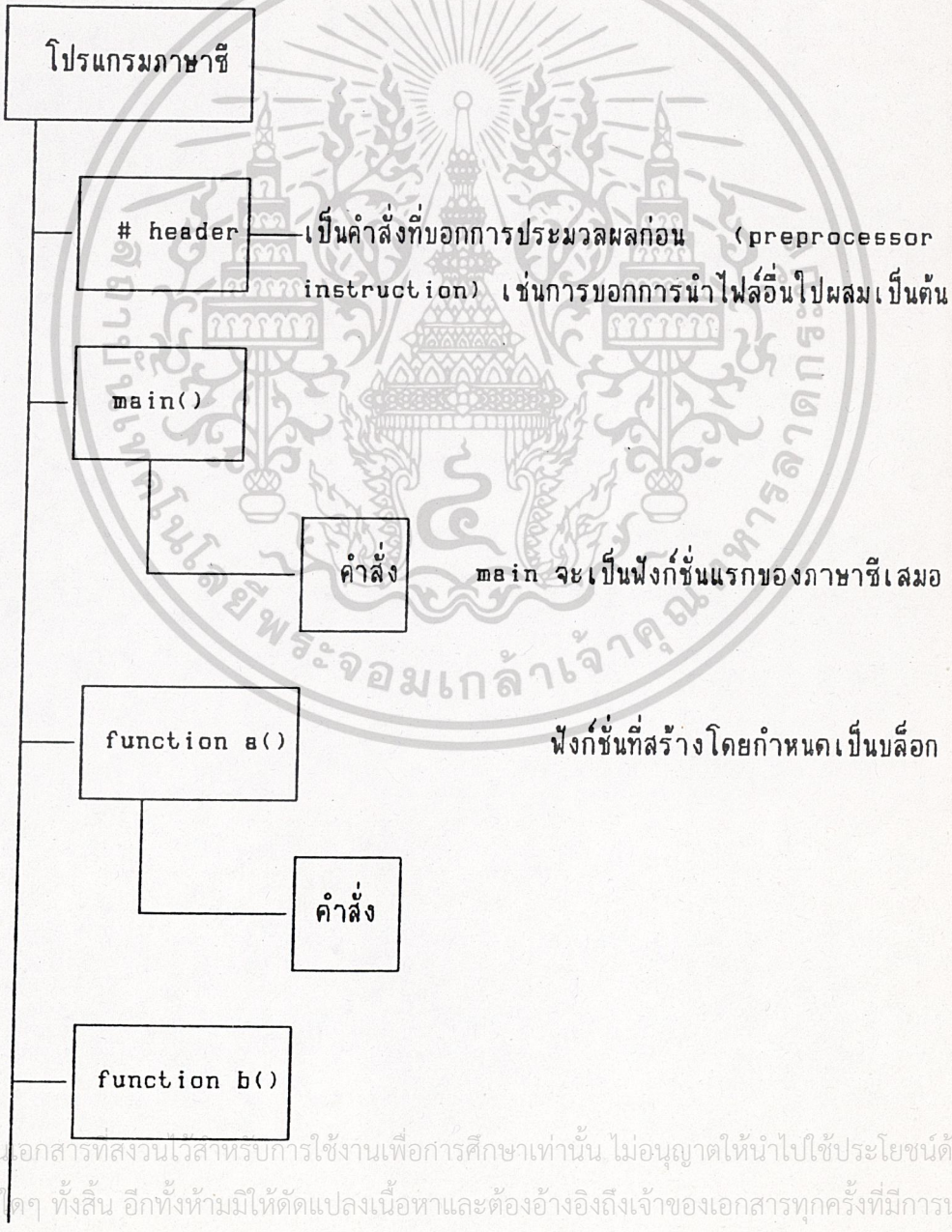
ลำดับการทำงานของการใช้ c เป็นดังรูปที่ 2.8.1



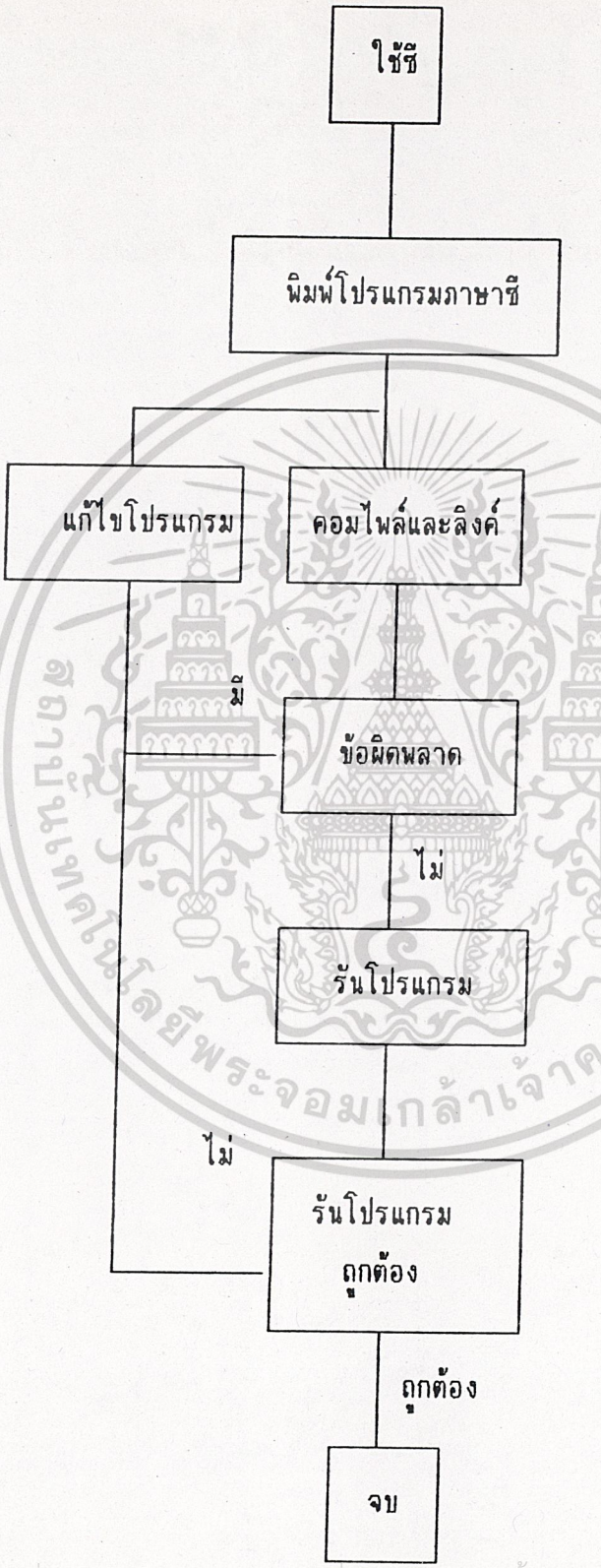
รูปที่ 2.8.1 ฝังการใช้ซีคอมไพเลอร์

ขั้นตอนการใช้ภาษาซี อาจยุ่งยากกว่าภาษาเบสิกในส่วนที่ต้องมีการคอมไพล์และลิงค์ ซึ่งถ้าหากโปรแกรมที่เขียนมีขนาดใหญ่ก็จะใช้เวลาในการคอมไพล์มาก เมื่อคอมไพล์แล้วจะได้ไฟล์ .OBJ คือเป็นไฟล์ออบเจคที่เป็นรหัส และเมื่อทำการลิค์กับไฟล์ห้องสมุด (LIB) ก็จะได้ไฟล์ .EXE หรือที่เรียกว่า เอ็กซีคิวต์ ไฟล์ที่พร้อมสำหรับการใช้งานได้ต่อไป

โครงสร้างภาษาซีเขียนเป็นไดอะแกรมได้ดังรูปที่ 2.8.2



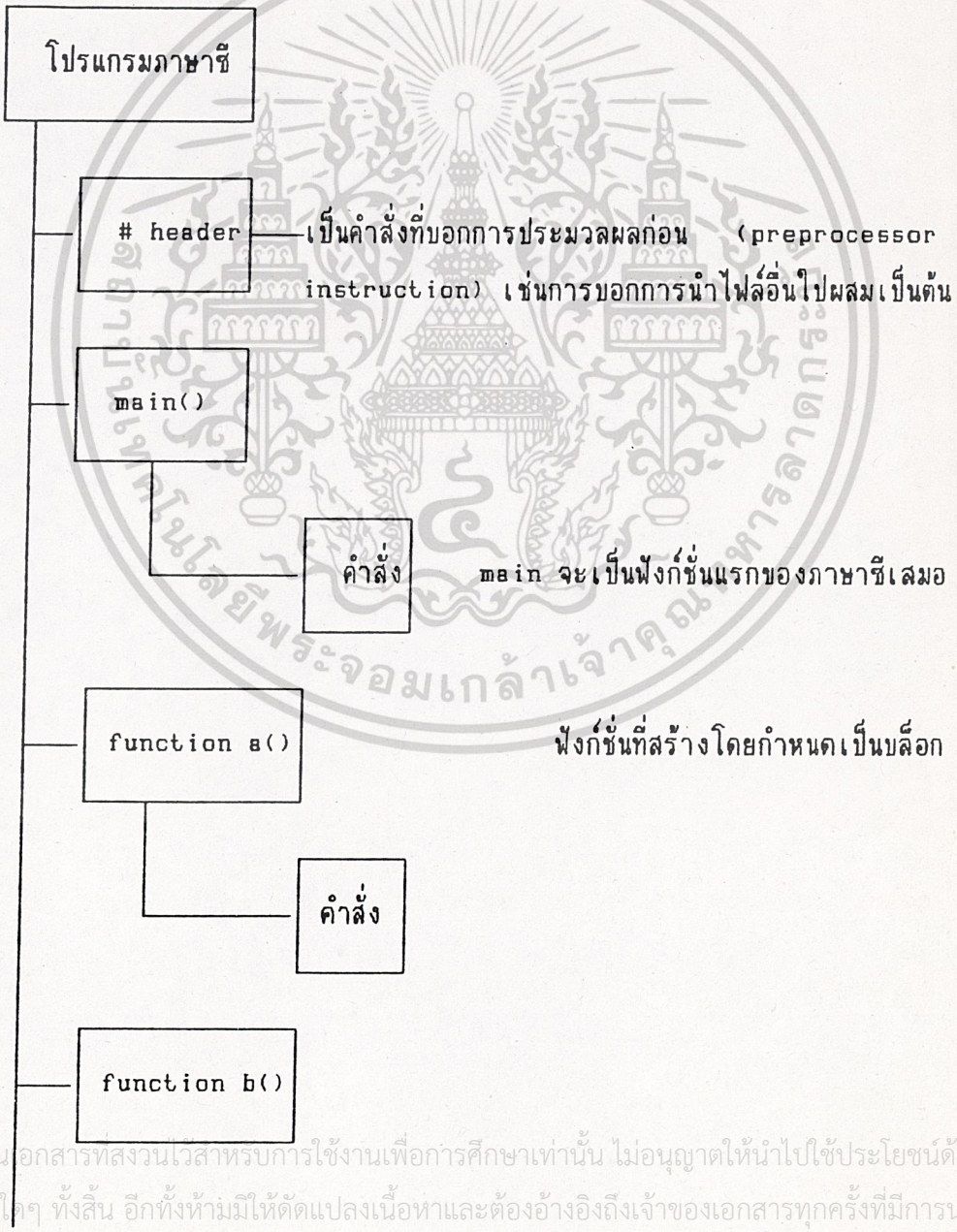
ลำดับการทำงานของการใช้ C เป็นดังรูปที่ 2.8.1

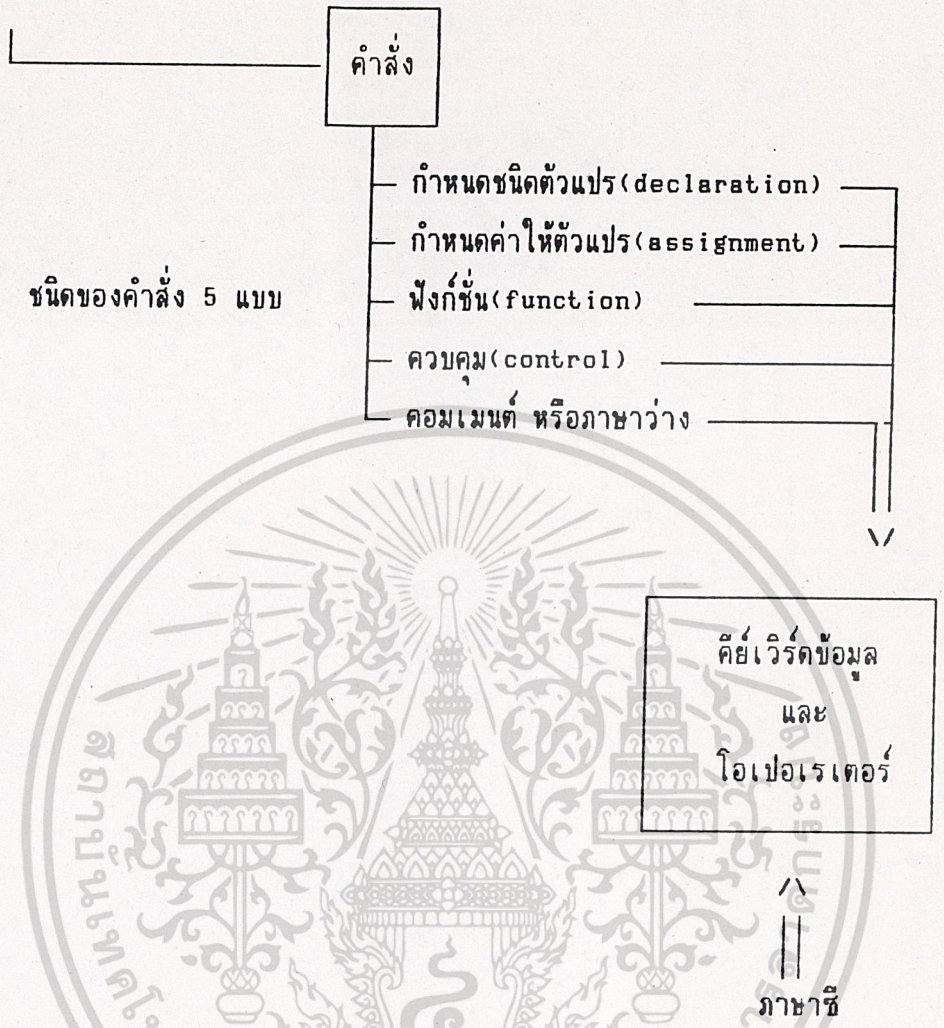


รูปที่ 2.8.1 ฝังการใช้ซีคอมไพเลอร์

ขั้นตอนการใช้ภาษาซี อาจยุ่งยากกว่าภาษาเบสิกในส่วนที่ต้องมีการคอมไพล์และลิงค์ ซึ่งถ้าหากโปรแกรมที่เขียนมีขนาดใหญ่ก็จะใช้เวลาในการคอมไพล์มาก เมื่อคอมไพล์แล้วจะได้ไฟล์ .OBJ คือเป็นไฟล์ออบเจคที่เป็นรหัส และเมื่อทำการลิค์กับไฟล์ห้องสมุด (LIB) ก็จะได้ไฟล์ .EXE หรือที่เรียกว่า เอ็กซีคิวต์ ไฟล์ที่พร้อมสำหรับการใช้งานได้ต่อไป

โครงสร้างภาษาซีเขียนเป็นไดอะแกรมได้ดังรูปที่ 2.8.2





รูปที่ 2.8.2 ไดอะแกรมโครงสร้างโปรแกรมภาษาซี

หากอธิบายตามโปรแกรมที่ยกตัวอย่างตามลำดับจะได้รายละเอียดดังนี้

`#include <stdio.h>` มีความหมายถึง การบอกให้ ซีคอมไพเลอร์ให้นำไฟล์ อินพุทเอาท์พุทมาตรฐาน (standerd I-O) ซึ่งกำหนดฟังก์ชั่นของการติดต่ออินพุทเอาท์พุทมารวมกับไฟล์นี้

`maom()` เป็นฟังก์ชั่นที่จะต้องมีในทุกโปรแกรม โดยเครื่องหมายวงเล็บจะไว้บอกค่าตัวแปรที่จะผ่านค่าเข้าออก ในที่นี้ไม่มี

เครื่องหมาย { } และ { } มีความหมายเป็นเรื่องรูปโครงที่สำคัญภาษาซีใช้ { และ } กำหนดบล็อกของฟังก์ชั่น โดยหลักการทั่วไป ภาษาซีจะมีฟังก์ชั่นภายใน

ให้เรียกใช้จำกัด แต่สามารถเรียกจากไฟล์อื่นที่ # include ไว้ได้

ฟังก์ชั่นของภาษาซีจะมี ()ตามมา เช่น main(),printf() จะเห็นว่า () คือการแสดงว่ามีอร์กิวเมนต์ หรือตัวผ่านค่าของฟังก์ชั่นหรือไม่ เช่น main() ซึ่งในวงเล็บจะไม่มีสัญญลักษณ์ ซึ่งหมายถึงไม่มีตัวผ่านค่า แต่ printf("I think") หมายความว่าข้อความ "I think" เป็นอาร์คิวเมนต์ที่จะผ่านค่าไปยังฟังก์ชั่น printf()

```
/* ข้อความ */ หมายถึงการคอมเมนต์เหมือน REM ในเบสิก
{           หมายถึงการเริ่มต้นบล็อกของฟังก์ชั่น
int  num;   หมายถึงการกำหนดชนิดตัวแปร(declaration)โดยกำหนด
            ตัวแปร num เป็นตัวแปรชนิดเลขจำนวนเต็ม
printf      เป็นฟังก์ชั่นภายในให้พิมพ์ข้อความในวงเล็บ
\n          หมายถึงระหัสควบคุมที่บอก ซีให้ขึ้นบรรทัดใหม่
%d          เป็นฟอร์แมตที่จะพิมพ์ตัวเลข 1 ตัว
}           หมายถึงการสิ้นสุดของบล็อก
```

บทที่ 3

แผนงานและการดำเนินโครงการ

3.1 วางแผนโครงการ

คณะผู้จัดทำได้ปรึกษาและตกลงกันว่าจะทำโครงการเกี่ยวกับการประยุกต์ไมโครคอมพิวเตอร์ที่มีอยู่แพร่หลายในปัจจุบันมาควบคุมขบวนการในอุตสาหกรรม

3.2 ระยะเวลาการดำเนินโครงการ

ใช้เวลาในการดำเนินโครงการ 1 ปีการศึกษาหรือ 2 ภาคการศึกษา จึงได้แบ่งระยะเวลาในการดำเนินงานออกเป็น 2 ด้าน ดังนี้คือ

ภาคการศึกษาที่ 1:

ศึกษาและดำเนินโครงการทางด้าน HARD WARE

ภาคการศึกษาที่ 2:

ศึกษาและดำเนินโครงการทางด้าน SOFT WARE

การดำเนินโครงการ

ภาคการศึกษาที่ 1

เป้าหมาย การ์ด INTERFACE MULTI I/O PORT.

