ปีการศึกษา 2533

ระบบแปลงสัญญาณภาพเป็นดิจิตอล

โดย

วิเชียร เต็มอนุภาพกุล

อาจารย์ที่ปรึกษา

ดร. วัลลภ สุระกำพลธร
รศ. ดร. สิทธิชัย โภไคยอุดม

ปริญญานิพนธ์ปีการศึกษา 2533

เรื่อง ระบบการแปลงสัญญาณภาพเป็นดิจิตอล

จัดทำโดย

วิเชียร เต็มอนุภาพกุล

_____อาจารย์ที่ปรึกษา

(ดร. วัลลภ สุระกำพลธร)

_____อาจารย์ที่ปรึกษา

(รศ. ดร. สิทธิชัย โภไคยอุดม)

ระบบแปลงสัญญาณภาพเป็นดิจิตอล

STAND ALONE DIGITIZER

วิเชียร เต็มอนุภาพกุล 291202

อาจารย์ที่ปรึกษา
ดร.วัลลภ สุระกำพลธร
รศ.ดร.สิทธิชัย โภไคยอุดม
ปีการศึกษา 2532

บทคัดย่อ

รายงานนี้เสนอ การออกแบบระบบการประมวลผลสัญญาณภาพแบบเวลาจริง (REAL TIME IMAGE PROCCESSING SYSTEM) โดยจะทำการแปลงสัญญาณภาพที่ได้มาจากกล้องโทรทัศน์หรือวิดีโอให้เป็นสัญญาณดิจิตอลเก็บไว้ภายใน STATIC RAM มีคุณสมบัติที่สำคัญ คือ

1) สามารถเก็บสัญญาณภาพแบบเวลาจริง (REAL TIME IMAGE ACQUISITION)
2) สามารถเก็บและแสดงภาพได้ด้วยตัวของมันเอง โดยใช้ SINGLE CHIP ตระกูล 8031 ในการควบคุม
3) สามารถจะส่งผ่านข้อมูลภาพไปมาระหว่างคอมพิวเตอร์ได้
4) สามารถที่ต่อผ่านโมเดม เพื่อส่งข้อมูลตามสายโทรศัพท์ ในกรณีที่ต้องการส่งสัญญาณภาพไปสถานที่ไกลๆ
5) เนื่องจากที่ใช้ SINGLE CHIP ตระกูล 8031 ในการควบคุม ทำให้เราสามารถที่จะประยุกต์ แก้ไขวงจร และ โปรแกรมอีกเล็กน้อยก็สามารถที่จะนำไปใช้ทางด้านอื่นได้อีก ตัวอย่างเช่น ทำการเก็บภาพความละเอียดสูง โดยเก็บสัญญาณภาพทั้งสัญญาณสแกนคู่และคี่ ทำการหมุน ย่อ ขยายภาพ ฯลฯ

สารบัญ

		หน้าที่
บทที่ 1	บทนำ	1
บทที่ 2	ระบบสัญญาณโทรทัศน์ (COMPOSITE SIGNAL SYSTEM)	3
บทที่ 3	การคำนวณและการสร้าง	10
บทที่ 4	การทดลองและผลการทดลอง	42
บทที่ 5	สรุปผลและวิจารณ์	45
ภาคผนวก	คู่มือการใช้ซิงเกิลบอร์ด 8031	46
	- บทนำ	48
	- การทำงานภายใน MCS -51	49
	- โครงสร้างหน่วยความจำ, แอดเดรสซิงโหมดและตรรกศาสตร์	56
	- ชุดคำสั่ง	62
	- ซิงเกิลบอร์ด MCS -51	72

บทที่ 1

บทนำ

เรื่องราวเกี่ยวกับวิธีการประมวลผลภาพได้รับความสนใจเป็นอย่างมาก ทั้งนี้เนื่องมาจากจินตนาการของมนุษย์ที่สำคัญ 2 ประการ

1.1 การปรับปรุงภาพให้ดีขึ้นเพื่อการตีความของมนุษย์เองเช่น ภาพที่ได้จากดาวเทียม LANDSAT (เป็นดาวเทียมเพื่อการสำรวจทรัพยากร) ภาพที่ส่งมาจะมีลักษณะไม่ชัดเจนยากต่อการตีความว่าพื้นที่ส่วนใด คือ พื้นที่ทางการเกษตร พื้นที่ส่วนใด คือ ป่าไม้ และ ฯลฯ ดังนั้นมนุษย์จึงได้พยายามหาวิธีการประมวลผลภาพ ซึ่งเป็นวิธีการทางซอฟแวร์ (SOFTWARE) โดยผ่านขบวนการทางคณิตศาสตร์ เพื่อให้ได้ภาพที่สามารถตีความได้ดียิ่งขึ้น

1.2 ความพยายามที่จะให้เครื่องจักรสามารถเข้าใจถึงภาพที่เห็นได้ ดัง, เช่น ความพยายามที่จะพัฒนาหุ่นยนต์ ให้มีความสามารถแยกแยะวัตถุต่างๆออกจากกันได้ ซึ่งเป็นเรื่องของศาสตร์ทางด้าน PATTERN RECOGNITION

ในปี ค.ศ. 1920 ได้มีการทดลองส่งภาพหนังสือพิมพ์ที่ผ่านการดิจิไตซ์ (DIGITIZED) ระหว่างกรุงลอนดอนกับกรุงนิวยอร์คผ่านใต้มหาสมุทรเป็นครั้งแรก ในการทดลองครั้งนั้นช่วยลดระยะเวลาในการส่งข้อมูล ซึ่งจากเดิมต้องใช้เวลาเป็นอาทิตย์ลงเหลือเพียง 3 ชั่วโมง โดยการนำข้อมูลภาพที่เป็นข้อมูลดิจิตอลมาเข้ารหัสแล้วส่งไปทางด้านรับ เมื่อรับสัญญาณมาก็ถอดรหัสออกแล้วพิมพ์ภาพในแบบ ฮาร์ฟ โทน (HALF TONE) ออกมา

ปัญหาในระยะแรกของระบบการประมวลผลภาพ คือ ระดับของความสว่าง (GRAY LEVEL) ซึ่งมีเพียง 5 ระดับ ทำให้ภาพที่ได้มีลักษณะหยาบแต่อย่างไรก็ตามปัญหานี้ก็ได้รับการแก้ไขเรื่อยๆ จนในปี ค.ศ. 1929 ก็สามารถพัฒนาได้เป็น 15 ระดับ ดังรูป 1.1

อย่างไรก็ตามระบบการประมวลภาพก็ยังใช้อยู่ในวงจำกัด ทั้งนี้เพราะการประมวลผลภาพใช้เนื้อที่หน่วยความจำสูงมาก ซึ่งคอมพิวเตอร์ในสมัยนั้นยังมีขีดความสามารถจำกัดอยู่ จะมีการใช้อยู่ก็ในเครื่องระดับเมนเฟม จนมาถึงปัจจุบันเทคโนโลยีทาง

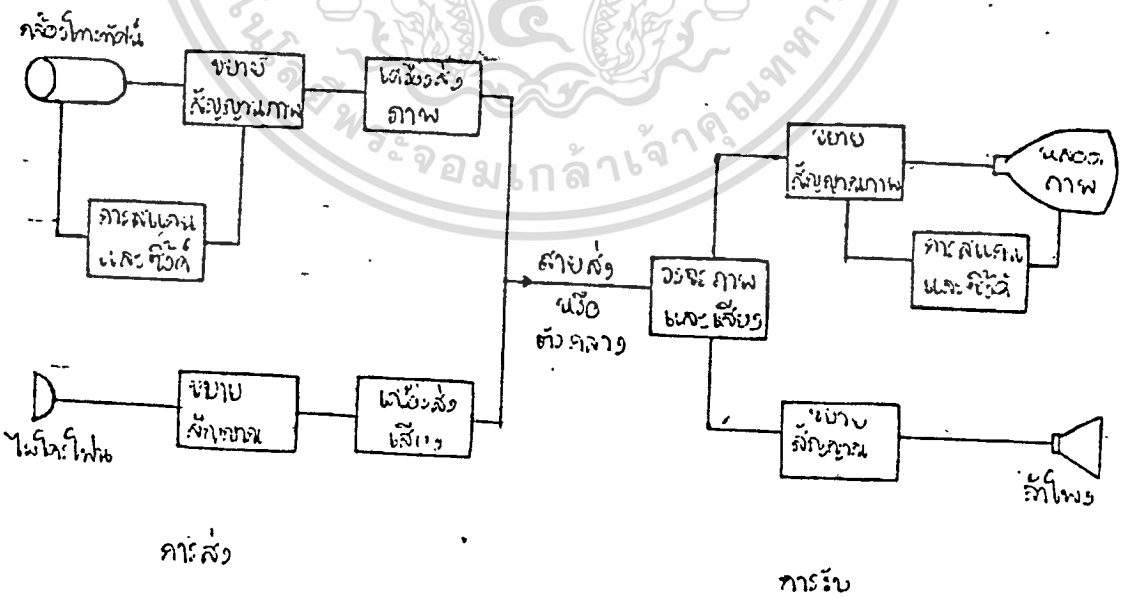
ด้าน VLSI ได้รับการพัฒนาไปไกลมาก คอมพิวเตอร์ส่วนบุคคลยุคใหม่มีขีดความสามารถสูงจึงได้มีการพัฒนาระบบประมวลผลภาพให้เข้ากับคอมพิวเตอร์ส่วนบุคคล หรือ ไมโครคอมพิวเตอร์กันอย่างจริงจัง ในต่างประเทศได้มีการนำระบบการประมวลผลภาพไปประยุกต์ใช้กับงานด้านออกแบบทรงผมให้เข้ากับใบหน้าลูกค้า โดยการถ่ายภาพใบหน้าของลูกค้าเก็บเป็นข้อมูล แล้วนำทรงผมแบบต่างๆ ซึ่งเก็บอยู่ในรูปข้อมูลของคอมพิวเตอร์มาทดลองผสมเข้ากับใบหน้าลูกค้า จนได้ทรงผมที่ถูกใจของลูกค้ามากที่สุด แล้วจึงค่อยแต่งทรงผม นอกจากนี้ยังมีการนำระบบการประมวลผลภาพไปประยุกต์ใช้กับงานด้านต่างๆอีกมากมาย สุดแล้วแต่จินตนาการของมนุษย์ชาติ ท่านผู้อ่านก็เป็นอีกท่านหนึ่งที่อาจจะมีความคิดที่จะนำไปประยุกต์ใช้กับงานของท่านบ้างก็เป็นไปได้

ในโครงการที่นำเสนอในปริญญานิพนธ์นี้ ได้พัฒนาระบบการประมวลผลภาพที่มีความละเอียด 512 x 256 จุด โดยสามารถที่จะทำการเก็บและแสดงสัญญาณภาพโดยตัวของมันเองโดยไม่ต้องพึ่งคอมพิวเตอร์อย่างใดเลย อีกทั้งยังสามารถที่จะส่งสัญญาณภาพที่ได้ทำการเก็บไว้ภายในแรม (RAM) ส่งข้อมูลต่างๆออกเป็นการส่งข้อมูลแบบอนุกรมไปยังคอมพิวเตอร์ส่วนตัวได้ แล้วทำการเก็บข้อมูลลงฟล๊อปปี้ดิสค์(FLOPPY DISK)อีกทีหนึ่ง สำหรับภาพที่ได้จากการเก็บจะมีความสว่าง 256 ระดับ

บทที่ 2

ระบบสัญญาณโทรทัศน์ (COMPOSITE SIGNAL SYSTEM)

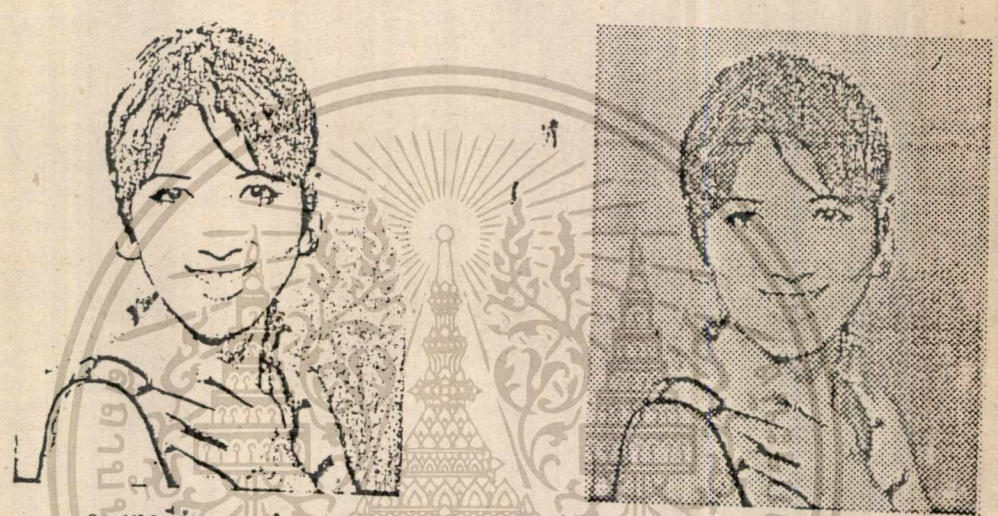
ในระบบโทรทัศน์นั้นจะต้องประกอบไปด้วย กล้องโทรทัศน์ทำหน้าที่ผลิตสัญญาณภาพ (VIDEO SIGNALS) และไมรโครโฟนทำหน้าที่ผลิตสัญญาณเสียง(AUDIO SIGNALS) โดยมีเครื่องแสดงผล(MONITOR) ทำหน้าที่แสดงผลภาพและเสียงที่สร้างโดยกล้องโทรทัศน์และ ไมรโครโฟนอีกครั้งหนึ่งดังรูป 2.1 และเพื่อให้การส่งและรับสัญญาณภาพได้อย่างถูกต้องจำเป็นต้องมีสัญญาณซิ้งค์ (SYNCHRONIZING PULSES) สำหรับการควบคุมการทำงานให้ดำเนินไปอย่างถูกต้อง ทั้งนี้ยังมีสัญญาณอื่นๆที่ช่วยให้เกิดภาพที่สมบูรณ์อีกโดยจะกล่าวละเอียดในส่วนต่อไป ในกรณีที่ต้องการส่งสัญญาณออกไปในระยะทางไกลๆสัญญาณเหล่านี้จะถูกมอดูเลท (MODULATE) ด้วยคลื่นพาหะ (CARRIER) ก่อน แล้วจึงทำการส่งไปได้ ในโครงการนี้เพียงแต่นำสัญญาณโทรทัศน์ที่ได้จากอุปกรณ์กำเหนิดสัญญาณโทรทัศน์ เช่น กล้องโทรทัศน์ , วิดีโอเทป ฯลฯ มาใช้ในขบวนการเก็บภาพเพียงอย่างเดียวโดยไม่มีการมอดูเลท ดังนั้นเนื้อหาที่จะกล่าวต่อไปเน้นเฉพาะส่วนของสัญญาณภาพเท่านั้น