1. ศึกษาระบบและการทำงานของไมโครคอมพิวเตอร์ ว่าจะออกแบบการ์ดอินเตอร์เฟสอย่างไรและเลือกวิธีการพร้อมทั้งอุปกรณ์, วงจรและราคา
2. ศึกษาปัญหาและคาดหวังว่าจะเกิดอาการแทรกซ้อนและแนวทางแก้ไข และศึกษาความน่าจะเป็นที่จะแก้ปัญหา
3. รวบรวมข้อมูล ราคา ปัญหา ของโครงการ และตัดสินใจดำเนินการ
4. จัดซื้ออุปกรณ์และทำการทดลอง
5. ดำเนินการสร้าง ลงบน POTOTYPE BOARD โดยใช้ WIRE LAB
6. ทดสอบและตรวจสอบ HARD WARE
7. รวบรวมข้อมูลทางด้าน HARD WARE
8. ทำ REPORT เสนออาจารย์ที่ปรึกษา
9. โครงการทางด้าน HARD WARE สำเร็จ

การดำเนินโครงการ

ภาคการศึกษาที่ 2

เป้าหมาย สามารถใช้ SOFT WARE ควบคุมการทำงานและแสดงผลแบบ MENU

1. คณะผู้จัดทำปรึกษาเพื่อทำการเลือก SOFT WARE ที่เหมาะสมกับโครงการ สรุปข้อดีและข้อเสีย ทำการตัดสินใจเลือก SOFT WARE ที่จะใช้ในโครงการ
2. เขียน PROGRAM ทดสอบการดีโค้ด พอร์ต
3. เขียน PROGRAME ติดต่อ ADC
4. เขียน PROGRAM MENU
5. รวบรวมและแก้ไข PROGRAM เพื่อความเหมาะสม
6. ทดสอบ PROGRAME ทั้งหมด
7. รวบรวมข้อมูลและทำ REPORT
8. โครงการทางด้าน SOFT WARE สำเร็จผล

บทที่ 4

การทดลอง

การทดลองที่ I/O PORT

วัตถุประสงค์ เพื่อเป็นการทดสอบวงจร DECODE แลการส่งข้อมูลเข้าออก และข้อมูลจากวงจรที่ออกแบบ

เครื่องมือและอุปกรณ์

1. OSSCILO SCOPE
2. MULTI METER
3. PROTOTYPE BOARD
4. IBM PC OR EQUIVALEANCE
5. TURBO C V 2.0 PROGRAMING

ลำดับขั้นการทดลอง

1. ต่อวงจร ตามรูปที่ 1
2. ป้อน PROGRAME ดังตัวอย่างลงใน TURBO C
3. เสียบการ์ด INTERFACE ลงไปบน SLOT ของ IBM PC

สรุปผลการทดลอง

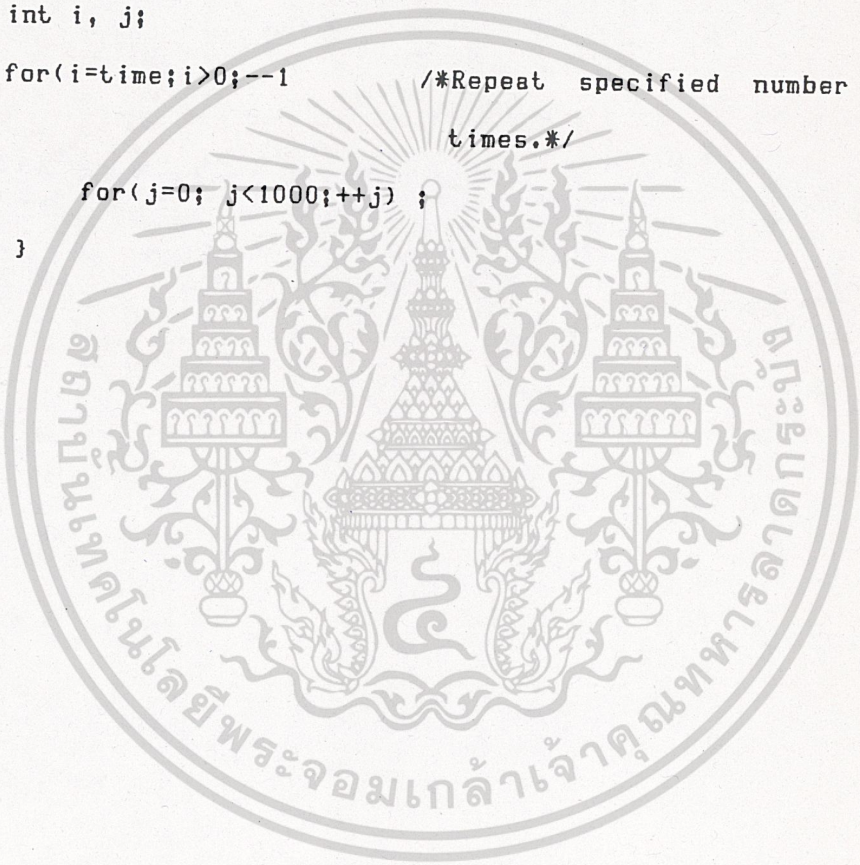
จะเห็นได้ว่า PROGRAME จะอ่านค่าจากที่ DIP SWITCH และมาแสดงที่ LED

```
#define   POORT_A   0x300      /* Port A on the 8255. */
#define   PORT_C    0x302      /* Port C on the 8255. */
#define   CTRL      0x303      /* 8255 control register. */

main()
     {
     char input ;
     int off_time , on_time = 8 ;

     outp(CTRL,  0x90) ;        /*  Set  the 8255 to mode 0 with
                                    port A as */
                                /*  an  input  port  and   other
                                    ports as */
                                /*  output  ports.  */
while   (!kbhit())              /*  Repeat  loop  until  the
                                    keboard is hit. */
    {
    input = inp (PORT_ A) ;    /* Read from DIP switches. */
    off_time = input & 0xf0 ; /*High bits Used for off time. */
    ofdf_time = off_time >> 4;/*Shift bits to lower half of */
                              /* off_time.*/
    outp(PORT_C,INPUT) ;       /* Trun LEDs ON. */
    delay(on_time) ;           /* Wait a while. */
    outp(POTY_C,0x00) ;        /* Turn LEDs off. */
    delay(off_time) ;          /* Wait awhilw. */
    }
}
```

```
delay(time)
      int time ;
      {
      int i, j;
      for(i=time;i>0;--1          /*Repeat  specified  number of
                                    times.*/
           for(j=0; j<1000;++j) ;
      }
```



บทที่ 5

วิจารณ์และสรุปผลการทดลอง

สรุปผลการทดลอง

เนื่องจากการดำเนินงานในโครงการนี้จะมี 2 ส่วนคือ HARD WARE และ SOFT WARE ทำให้มีปัญหาอยู่บ้าง เพราะบางครั้งงานทั้งสองด้านเกี่ยวข้องกันตลอดจึงทำให้มีปัญหาในด้านเวลาดำเนินการอยู่บ้าง แต่อย่างไรก็ตามเวลาก็ไม่ใช่ปัญหาหลักในการดำเนินโครงการ

ปัญหาในโครงการนี้มีปัญหาอยู่ที่ระบบ GOUND อยู่บ้าง เพราะในโครงการนี้จะมีทงสัญญาณ คือ อนาล็อกและดิจิตอล ซึ่งทำให้มีการรบกวนอยู่บ้าง และอีกประการหนึ่งก็คือการรบกวนซึ่งเกิดจากการพัน WIRE LAB แต่ปัญหาดังกล่าวมิได้มีผลต่อโครงการมากนัก

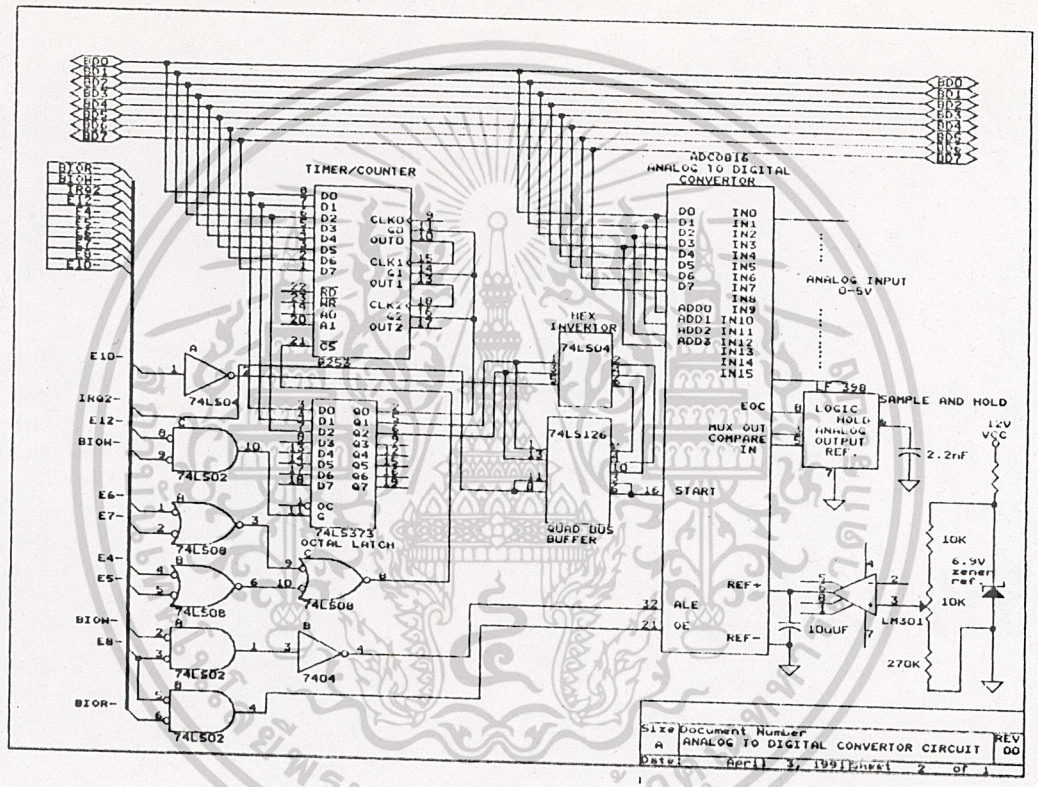
SOFT WARE ที่ใช้ในโครงการเป็นภาษาคอมพิวเตอร์ที่อาศัยหลักการทางวิธีการโปรแกรมสมัยใหม่ ที่เรียกว่า โปรแกรมโครงการ การออกแบบระบบซอฟต์แวร์จึงมีรูปแบบการออกแบบที่ง่ายเป็นโมดุล และสามารถนำมาใช้งานได้ง่าย ผู้พัฒนาโปรแกรมจึงรู้สึกว่าการใช้ภาษา C เป็นเรื่องที่เหมาะสมในการพัฒนา

แนวทางพัฒนาต่อโครงการการนี้

จากการดำเนินโครงการนี้แนวทางพัฒนาต่อคือ

- เพิ่ม DAC ให้กับระบบ

ภาคผนวก


TIMER/COUNTER
ADC0816
ANALOG TO DIGITAL
CONVERTOR
ANALOG INPUT
0-5V
HEX
INVERTOR
74LS04
74LS126
QUAD BUS
BUFFER
74LS373
OCTAL LATCH
74LS08
74LS02
7404
SAMPLE AND HOLD
LM301
2.2nF
100UF
10K
270K
6.9V
zener
ref
+12V
VCC
START
ALE
OE
EOC
REF+
REF-
MUX OUT
COMPARE
IN
LOGIC
HOLD
ANALOG
OUTPUT
REF.
Size
A
Document Number
ANALOG TO DIGITAL CONVERTOR CIRCUIT
REV
00
Date:
April 3, 1991
Sheet 2 of 3

ภาคผนวก ก.

ตารางก.1 รหัสแอสกี

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0		30	1E	▲	60	3C	<	90	5A	Z
1	1	☺	31	1F	▼	61	3D	=	91	5B	[
2	2	●	32	20		62	3E	>	92	5C	\
3	3	♥	33	21	!	63	3F	?	93	5D	]
4	4	♦	34	22	"	64	40	@	94	5E	^
5	5	♣	35	23	#	65	41	A	95	5F	_
6	6	♠	36	24	$	66	42	B	96	60	`
7	7	•	37	25	%	67	43	C	97	61	a
8	8	◘	38	26	&	68	44	D	98	62	b
9	9	○	39	27	'	69	45	E	99	63	c
10	A	◙	40	28	(	70	46	F	100	64	d
11	B	♂	41	29	)	71	47	G	101	65	e
12	C	♀	42	2A	*	72	48	H	102	66	f
13	D	♪	43	2B	+	73	49	I	103	67	g
14	E	♫	44	2C	,	74	4A	J	104	68	h
15	F	☼	45	2D	-	75	4B	K	105	69	i
16	10	►	46	2E	.	76	4C	L	106	6A	j
17	11	◄	47	2F	/	77	4D	M	107	6B	k
18	12	↕	48	30	0	78	4E	N	108	6C	l
19	13	!!	49	31	1	79	4F	O	109	6D	m
20	14	π	50	32	2	80	50	P	110	6E	n
21	15	§	51	33	3	81	51	Q	111	6F	o
22	16	▬	52	34	4	82	52	R	112	70	p
23	17	↨	53	35	5	83	53	S	113	71	q
24	18	↑	54	36	6	84	54	T	114	72	r
25	19	↓	55	37	7	85	55	U	115	73	s
26	1A	→	56	38	8	86	56	V	116	74	t
27	1B	←	57	39	9	87	57	W	117	75	u
28	1C	L	58	3A	:	88	58	X	118	76	v
29	1D	↔	59	3B	;	89	59	Y	119	77	w

ตาราง ก.1 (ต่อ)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
120	78	x	154	9A	Ü	188	BC	╝	222	DE	▌
121	79	y	155	9B	¢	189	BD	╜	223	DF	▀
122	7A	z	156	9C	£	190	BE	╛	224	E0	α
123	7B	{	157	9D	¥	191	BF	┐	225	E1	ß
124	7C	|	158	9E	Pt	192	C0	└	226	E2	Γ
125	7D	}	159	9F	ƒ	193	C1	┴	227	E3	π
126	7E	~	160	A0	á	194	C2	┬	228	E4	Σ
127	7F	⌂	161	A1	í	195	C3	├	229	E5	σ
128	80	Ç	162	A2	ó	196	C4	─	230	E6	µ
129	81	ü	163	A3	ú	197	C5	┼	231	E7	τ
130	82	é	164	A4	ñ	198	C6	╞	232	E8	Φ
131	83	â	165	A5	Ñ	199	C7	╟	233	E9	θ
132	84	ä	166	A6	ª	200	C8	╚	234	EA	Ω
133	85	à	167	A7	º	201	C9	╔	235	EB	δ
134	86	å	168	A8	¿	202	CA	╩	236	EC	∞
135	87	ç	169	A9	⌐	203	CB	╦	237	ED	φ
136	88	ê	170	AA	¬	204	CC	╠	238	EE	ε
137	89	ë	171	AB	½	205	CD	═	239	EF	∩
138	8A	è	172	AC	¼	206	CE	╬	240	F0	≡
139	8B	ï	173	AD	¡	207	CF	╧	241	F1	±
140	8C	î	174	AE	«	208	D0	╨	242	F2	≥
141	8D	ì	175	AF	»	209	D1	╤	243	F3	≤
142	8E	Ä	176	B0	▒	210	D2	╥	244	F4	⌠
143	8F	Å	177	B1	▒	211	D3	╙	245	F5	⌡
144	90	É	178	B2	▓	212	D4	╘	246	F6	÷
145	91	æ	179	B3	│	213	D5	╒	247	F7	≈
146	92	Æ	180	B4	┤	214	D6	╓	248	F8	°
147	93	ô	181	B5	╡	215	D7	╫	249	F9	∙
148	94	ö	182	B6	╢	216	D8	╪	250	FA	·
149	95	ò	183	B7	╖	217	D9	┘	251	FB	√
150	96	û	184	B8	╕	218	DA	┌	252	FC	n
151	97	ù	185	B9	╣	219	DB	▒	253	FD	²
152	98	ÿ	186	BA	║	220	DC	▄	254	FE	■
153	99	Ö	187	BB	╗	221	DD	▌	255	FF	

หมายเหตุ Dec หมายถึงเลขฐานสิบ
Hex หมายถึงเลขฐานสิบหก
Char หมายถึงตัวอักขระ

National Semiconductor

PRELIMINARY

DM74AS245A Octal TRI-STATE® Bus Transceivers

General Description

This advanced Schottky device contains 8 pairs of TRI-STATE logic elements configured as octal bus transceivers. These circuits are designed for use in memory, microprocessor systems and in asynchronous bidirectional data buses. Two way communication between buses is controlled by the (DIR) input. Data transmits either from the A bus to the B bus or from the B bus to the A bus. Both the driver and receiver outputs can be disabled via the ($\overline{G}$) enable input which causes outputs to enter the high impedance mode so that the buses are effectively isolated.

Features

- Advanced oxide-isolated, ion implanted Schottky TTL process
- Non-inverting logic output
- TRI-STATE outputs independently controlled on A and B buses
- Low output impedance to drive terminated transmission lines to 133Ω
- Switching response specified into 500Ω/50 pF
- Specified to interface with CMOS at $V_{OH} = V_{CC} - 2V$
- PNP inputs reduce input loading
- Switching specifications guaranteed over full temperature and V_{CC} range

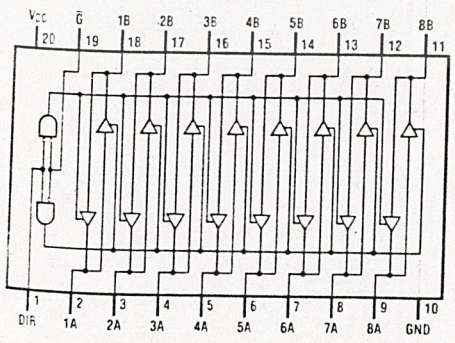
Absolute Maximum Ratings

Supply Voltage, V_{CC}	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM74AS	0°C to 70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device can not be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Connection Diagram

Dual-In-Line Package



TL/F/6299-1

DM74AS245A (J, N)

Function Table

Control Inputs		Operation
$\overline{G}$	DIR	
L	L	B Data to A Bus
L	H	A Data to B Bus
H	X	Hi-Z

This document contains information on a product under development. NSC reserves the right to change or discontinue this product without notice.

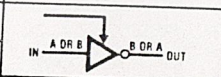
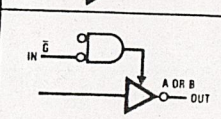
Recommended Operating Conditions

Symbol	Parameter	DM74AS245A			
		Min	Typ	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	
V_{IH}	High Level Input Voltage	2			
V_{IL}	Low Level Input Voltage			0.8	
I_{OH}	High Level Output Current			−15	
I_{OL}	Low Level Output Current			48	

Electrical Characteristics over recommended operating free air temperature range (Note 1)

Symbol	Parameter	Conditions		Min	Typ	Max	Un
V_{IK}	Input Clamp Voltage	V_{CC} = 4.5V, I_{IN} = −18 mA				−1.2	
V_{OH}	High Level Output Voltage	V_{CC} = 4.5V, I_{OH} = −3 mA		2.4	3.2		
		V_{CC} = 4.5V, I_{OH} = −15 mA		2.0	2.3		
		I_{OH} = −2 mA, V_{CC} = 4.5V to 5.5V		V_{CC} − 2			
V_{OL}	Low Level Output Voltage	V_{CC} = 4.5V, I_{OL} = Max			0.35	0.5	
I_I	Input Current at Max Input Voltage	V_{CC} = 5.5V, V_{IN} = 7V (V_{IN} = 5.5V for A or B Ports)				0.1	m
I_{IH}	High Level Input Current	V_{CC} = 5.5V, V_{IN} = 2.7V	Control Inputs			20	
			A or B Ports			50	
I_{IL}	Low Level Input Current	V_{CC} = 5.5V, V_{IN} = 0.4V	Control Inputs			−0.1	m
			A or B Ports			−0.75	
I_O	Output Drive Current	V_{CC} = 5.5V, V_{OUT} = 2.25V		−30		−112	m
I_{CC}	Supply Current	V_{CC} = 5.5V	Outputs High		62		m
			Outputs Low		75		
			TRI-STATE		79		

Switching Characteristics over recommended operating free air temperature range (Notes 1 and 2)

Parameter (Propagation Delay Time)	Circuit Configuration	DM74AS245A			Un
		Min	Typ	Max	
t_{PLH}, High-to-Low Level Output			6		n
t_{PHL}, High-to-Low Level Output			5		n
t_{PZL}, Output Enable to Low Level			8		n
t_{PZH}, Output Enable to High Level			8		n
t_{PLZ}, Output Disable from Low Level			5		n
t_{PHZ}, Output Disable from High Level			4.5		n

Note 1: See Section 1 for test waveforms and output load.

Note 2: Switching characteristic conditions are V_{CC} = 4.5V to 5.5V, R_L = 500Ω, C_L = 50 pF.

National Semiconductor

DM54LS154/DM74LS154 4-Line to 16-Line Decoders/Demultiplexers

General Description

Each of these 4-line-to-16-line decoders utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, G1 and G2, are low. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. These demultiplexers are ideally suited for implementing high-performance memory decoders. All inputs are buffered and input clamping diodes are provided to minimize transmission-line effects and thereby simplify system design.

Features

- Decodes 4 binary-coded inputs into one of 16 mutually exclusive outputs
- Performs the demultiplexing function by distributing data from one input line to any one of 16 outputs
- Input clamping diodes simplify system design
- High fan-out, low-impedance, totem-pole outputs
- Typical propagation delay
  3 levels of logic 23 ns
  Strobe 19 ns
- Typical power dissipation 45 mW

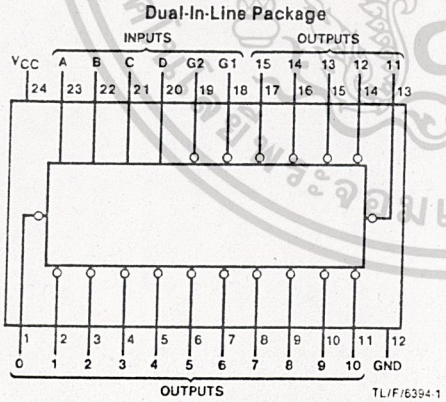
Absolute Maximum Ratings (Note 1)

Supply Voltage	7V
Input Voltage	7V
Storage Temperature Range	−65°C to 150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

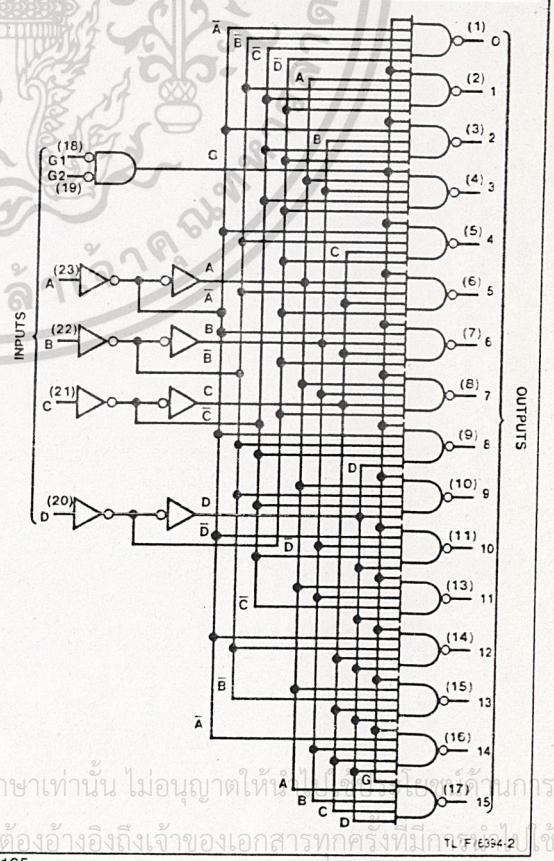
Connection and Logic Diagrams

Dual-In-Line Package



TL/F/6394-1

54LS154 (J) 74LS154 (N)



TL/F/6394-2

Recommended Operating Conditions

Symbol	Parameter	DM54LS154			DM74LS154		
		Min	Nom	Max	Min	Nom	Max
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25
V_{IH}	High Level Input Voltage	2			2		
V_{IL}	Low Level Input Voltage			0.7			0.8
I_{OH}	High Level Output Current			−0.4			−0.4
I_{OL}	Low Level Output Current			4			8
T_A	Free Air Operating Temperature	−55		125	0		70

Electrical Characteristics over recommended operating free air temperature (unless otherwise note

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	U
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	m
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µ
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.4	m
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	m
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max (Note 3)			9	14	m

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 3: I_{CC} is measured with all outputs open and all inputs grounded.

Switching Characteristics at $V_{CC} = 5V$ and $T_A = 25°C$ (See Section 1 for Test Waveforms and Output Load)

Parameter	From (Input) To (Output)	$R_L = 2\ k\Omega$						Units
		$C_L = 15\ pF$			$C_L = 50\ pF$			
		Min	Typ	Max	Min	Typ	Max	
t_{PLH} Propagation Delay Time Low to High Level Output	Data to Output		18	30		22	35	ns
t_{PHL} Propagation Delay Time High to Low Level Output	Data to Output		18	30		24	35	ns
t_{PLH} Propagation Delay Time Low to High Level Output	Strobe to Output		12	20		15	25	ns
t_{PHL} Propagation Delay Time High to Low Level Output	Strobe to Output		16	25		23	35	ns

Function Table

Inputs						Outputs														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = High Level, L = Low Level, X = Don't Care

D

8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2.6 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  - —Standard Temperature Range
  - —Extended Temperature Range

The Intel® 8253 is a programmable counter/timer device designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2.6 MHz. All modes of operation are software programmable.

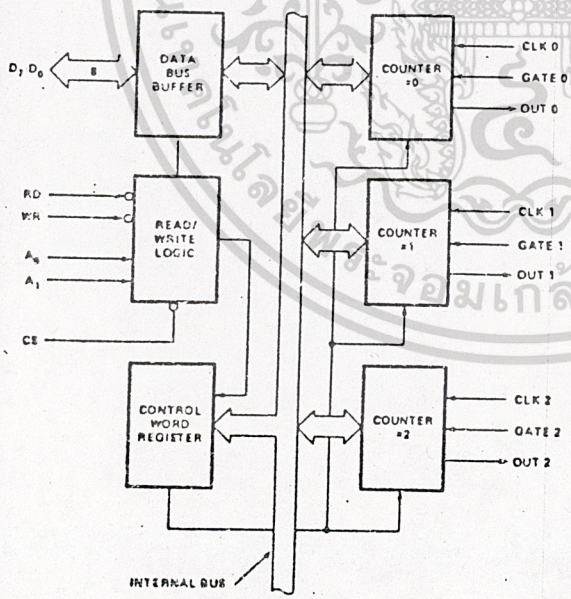


Figure 1. Block Diagram

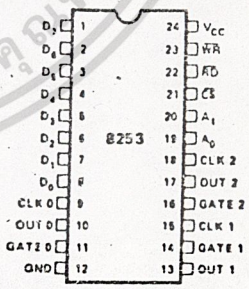


Figure 2. Pin Configuration

242

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground ..... −0.5 V to +7 V
Power Dissipation ..... 1 Watt

*NOTICE: Stresses above those listed under "Absolu Maximum Ratings" may cause permanent damage to t device. This is a stress rating only and functional ope tion of the device at these or any other conditions abo those indicated in the operational sections of th specification is not implied. Exposure to absolute ma mum rating conditions for extended periods may affe device reliability.

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5V ± 1

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.2	V_{CC}+.5V	V	
V_{OL}	Output Low Voltage		0.45	V	Note 1
V_{OH}	Output High Voltage	2.4		V	Note 2
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to .45V
I_{CC}	V_{CC} Supply Current		140	mA	

CAPACITANCE (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%, GND = 0V) *

Bus Parameters (Note 3)

READ CYCLE

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AR}	Address Stable Before $\overline{READ}$	50		30		ns
t_{RA}	Address Hold Time for $\overline{READ}$	5		5		ns
t_{RR}	$\overline{READ}$ Pulse Width	400		300		ns
t_{RD}	Data Delay From $\overline{READ}$[4]		300		200	ns
t_{DF}	$\overline{READ}$ to Data Floating	25	125	25	100	ns
t_{RV}	Recovery Time Between READ and Any Other Control Signal	1		1		μs

A.C. CHARACTERISTICS (Continued)

WRITE CYCLE

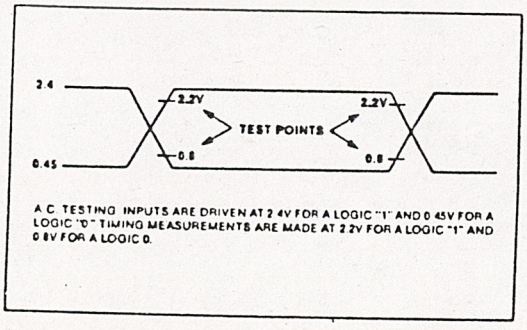
Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{AW}	Address Stable Before $\overline{WRITE}$	50		30		ns
t_{WA}	Address Hold Time for $\overline{WRITE}$	30		30		ns
t_{WW}	$\overline{WRITE}$ Pulse Width	400		300		ns
t_{DW}	Data Set Up Time for $\overline{WRITE}$	300		250		ns
t_{WD}	Data Hold Time for $\overline{WRITE}$	40		30		ns
t_{RV}	Recovery Time Between $\overline{WRITE}$ and Any Other Control Signal	1		1		μs

CLOCK AND GATE TIMING

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
t_{CLK}	Clock Period	380	dc	380	dc	ns
t_{PWH}	High Pulse Width	230		230		ns
t_{PWL}	Low Pulse Width	150		150		ns
t_{GW}	Gate Width High	150		150		ns
t_{GL}	Gate Width Low	100		100		ns
t_{GS}	Gate Set Up Time to CLK↑	100		100		ns
t_{GH}	Gate Hold Time After CLK↑	50		50		ns
t_{OD}	Output Delay From CLK↓[4]		400		400	ns
t_{ODG}	Output Delay From Gate↓[4]		300		300	ns

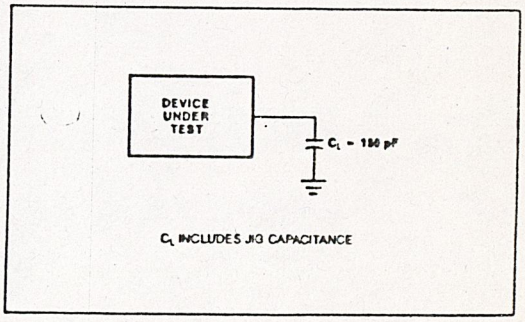
NOTES:
1. I_{OL} = 2.2 mA.
2. I_{OH} = −400 μA.
3. AC timings measured at V_{OH} 2.2, V_{OL} = 0.8.
4. C_L = 150pF.
* For Extended Temperature EXPRESS, use M8253 electrical parameters.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. TESTING INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". TIMING MEASUREMENTS ARE MADE AT 2.2V FOR A LOGIC "1" AND 0.8V FOR A LOGIC 0.

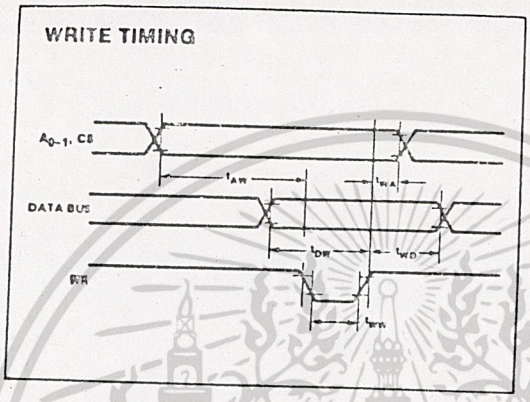
A.C. TESTING LOAD CIRCUIT

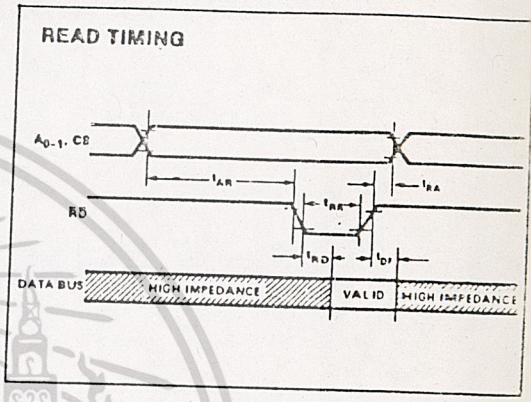


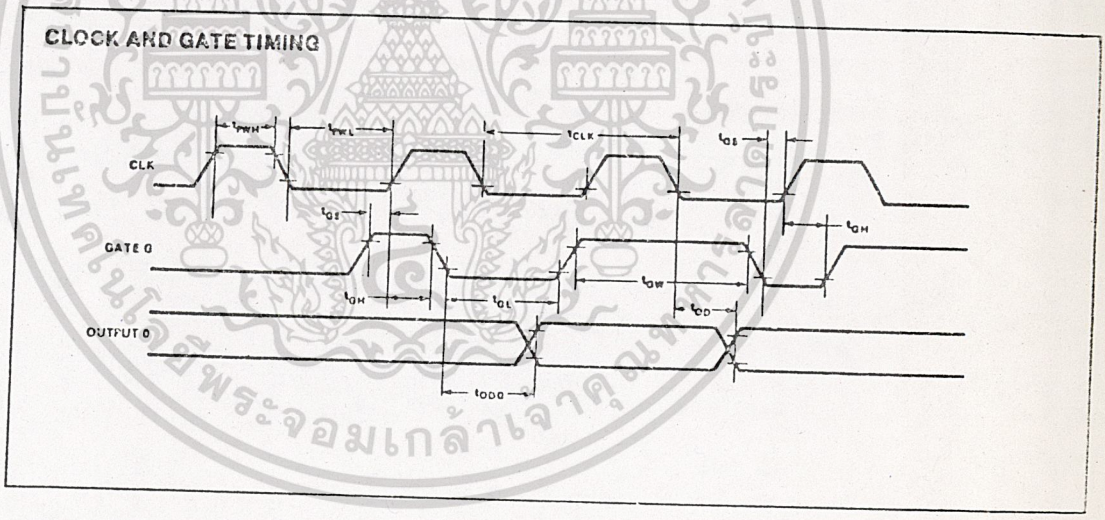
C_L INCLUDES JIG CAPACITANCE

244

WAVEFORMS







National Semiconductor

LF198/LF298/LF398, LF198A/LF398A Monolithic Sample and Hold Circuits

General Description

The LF198/LF298/LF398 are monolithic sample and hold circuits which utilize BI-FET technology to obtain ultra-high dc accuracy with fast acquisition of signal and low droop rate. Operating as a unity gain follower, dc gain accuracy is 0.002% typical and acquisition time is as low as 6μs to 0.01%. A bipolar input stage is used to achieve low offset voltage and wide bandwidth. Input offset adjust is accomplished with a single pin and does not degrade input offset drift. The wide bandwidth allows the LF198 to be included inside the feedback loop of 1 MHz op amps without having stability problems. Input impedance of $10^{10}\Omega$ allows high source impedances to be used without degrading accuracy.

P-channel junction FET's are combined with bipolar devices in the output amplifier to give droop rates as low as 5 mV/min with a 1μF hold capacitor. The JFET's have much lower noise than MOS devices used in previous designs and do not exhibit high temperature instabilities. The overall design guarantees no feedthrough from input to output in the hold mode even for input signals equal to the supply voltages.