รูป 2.1 ระบบการส่ง-รับโทรทัศน์

2.1 ส่วนประกอบของภาพ (PICTURE ELEMENT)

ภาพขาวดำแต่ละภาพถ้าลองขยายให้ใหญ่ขึ้น จะพบว่าภาพประกอบด้วยจุดดำและจุดขาวมากมาย ดังเช่นภาพจากหนังสือพิมพ์ การจัดเรียงตัวของจุดขาวและจุดดำทำให้เกิดภาพขึ้นได้จุดเหล่านี้เรียกว่า ส่วนประกอบของภาพ สำหรับในพื้นที่ที่เท่ากันแล้วภาพใดที่มีจำนวนระดับของความสว่าง (LEVEL OF BRIGHTNESS) มากกว่าจะเป็นภาพที่คมชัดและกลมกลืนมากกว่า สิ่งนี้เป็นสาเหตุที่ทำให้เราไม่สามารถใช้ จอภาพโมโนโครมของ IBM PC/XT แสดงผลได้เพราะมีจำนวนระดับความสว่างเพียง 2 ระดับ คือ เขียว และ ดำ เท่านั้น



ก) หากมีจำนวนจุดดำมาก ภาพจะมองดูละเอียด

ข) หากมีจำนวนจุดดำน้อย ภาพจะมองดูหยาบ

รูปที่ 2.2 เปรียบเทียบภาพที่มีพื้นที่เท่ากัน

2.2 การสแกน (SCANNING)

จากที่ได้กล่าวมาแล้วข้างต้นว่า ภาพประกอบด้วยจำนวนส่วนประกอบภาพมากมาย ซึ่งแต่ละจุดภาพที่ส่งไปจะบอกว่าเป็นจุดขาวหรือดำก็แสดงโดยสัญญาณภาพ ทางด้านส่งจะส่งทีละจุดเป็นลำดับแยกกันไป ทางด้านรับก็จะนำจุดต่างๆเหล่านี้มาเรียงกันใหม่ให้เป็นภาพขึ้นมา วิธีนี้เรียกว่า การสแกน เส้นประกอบกันเป็นภาพในจอโทรทัศน์นี้เรียกว่า เส้นสแกน ปัจจุบันในโลกเรามีโทรทัศน์ใช้กันอยู่มี 2 ระบบ คือ

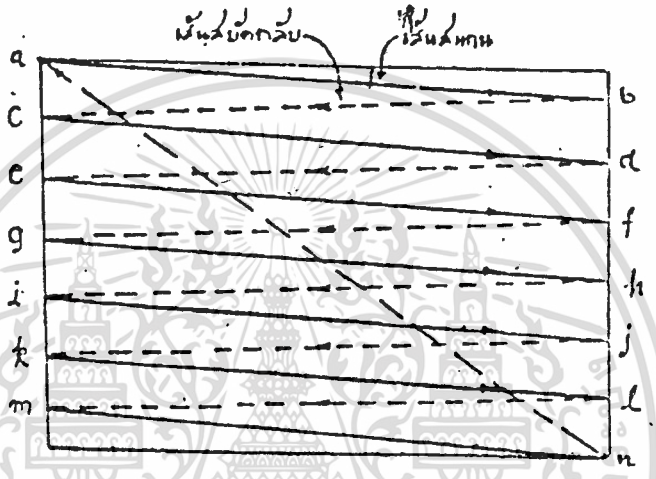
2.2.1 โทรทัศน์ระบบอเมริกา มีจำนวนเส้นการสแกน 525 เส้นต่อภาพ และทำการส่ง 30 ภาพต่อวินาที ดังนั้นความถี่ที่ใช้ในการสแกนเท่ากับ (525)x(30) = 15,750 Hz (PAL)

2.2.2 โทรทัศน์ระบบยุโรป (STSC) มีจำนวนเส้นสแกน 625 เส้นต่อภาพ และทำการส่ง 25 ภาพต่อวินาที ดังนั้นความถี่ที่ใช้ในการสแกนเท่ากับ (625)x(25) =

15,625 Hz

การสแกนมีด้วยกัน 2 วิธี คือ การสแกนแบบก้าวหน้า (PROGRESSIVE SCANNING) และ วิธีการสแกนแบบสลับเส้น (INTERLACED SCANNING)

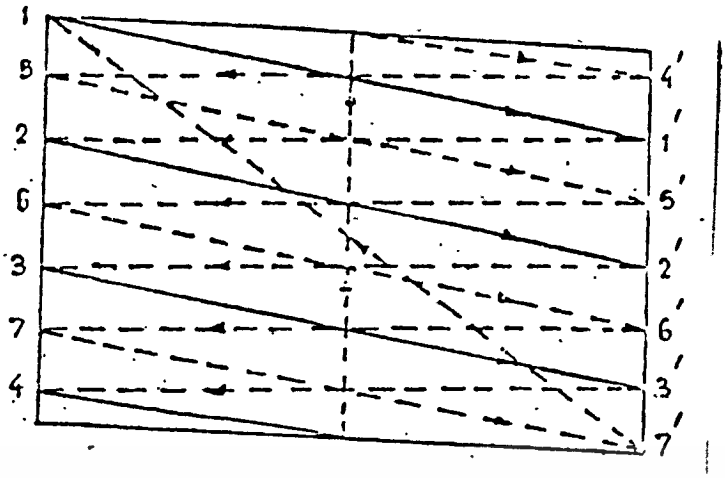
จากรูป 2.3 จะเห็นว่าได้ว่าการสแกนเริ่มจาก a ---> b , c ---> d e ---> f จนถึงสุดท้าย k ---> l ซึ่งเป็นการสแกนตามลำดับจากซ้ายไปขวาและข้างบนลงข้างล่างเหมือนการอ่านหนังสือหรือการพิมพ์ดีดนั่นเอง เป็นการสแกนแบบที่ใช้ในออสซิสโคป เรียกว่า การสแกนแบบก้าวหน้า



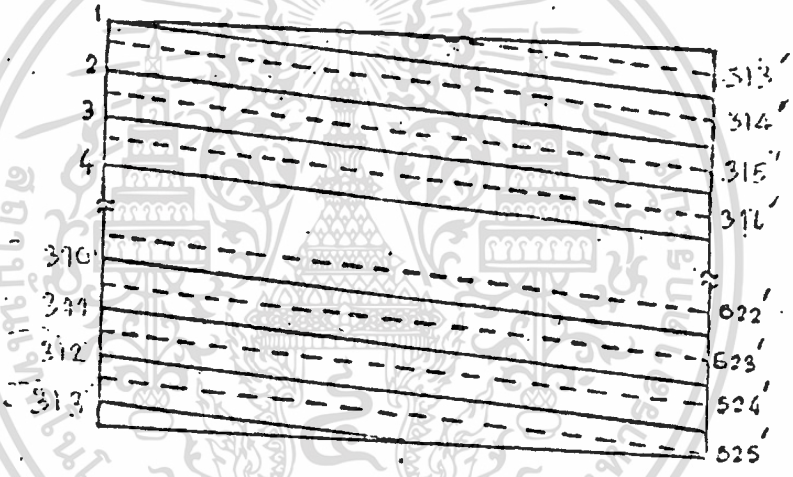
รูปที่ 2.3 การสแกนแบบก้าวหน้า

จากรูป 2.4 (ก) จะเห็นว่าการสแกนเริ่มจาก 1,2,3,4 เรียกว่า ฟิลด์คี่ (ODD FIELD) และในระหว่างเส้นต่อเส้นก็จะเว้นช่องว่างให้พอสแกนได้อีกครั้งหนึ่ง จากนี้ก็จะเริ่ม 5,6,7 ใหม่อีกครั้งหนึ่ง เรียกว่า ฟิลด์คู่ (EVEN FIELD) เป็นการสแกนแบบเส้นเว้นเส้น ซึ่งต้องใช้การสแกนในแนวตั้ง 2 ครั้ง ดังนั้นถ้าต้องการส่งภาพ 25 ภาพต่อวินาที ก็ต้องส่ง 50 ครั้ง (เป็นระบบแบบยุโรป) นั่นคือ เป็นการส่งภาพแบบหยาบๆไป 50 ภาพนั่นเอง วิธีนี้เราเรียกว่าวิธีการสแกนแบบสลับเส้น

ในการสแกนแบบสลับเส้นนี้ต้องใช้การสแกนแนวตั้ง 2 ครั้ง การสแกนแนวตั้ง 1 ครั้ง เรียกว่า การสแกน 1 ฟิลด์ และการสแกน 2 ฟิลด์ เรียกว่า 1 เฟรม (FRAME) ในระบบ 625 เส้นจากรูป 2.4 (ข) การสแกน 1 ฟิลด์มี 312.5 เส้น เนื่องจากตาของมนุษย์มีคุณสมบัติในการคงอยู่ของภาพ (PRESISTANCE OF IMAGE) และระยะเวลาในการเรืองแสงของฟอสเฟอร์ที่จอภาพ ทำให้ภาพของฟิลด์ที่หนึ่งยังคงอยู่



(ก)



(ข)

รูปที่ 2.4 การสแกนแบบสลับเส้น

ในขณะที่ฟิลด์ที่ 2 สแกนเสร็จแล้ว ภาพที่มองเห็นจึงมีจำนวน 625 เส้น ข้อดีของวิธีนี้ก็คือ ทำให้ลดการกระพริบของภาพ (FLICKER) ได้ถึงเท่าตัว

ในการสแกนทั้งแบบก้าวหน้าและแบบสลับเส้น เมื่อสแกนไปสุดของแต่ละเส้นแล้ว ต้องรีบกลับมาเริ่มเส้นใหม่ ทั้งแนวตั้งและแนวนอน ระยะเวลาในการวิ่งกลับมาเริ่มใหม่นั้นยิ่งน้อยเท่าไรยิ่งดี จากรูป 2.3 คือ เส้นประจาก b ---> c ,d ---> e เส้นนี้เรียกว่าเส้นสแกนกลับ (RETRACE OR FLYBACK) เส้นนี้ไม่มีความจำเป็นในการประกอบเป็นภาพจึงมีวงจรควบคุมไม่ให้ปรากฏที่จอภาพ

ในโครงการนี้จะทำการดิจิไตซ์ (DIGITIZE) เพียงฟิลด์เดียว จากสัญญาณ

ที่ส่งมาโดยวิธีการสแกนสลับเส้น ดังนั้นก่อนที่จะทำการเก็บภาพ ภาพที่แสดงบนจอภาพโทรทัศน์จะเป็นวิธีการสแกนแบบสลับเส้น เมื่อทำการเก็บภาพที่แสดงบนจอภาพโทรทัศน์ จะเป็นภาพของข้อมูลที่เก็บอยู่ในหน่วยความจำ และ แสดงผลโดยวิธีการสแกนแบบก้าวหน้า

2.3 สัญญาณโทรทัศน์ (COMPOSITE SIGNAL)

เพื่อที่ทำให้เกิดภาพที่สมบูรณ์ สัญญาณโทรทัศน์ต้องประกอบด้วยสัญญาณต่างๆดังนี้

2.3.1 สัญญาณภาพและสัญญาณเสียง (VIDEO AND AUDIO SIGNAL)

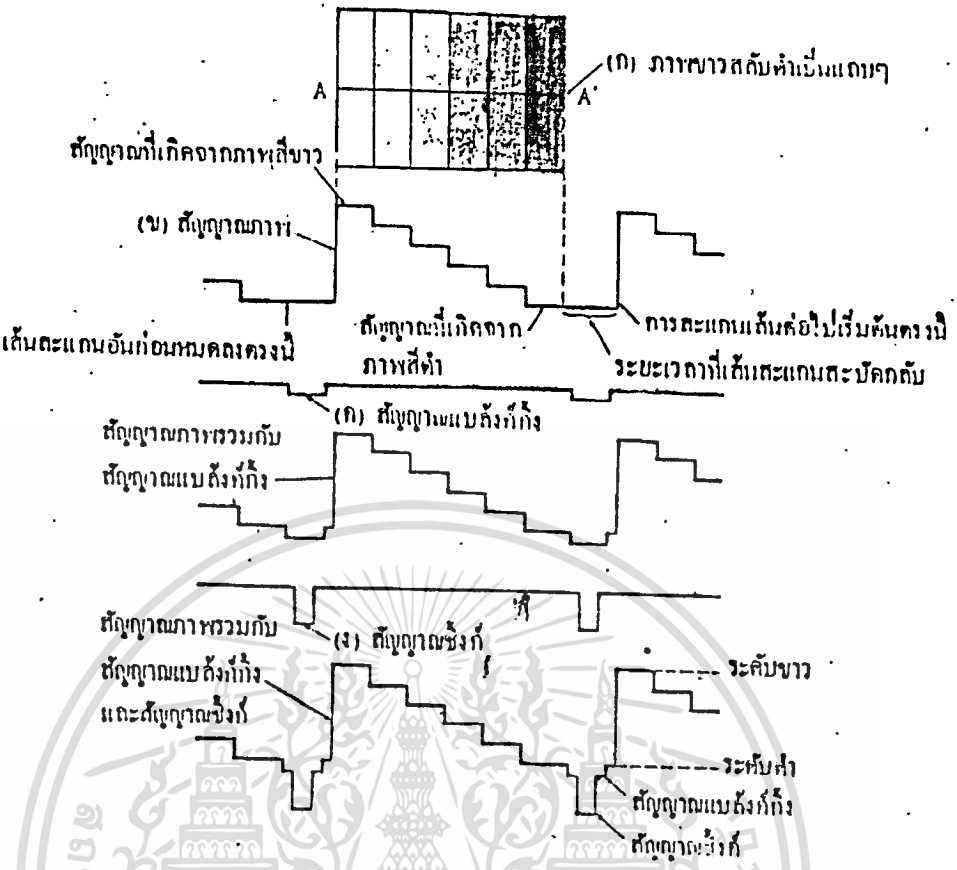
เป็นสัญญาณที่ใช้เพื่อทำให้เกิดภาพและเสียงตามต้องการ ในโครงการนี้เราใช้เพียงสัญญาณภาพอย่างเดียว

2.3.2 สัญญาณแบล้งกลิ้ง (BLINKING PULSE)

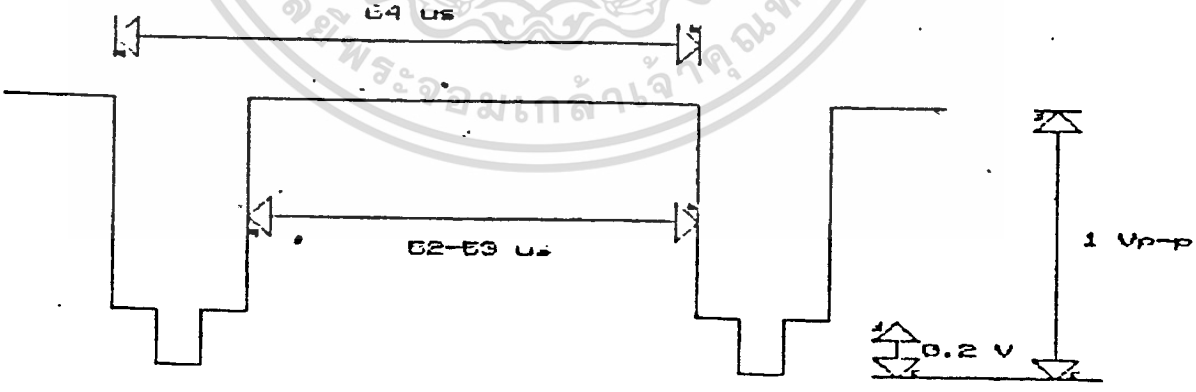
เป็นสัญญาณที่ใช้เพื่อลบเส้นสแกนสะบัดกลับทิ้งทั้งในแนวนอนและแนวตั้ง เพื่อมิให้สังเกตเห็นได้ชัดเจนทางจอภาพ สำหรับโทรทัศน์ระบบอเมริกาสัญญาณแบล้งกลิ้งระหว่างเส้นสแกน (แบล้งกลิ้งทางแนวนอน) จะมีขนาดประมาณ 10 ไมโครวินาที ทำนองเดียวกันสัญญาณแบล้งกลิ้งระหว่างฟิลด์ (แบล้งกลิ้งทางแนวตั้ง) จะมีขนาดประมาณ 1,250 ไมโครวินาที ส่วนสัญญาณแบล้งกลิ้งของโทรทัศน์ระบบยุโรปก็มีค่าประมาณนี้ (ดูรูปที่ 2.5)

2.3.3 สัญญาณซิ้งค์

เป็นสัญญาณที่ใช้เพื่อช่วยทำให้วงจรหักเหทางแนวนอนและแนวตั้ง (เรื่องราวเกี่ยวกับวงจรหักเหสามารถค้นคว้าได้จากตำราโทรทัศน์ทั่วไป) ในเครื่องส่งและเครื่องรับโทรทัศน์มีความถี่ตรงกันตลอดเวลา ในระบบยุโรปสัญญาณซิ้งค์ทางแนวนอนมีความถี่ 15,625 Hz ซึ่งเท่ากับความถี่ของวงจรหักเหทางแนวนอนและสัญญาณซิงค์ทางแนวตั้งก็มีความถี่ 50 Hz ซึ่งเท่ากับความถี่ของวงจรทางแนวตั้งเหมือนกัน เนื่องจากว่าความถี่ของสัญญาณซิ้งค์มีค่าเท่ากับความถี่ของสัญญาณแบล้งกลิ้งพอดี จึงจำเป็นต้องป้องกันการรบกวนที่อาจเกิดขึ้นโดยการกำหนดขนาดของคซิ้งค์พัลซ์ให้น้อยกว่าขนาดของแบล้งกลิ้ง คือทำให้ซิงค์พัลซ์ทางแนวนอนมีขนาดเพียง 5 ไมโครวินาที และ ซิ้งค์ทางแนวตั้งมีขนาดประมาณ 190 ไมโครวินาทีเท่านั้น นอกจากนี้ยังใช้วิธีส่งซิ้งค์เหล่านี้ปนไปกับแบล้งกลิ้งพัลซ์อีกด้วย โดยให้ฐานของซิ้งค์พัลซ์อยู่ทับซอบบนของแบล้งกลิ้งพัลซ์อีกชั้นหนึ่ง เมื่อจัดขอบเขตความต่างศักย์ให้อยู่ระดับสูงสุดของแบล้งกลิ้งพัลซ์เป็นระดับดำมืดจนมองไม่เห็นแล้ว ระดับของสัญญาณซิ้งค์พัลซ์ที่อยู่บนยอดสูงสุดของแบล้งกลิ้งพัลซ์ก็จะเป็นดำมืดสนิทไปด้วย และไม่ทำให้เกิดการรบกวนภาพที่จะปรากฏแต่อย่างใด (ดูรูป 2.5 , 2.6 ,2.7)



รูปที่ 2.5 รูปร่างของสัญญาณโทรทัศน์ที่เกิดจากภาพขาวสลับดำเป็นแถบๆ



รูปที่ 2.6 ขนาดและช่วงเวลาของสัญญาณภาพ

รูปที่ 2.7 สัญญาณภาพรวม

บทที่ 3

การคำนวณและการสร้าง

สำหรับรายละเอียดแนวความคิดและการสร้างนั้น ขอแบ่งรายละเอียดเป็นส่วน 2 ส่วน เพื่อง่ายแก่การทำความเข้าใจวงจร โดยจะแบ่งออกเป็น 2 ส่วน ดังนี้

1 อธิบายแยกออกเป็นส่วนของวงจรย่อยๆ

2 อธิบายถึงการรวมส่วนย่อยๆเหล่านั้น เพื่อเป็นวงจรลมบูรณ์แบบ

3.1 อธิบายแยกออกเป็นส่วนของวงจรย่อยๆ

ในส่วนย่อยเหล่านั้นสามารถที่แบ่งออกเป็นส่วนต่างๆได้ ดังนี้

3.1.1 ภาคแยกสัญญาณสแกนแนวนอนและแนวตั้ง

3.1.2 ภาคแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

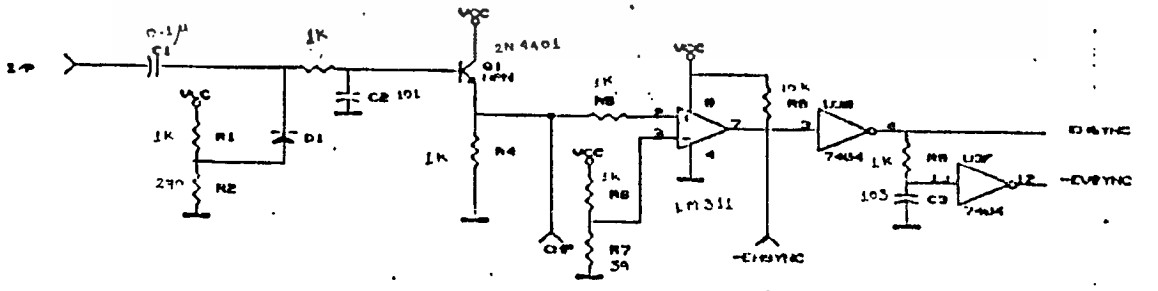
3.1.3 ภาควงจรนับ (COUNTER)

3.1.4 ภาคหน่วยเก็บความจำความถี่สูง (HIGH FREQUENCY RAM)

3.1.5 ภาคควบคุมและรับคำสั่ง (CONTROL AND KEY)

3.1.6 ภาคแปลงสัญญาณดิจิตอลเป็นอนาลอก

3.1.1 ภาคแยกสัญญาณสแกนแนวนอนและแนวตั้ง



รูป 3.1 วงจรแยกสัญญาณสแกนแนวนอนและแนวตั้ง

จากที่ได้ทราบมาแล้วสัญญาณโทรทัศน์ หรือ สัญญาณคอมโพสิท (COMPOSITE) นั้นประกอบไปด้วยสัญญาณสแกนในแนวนอน (H SYNC) และ สัญญาณสแกนในแนวตั้ง (V SYNC)

หลักการทำงาน

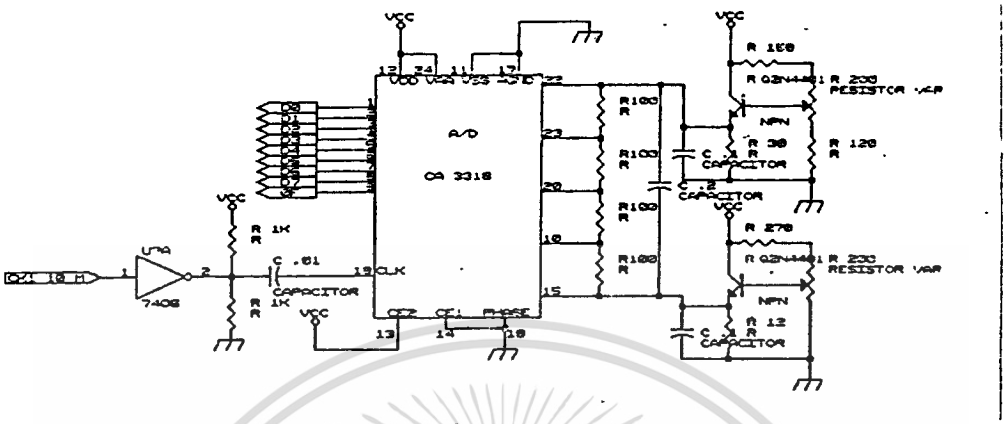
เริ่มต้นจากมีสัญญาณโทรทัศน์เป็นสัญญาณอินพุทผ่าน C1 ซึ่ง C1 มีหน้าที่คัปปลิ้ง (Coupling) ให้ปลายของสัญญาณซิ้งค์อยู่ที่ระดับกราวนด์พอดี R1, R2, D1 ทำหน้าที่ยกระดับสัญญาณดีซีของสัญญาณโทรทัศน์ให้สูงขึ้น (CLAMPING SIGNAL) ประมาณ 0.6 โวลท์ทั้งนี้เนื่องจากเมื่อสัญญาณผ่านQ1 ซึ่งเป็นบัฟเฟอร์แล้วจะทำให้ศักดาที่จุด A สูงกว่าที่จุด B อยู่ประมาณ 0.6 โวลท์ ดังนั้นเราจึงต้องชดเชยโดยการเพิ่มสัญญาณดีซีเข้าที่จุด A ออปแอมป์ LM 311N ซึ่งเป็นออปแอมป์ประเภทโวท์เตจคอมพาราเตอร์ (VOLTAGE COMPARATER) ใช้ในการเปรียบเทียบสัญญาณที่ต่ำกว่า 0.2 โวลท์ ซึ่งเป็นสัญญาณซิงค์ โดยเมื่อมีสัญญาณซิงค์ผ่านเข้ามาจะทำให้เอาท์พุทที่ขา 7 ของ LM 311N มีค่าลอจิกเป็น 0 ดังนั้นเราจะได้สัญญาณสแกนแนวนอน แต่เนื่องจากในสัญญาณสแกนในแนวตั้งนั้นจะมีสัญญาณสแกนในแนวนอนประปนมาด้วย ทำให้เราจำเป็นทำการอินทิเกรตสัญญาณ หรือ หาพื้นที่ใต้กราฟของสัญญาณ ซึ่งในจุดที่เป็นสัญญาณสแกนในแนวตั้งนั้นจะมีพื้นที่ใต้กราฟสูงที่จุดอื่นๆ สามารถสังเกตได้จากภาพของสัญญาณที่ให้มา นอกจากนี้ยังมีอินเวอร์เตอร์ที่จะกลับเฟสสัญญาณและจะทำให้สัญญาณสแกนในแนวตั้งเป็น 0 หรือ 1 โดยแท้จริง

3.1.2 ภาคการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (A/D)

A/D เป็นส่วนการแปลงสัญญาณอนาลอก (มีทั้งค่า ศักดา, กระแส, ความถี่ ฯลฯ) ให้เป็นสัญญาณดิจิตอล ในการออกแบบที่จะใช้ A/D นี้ควรจะทราบถึงความหมายของคำต่างๆ ดังนี้