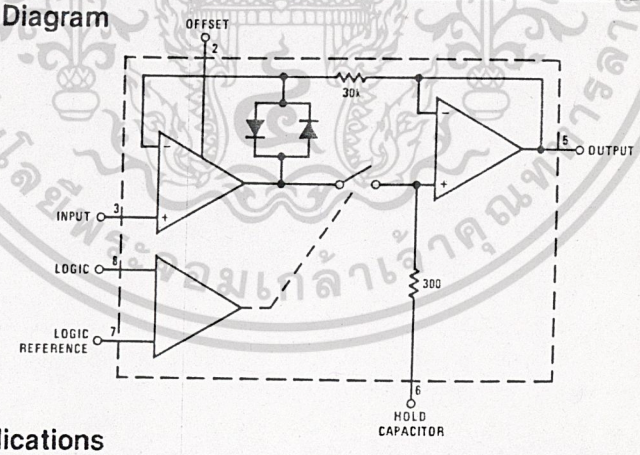
Features

- Operates from ±5V to ±18V supplies
- Less than 10μs acquisition time
- TTL, PMOS, CMOS compatible logic input
- 0.5 mV typical hold step at C_h = 0.01μF
- Low input offset
- 0.002% gain accuracy
- Low output noise in hold mode
- Input characteristics do not change during hold mode
- High supply rejection ratio in sample or hold
- Wide bandwidth

Logic inputs on the LF198 are fully differential with low input current, allowing direct connection to TTL, PMOS, and CMOS. Differential threshold is 1.4V. The LF198 will operate from ±5V to ±18V supplies. It is available in an 8-lead TO-5 package.

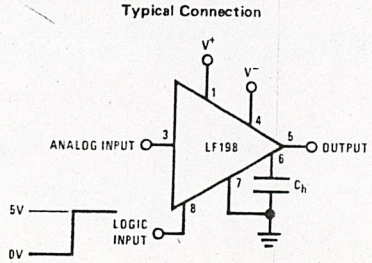
An "A" version is available with tightened electrical specifications.

Functional Diagram

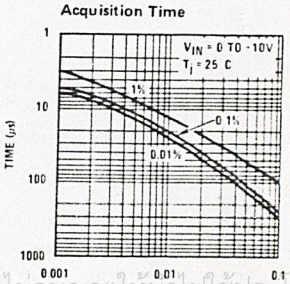


Typical Applications

Typical Connection



Acquisition Time



Absolute Maximum Ratings

Supply Voltage	±18V
Power Dissipation (Package Limitation) (Note 1)	500 mW
Operating Ambient Temperature Range	
LF198/LF198A	−55°C to +125°C
LF298	−25°C to +85°C
LF398/LF398A	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Input Voltage	Equal to Supply Voltage
Logic To Logic Reference Differential Voltage (Note 2)	+7V
Output Short Circuit Duration	Indefinite
Hold Capacitor Short Circuit Duration	10
Lead Temperature (Soldering, 10 seconds)	300

Electrical Characteristics (Note 3)

PARAMETER	CONDITIONS	LF198/LF298			LF398			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
Input Offset Voltage, (Note 6)	T_j = 25°C		1	3		2	7	mV
	Full Temperature Range			5			10	mV
Input Bias Current, (Note 6)	T_j = 25°C		5	25		10	50	nA
	Full Temperature Range			75			100	nA
Input Impedance	T_j = 25°C		10^{10}			10^{10}		Ω
Gain Error	T_j = 25°C, R_L = 10k		0.002	0.005		0.004	0.01	%
	Full Temperature Range			0.02			0.02	%
Feedthrough Attenuation Ratio at 1 kHz	T_j = 25°C, C_h = 0.01μF	86	96		80	90		dB
Output Impedance	T_j = 25°C, "HOLD" mode		0.5	2		0.5	4	Ω
	Full Temperature Range			4			6	Ω
"HOLD" Step, (Note 4)	T_j = 25°C, C_h = 0.01μF, V_{OUT} = 0		0.5	2.0		1.0	2.5	mV
Supply Current, (Note 6)	$T_j \geq$ 25°C		4.5	5.5		4.5	6.5	mA
Logic and Logic Reference Input Current	T_j = 25°C		2	10		2	10	μA
Leakage Current into Hold Capacitor (Note 6)	T_j = 25°C, (Note 5) Hold Mode		30	100		30	200	pA
Acquisition Time to 0.1%	ΔV_{OUT} = 10V, C_h = 1000 pF		4			4		μs
	C_h = 0.01μF		20			20		μs
Hold Capacitor Charging Current	$V_{IN} - V_{OUT}$ = 2V		5			5		mA
Supply Voltage Rejection Ratio	V_{OUT} = 0	80	110		80	110		dB
Differential Logic Threshold	T_j = 25°C	0.8	1.4	2.4	0.8	1.4	2.4	V

Electrical Characteristics (Continued) (Note 3)

PARAMETER	CONDITIONS	LF198A			LF398A			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
Input Offset Voltage, (Note 6)	$T_j = 25°C$		1	1		2	2	mV
	Full Temperature Range			2			3	mV
Input Bias Current, (Note 6)	$T_j = 25°C$		5	25		10	25	nA
	Full Temperature Range			75			50	nA
Input Impedance	$T_j = 25°C$		10^{10}			10^{10}		Ω
Gain Error	$T_j = 25°C$, $R_L = 10k$		0.002	0.005		0.004	0.005	%
	Full Temperature Range			0.01			0.01	%
Feedthrough Attenuation Ratio at 1 kHz	$T_j = 25°C$, $C_h = 0.01\mu F$	86	96		86	90		dB
Output Impedance	$T_j = 25°C$, "HOLD" mode		0.5	1		0.5	1	Ω
	Full Temperature Range			4			6	Ω
"HOLD" Step, (Note 4)	$T_j = 25°C$, $C_h = 0.01\mu F$, $V_{OUT} = 0$		0.5	1		1.0	1	mV
Supply Current, (Note 6)	$T_j \geq 25°C$		4.5	5.5		4.5	6.5	mA
Logic and Logic Reference Input Current	$T_j = 25°C$		2	10		2	10	μA
Leakage Current into Hold Capacitor (Note 6)	$T_j = 25°C$, (Note 5) Hold Mode		30	100		30	100	pA
Acquisition Time to 0.1%	$\Delta V_{OUT} = 10V$, $C_h = 1000$ pF		4	6		4	6	μs
	$C_h = 0.01\mu F$		20	25		20	25	μs
Hold Capacitor Charging Current	$V_{IN} - V_{OUT} = 2V$		5			5		mA
Supply Voltage Rejection Ratio	$V_{OUT} = 0$	90	110		90	110		dB
Differential Logic Threshold	$T_j = 25°C$	0.8	1.4	2.4	0.8	1.4	2.4	V

Note 1: The maximum junction temperature of the LF198/LF198A is 150°C, for the LF298, 115°C, and for the LF398/LF398A, 100°C. When operating at elevated ambient temperature, the power dissipation must be derated based on a thermal resistance (Θ_{jA}) of 150°C/W.

Note 2: Although the differential voltage may not exceed the limits given, the common-mode voltage on the logic pins may be equal to the supply voltages without causing damage to the circuit. For proper logic operation, however, one of the logic pins must always be at least 2V below the positive supply and 3V above the negative supply.

Note 3: Unless otherwise specified, the following conditions apply. Unit is in "sample" mode, $V_S = \pm 15V$, $T_j = 25°C$, $-11.5V \leq V_{IN} \leq +11.5V$, $C_h = 0.01\mu F$, and $R_L = 10\ k\Omega$. Logic reference voltage = 0V and logic voltage = 2.5V.

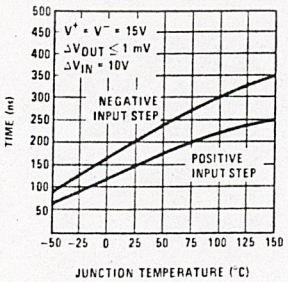
Note 4: Hold step is sensitive to stray capacitive coupling between input logic signals and the hold capacitor. 1 pF, for instance, will create an additional 0.5 mV step with a 5V logic swing and a 0.01μF hold capacitor. Magnitude of the hold step is inversely proportional to hold capacitor value.

Note 5: Leakage current is measured at a *junction* temperature of 25°C. The effects of junction temperature rise due to power dissipation or elevated ambient can be calculated by doubling the 25°C value for each 11°C increase in chip temperature. Leakage is guaranteed over full input signal range.

Note 6: These parameters guaranteed over a supply voltage range of ±5 to ±18V.

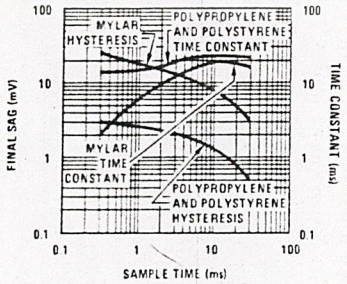
Typical Performance Characteristics

Aperture Time*

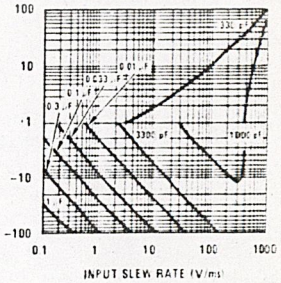


*See Definition of Terms

Capacitor Hysteresis

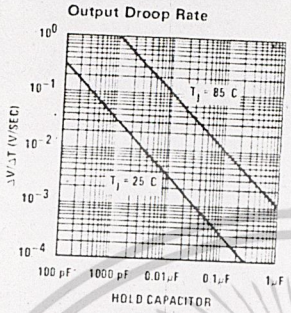


Dynamic Sampling Error

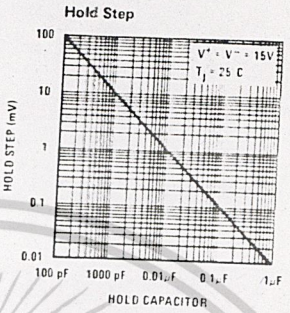


Typical Performance Characteristics (Continued)

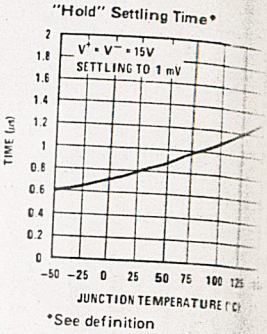
Output Droop Rate



Hold Step

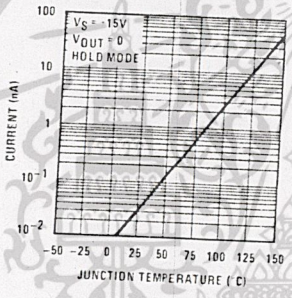


"Hold" Settling Time*

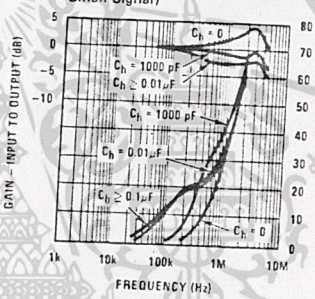


*See definition

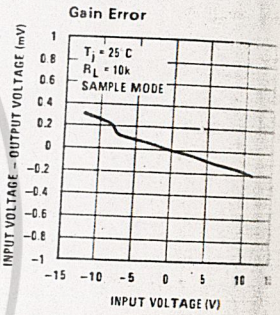
Leakage Current into Hold Capacitor



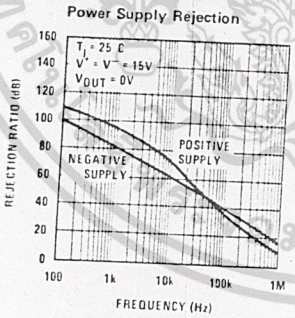
Phase and Gain (Input to Output, Small Signal)



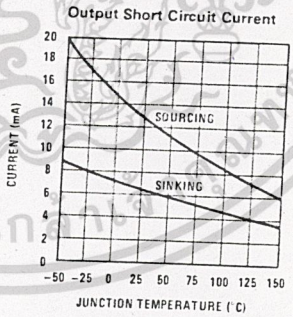
Gain Error



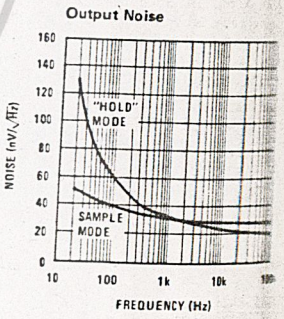
Power Supply Rejection



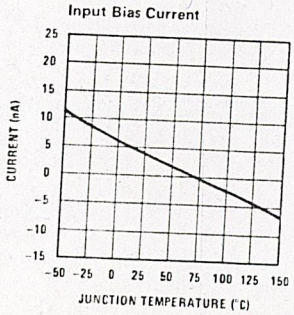
Output Short Circuit Current



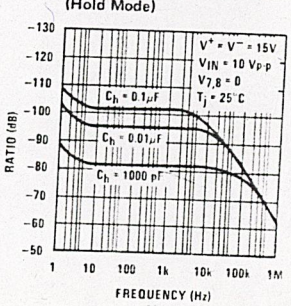
Output Noise



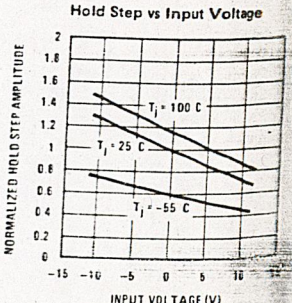
Input Bias Current



Feedthrough Rejection Ratio (Hold Mode)



Hold Step vs Input Voltage



Application Hints

Hold Capacitor

Hold step, acquisition time, and droop rate are the major trade-offs in the selection of a hold capacitor value. Size and cost may also become important for larger values. Use of the curves included with this data sheet should be helpful in selecting a reasonable value of capacitance. Keep in mind that for fast repetition rates or tracking fast signals, the capacitor drive currents may cause a significant temperature rise in the LF198.

A significant source of error in an accurate sample and hold circuit is dielectric absorption in the hold capacitor. A mylar cap, for instance, may "sag back" up to 0.2% after a quick change in voltage. A long "soak" time is required before the circuit can be put back into the hold mode with this type of capacitor. Dielectrics with very low hysteresis are polystyrene, polypropylene, and Teflon. Other types such as mica and polycarbonate are not nearly as good. Ceramic is unusable with $> 1\%$ hysteresis. The advantage of polypropylene over polystyrene is that it extends the maximum ambient temperature from 85°C to 100°C. "NPO" or "COG" capacitors are now available for 125°C operation and also have low dielectric absorption. For more exact data, see the curve labeled dielectric absorption error vs sample time. The hysteresis numbers on the curve are final values, taken after full relaxation. The hysteresis error can be significantly reduced if the output of the LF198 is digitized quickly after the hold mode is initiated. The hysteresis relaxation time constant in polypropylene, for instance, is 10–50 ms. If A-to-D conversion can be made within 1 ms, hysteresis error will be reduced by a factor of ten.

DC and AC Zeroing

DC zeroing is accomplished by connecting the offset adjust pin to the wiper of a 1 kΩ potentiometer which has one end tied to V^+ and the other end tied through a resistor to ground. The resistor should be selected to give ≈0.6 mA through the 1k potentiometer.

AC zeroing (hold step zeroing) can be obtained by adding an inverter with the adjustment pot tied input to output. A 10 pF capacitor from the wiper to the hold capacitor will give ±4 mV hold step adjustment with a 0.01μF hold capacitor and 5V logic supply. For larger logic swings, a smaller capacitor (< 10 pF) may be used.

Logic Rise Time

For proper operation, logic signals into the LF198 must have a minimum dV/dt of 1.0 V/μs. Slower signals will cause excessive hold step. If a R/C network is used in front of the logic input for signal delay, calculate the slope of the waveform at the threshold point to ensure that it is at least 1.0 V/μs.

Sampling Dynamic Signals

Sample error due to moving input signals probably causes more confusion among sample-and-hold users than any other parameter. The primary reason for this is that many users make the assumption that the sample and hold amplifier is truly locked on to the input signal while in the sample mode. In actuality, there are finite phase delays through the circuit creating an input-output differential for fast moving signals. In addition, although the output may have settled, the hold capacitor has an additional lag due to the 300Ω series resistor on the chip. This means that at the moment the "hold" command arrives, the hold capacitor voltage may be somewhat different than the actual analog input. The effect of these delays is opposite to the effect created by delays in the logic which switches the circuit from sample to hold. For example, consider an analog input of 20 Vp-p at 10 kHz. Maximum dV/dt is 0.6 V/μs. With no analog phase delay and 100 ns logic delay, one could expect up to (0.1μs)(0.6V/μs) = 60 mV error if the "hold" signal arrived near maximum dV/dt of the input. A positive-going input would give a +60 mV error. Now assume a 1 MHz (3 dB) bandwidth for the overall analog loop. This generates a phase delay of 160 ns. If the hold capacitor sees this exact delay, then error due to analog delay will be (0.16μs)(0.6 V/μs) = −96 mV. Total output error is +60 mV (digital) −96 mV (analog) for a total of −36 mV. To add to the confusion, analog delay is proportional to hold capacitor value while digital delay remains constant. A family of curves (dynamic sampling error) is included to help estimate errors.

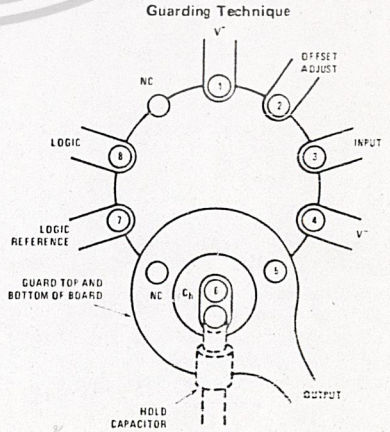
A curve labeled Aperture Time has been included for sampling conditions where the input is steady during the sampling period, but may experience a sudden change nearly coincident with the "hold" command. This curve is based on a 1 mV error fed into the output.

A second curve, Hold Settling Time indicates the time required for the output to settle to 1 mV after the "hold" command.

Digital Feedthrough

Fast rise time logic signals can cause hold errors by feeding externally into the analog input at the same time the amplifier is put into the hold mode. To minimize this problem, board layout should keep logic lines as far as possible from the analog input. Grounded guarding traces may also be used around the input line, especially if it is driven from a high impedance source. Reducing high amplitude logic signals to 2.5V will also help.

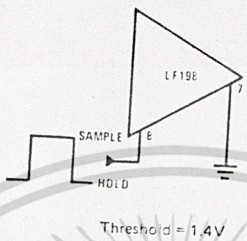
Guarding Technique



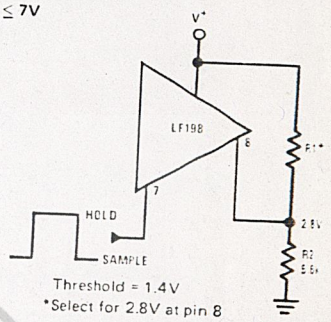
Use 10-pin layout. Guard around C_h is tied to output.

Logic Input Configurations

TTL & CMOS
$3V \leq V_L$ (Hi State) $\leq 7V$

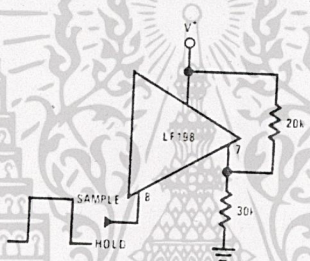


Threshold = 1.4V

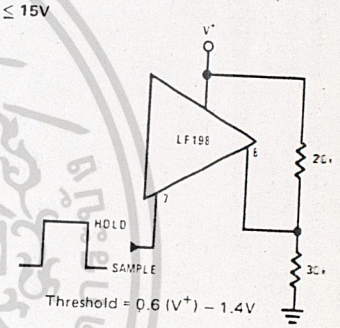


Threshold = 1.4V
*Select for 2.8V at pin 8

CMOS
$7V \leq V_L$ (Hi State) $\leq 15V$

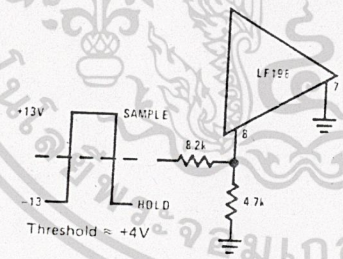


Threshold = 0.6 (V^+) + 1.4V

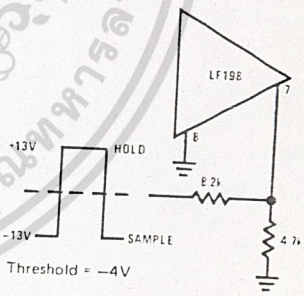


Threshold = 0.6 (V^+) − 1.4V

Op Amp Drive



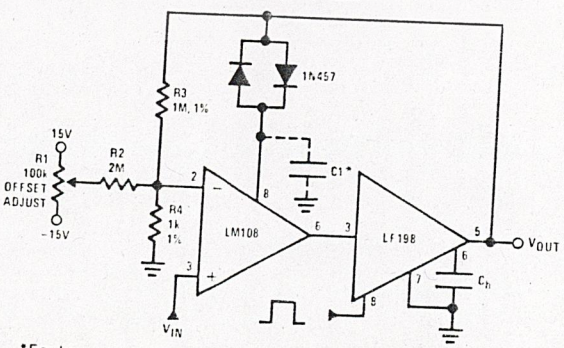
Threshold = +4V



Threshold = −4V

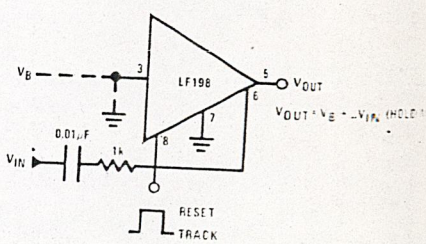
Typical Applications (Continued)

X1000 Sample & Hold



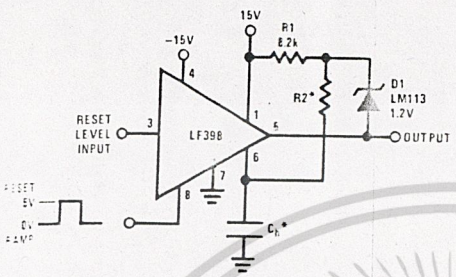
*For lower gains, the LM108 must be frequency compensated
Use $\approx \frac{100}{A_V}$ pF from comp 2 to ground

Sample and Difference Circuit
(Output Follows Input in Hold Mode)



Typical Applications (Continued)

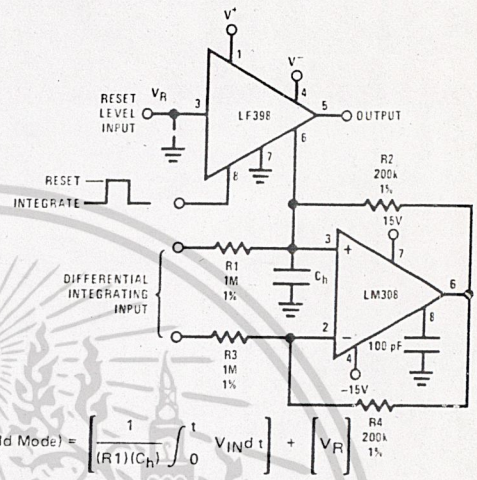
Ramp Generator with Variable Reset Level



*Select for ramp rate $R \geq 10k$

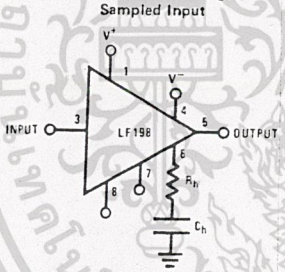
$$\frac{\Delta V}{\Delta T} = \frac{1.2V}{(R2)(C_h)}$$

Integrator with Programmable Reset Level



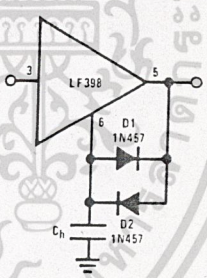
$$V_{OUT}\ (\text{Hold Mode}) = \left[\frac{1}{(R1)(C_h)}\int_0^t V_{IN}\,dt\right] + \left[V_R\right]$$

Output Holds at Average of Sampled Input

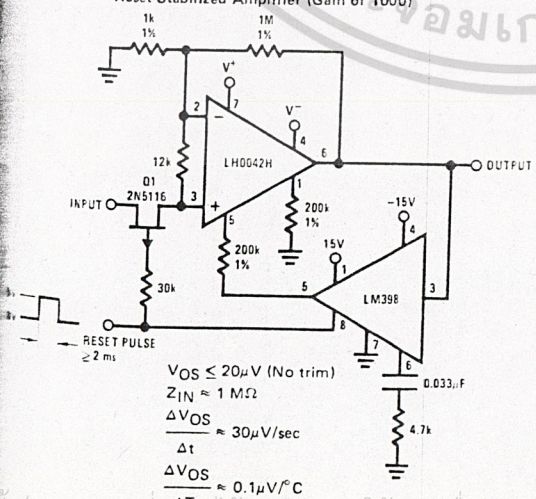


$$\text{Select } (R_h)(C_h) >> \frac{1}{2\pi f_{IN}\ (\text{Min})}$$

Increased Slew Current



Reset Stabilized Amplifier (Gain of 1000)



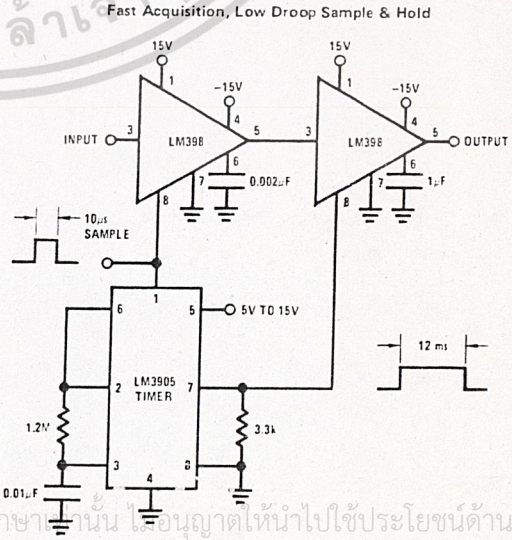
$V_{OS} \leq 20\mu V$ (No trim)

$Z_{IN} \approx 1\ M\Omega$

$$\frac{\Delta V_{OS}}{\Delta t} \approx 30\mu V/\text{sec}$$

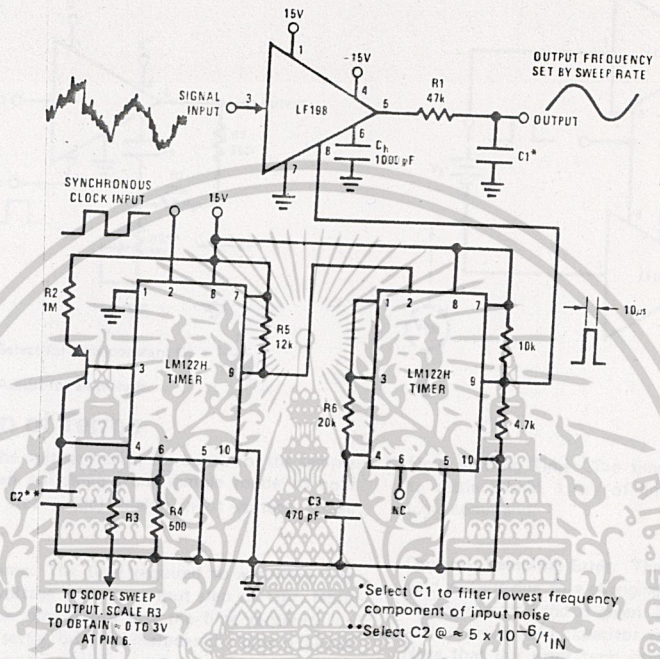
$$\frac{\Delta V_{OS}}{\Delta T} \approx 0.1\mu V/^{\circ}C$$

Fast Acquisition, Low Droop Sample & Hold



Typical Applications (Continued)

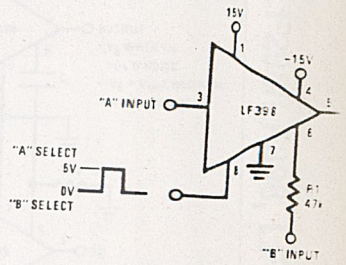
Synchronous Correlator for Recovering Signals Below Noise Level



*Select C1 to filter lowest frequency component of input noise

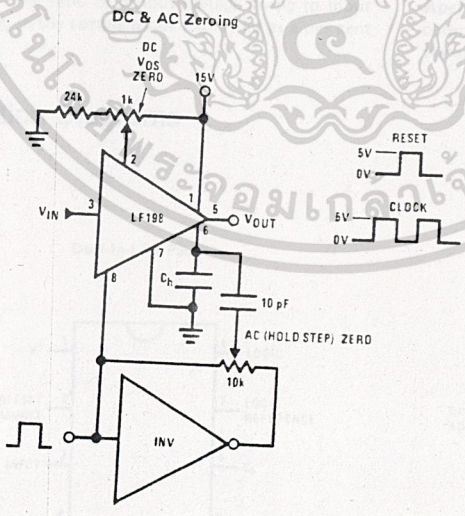
**Select C2 @ ≈ $5 \times 10^{-6}/f_{IN}$

2-Channel Switch

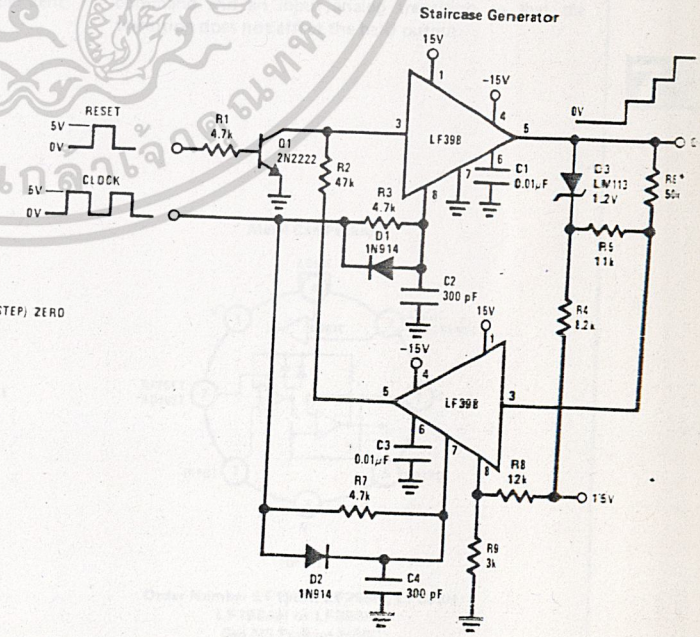


	A	B
Gain	1 ±0.02%	1 ±0.2
Z_{IN}	$10^{10}\Omega$	47 kΩ
BW	≃ 1 MHz	≈400 k
Crosstalk @ 1 kHz	−90 dB	−90 dB
Offset	≤ 6 mV	≤ 75 m

DC & AC Zeroing



Staircase Generator



*Select for step height 50k → ≃ 1V Step

National Semiconductor

Operational Amplifiers/Buffers

LM101A/LM201A/LM301A Operational Amplifiers

General Description

The LM101A series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. Advanced processing techniques make possible an order of magnitude reduction in input currents, and a redesign of the biasing circuitry reduces the temperature drift of input current. Improved specifications include:

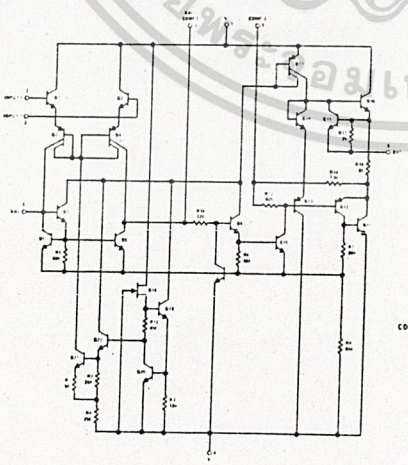
- Offset voltage 3 mV maximum over temperature (LM101A/LM201A)
- Input current 100 nA maximum over temperature (LM101A/LM201A)
- Offset current 20 nA maximum over temperature (LM101A/LM201A)
- Guaranteed drift characteristics
- Offsets guaranteed over entire common mode and supply voltage ranges
- Slew rate of 10V/μs as a summing amplifier

This amplifier offers many features which make its application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, freedom from oscillations and compensation with a single 30 pF capacitor. It has advantages over internally compensated amplifiers in that the frequency compensation can be tailored to the particular application. For example, in low frequency circuits it can be overcompensated for increased stability margin. Or the compensation can be optimized to give more than a factor of ten improvement in high frequency performance for most applications.

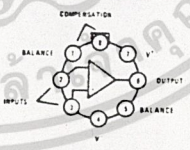
In addition, the device provides better accuracy and lower noise in high impedance circuitry. The low input currents also make it particularly well suited for long interval integrators or timers, sample and hold circuits and low frequency waveform generators. Further, replacing circuits where matched transistor pairs buffer the inputs of conventional IC op amps, it can give lower offset voltage and drift at a lower cost.

The LM101A is guaranteed over a temperature range of −55°C to +125°C, the LM201A from −25°C to +85°C, and the LM301A from 0°C to 70°C.

Schematic ** and Connection Diagrams (Top Views)

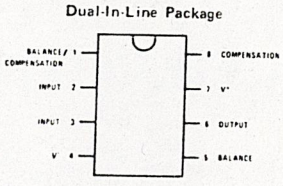


Metal Can Package



Order Number LM101AH, LM201AH or LM301AH
See NS Package H08C

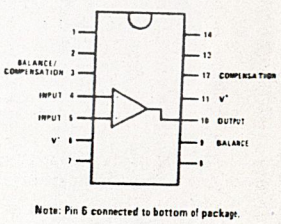
Dual-In-Line Package



Order Number
LM101AJ, LM201AJ, LM301AJ
See NS Package J08A

Order Number LM301AN
See NS Package N08A

Dual-In-Line Package



Note: Pin 6 connected to bottom of package.

Order Number LM101AJ-14
LM201AJ-14 or LM301AJ-14
See NS Package J14A

**Pin connections shown are for metal can.

Absolute Maximum Ratings

	LM101A/LM201A	LM301A
Supply Voltage	±22V	±18V
Power Dissipation (Note 1)	500 mW	500 mW
Differential Input Voltage	±30V	±30V
Input Voltage (Note 2)	±15V	±15V
Output Short Circuit Duration (Note 3)	Indefinite	Indefinite
Operating Temperature Range	−55°C to +125°C (LM101A) −25°C to +85°C (LM201A)	0°C to +70°C
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C	300°C

Electrical Characteristics (Note 4)

PARAMETER	CONDITIONS	LM101A/LM201A			LM301A			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	
Input Offset Voltage LM101A, LM201A, LM301A	$T_A = 25°C$ $R_S \leq 50\ k\Omega$		0.7	2.0		2.0	7.5	mV
Input Offset Current	$T_A = 25°C$		1.5	10		3.0	50	nA
Input Bias Current	$T_A = 25°C$		30	75		70	250	nA
Input Resistance	$T_A = 25°C$	1.5	4.0		0.5	2.0		MΩ
Supply Current	$T_A = 25°C$ $V_S = \pm 20V$ $V_S = \pm 15V$		1.8	3.0		1.8	3.0	mA mA
Large Signal Voltage Gain	$T_A = 25°C$, $V_S = \pm 15V$ $V_{OUT} = \pm 10V$, $R_L \geq 2\ k\Omega$	50	160		25	160		V/mV
Input Offset Voltage	$R_S \leq 50\ k\Omega$ $R_S \leq 10\ k\Omega$			3.0			10	mV mV
Average Temperature Coefficient of Input Offset Voltage	$R_S \leq 50\ k\Omega$ $R_S \leq 10\ k\Omega$		3.0	15		6.0	30	μV/°C μV/°C
Input Offset Current	$T_A = T_{MAX}$ $T_A = T_{MIN}$			20			70	nA nA
Average Temperature Coefficient of Input Offset Current	$25°C \leq T_A \leq T_{MAX}$ $T_{MIN} \leq T_A \leq 25°C$		0.01 0.02	0.1 0.2		0.01 0.02	0.3 0.6	nA/°C nA/°C
Input Bias Current				0.1			0.3	μA
Supply Current	$T_A = T_{MAX}$, $V_S = \pm 20V$		1.2	2.5				mA
Large Signal Voltage Gain	$V_S = \pm 15V$, $V_{OUT} = \pm 10V$, $R_L \geq 2k$	25			15			V/mV
Output Voltage Swing	$V_S = \pm 15V$ $R_L = 10\ k\Omega$ $R_L = 2\ k\Omega$	±12 ±10	±14 ±13		±12 ±10	±14 ±13		V V
Input Voltage Range	$V_S = \pm 20V$ $V_S = \pm 15V$	±15	+15, −13			+15, −13		V V
Common-Mode Rejection Ratio	$R_S \leq 50\ k\Omega$ $R_S \leq 10\ k\Omega$	80	96		70	90		dB dB
Supply Voltage Rejection Ratio	$R_S \leq 50\ k\Omega$ $R_S \leq 10\ k\Omega$	80	96		70	96		dB dB

Note 1: The maximum junction temperature of the LM101A is 150°C, and that of the LM201A/LM301A is 100°C. For operating at elevated temperatures, devices in the TO-5 package must be derated based on a thermal resistance of 150°C/W, junction to ambient, or 45°C/W, junction to case. The thermal resistance of the dual-in-line package is 187°C/W, junction to ambient.

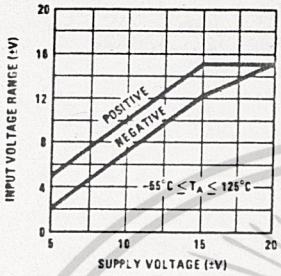
Note 2: For supply voltages less than ±15V, the absolute maximum input voltage is equal to the supply voltage.