รีโซลูชั่น หมายถึง จำนวนขั้นที่แบ่งจากช่วงระหว่างระดับสัญญาณต่ำสุดถึงสูงสุด ซึ่งมันจะเป็นตัวกำหนดความกว้างของระดับสัญญาณในแต่ละขั้นอีกด้วย การบอกรีโซลูชั่นจะบอกเป็นจำนวนบิทของเอาท์พุท เช่น ถ้ามีรีโซลูชั่น 8 บิท ก็จะมีจำนวนขั้นทั้งหมด 256 ขั้น

LSB (LEAST SIGNIFICANT BIT) เป็นบิทที่มีความสำคัญน้อยที่สุด (ถูกถ่วงน้อยที่สุด) คือ บิทที่จะเปลี่ยนแปลงได้ด้วยค่าศักดาอินพุทเพียงเล็กน้อย คือ เท่ากับช่วงของศักดาที่วัดจากระดับเปรียบเทียบ (VOLTAGE REFERENCE,V-) ต่ำสุดจนถึง



รูป 3.2 วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

ระดับแรกที่แบ่งไว้ เช่น 00000000-00000001

MSB (MOST SIGNIFICANT BIT) เป็นบิทที่มีความสำคัญมากที่สุด (ถูกถ่วงมากที่สุด) คือ เป็นบิทที่จะเปลี่ยนแปลงได้ด้วยค่าอินพุทมาก คือ เท่ากับช่วงที่แบ่งไว้ ซึ่งจะเท่ากับครึ่งหนึ่งของจำนวนช่วงทั้งหมด เช่น 00000000-10000000

ความแม่นยำ (ACCURACY) เป็นผลรวมของความผิดพลาด (ERROR) ทั้งหมดที่เกิดขึ้น คือ จากความไม่เป็นเชิงเส้น (NON-LINEARITY) , ความผิดพลาดระดับต่ำ (ZERO SCALE ERROR) ความผิดพลาดระดับสูง (FULL SCALE ERROR) , ความผิดพลาดจากอุณหภูมิ (TEMPERATURE DRIFT)

เวลาในการแปลง (CONVERTION TIME) เป็นเวลาที่ใช้ในการแปลงสัญญาณจากอินพุทจนได้เอาท์พุท

ศักดาเปรียบเทียบ (REFERENCE VOLTAGE) มีด้านต่ำ (V-) และด้านสูง (V+) คือ เป็นช่วงขอบเขตศักดาที่ต้องการแบ่งให้เป็นขั้นโดย

ด้านต่ำ จะเป็นระดับศักดาที่จะให้ค่าเอาท์พุทเป็น 0 หมด

ด้านสูง จะเป็นระดับศักดาที่จะให้ค่าเอาท์พุทเป็น 1 หมด

สำหรับโครงงานนี้ใช้ A/D ที่เป็นไอซีสำเร็จรูป CA3318E

V- เป็นระดับสัญญาณภาพที่ มืดที่สุด

V+ เป็นระดับสัญญาณภาพที่ สว่างที่สุด

สำหรับวงจรสามารถดูได้ในรูป 3.2 ในการออกแบบสามารถดูได้จากดาต้าบุค (DATA BOOK) ไอซีตัวนี้มีข้อดีที่สามารถแบ่งแยกกราวด์อนาลอกและกราวด์ดิจอตอลออกกันได้อย่างเด็ดขาด ทำให้ลดปัญหาการกวนกันได้ มีรีโซลูชั่น 8 บิท หรือ จะมีรีโซลูชั่น 256 ขั้น

3.1.3 วงจรนับ (COUNTER)

ในส่วนของวงจรนับนั้นจะมีหน้าที่ในการนับ คือ ใน 1 สแกนแนวนอนนั้นจะต้องนับให้ได้ 512 จุดภายในเวลา 52 ไมโครวินาที โดยเมื่อนับได้ครบ 512 แล้วจะต้องหยุดนับเองโดยอัตโนมัติ เวลาที่เหลืออยู่จาก 52 ไมโครวินาทีนั้นก็ปล่อยทิ้งไป และจะต้องนับทั้งสิ้นทั้งหมด 256 เส้นสแกนแนวนอน ใน 1 เส้นสแกนแนวตั้ง โดยปกตินั้นใน 1 เส้นสแกนแนวตั้งจะมีเส้นสแกนแนวนอนอยู่เท่ากับ 312.5 เส้น ฉะนั้นเราจึงต้องตัดเส้นสแกนในแนวนอนบางเส้นทิ้งไป โดยเราจะทำการตัดเส้นสแกนในแนวนอนในช่วงต้น และช่วงท้ายทิ้งไป ซึ่งในส่วนนี้ถูกควบคุม และ นับโดย 8031

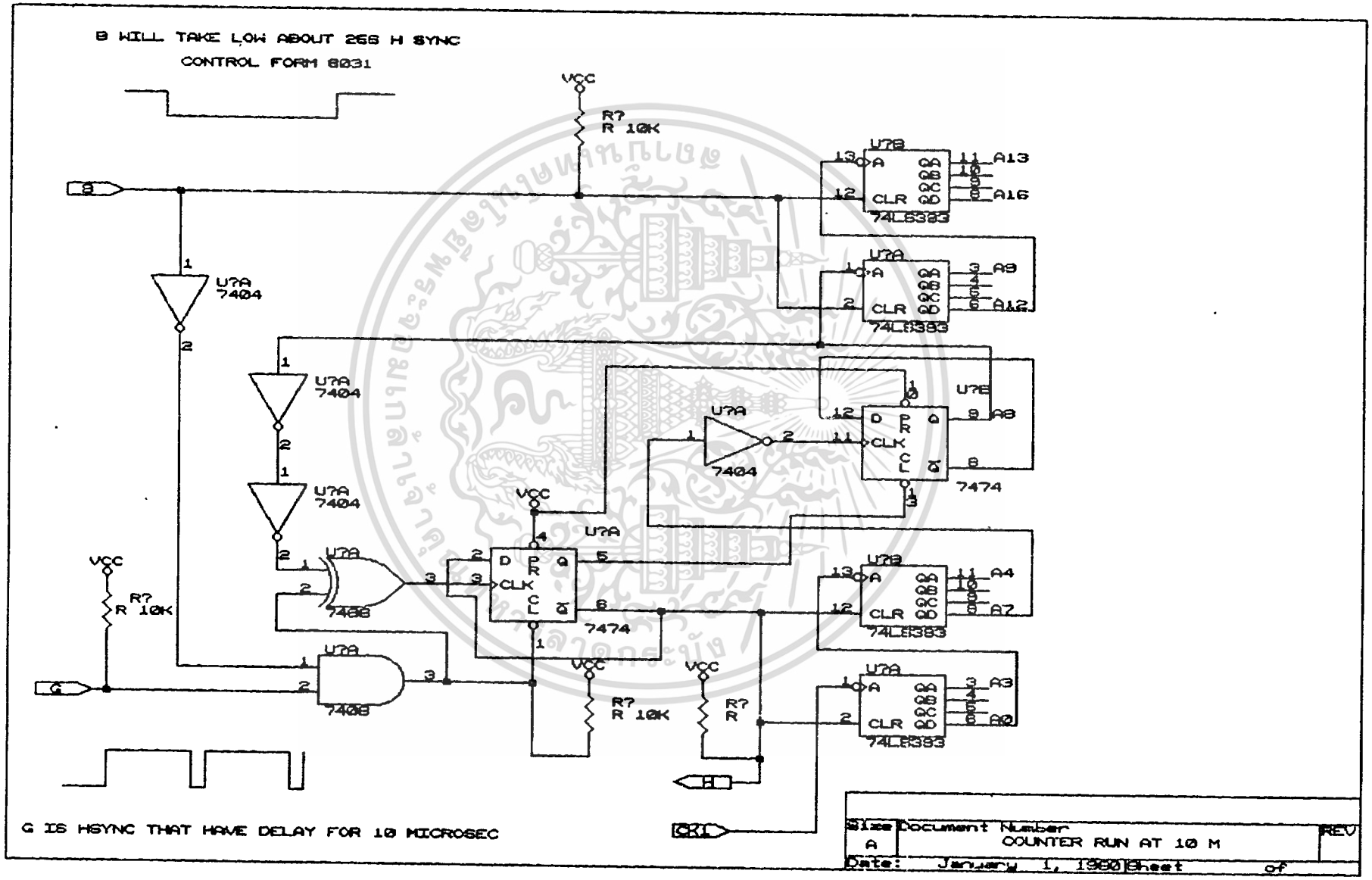
ข้อระวังในการออกแบบ

1 การทำงานของ IC แต่ละตัว โดยเฉพาะ IC ที่ใช้ขอบขาในการทำงานนั้น ควรพิจารณาสัญญาณด้วยความรอบคอบต้องให้ซิงค์กับสัญญาณนาฬิกา หรือ สัญญาณอื่นๆที่นำมาทำเป็นสัญญาณในการเปรียบเทียบ

2 ควรจะมีความต้านทาน ทำการพูลอัพ (PULL UP) ไว้ทุกขาของสัญญาณที่ควบคุมการทำงานของ IC หรือ สายสัญญาณที่เราจะนำไปใช้ควบคุม IC ตัวอื่นๆอีกที

หลักการทำงาน

จะเห็นว่ามีส่วนควบคุมการทำงานอยู่ 2 เส้น คือ สายสัญญาณที่ได้จากการนับเส้นสแกนในแนวนอน 256 เส้น จาก 312.5 เส้นใน 1 เส้นสแกนในแนวตั้งโดยถ้าอยู่ในช่วง 256 เส้นนั้นจะมีสถานะเป็น LOW นอกนั้นจะอยู่ในสถานะ HIGH (สาย B) กับสายสัญญาณสแกนในแนวนอนที่ได้รับการหน่วงทิ้งไปแล้ว 10 ไมโครวินาที นับจากขอบขาของสัญญาณซิงค์ในแนวนอน (สาย G) ในขณะที่ B เป็น HIGH นั้น จะทำให้ 74393 ทุกตัวอยู่ในสถานะที่ CLEAR หมด คือ ค่าแอดเดรสที่เป็นค่าผลลัพธ์ (OUTPUT) ของ


B WILL TAKE LOW ABOUT 256 H SYNC
CONTROL FORM 8031
VCC
R?
R 10K
U?A
7404
U?A
7408
U?A
7474
74LS383
A13
A16
A9
A12
A8
A4
A7
A3
A0
B
G
H
CRT
G IS HSYNC THAT HAVE DELAY FOR 10 MICROSEC
Size
A
Document Number
COUNTER RUN AT 10 M
REV
Date: January 1, 1980
Sheet
of

74393 เป็นสถานะ 00 หมด เมื่อ B เปลี่ยนสถานะเป็น LOW 7474 ก็จะรอสัญญาณ G ว่าจะขึ้นเป็น HIGH เมื่อไร เมื่อขึ้นเป็น HIGH ได้แล้ว ขา CLEAR ของ 74393 ก็จะเป็น LOW ทำให้พร้อมในการที่จะนับถ้ามีสัญญาณนาฬิกาขอบขาลงมาที่ขา CK ของ 74393 เมื่อทำการนับครบ 512 แล้ว 7474(2) ก็จะส่งสัญญาณมาที่ 7486 มาทำให้ 7474(1) เกิดการท๊อกเกิล (TOGGLE) เป็นผลให้ 74393 ตัวที่ 1 และ 7474 ตัวที่ 2 อยู่ในสถานะ CLEAR อีกครั้ง ซึ่งจะทำการนับ 512 จุดเช่นนี้ไปเรื่อยๆ เป็นจำนวนทั้งสิ้น 256 เส้นสแกนในแนวนอน จาก 312.5 เส้นใน 1 เส้นสแกนในแนวตั้ง

สามารถดูการออกแบบวงจรนับได้ในรูป 3.3

3.1.4 ภาคหน่วยเก็บความจำความถี่สูง (HIGH FREQUENCY RAM)

เนื่องมาจากการที่เราทำการแซมปลิ้งสัญญาณด้วยความถี่สูง เพื่อให้ได้ 512 จุดภายใน 52 ไมโครวินาทีนั้นจะต้องใช้ความถี่ในการแซมปลิ้งถึง 10MHz หรือ เวลาในการอ่าน และ เขียนบนหน่วยความจำเพียง 100 นาโนวินาที เท่านั้น โดยปกติแล้วหน่วยความจำที่มีขายในท้องตลาด จะมีความเร็ว 120 , 150 และ 200 นาโนวินาที ส่วนหน่วยความจำที่ความเร็วขนาด 100 นาโนวินาทีนั้นมีขายอยู่ต่างประเทศ และ มีราคาแพง ทำให้เรามีความจำเป็นจะต้องหาเทคนิคและกรรมวิธีที่จะให้หน่วยความจำของเราสามารถที่จะอ่านหรือเขียนได้เร็วขึ้น โดยจะต้องรักษาคุณสมบัติการอ่านและเขียนในแบบปกติ คือ

1 จะต้องสามารถที่จะทำการอ่านและเขียน โดย 8031ได้ โดยมีเวลาในการอ่าน และ การเขียนข้อมูลที่ขึ้นกับเวลาการอ่าน และ เขียนของ 8031โดยตรง

2 จะต้องสามารถที่จะทำการอ่านและการเขียนด้วยความเร็วสูงเนื่องมาจากการแซมปลิ้งความถี่สูงได้ เป็นความถี่ในการเก็บสัญญาณภาพ

ข้อระวังในการออกแบบ

1 จะต้องระวังอย่างมากในเรื่องขอบขาของสัญญาณควรจะทำตารางเวลาการทำงานของ IC แต่ละไว้ด้วย เพื่อที่จะทำความเข้าใจได้ง่าย และ ไม่สับสน