Note 3: Continuous short circuit is allowed for case temperatures to 125°C and ambient temperatures to 75°C for LM101A/LM201A, and 70°C and 55°C respectively for LM301A.

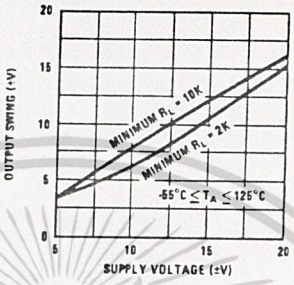
Note 4: Unless otherwise specified, these specifications apply for C1 = 30 pF, ±5V ≤ V_S ≤ ±20V and −55°C ≤ T_A ≤ +125°C (LM101A), ±5V ≤ V_S ≤ ±20V and −25°C ≤ T_A ≤ +85°C (LM201A), ±5V ≤ V_S ≤ ±15V and 0°C ≤ T_A ≤ +70°C (LM301A).

Guaranteed Performance Characteristics LM101A/LM201A

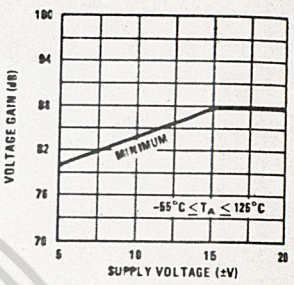
Input Voltage Range



Output Swing

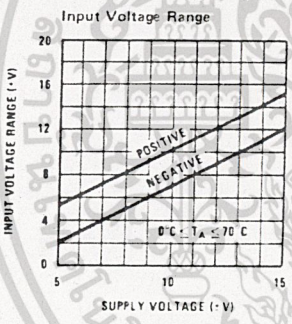


Voltage Gain

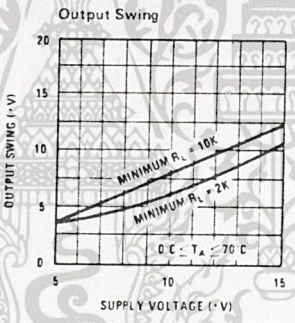


Guaranteed Performance Characteristics LM301A

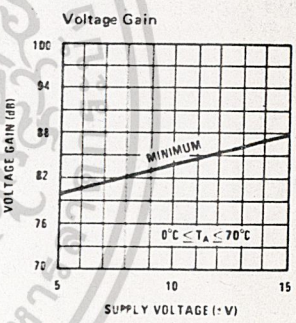
Input Voltage Range



Output Swing

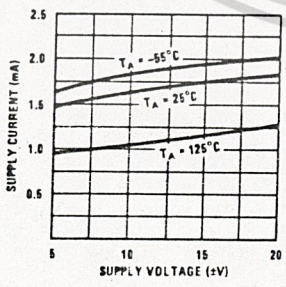


Voltage Gain

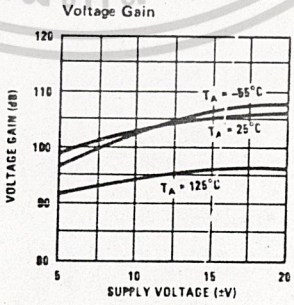


Typical Performance Characteristics

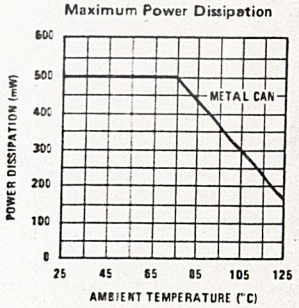
Supply Current



Voltage Gain

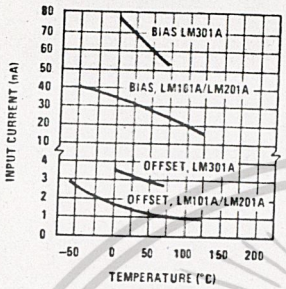


Maximum Power Dissipation

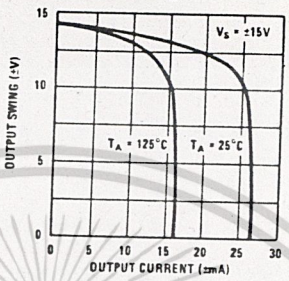


Typical Performance Characteristics (Continued)

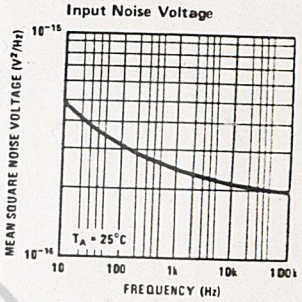
Input Current, LM101A/ LM201A/LM301A



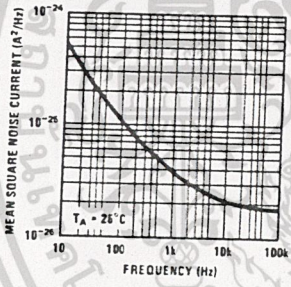
Current Limiting



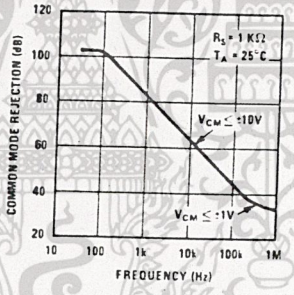
Input Noise Voltage



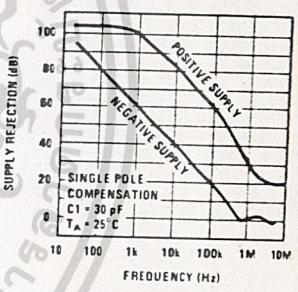
Input Noise Current



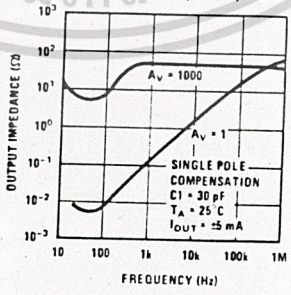
Common Mode Rejection



Power Supply Rejection

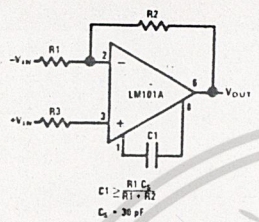


Closed Loop Output Impedance

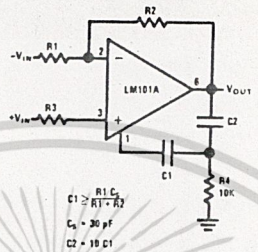


Typical Performance Characteristics for Various Compensation Circuits**

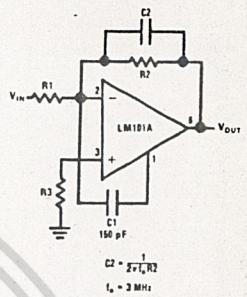
Single Pole Compensation



Two Pole Compensation

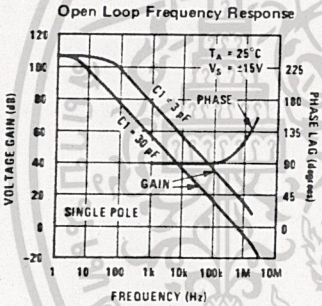


Feedforward Compensation

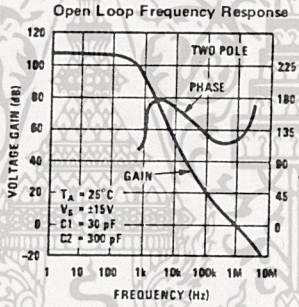


**Pin connections shown are for metal can.

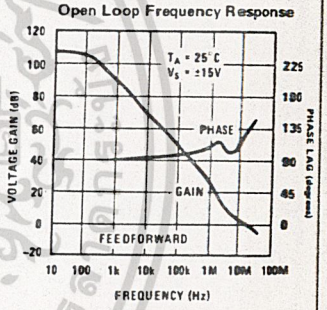
Open Loop Frequency Response



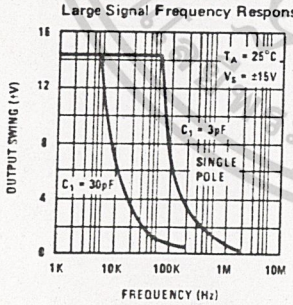
Open Loop Frequency Response



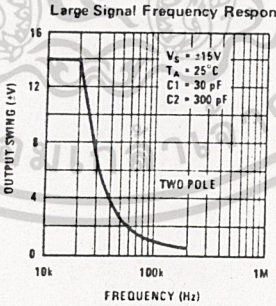
Open Loop Frequency Response



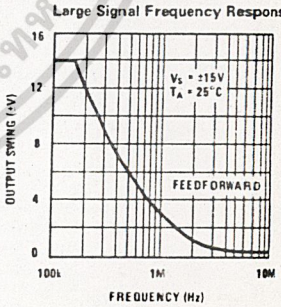
Large Signal Frequency Response



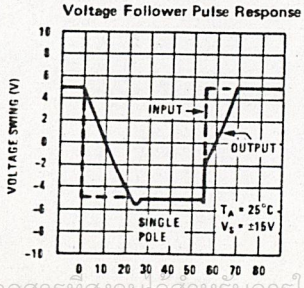
Large Signal Frequency Response



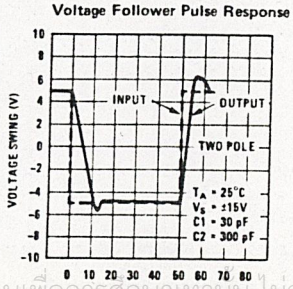
Large Signal Frequency Response



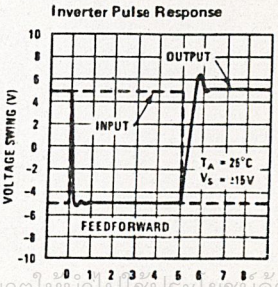
Voltage Follower Pulse Response



Voltage Follower Pulse Response

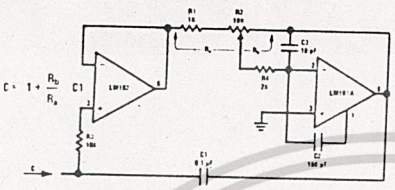


Inverter Pulse Response

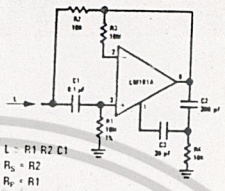


Typical Applications **

Variable Capacitance Multiplier



Simulated Inductor

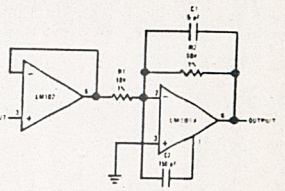


$L = R1\,R2\,C1$

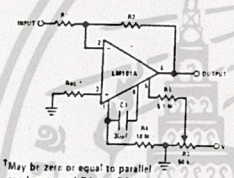
$R_S = R2$

$R_P = R1$

Fast Inverting Amplifier With High Input Impedance

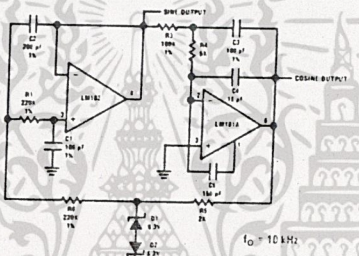


Inverting Amplifier with Balancing Circuit



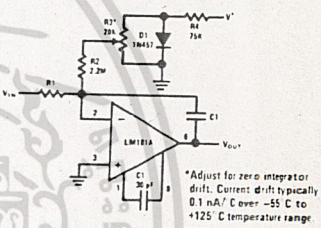
†May be zero or equal to parallel combination of R1 and R2 for minimum offset

Sine Wave Oscillator



$f_O = 10$ kHz

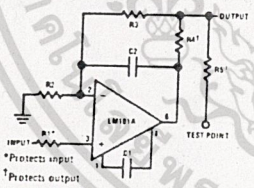
Integrator with Bias Current Compensation



*Adjust for zero integrator drift. Current drift typically 0.1 nA/°C over −55°C to +125°C temperature range

Application Hints **

Protecting Against Gross Fault Conditions



*Protects input

†Protects output

‡Protects output—not needed when R6 is used

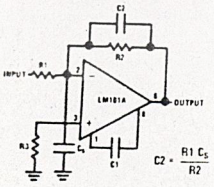
Although the LM101A is designed for trouble free operation, experience has indicated that it is wise to observe certain precautions given below to protect the devices from abnormal operating conditions. It might be pointed out that the advice given here is applicable to practically any IC op amp, although the exact reason why may differ with different devices.

When driving either input from a low-impedance source, a limiting resistor should be placed in series with the input lead to limit the peak instantaneous output current of the source to something less than 100 mA. This is especially important when the inputs go outside a piece of equipment where they could accidentally be connected to high voltage sources. Large capacitors on the input (greater than 0.1 μF) should be treated as a low source impedance and isolated with a resistor. Low impedance sources do not cause a problem unless their output voltage exceeds the supply voltage. However, the supplies go to zero when they are turned off, so the isolation is usually needed.

The output circuitry is protected against damage from shorts to ground. However, when the amplifier output is connected to a test point, it should be isolated by a limiting resistor, as test points frequently get shorted to bad places. Further, when the amplifier drives a load external to the equipment, it is also advisable to use some sort of limiting resistance to preclude mishaps.

Precautions should be taken to insure that the power supplies for the integrated circuit never become reversed—even under transient conditions. With reverse voltages greater than 1V, the IC will conduct excessive current, fuzing internal aluminum interconnects. If there is a possibility of this happening, clamp diodes with a high peak current rating should be installed on the supply lines. Reversal of the voltage between V^+ and V^- will always cause a problem, although reversals with respect to ground may also give difficulties in many circuits.

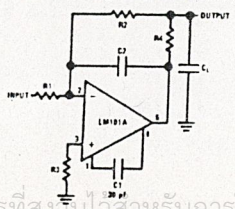
Compensating For Stray Input Capacitances Or Large Feedback Resistor



$C2 = \frac{R1\,C_S}{R2}$

The minimum values given for the frequency compensation capacitor are stable only for source resistances less than 10 kΩ, stray capacitances on the summing junction less than 5 pF and capacitive loads smaller than 100 pF. If any of these conditions are not met, it becomes necessary to overcompensate the amplifier with a larger compensation capacitor. Alternately, lead capacitors can be used in the feedback network to negate the effect of stray capacitance and large feedback resistors or an RC network can be added to isolate capacitive loads.

Isolating Large Capacitive Loads

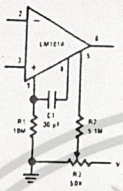


Although the LM101A is relatively unaffected by supply bypassing, this cannot be ignored altogether. Generally it is necessary to bypass the supplies to ground at least once on every circuit card, and more bypass points may be required if more than five amplifiers are used. When feed-forward compensation is employed, however, it is advisable to bypass the supply leads of each amplifier with low inductance capacitors because of the higher frequencies involved.

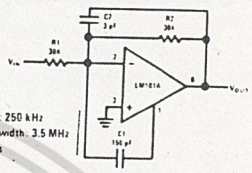
**Pin connections shown are for metal can.

Typical Applications** (Continued)

Standard Compensation and Offset Balancing Circuit

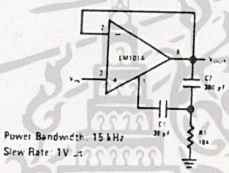


Fast Summing Amplifier



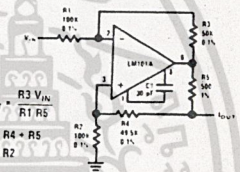
Power Bandwidth: 250 kHz
Small Signal Bandwidth: 3.5 MHz
Slew Rate: 10V/μs

Fast Voltage Follower



Power Bandwidth: 15 kHz
Slew Rate: 1V/μs

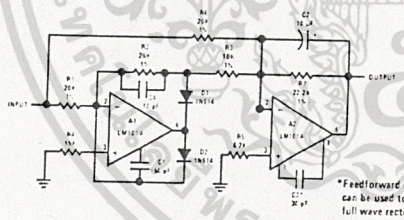
Bilateral Current Source



$$I_{OUT} = \frac{R3\ V_{IN}}{R1\ R5}$$

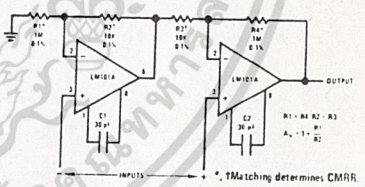
R3 = R4 + R5
R1 = R2

Fast AC/DC Converter*



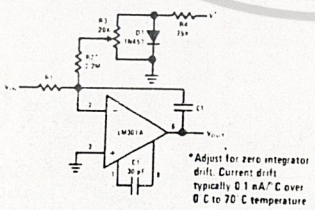
*Feedforward compensation can be used to make a fast full wave rectifier without a filter.

Instrumentation Amplifier



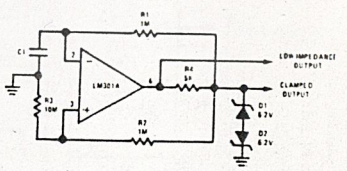
*, †Matching determines CMRR.

Integrator with Bias Current Compensation

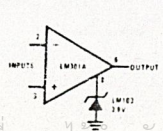


*Adjust for zero integrator drift. Current drift typically 0.1 nA/°C over 0°C to 70°C temperature range.