2 เนื่องจากความจำเป็นที่จะต้องใช้ 74374 ซึ่งเป็น IC ที่ต้องใช้ขอบขาของสัญญาณเพื่อที่จะทำการ แลทช์ ค่าข้อมูลจากอินพุท ให้ออกมาที่เอาท์พุทนั้น ถ้าหากมีการแลทช์ค่าข้อมูลค่าข้อมูลในขณะที่ค่าข้อมูลในอินพุทมีสถานะไม่แน่นอนนั้นจะทำให้เกิดปัญหาอย่างมาก เพราะฉะนั้นควรจะมีการหน่วงสัญญาณออกไปเล็กน้อย โดยใช้ อินเวอร์เตอร์

หรือ แอนด์ หรือ ออร์ เกทก็ได้

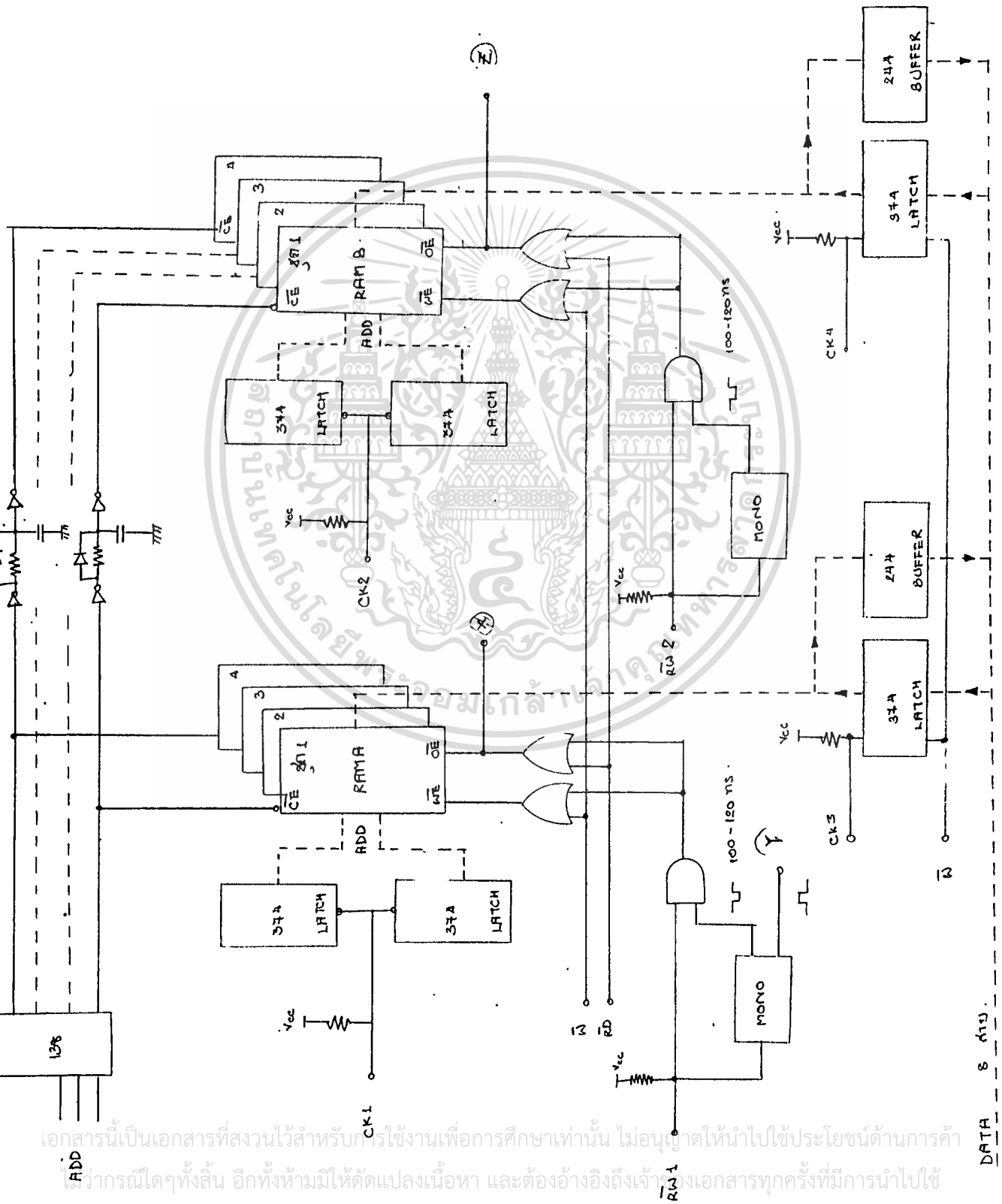
เนื่องมาจากในเทอมที่ 2 นี้ได้มีการออกแบบวงจรใหม่ โดยมีหลักการที่แตกต่างไปจากวงจรในแบบแรก ผมขอทิ้งแบบวงจรแบบแรกไว้เป็นแนวคิด ซึ่งก็คงมีประโยชน์ไม่มากก็น้อย

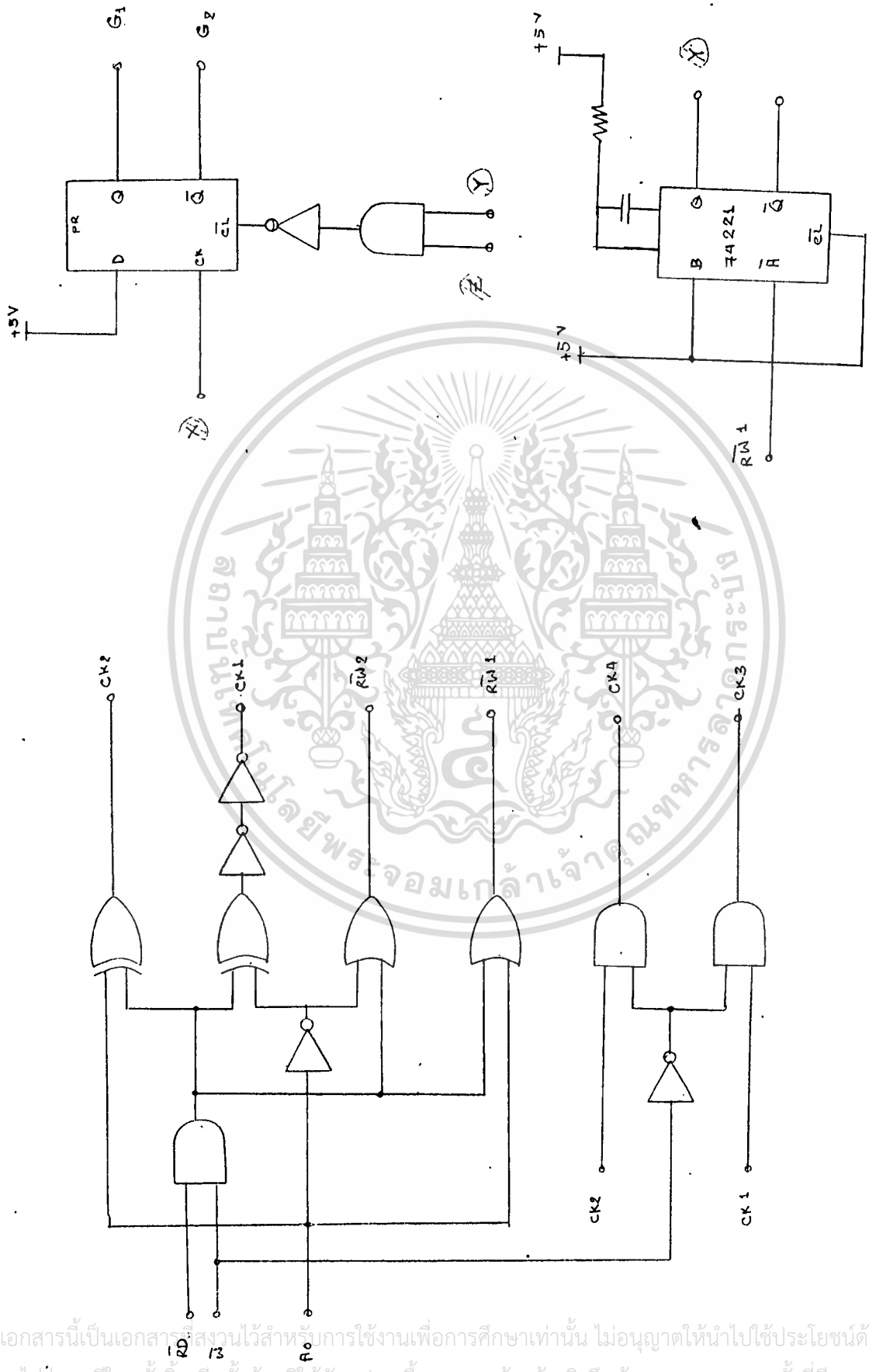
หลักการทำงานของวงจรหน่วยเก็บความจำความถี่สูงเทอม 1

เราจะทำการออกแบบวงจรนี้เราจะต้องรักษาคุณสมบัติในความเป็น RAM แบบปกติ คือ สามารถที่ทำการอ่าน และ เขียนข้อมูลได้ มีการติดต่อส่งผ่านข้อมูลกับ 8031 ได้ โดย HIGH FREQUENCY RAM มีหลักการดังนี้ เนื่องจากเวลา READ/WRITE CYCLE TIME มีค่ามากกว่าความเร็วที่เราทำการแซมปลิ้งข้อมูล ทำให้เวลาในข้อมูลแต่ละตัวของการแซมปลิ้งมีค่าน้อยกว่าเวลา READ/WRITE CYCLE TIME ของ RAM ซึ่งก็หมายความว่า เราไม่สามารถที่ทำการอ่านเขียนข้อมูลได้อย่างสมบูรณ์แบบและถูกต้อง ดังนั้นเราจึงใช้หลักการที่ให้ RAM สลับกันทำงานเป็นผลให้มันมีความสามารถในการอ่าน และเขียนข้อมูลที่ต่อเนื่องกันมา (ไม่มีการข้าม ADDRESS) เป็น 2 เท่าของ RAM ปกติ โดยเราใช้ A_0 เป็นตัวเลือก RAMA หรือ RAMB ฉะนั้นข้อมูลที่มีค่าแอดเดรสคู่จะอยู่ที่ RAMA และ แอดเดรสคี่จะอยู่ที่ RAMB สำหรับโมโนสเตเบิ้ลที่ทำขึ้นมาเพื่อเพิ่มเวลาในการอ่านและเขียนข้อมูลให้มากขึ้นให้ได้ประมาณ 100-120 นาโนวินาที ตามค่าที่บอกมาของ RAM ตัวนั้น นอกจากนี้เราจำเป็นต้องค้างค่าข้อมูล(LATCH) ค่าแอดเดรสไว้โดยใช้ IC 74374 ซึ่งจะทำงานที่ขอบขาขึ้นของสัญญาณนาฬิกา เพื่อที่ค้างค่าแอดเดรสไว้ไม่ให้เปลี่ยนแปลงเมื่อมีการสลับกันทำงานของ RAM สำหรับค่าของข้อมูล(DATA)ก็ค้างค่าข้อมูลโดยใช้ IC 74374 เช่นเดียวกัน แต่ในการอ่านค่าข้อมูลนั้นจะใช้บัฟเฟอร์ 74244 แทน โดยใช้สัญญาณ G1 และ G2 เป็นตัวเลือก โดยที่ G2 = G1 หลักการที่ 74244 จะเปิดให้ค่าข้อมูลออกนั้นมีดังนี้ G1 จะลงLOWนั้นได้นั้นสัญญาณ OE ของ RAMA นั้นจะต้องลง LOW เช่นกัน และ G1 จะลง LOW ได้นั้น OE ของ RAMB จะต้องขึ้นเป็น HIGH แล้ว เมื่อ G1 ลง LOW แล้วจะขึ้น HIGH ได้ก็ต่อเมื่อสัญญาณ OE ของ RAMA ขึ้น HIGH สำหรับวงจรที่ออกแบบมานั้น ถ้าหากโมโนสเตเบิ้ลของทั้ง 2 ตัวมีเวลาเท่ากันแล้ว เวลาของ G1 จะเท่ากับเวลา A_0

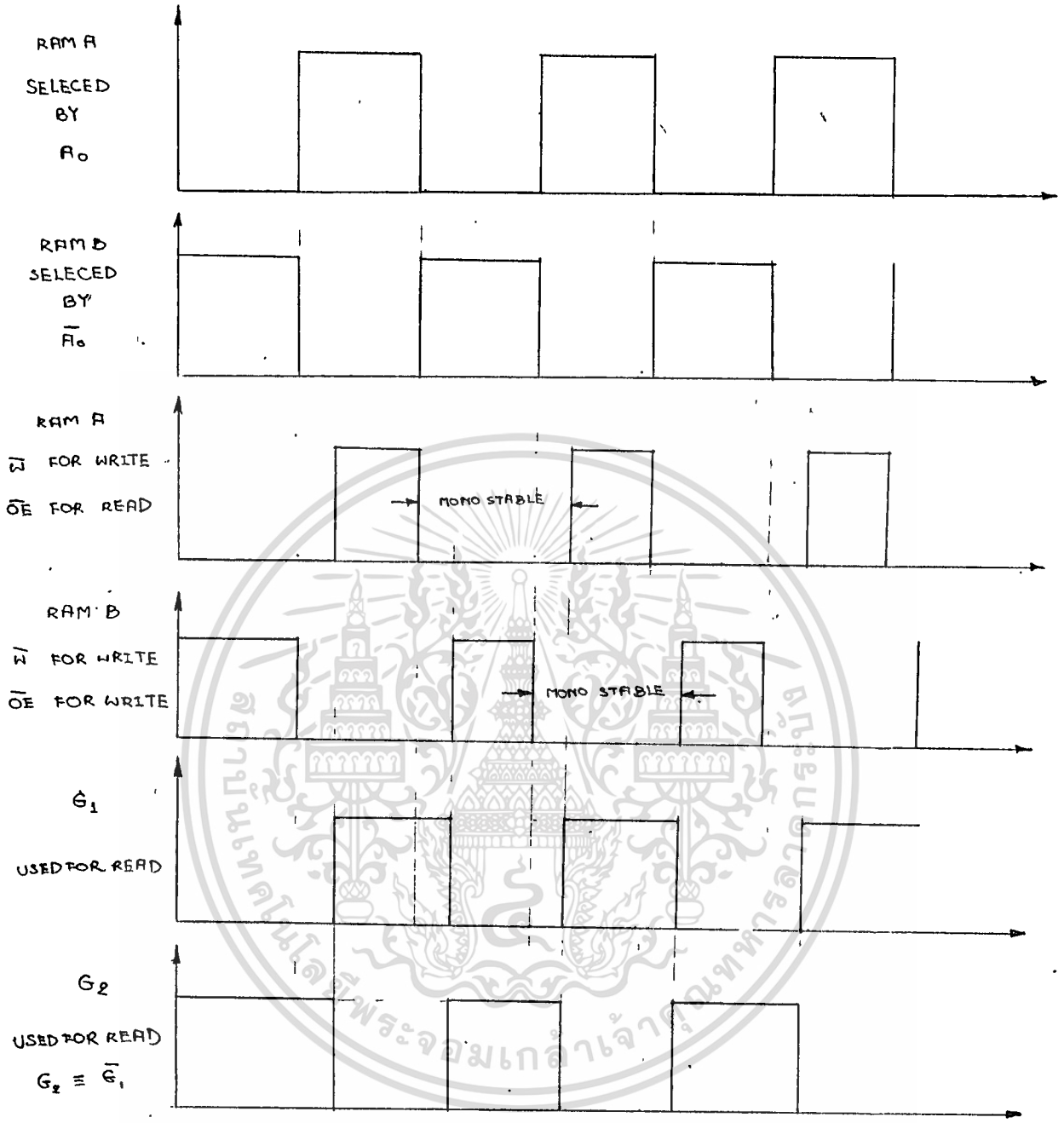
หลังจากที่ RAMA RAMB ของชุด 1 เต็มแล้ว ชุดที่ 2 ก็จะเริ่มทำงานแทน โดยสามารถเลือกได้จาก 74138 จะเห็นว่ามีวงจรหน่วงเวลาก่อนที่จะเข้าขา CE ของ

รูป 3.4 วงจรหน่วยเก็บความจำความถี่สูงเทอม 1





รูป 3.5 วงจรหน่วยเก็บความจำความถี่สูงเทอม 1 ส่วนควบคุม



รูป 3.6 ตารางเวลาและการทำงานวงจรหน่วยเก็บความจำความถี่สูง

RAMB นั้นก็เพื่อที่หน่วงเวลาให้ BYTE สุดท้ายของ RAMB ทำงานให้เสร็จบริบูรณ์ก่อนที่จะไปทำงานต่อในชุดที่ 2

สำหรับรูปการออกแบบในเทอม 1 นี้ สามารถดูในรูปที่ 3.4 ,3.5 ,3.6 ตามลำดับ

หลักการทำงานวงจรหน่วยความจำความถี่สูงแบบที่ 2

แนวคิดในการออกแบบวงจรที่ 2 คือจะทำให้มีการเก็บข้อมูลมาไว้ก่อนแล้วจึงทำการเขียนพร้อมๆกัน และในขณะที่ทำการเขียนอยู่นั้นก็ให้ทำการเก็บค่าข้อมูลชุดใหม่ไปด้วย เมื่อเขียนค่าข้อมูลชุดเก่าเสร็จก็จะทำการเขียนค่าข้อมูลชุดใหม่ต่อไปเลย ทำงานเช่นนี้ไปเรื่อยๆ ... สำหรับในการอ่านนั้นก็จะทำการดีโค๊ดแอดเดรส A0 A1 เพื่อส่งสัญญาณไปที่ OUTPUT ENABLE ของ 74244 ซึ่งทำหน้าที่เป็นบัฟเฟอร์ ขาดาต้าของหน่วยความจำแต่ละตัว ให้มีเอาท์พุทออกมาตามลำดับ

หลักการทำงานกล่าวโดยละเอียดสามารถแบ่งออกได้เป็น 2 ส่วน

<u>*การเขียน*</u> เราใช้ชิพรีจิสเตอร์ (SHILF REGESTER) เป็นเก็บดาต้าที่เกิดจากการแซมปลิ้งของสัญญาณนาฬิกาที่มาจากวงจรแปลงอนาลอกเป็นดิจิตอลและ ที่เกิดจากเขียนของ 8031 เอง โดย D0 เก็บไว้ในชิพรีจิสเตอร์ในตัวที่ 1 ถัดไปเรื่อยๆตามลำดับจนถึง D7 ก็เป็นชิพรีจิสเตอร์ตัวที่ 8 หลังจากที่ทำการเก็บครบ 4 ตัวแล้ว ก็จะทำการแลทช์ค่าดาต้า ตั้งแต่ D0 ถึง D7 ทั้งหมด 4 ชุดไว้ที่ 74374 นอกจากนี้ยังจะต้องแลทช์ค่าแอดเดรสไว้เช่นกัน โดยเมื่อนับถึง หน่วยความจำที่ 4 (A0 A1 = 11) ก็จะทำการแลทช์ค่าแอดเดรสไว้ แล้วใช้ขอบขาของสัญญาณดังกล่าวเป็นสัญญาณในการเขียนหน่วยความจำทั้ง 4 ตัวพร้อมกัน

<u>*การอ่าน*</u> การอ่านจะใช้การดีโค๊ดจาก A0 A1 เพื่อที่จะไปทำการเปิดเกทให้มีเอาท์พุทออกที่ขาเอาท์พุทของ 74244 ซึ่งทำหน้าที่เป็นบัฟเฟอร์ โดยให้มีเอาท์ออกมาจากหน่วยความจำตัวที่ 1 ที่ 2 จนถึงที่ 4 ตามลำดับ ตามค่าที่ดีโค๊ดได้จาก A0 A1

วงจรรวมของวงจรหน่วยความจำความถี่ในเทอม 2 สามารถดูได้จากรูป 3.8 ,3.9 ,3.10

ในส่วนของวงจร HIGH FREQUENCY RAM แบบที่ 2 มีส่วนควบคุมที่น่าสนใจอยู่ส่วนหนึ่ง คือ ส่วนของวงจรที่แบ่งแยกการอ่านแบะการเขียน โดนแบ่งเป็นดังนี้

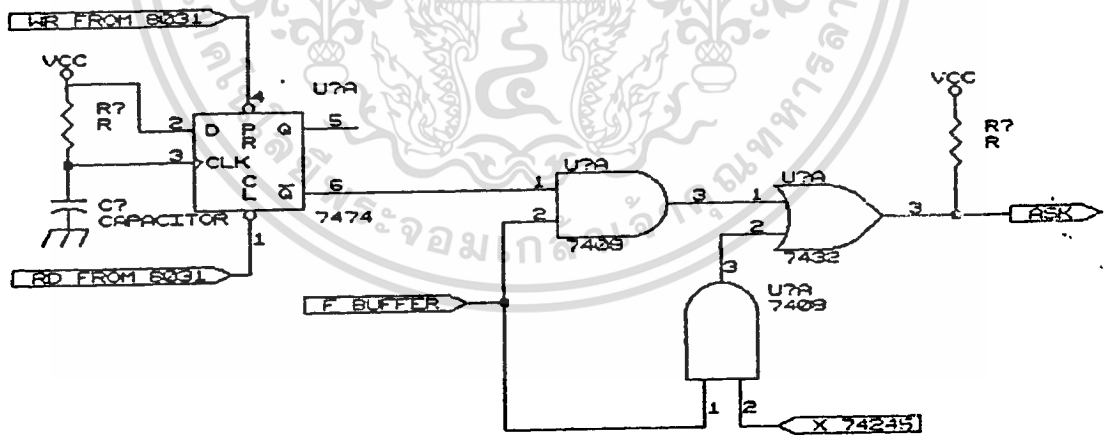
1 การอ่าน และ การเขียน ที่เกิดจากการแซมปลิ้งจากการแปลงอนาลอก