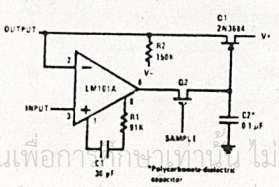
Low Frequency Square Wave Generator



Voltage Comparator for Driving RTL Logic or High Current Driver

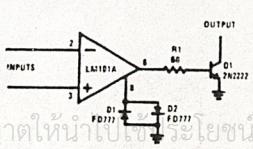


Low Drift Sample and Hold



*Polycarbonate dielectric capacitor

Voltage Comparator for Driving DTL or TTL Integrated Circuits



**Pin connections shown are for metal can.

National Semiconductor

A to D, D to A

ADC0816, ADC0817 8-Bit μP Compatible A/D Converters with 16-Channel Multiplexer

General Description

The ADC0816, ADC0817 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 16-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 16-channel multiplexer can directly access any one of 16-single-ended analog signals, and provides the logic for additional channel expansion. Signal conditioning of any analog input signal is eased by direct access to the multiplexer output, and to the input of the 8-bit A/D converter.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

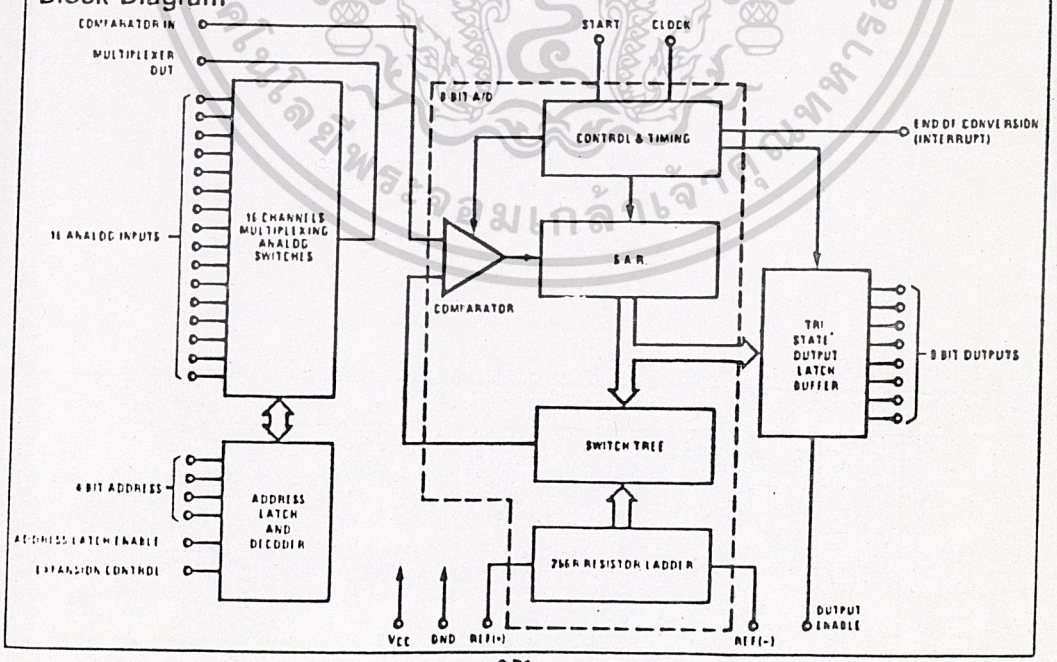
The design of the ADC0816, ADC0817 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0816, ADC0817 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For similar performance in an 8-channel, 28-pin, 8-bit A/D converter, see the ADC0808, ADC0809 data sheet. (See AN-258 for more information.)

Features

- Resolution — 8-bits
- Total unadjusted error — ± 1/2 LSB and ± 1 LSB
- No missing codes
- Conversion time — 100 μs
- Single supply — 5 V_{DC}
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- 16-channel multiplexer with latched control logic
- Easy interface to all microprocessors, or operates "stand alone"
- Outputs meet T^2L voltage level specifications
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Standard hermetic or molded 40-pin DIP package
- Temperature range −40°C to +85°C or −55°C to +125°C
- Low power consumption — 15 mW
- Latched TRI-STATE® output
- Direct access to "comparator in" and "multiplexer out" for signal conditioning

TRI-STATE® is a registered trademark of National Semiconductor Corp.

Block Diagram



Absolute Maximum Ratings (Notes 1 and 2)

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage at Any Pin Except Control Inputs	−0.3V to (V_{CC} + 0.3V)
Voltage at Control Inputs (START, OE, CLOCK, ALE, EXPANSION CONTROL, ADD A, ADD B, ADD C, ADD D)	−0.3V to 15V
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A = 25°C	875 mW
Lead Temperature (Soldering, 10 seconds)	300°C

Operating Ratings (Notes 1 and 2)

Temperature Range (Note 1) ADC0816CJ	$T_{MIN} \le T_A \le T_{MAX}$ −55°C ≤ T_A ≤ +125°C
ADC0816CCJ, ADC0816CCN, ADC0817CCN	−40°C ≤ T_A ≤ +85°C
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.0 V_{DC}
Voltage at Any Pin Except Control Inputs	0V to V_{CC}
Voltage at Control Inputs (START, OE, CLOCK, ALE, EXPANSION CONTROL, ADD A, ADD B, ADD C, ADD D)	0V to 15V

Electrical Characteristics

Converter Specifications: V_{CC} = 5 V_{DC} = $V_{REF(+)}$, $V_{REF(-)}$ = GND, V_{IN} = $V_{COMPARATOR\ IN}$, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK} = 640 kHz unless otherwise stated.

	Parameter	Conditions	Min	Typ	Max	Units
	ADC0816					
	Total Unadjusted Error (Note 5)	25°C			±1/2	LSB
		T_{MIN} to T_{MAX}			±3/4	LSB
	ADC0817					
	Total Unadjusted Error (Note 5)	0°C to 70°C			±1	LSB
		T_{MIN} to T_{MAX}			±1 1/4	LSB
	Input Resistance	From Ref(+) to Ref(−)	1.0	4.5		kΩ
	Analog Input Voltage Range	(Note 4) V(+) or V(−)	GND−0.10		V_{CC}+0.10	V_{DC}
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	V_{CC}+0.1	V
$\frac{V_{REF(+)}+V_{REF(-)}}{2}$	Voltage, Center of Ladder		$V_{CC}/2$−0.1	$V_{CC}/2$	$V_{CC}/2$+0.1	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V
	Comparator Input Current	f_c = 640 kHz, (Note 6)	−2	±0.5	2	μA

Electrical Characteristics

Digital Levels and DC Specifications: ADC0816CJ 4.5V ≤ V_{CC} ≤ 5.5V, −55°C ≤ T_A ≤ +125°C unless otherwise noted. ADC0816CCJ, ADC0816CCN, ADC0817CCN 4.75V ≤ V_{CC} ≤ 5.25V, −40°C ≤ T_A ≤ +85°C unless otherwise noted.

	Parameter	Conditions	Min	Typ	Max	Units
	ANALOG MULTIPLEXER					
R_{ON}	Analog Multiplexer ON Resistance	(Any Selected Channel)				
		T_A = 25°C, R_L = 10k		1.5	3	kΩ
		T_A = 85°C			6	kΩ
		T_A = 125°C			9	kΩ
ΔR_{ON}	Δ ON Resistance Between Any 2 Channels	(Any Selected Channel) R_L = 10k		75		Ω
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 5V,				
		T_A = 25°C		10	200	nA
		T_{MIN} to T_{MAX}			1.0	μA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 0,				
		T_A = 25°C	−200			nA
		T_{MIN} to T_{MAX}	−1.0			μA
	CONTROL INPUTS					
$V_{IN(1)}$	Logical "1" Input Voltage		V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage				1.5	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN} = 15V			1.0	μA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN} = 0	−1.0			μA
I_{CC}	Supply Current	f_{CLK} = 640 kHz		0.3	3.0	mA

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0816CJ — $4.5V \leq V_{CC} \leq 5.5V$, $-55°C \leq T_A \leq +125°C$ unless otherwise noted. ADC0816CCJ, ADC0816CCN, ADC0817CCN — $4.75V \leq V_{CC} \leq 5.25V$, $-40°C \leq +85°C$ unless otherwise noted.

Parameter		Conditions	Min.	Typ.	Max.	Units
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	$I_O = -360\mu A$, $T_A = 85°C$ $I_O = -300\mu A$, $T_A = 125°C$	$V_{CC} - 0.4$			V
$V_{OUT(0)}$	Logical "0" Output Voltage	$I_O = 1.6mA$			0.45	V
$V_{OUT(0)}$	Logical "0" Output Voltage EOC	$I_O = 1.2mA$			0.45	V
I_{OUT}	TRI-STATE® Output Current	$V_O = V_{CC}$ $V_O = 0$	 -3.0		3.0	μA μA

Electrical Characteristics

Timing Specifications: $V_{CC} = V_{REF(+)} = 5V$, $V_{REF(-)} = GND$, $t_r = t_f = 20$ ns and $T_A = 25°C$ unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	*(Figure 5)*		100	200	ns
t_{WALE}	Minimum ALE Pulse Width	*(Figure 5)*		100	200	ns
t_s	Minimum Address Set-Up Time	*(Figure 5)*		25	50	ns
t_H	Minimum Address Hold Time	*(Figure 5)*		25	50	ns
t_D	Analog MUX Delay Time From ALE	$R_S = 0\Omega$ *(Figure 5)*		1	2.5	μs
t_{1H}, t_{H0}	OE Control to Q Logic State	$C_L = 50$ pF, $R_L = 10k$ *(Figure 8)*		125	250	ns
t_{1H}, t_{0H}	OE Control to Hi-Z	$C_L = 10$ pF, $R_L = 10k$ *(Figure 8)*		125	250	ns
t_c	Conversion Time	$f_c = 640$ kHz, *(Figure 5)* (Note 7)	90	100	116	μs
f_c	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	*(Figure 5)*	0		8 + 2 μs	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE® Output Capacitance	At TRI-STATE Outputs. (Note 7)		10	15	pF

Note 1: Absolute maximum ratings are those values beyond which the life of the device may be impaired.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: A zener diode exists, internally, from V_{CC} to GND and has a typical breakdown voltage of 7 V_{DC}.

Note 4: Two on-chip diodes are tied to each analog input which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. The spec allows 100 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 100 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.900 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Total unadjusted error includes offset, full-scale, and linearity errors. See *Figure 3*. None of these A/Ds requires a zero or full-scale adjust. However, if an all zero code is desired for an analog input other than 0.0V, or if a narrow full-scale span exists (for example: 0.5V to 4.5V full-scale) the reference voltages can be adjusted to achieve this. See *Figure 13*.

Note 6: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence *(Figure 6)*. See paragraph 4.0.

Note 7: The outputs of the data register are updated one clock cycle before the rising edge of EOC.

Functional Description

Multiplexer: The device contains a 16-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table I shows the input states for the address line and the expansion control line to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE I

SELECTED ANALOG CHANNEL	ADDRESS LINE				EXPANSION CONTROL
	D	C	B	A	
IN0	L	L	L	L	H
IN1	L	L	L	H	H
IN2	L	L	H	L	H
IN3	L	L	H	H	H
IN4	L	H	L	L	H
IN5	L	H	L	H	H
IN6	L	H	H	L	H
IN7	L	H	H	H	H
IN8	H	L	L	L	H
IN9	H	L	L	H	H
IN10	H	L	H	L	H
IN11	H	L	H	H	H
IN12	H	H	L	L	H
IN13	H	H	L	H	H
IN14	H	H	H	L	H
IN15	H	H	H	H	H
All Channels OFF	X	X	X	X	L

X = don't care

Additional single-ended analog signals can be multiplexed to the A/D converter by disabling all the multiplexer inputs using the expansion control. The additional external signals are connected to the comparator input and the device ground. Additional signal conditioning (i.e., prescaling, sample and hold, instrumentation amplification, etc.) may also be added between the analog input signal and the comparator input.

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach (*Figure 1*) was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached + 1/2 LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

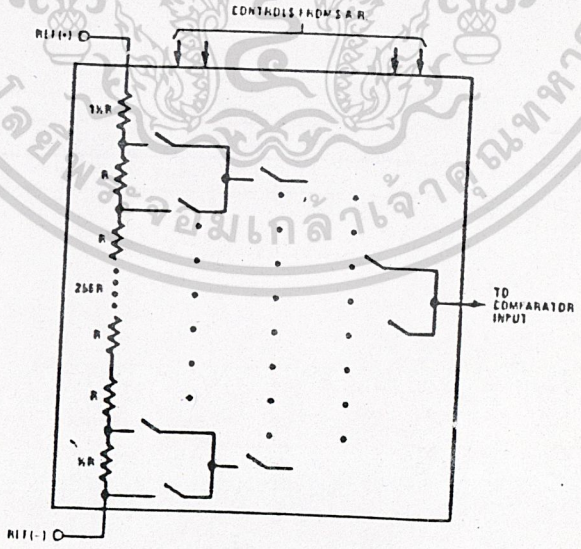


FIGURE 1. Resistor Ladder and Switch Tree

Functional Description (Continued)

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0816, ADC0817, the approximation technique is extended to 8 bits using the 256R network.

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0816 as measured using the procedures outlined in AN-179.

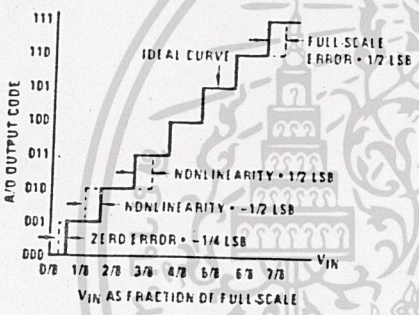


FIGURE 2. 3-Bit A/D Transfer Curve

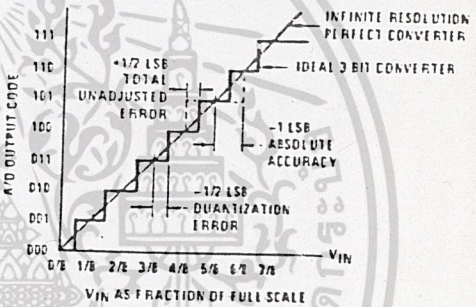


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

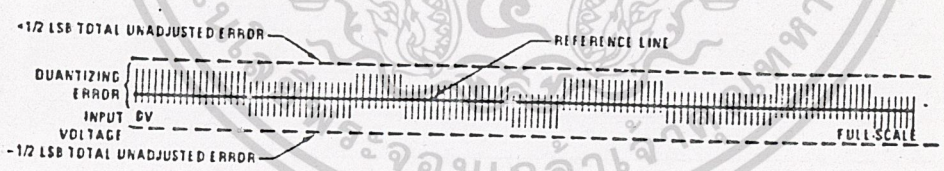
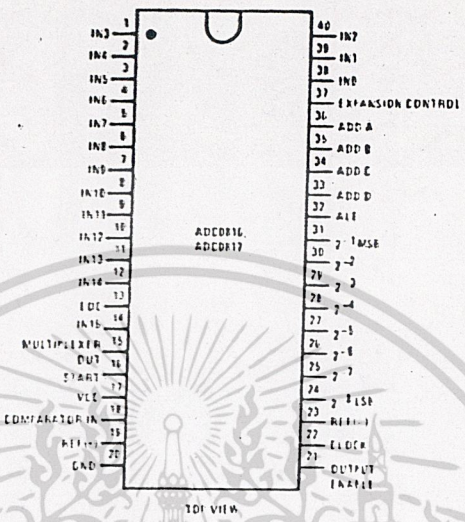


FIGURE 4. Typical Error Curve

Connection Diagram

Dual-In-Line Package



Timing Diagram

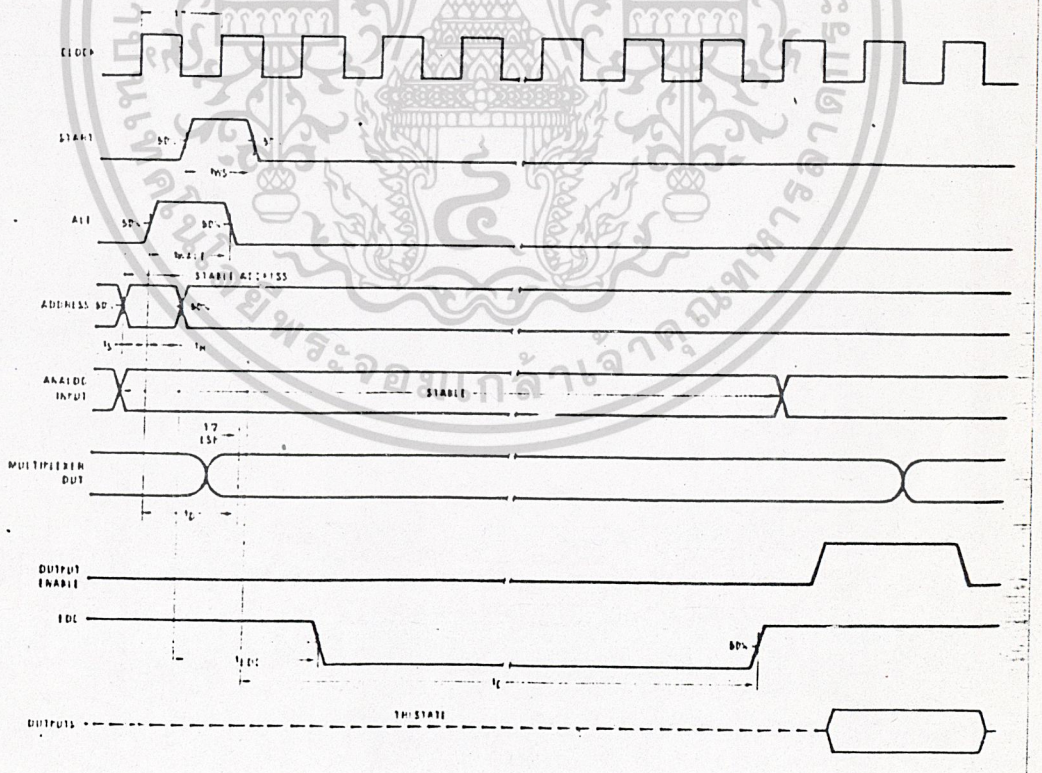


FIGURE 5

Typical Performance Characteristics

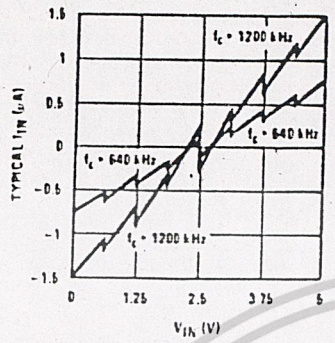


FIGURE 6. Comparator I_{IN} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)

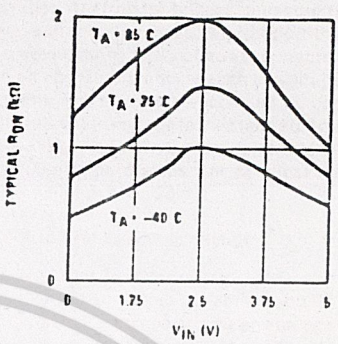
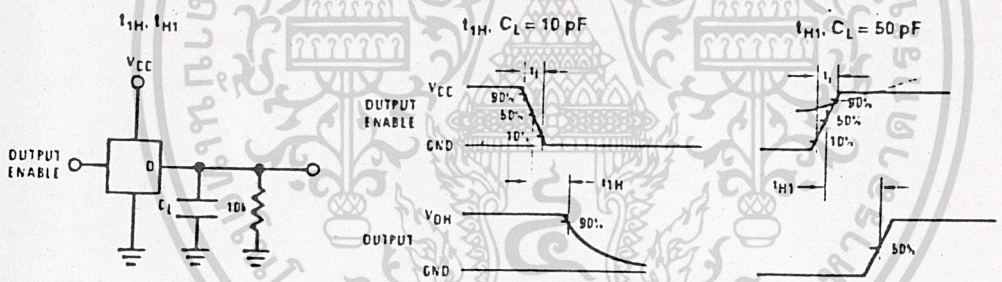


FIGURE 7. Multiplexer R_{ON} vs V_{IN} ($V_{CC} = V_{REF} = 5V$)

TRI-STATE® Test Circuits and Timing Diagrams



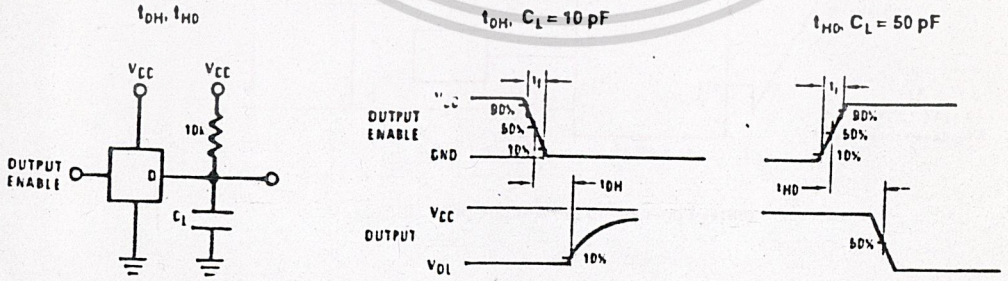


FIGURE 8

Applications Information

OPERATION

1.0 Ratiometric Conversion

The ADC0816, ADC0817 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0816 is expressed by the equation

$$\frac{V_{IN}}{V_{fs} - V_Z} = \frac{D_X}{D_{MAX} - D_{MIN}} \quad (1)$$

V_{IN} = Input voltage into the ADC0816
V_{fs} = Full-scale voltage
V_Z = Zero voltage
D_X = Data point being measured
D_{MAX} = Maximum data limit
D_{MIN} = Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0816, ADC0817 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs, *(Figure 9)*.