เป็นดิจิตอล

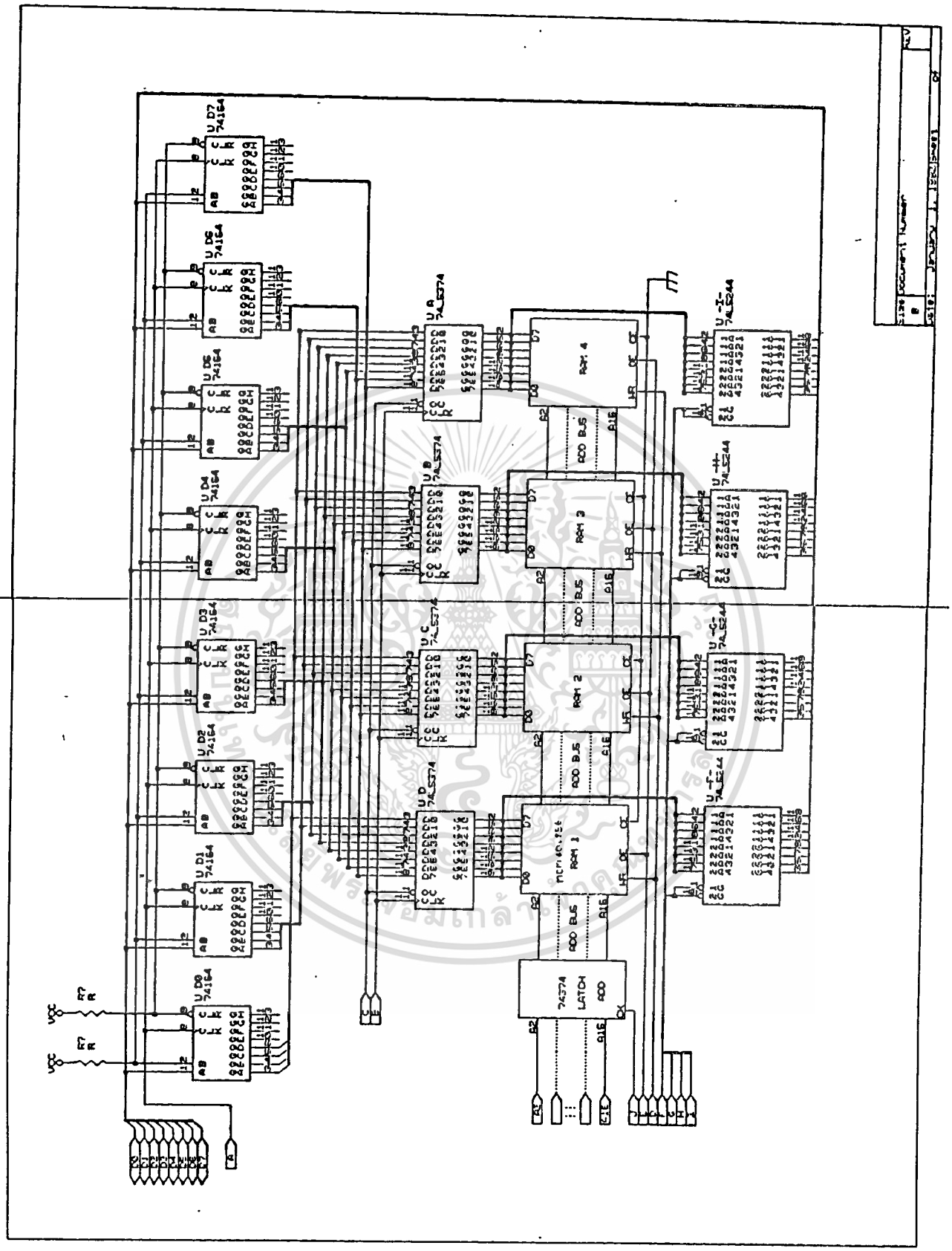
2 การอ่าน และ การเขียน อันเกิดจาก 8031

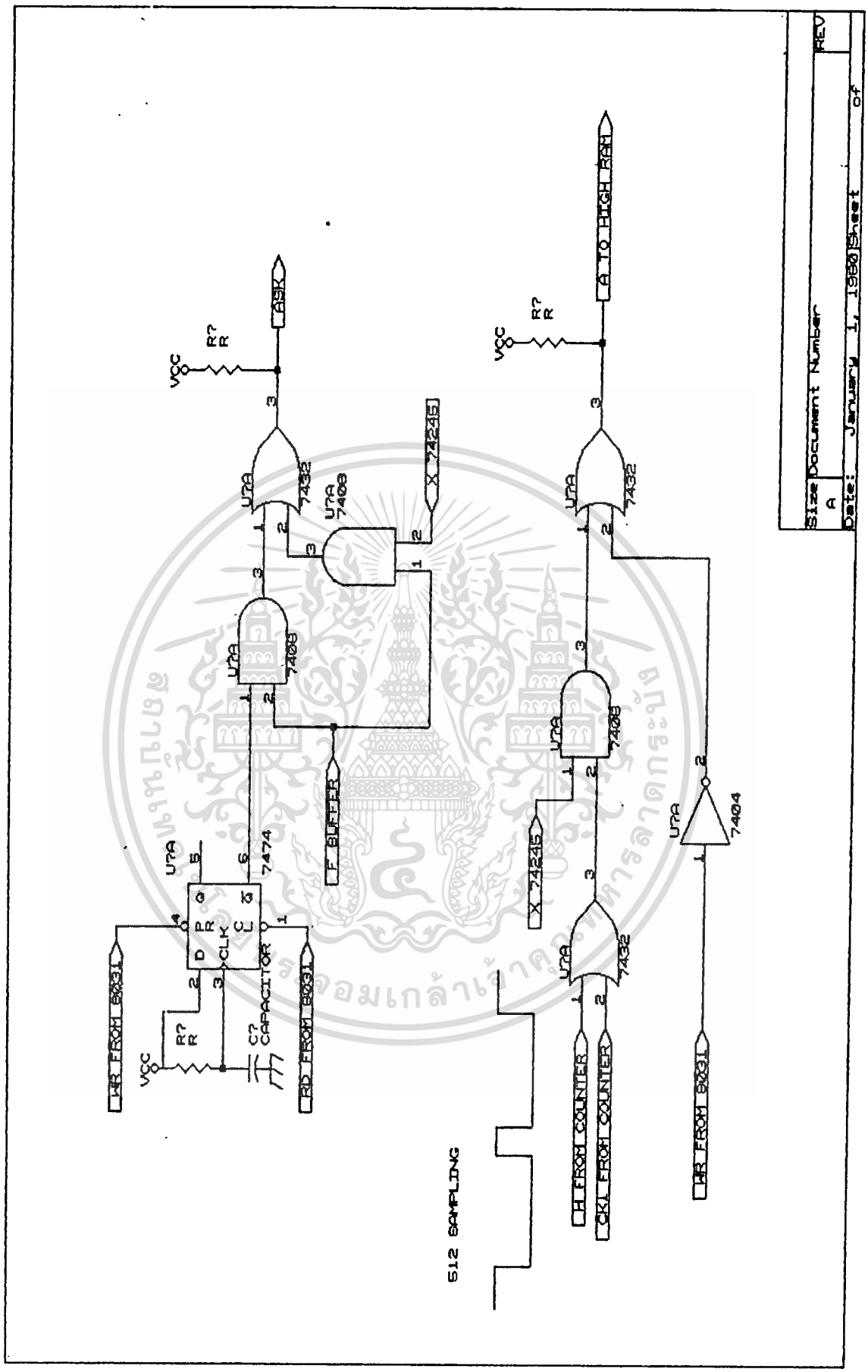
ดูรูปได้จากรูป 3.7 และ จากตาราง

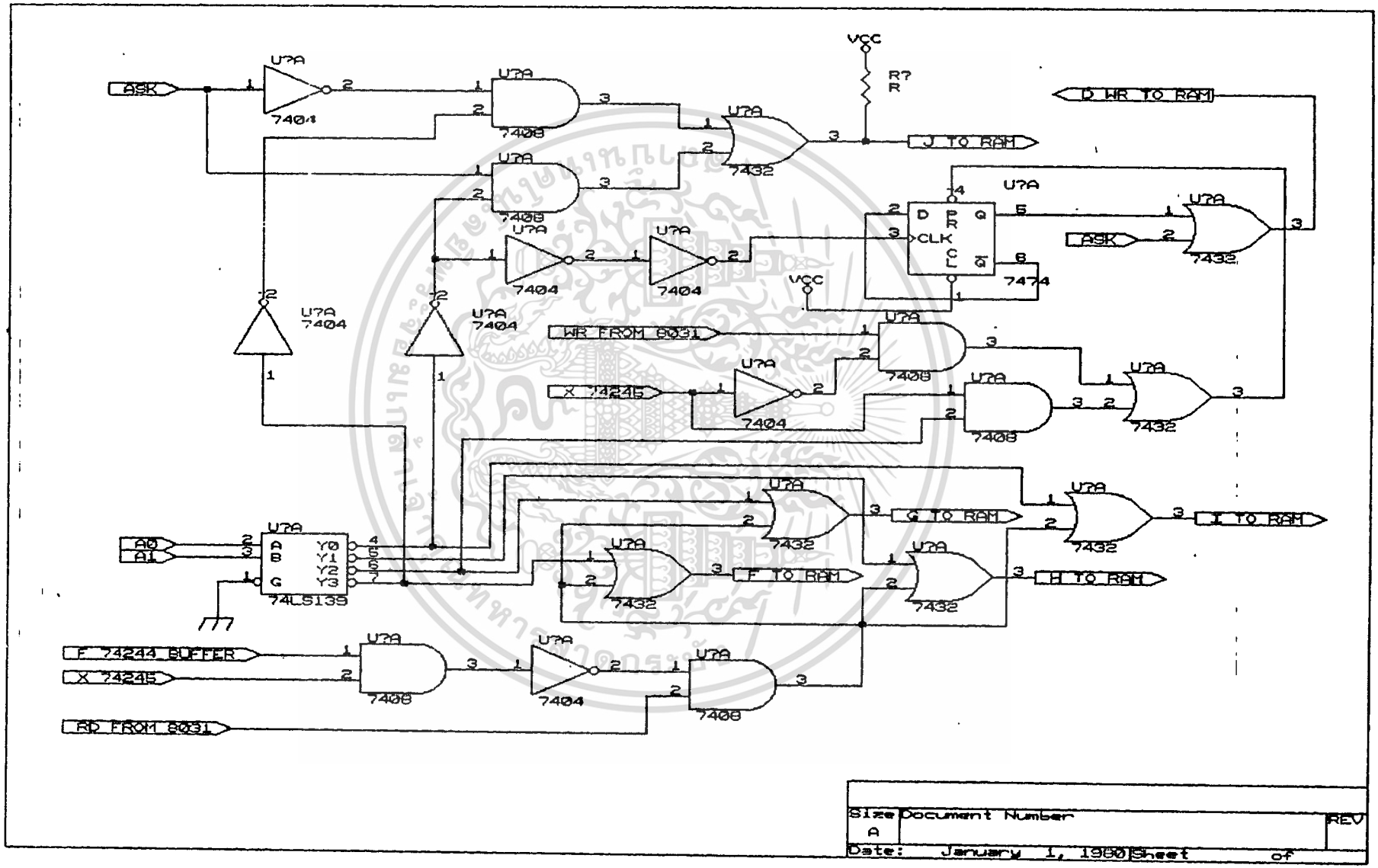
BUFFER	74245 (X)	O/P
0	0	ไม่อนุญาตให้เกิดขึ้น
0	1	เกิดการเขียนจาก A/D
1	0	เกิดการเขียนจาก 8031 หรือเกิดการอ่านจาก 8031
1	1	เกิดการอ่านจาก D/A



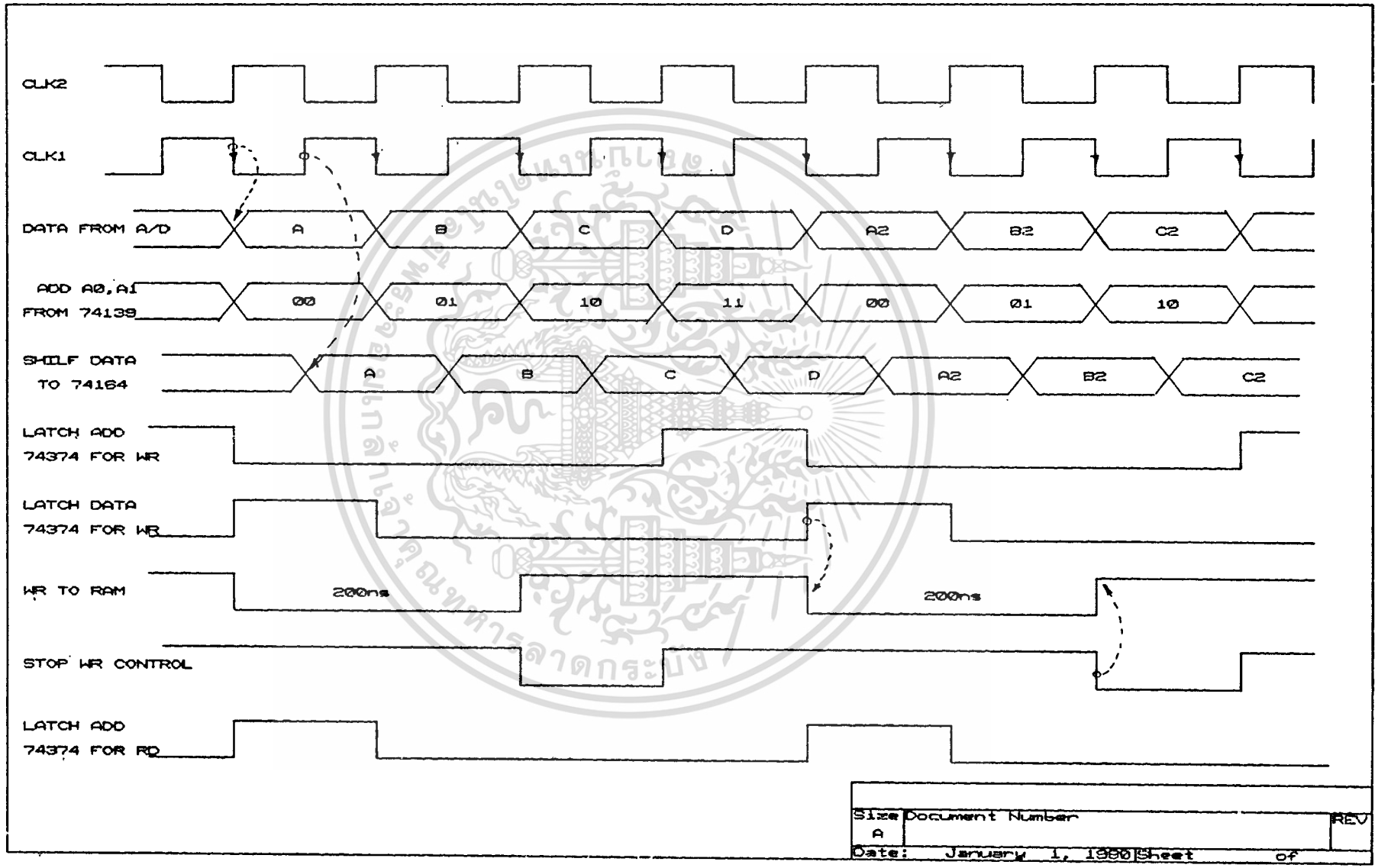
รูป 3.7 วงจรถอดรหัสการอ่านและการเขียนทุกกรณี







รูป 3.9 (ข) ส่วนควบคุมวงจรหน่วยเก็บความจำความเร็วสูงแคช 2



รูป 3.10 ตารางเวลาและการทำงานวงจรหน่วยเก็บความจำความถี่สูง

ภาคควบคุมและรับคำสั่ง (CONTROL AND KEY FROM 8031)

ในส่วนนี้จะอธิบายการทำงานของโปรแกรม ที่สามารถแยกย่อยออกเป็นส่วนได้ดังนี้

3.1.5.1 การรับการกดปุ่ม (KEY)

โดยการใช้ อินเตอร์รัพธ์ 1 เป็นตัวรับสัญญาณให้ไปทำงานในส่วนโปรแกรมการรับ KEY โดยโปรแกรมนี้จะทำหน้าที่เป็นตัวรับว่าตอนนี้ต้องการทำอะไร แล้วจะกระโดดไปทำในงานส่วนโปรแกรมย่อยนั้น ซึ่งเราสามารถสั่งการทำงานได้โดยการกดแป้นกดปุ่ม

3.1.5.2 ส่วนสร้างสัญญาณในสแกนในแนวนอนและแนวตั้ง

เนื่องจากการที่เราใช้ 8031 ในการควบคุมการทำงาน ดังนั้นเราสามารถใช้ 6845 ซึ่งเป็น IC สำเร็จรูปที่จะควบคุมการทำงานจอคอมพิวเตอร์ ที่สามารถสร้างสัญญาณสแกนแนวนอน และ สัญญาณสแกนแนวตั้ง ให้มีรูปแบบเป็นแบบอินเตอร์แลทซ์ (INTERLANCE) หรือเป็นในรูปแบบนอน อินเตอร์แลทซ์ (NON-INTERLANCE) ได้โดยการเปลี่ยนดาต้าในรีจิสเตอร์ภายใน 6845 ซึ่งเหมาะสมเป็นอย่างมากในการขยายการทำงานในอนาคต สำหรับโปรแกรมในส่วนนี้จะอยู่ในช่วงต้นของโปรแกรมเลย คือ จะอยู่ทำงานทันทีหลังจากการเปิดเครื่องใหม่ๆ

3.1.5.3 โปรแกรมย่อยส่วนการเก็บค่าข้อมูล (SAVE)

เป็นส่วนที่ควบคุมเกทต่างๆให้อยู่ในสภาวะที่พร้อมที่ทำการเก็บค่าข้อมูลที่ได้จากการแซมปลิ้งสัญญาณโทรทัศน์ แล้วนำมาเก็บไว้ใน HIGH FREQUENCY RAM ในโปรแกรมส่วนนี้ นอกจากที่จะส่งสัญญาณควบคุมในการเปิดเกทแล้ว ยังต้องมีโปรแกรมในส่วนการนับสัญญาณสแกนแนวนอนเป็นจำนวน 256 เส้นจาก 312.5 เส้นใน 1 สแกนแนวตั้ง โดยเราใช้ ไทมเมอร์/เคาท์เตอร์ 1 โหมด 2 ที่เป็น 8 บิท ออโตรีโหลด โดยใช้ TL1 เป็นจำนวนเส้นสแกนแนวนอนช่วงต้นที่ไม่ต้องการให้มีการเก็บค่าข้อมูล และส่งข้อมูลออกมาเป็น HIGH ส่วน TH1 เป็นจำนวนเส้นสแกนแนวนอนที่ต้องการเก็บค่าข้อมูลเข้าไปในหน่วยเก็บความจำความเร็วสูง คือ มีทั้งหมด 256 เส้น และเปลี่ยนสถานะจาก HIGH เป็น LOW (สายสัญญาณ B) ถึงอย่างไรก็ดีการที่จะเริ่มนับได้นั้นจะต้องอาศัย เส้นสแกนแนวตั้งมาเป็นตัวกระตุ้นให้เกิดการนับเกิดขึ้น ดังนั้นจึงใช้ พอร์ท P3.2 เป็นตัวตรวจ