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc., are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a system reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC} = V_{REF} = 5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

2.0 Resistor Ladder Limitations

The voltages from the resistor ladder are compared to the selected input 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder, Ref(−), should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 10 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V reference is used, the supply should be adjusted to the same voltage within 0.1V.

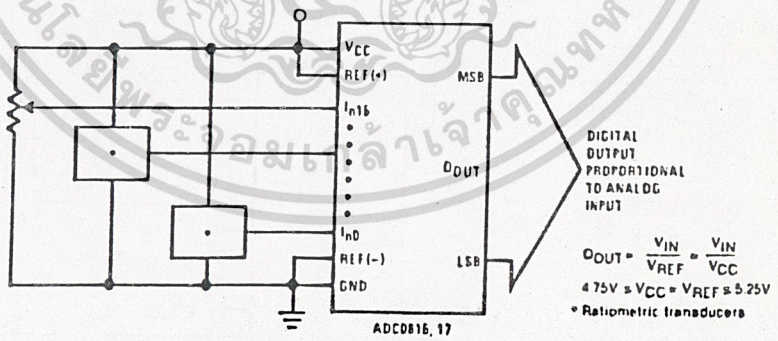


FIGURE 9. Ratiometric Conversion System

Applications Information (Continued)

The ADC0816 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 11* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 12*. The LM301 is overcompensated to insure stability when loaded by the 10 μF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 13*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB to be half the size of the LSB in a 5V reference system.

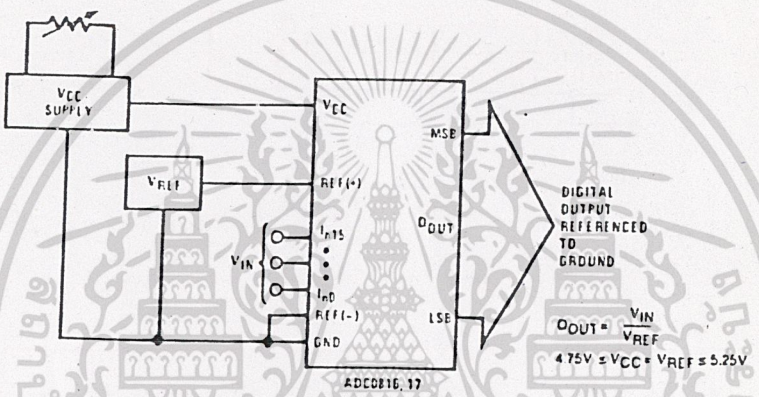


FIGURE 10. Ground Referenced Conversion System Using Trimmed Supply

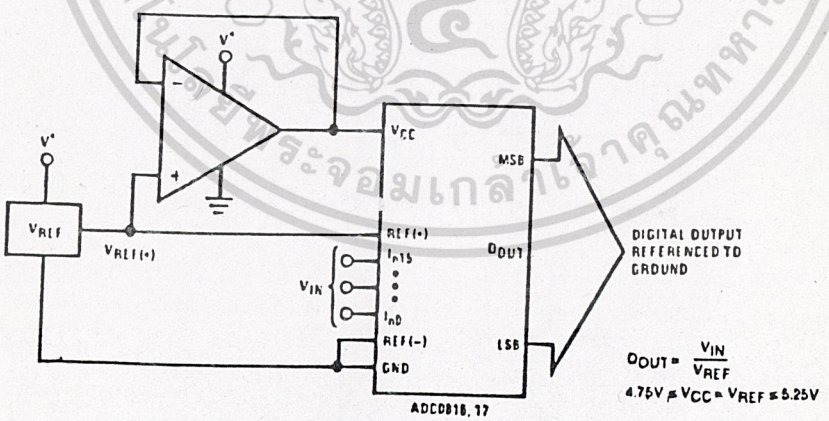


FIGURE 11. Ground Referenced Conversion System with Reference Generating V_{CC} Supply

Applications Information (Continued)

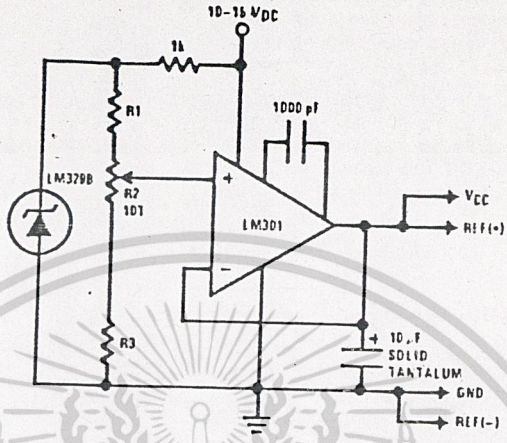


FIGURE 12. Typical Reference and Supply Circuit

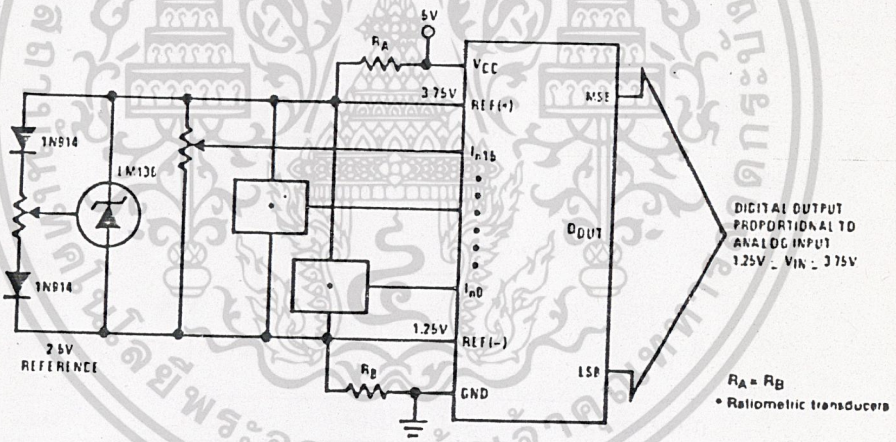


FIGURE 13. Symmetrically Centered Reference

3.0 Converter Equations

The transition between adjacent codes N and N + 1 is given by:

$$V_{IN} = \left\{(V_{REF(+)} - V_{REF(-)})\left[\frac{N}{256} + \frac{1}{512}\right] \pm V_{TUE}\right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} = \left\{(V_{REF(+)} - V_{REF(-)})\left[\frac{N}{256}\right] \pm V_{TUE}\right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

where: V_{IN} = Voltage at comparator input
$V_{REF(+)}$ = Voltage at Ref(+)
$V_{REF(-)}$ = Voltage at Ref(−)
V_{TUE} = Total unadjusted error voltage (typically $V_{REF(+)} - 512$)

Applications Information (Continued)

4.0 Analog Comparator Inputs

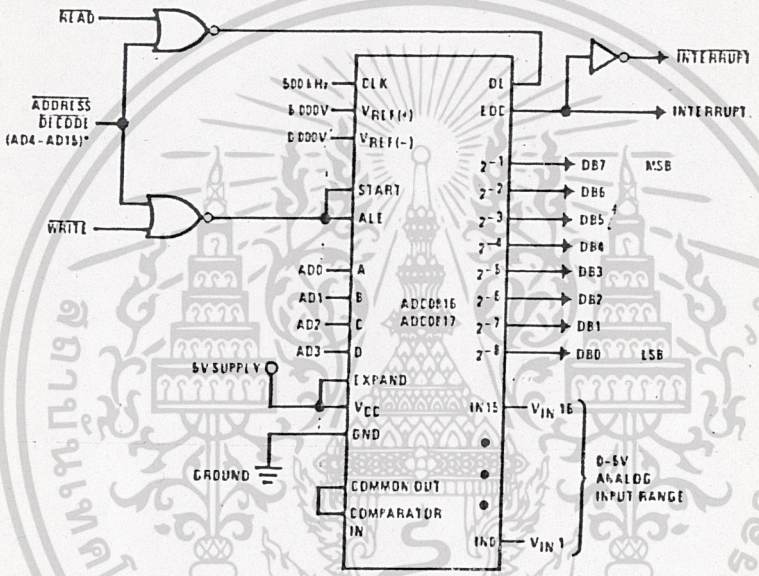
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6*.

If no filter capacitors are used at the analog or comparator inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally. See AN-258 for further discussion.

Typical Application



* Address latches needed for 8085 and SC/MP interfacing the ADC0816, 17 to a microprocessor

Microprocessor Interface Table

PROCESSOR	$\overline{\text{READ}}$	$\overline{\text{WRITE}}$	INTERRUPT (COMMENT)
8080	$\overline{\text{MEMR}}$	$\overline{\text{MEMW}}$	INTR (Thru RST Circuit)
8085	$\overline{\text{RD}}$	$\overline{\text{WR}}$	INTR (Thru RST Circuit)
Z-80	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{INT}}$ (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA•Φ2•R/W	VMA•Φ2•$\overline{\text{R/W}}$	$\overline{\text{IRQA}}$ or $\overline{\text{IRQB}}$ (Thru PIA)

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C		−55°C to +125°C
Error	± 1/2 Bit Unadjusted	ADC0816CCN	ADC0816CCJ	ADC0816CJ
	± 1 Bit Unadjusted	ADC0817CCN		
Package Outline		N40A Molded DIP	J40A Hermetic DIP	J40A Hermetic DIP

Applications Information (Continued)

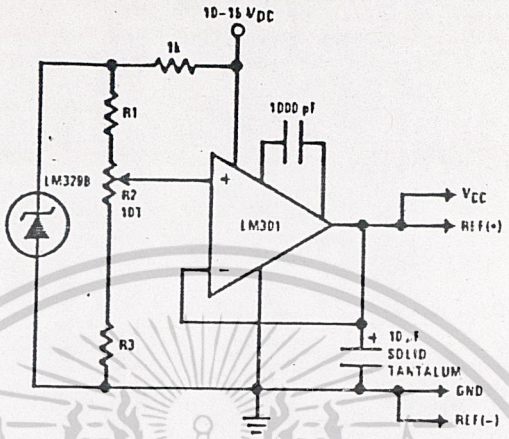


FIGURE 12. Typical Reference and Supply Circuit

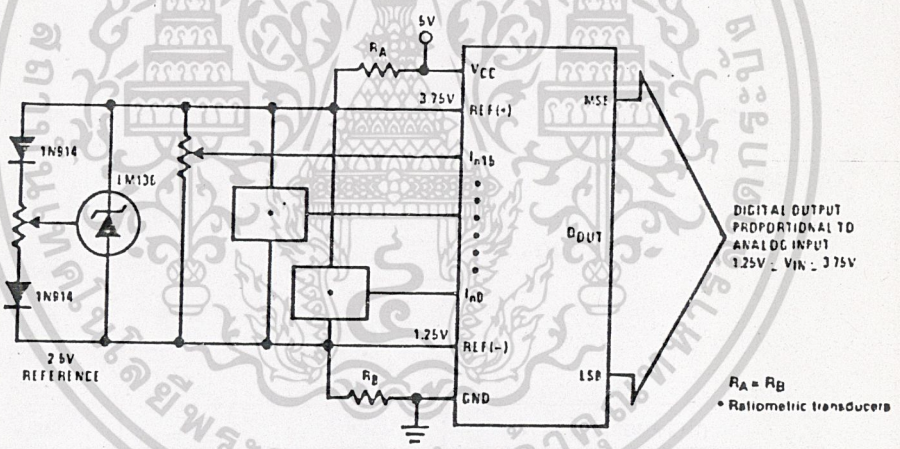


FIGURE 13. Symmetrically Centered Reference

3.0 Converter Equations

The transition between adjacent codes N and N + 1 is given by:

$$V_{IN} = \left\{(V_{REF(+)} - V_{REF(-)})\left[\frac{N}{256} + \frac{1}{512}\right] \pm V_{TUE}\right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} = \left\{(V_{REF(+)} - V_{REF(-)})\left[\frac{N}{256}\right] \pm V_{TUE}\right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

where: V_{IN} = Voltage at comparator input
$V_{REF(+)}$ = Voltage at Ref(+)
$V_{REF(-)}$ = Voltage at Ref(−)
V_{TUE} = Total unadjusted error voltage (typically $V_{REF(+)}$ - 512)

Applications Information (Continued)

4.0 Analog Comparator Inputs

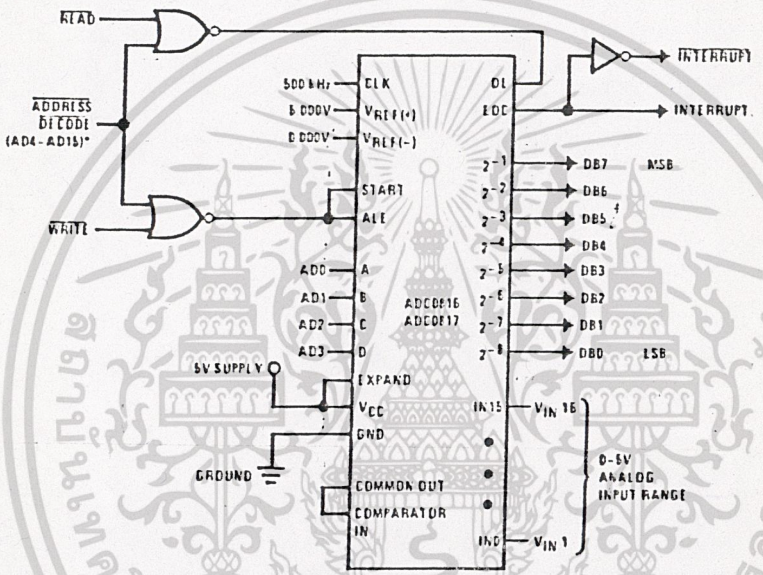
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6*.

If no filter capacitors are used at the analog or comparator inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally. See AN-258 for further discussion.

Typical Application



* Address latches needed for 8085 and SC/MP interfacing the ADC0816, 17 to a microprocessor

Microprocessor Interface Table

PROCESSOR	$\overline{\text{READ}}$	$\overline{\text{WRITE}}$	INTERRUPT (COMMENT)
8080	$\overline{\text{MEMR}}$	$\overline{\text{MEMW}}$	INTR (Thru RST Circuit)
8085	$\overline{\text{RD}}$	$\overline{\text{WR}}$	INTR (Thru RST Circuit)
Z-80	$\overline{\text{RD}}$	$\overline{\text{WR}}$	$\overline{\text{INT}}$ (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA • ϕ2 • R/$\overline{\text{W}}$	VMA • ϕ2 • R/$\overline{\text{W}}$	$\overline{\text{IRQA}}$ or $\overline{\text{IRQB}}$ (Thru PIA)

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C		−55°C to +125°C
Error	± 1/2 Bit Unadjusted	ADC0816CCN	ADC0816CCJ	ADC0816CJ
	± 1 Bit Unadjusted	ADC0817CCN		
Package Outline		N40A Molded DIP	J40A Hermetic DIP	J40A Hermetic DIP

กิตติกรรมประกาศ

คณะผู้เขียนปริญญานิพนธ์นี้ ต้องขอขอบคุณ อาจารย์ วิริยะ กรองรัตน์ ซึ่งท่านได้กรุณาให้เกียรติรับเป็นอาจารย์ที่ปรึกษา โปรเจ็คนี้ และท่านยังได้ช่วยชี้แนะจุดประสงค์ และคอยให้คำปรึกษา ตั้งแต่เริ่มต้นจนจนจบ ด้วยดีเสมอมา

และขอขอบคุณคณาจารย์ภาค เทคโนโลยีการวัดคุมทางอุตสาหกรรมทุกๆท่านที่กรุณาช่วยเหลือในการทำ ปริญญานิพนธ์ ในครั้งนี้เป็นอย่างดียิ่ง

บรรณานุกรม

1. BRIAN W. DEMIGHAN AND DENNIS M. RITCHIE, THE C PROGRAMMING LANGUAGE, PRENTICE-HALL, INC., 1983

2. TURBO C USER'S GUIDE, VERSION 1.5, BORLAND INTERNATIONAL, INC., 1987

3. บุญเลิศ เอี่ยมทัศนา, ยืน ภู่วรวรรณ, สมนึก คีรีโต, โปรแกรมคอมพิวเตอร์ภาษาซี บริษัท ซีเอ็ดยูเคชั่น จำกัดม 2521

4. ผศ. ดร. ดวงแก้ว สวามิภักดิ์ การโปรแกรม c, บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2521

5. ทินกร ดุ๊ก ,ธานินทร์ ถาวรศาสนวงศ์ , การอินเทอร์เฟส IBM/PC ฟิสิกส์เซ็นเตอร์การพิมพ์

6. JOHN UFFENBECK, THE 8086/8088 FAMILY DESIGN PROGRAMMING AND INTERFACING, PRENTICE-HILL, INC., 1787

7. IBM, IBM TECHNICAL REFERENCE, 1983