สอบว่ามีสัญญาณแนวตั้งมาหรือยัง

3.1.5.4 การแสดงผล (DISPLAY)

มีการทำงานเช่นเดียวกับการเก็บค่าข้อมูล คือ ไปทำการเปิดเกทต่างๆ และก็นับสัญญาณสแกนแนวนอนให้ได้ 256 เส้นเช่นเดียวกัน แล้วทำการแสดงผล โดยใช้สัญญาณสแกนแนวนอน และ แนวตั้งจาก ในแบบการสแกนแบบก้าวหน้า จาก 6845 ที่ได้ทำการโปรแกรมไว้แล้ว

3.1.5.5 การส่งข้อมูลออกไปในรูปแบบอนุกรม (SENT SERIAL)

เหตุผลที่สำคัญที่ทำให้เราเลือก 8031 เป็นตัวควบคุมนั้นนอกมาจากเรื่องที่ใช้อุปกรณ์ในการทำงานร่วมน้อย และ การต่อพอร์ทไปควบคุมนั้นไม่จำเป็นต้องใช้ 8255 ดังเช่น Z80 แล้วนั้นเหตุผลที่สำคัญอีกอย่างหนึ่ง คือ 8031 มีพอร์ททีทำหน้าที่เป็น พอร์ทที่ส่งข้อมูลออกไปตรง และมีโหมดให้เลือกมากมายไม่ว่าจะเป็นในรูปแบบที่มีการกำหนดไว้แน่นอน หรือ เปลี่ยนค่าความถี่ในการส่งได้ หรือ จะกำหนดให้มีพาราตี้บิท (PARITY BIT) สตาร์ทบิท (START BIT) หรือ สต๊อปบิท (STOP BIT) ได้ จะเห็นว่ามีประโยชน์อย่างมากมาย

โดยการโปรแกรมของเรานั้นเลือกทำในโหมด 1 คือ มี 10 บิท ซึ่งประกอบด้วย สตาร์ทบิท สต๊อปบิท และ ดาต้า 8 บิท สามารถที่จะกำหนดค่าความถี่ในการส่งค่าข้อมูลแบบอนุกรมได้ โดยโปรแกรมที่เราเขียนเราแบ่งเป็นในส่วนที่รับข้อมูลจากภายนอกเพื่อมากำหนดค่าความถี่ในการส่งแบบอนุกรม กับ ส่วนที่เรากำหนดค่าไว้แน่นอนเลย

3.1.5.6 การรับข้อมูลเข้ามาในรูปแบบอนุกรม (RECIEVE SERIAL)

ก็เป็นเช่นเดียวกับการส่ง แต่เปลี่ยนเป็นการรับเท่านั้น ยังอยู่ในโหมดเดียวกันกับการส่ง โปรแกรมมีลักษณะที่คล้ายกัน

สำหรับโปรแกรมรวมทั้งหมดมีดังนี้

```
0000                            CPU "8051.TBL"
0000                            HOF "BIN8"

0000                            ORG 0000H       ;RESET
0000 0144                       AJMP RESET
0013                            ORG 0013H       ;INT1 FOR KEY
0013 0135                       AJMP ASK
001B                            ORG 001BH       ;TIMER/COUNTER1 FOR SAVE,DISPLAY
001B 112B                       ACALL INTF
001D 32                         RETI
0023                            ORG 0023H       ;RI+TI FOR SERIAL
0023 A3                         INC DPTR
0024 C299                       CLR SCN1
0026 C298                       CLR SCN0
0028 04                         INC A
0029 32                         RETI
002B                            ORG 002BH
002B 7004       INTF:           JNZ TF_1
002D C291                       CLR P11
002F 04                         INC A
0030 32                         RETI
0031 D291       TF_1:           SETB P11
0033 E4                         CLR A
0034 32                         RETI
0035 D0F0       ASK:            POP B
0037 D0F0                       POP B
0039 75F000                     MOV B,#00H
003C C0F0                       PUSH B
003E 75F0A7                     MOV B,#0A7H
0041 C0F0                       PUSH B
0043 32                         RETI
0044 00         RESET:          NOP
0045 00                         NOP
0046 00                         NOP
0047 00                         NOP
0048 00                         NOP
0049 00                         NOP
004A 00                         NOP
004B 00                         NOP
004C 00                         NOP
004D 00                         NOP
004E 758120                     MOV SP,#20H
0051 75D000                     MOV PSW,#00H
0054 75A884                     MOV IE,#84H
0057 758805                     MOV TCON,#05H
005A 75B805                     MOV IP,#05
005D 759900                     MOV SBUF,#00H
0060 75907B                     MOV P1,#7BH
0063 7900                       MOV R1,#00H     ; RS = LOW FOR WRITE ADDRESS
                                                ; OF INTERNAL RAM
0065 7801                       MOV R0,#01H     ; RS = HIGH FOR WRITE DATA
                                                ; INTO INTERNAL RAM
0067 7400                       MOV A,#00H      ; RS0
0069 F3                         MOVX @R1,A
006A 749F                       MOV A,#9FH
006C F2                         MOVX @R0,A
006D 7401                       MOV A,#01H      ;RS1
006F F3                         MOVX @R1,A
```

```
0070 748B                       MOV A,#8BH
0072 F2                         MOVX @R0,A
0073 7402                       MOV A,#02H      ;RS2
0075 F3                         MOVX @R1,A
0076 7492                       MOV A,#92H
0078 F2                         MOVX @R0,A
0079 7403                       MOV A,#03H      ;RS3
007B F3                         MOVX @R1,A
007C 740B                       MOV A,#0BH
007E F2                         MOVX @R0,A
007F 7404                       MOV A,#04H      ;RS4
0081 F3                         MOVX @R1,A
0082 7426                       MOV A,#26H
0084 F2                         MOVX @R0,A
0085 7405                       MOV A,#05H      ;RS5
0087 F3                         MOVX @R1,A
0088 7400                       MOV A,#00H
008A F2                         MOVX @R0,A
008B 7406                       MOV A,#06H      ;RS6
008D F3                         MOVX @R1,A
008E 7422                       MOV A,#22H
0090 F2                         MOVX @R0,A
0091 7407                       MOV A,#07H      ;RS7
0093 F3                         MOVX @R1,A
0094 7423                       MOV A,#23H
0096 F2                         MOVX @R0,A
0097 7408                       MOV A,#08H      ;RS8
0099 F3                         MOVX @R1,A
009A 7400                       MOV A,#00H
009C F2                         MOVX @R0,A
009D 7409                       MOV A,#09H      ;RS9
009F F3                         MOVX @R1,A
00A0 7407                       MOV A,#07H
00A2 F2                         MOVX @R0,A
00A3 7590FF                     MOV P1,#0FFH
00A7                            ORG 00A7H
00A7 7590FF     KEY:            MOV P1,#0FFH
00AA E590       LOOP1:          MOV A,P1
00AC 64FF                       XRL A,#0FFH
00AE 60FA                       JZ LOOP1
00B0 5401                       ANL A,#01H
00B2 7024                       JNZ G00
00B4 5421                       ANL A,#21H
00B6 7022                       JNZ G01
00B8 5402                       ANL A,#02H
00BA 7020                       JNZ G02
00BC 5422                       ANL A,#22H
00BE 701E                       JNZ G03
00C0 5404                       ANL A,#04H
00C2 701C                       JNZ G04
00C4 5424                       ANL A,#24H
00C6 701A                       JNZ G05
00C8 5408                       ANL A,#08H
00CA 7018                       JNZ G06
00CC 5428                       ANL A,#28H
00CE 7016                       JNZ G07
00D0 5410                       ANL A,#10H
00D2 7014                       JNZ G08
00D4 5440                       ANL A,#40H
00D6 7012                       JNZ G09
00D8 410E       G00:            AJMP DISP_68
00DA 4103       G01:            AJMP DISP_CA
00DC 21DA       G02:            AJMP SAVE_CA
00DE 01EC       G03:            AJMP SAVE_SOMETHING
00E0 212D       G04:            AJMP SENT_F
00E2 01F2       G05:            AJMP SENT_V
```

```
00E4 21A0      GO6:            AJMP RECIEVE_F
00E6 2165      GO7:            AJMP RECIEVE_V
00E8 01EE      GO8:            AJMP RES1
00EA 01F0      GO9:            AJMP RES2
00EC 80B9      SAVE_SOMETHING: SJMP KEY
00EE 80B7      RES1:           SJMP KEY
00F0 80B5      RES2:           SJMP KEY



00F2 3092FD    SENT_V:         JNB P12,SENT_V
00F5 3095FD    LS2:            JNB P15,LS2
00F8 75A894    VARS:           MOV IE,#94H
00FB 758928                    MOV TMOD,#28H
00FE 758845                    MOV TCON,#45H
0101 75907E                    MOV P1,#7EH
0104 75D000                    MOV PSW,#00H
0107 79FF                      MOV R1,#0FFH
0109 2095FD    SS1:            JB P15,SS1      ; WAIT FOR YOU TOUCH KEY P1.5
                                               ; TO INPUT DATA TO SMOD IN
010C 3095FD    TS1:            JNB P15,TS1     ; PCON REGISTER
010F 7580FF                    MOV P0,#0FFH
0112 E3                        MOVX A,@R1
0113 5480                      ANL A,#80H
0115 F587                      MOV PCON,A
0117 79FF                      MOV R1,#0FFH
0119 75D000                    MOV PSW,#00H
011C 2095FD    SS2:            JB P15,SS2      ; WAIT FOR YOU TOUCH KEY P1.5
                                               ; TO INPUT DATA TO TH1
011F 3095FD    TS2:            JNB P15,TS2
0122 7580FF                    MOV P0,#0FFH
0125 E3                        MOVX A,@R1
0126 758BFF                    MOV TL1,#0FFH
0129 F58D                      MOV TH1,A
012B 8012                      SJMP REL
012D 3092FD    SENT_F:         JNB P12,SENT_F
0130 75A894                    MOV IE,#94H
0133 758928                    MOV TMOD,#28H
0136 758845                    MOV TCON,#45H
0139 758DFE                    MOV TH1,#0FEH
013C 758780                    MOV PCON,#80H
013F 759860    REL:            MOV SCON,#60H
0142 759080                    MOV P1,#80H
0145 758200                    MOV DPL,#00H
0148 758300                    MOV DPH,#00H
014B E0        SS3:            MOVX A,@DPTR
014C F599                      MOV SBUF,A
014E E4                        CLR A
014F 60FE      SS4:            JZ SS4          ; WAIT FOR INTETRRUPT FROM
0151 E582                      MOV A,DPL       ;                 RI OR TI
0153 70F6                      JNZ SS3
0155 E583                      MOV A,DPH       ;     TO CHECK IF DPTR = H0000
0157 70F2                      JNZ SS3         ;     THEN INCREMENT P1
0159 0590                      INC P1          ;
015B E590                      MOV A,P1        ;     TO CHECK IF ADDRESSING
015D 6402                      XRL A,#02H      ;     EXTERNAL MEMORY  MAXIMUM
015F 70EA                      JNZ SS3         ;     CONTAIN PLUS 1 THEN ACALL
0161 C2AC                      CLR IE4         ;     DISPLAY (MEMORY FULL)
0163 01A7                      AJMP KEY

0165 3093FD    RECIEVE_V:      JNB P13,RECIEVE_V
0168 3095FD    LR2:            JNB P15,LR2
016B 75A894    VARR:           MOV IE,#94H
016E 758928                    MOV TMOD,#28H
0171 758845                    MOV TCON,#45H
0174 75907E                    MOV P1,#7EH
```

```
0177 75D000                MOV PSW,#00H
017A 79FF                  MOV R1,#0FFH
017C 2095FD    SR1:        JB P15,SR1      ; WAIT FOR YOU TOUCH KEY P1.5
                                           ; TO INPUT DATA TO SMOD IN
017F 3095FD    TSR1:       JNB P15,TSR1    ; PCON REGISTER
0182 7580FF                MOV P0,#0FFH
0185 E3                    MOVX A,@R1
0186 5480                  ANL A,#80H
0188 F587                  MOV PCON,A
018A 75D000                MOV PSW,#00H
018D 79FF                  MOV R1,#0FFH
018F 2095FD    SR2:        JB P15,SR2      ; WAIT FOR YOU TOUCH KEY P1.5
                                           ; TO INPUT DATA TO TH1
0192 3095FD    TSR2:       JNB P15,TSR2
0195 7580FF                MOV P0,#0FFH
0198 E3                    MOVX A,@R1
0199 758BFF                MOV TL1,#0FFH
019C F58D                  MOV TH1,A
019E 8012                  SJMP RELR
01A0 3093FD    RECIEVE_F:  JNB P13,RECIEVE_F
01A3 75A894                MOV IE,#94H
01A6 758928                MOV TMOD,#28H
01A9 758845                MOV TCON,#45H
01AC 758DFE                MOV TH1,#0FEH
01AF 758780                MOV PCON,#80H
01B2 759870    RELR:       MOV SCON,#70H
01B5 759080                MOV P1,#80H
01B8 7582FF                MOV DPL,#0FFH
01BB 7583FF                MOV DPH,#0FFH
01BE E4        SR3:        CLR A
01BF 60FE      SR4:        JZ SR4          ; WAIT FOR RI INTERRUPT
01C1 E599                  MOV A,SBUF
01C3 F0                    MOVX @DPTR,A
01C4 E582                  MOV A,DPL
01C6 04                    INC A
01C7 70F5                  JNZ SR3
01C9 E583                  MOV A,DPH
01CB 04                    INC A
01CC 70F0                  JNZ SR3
01CE 0590                  INC P1
01D0 E590                  MOV A,P1
01D2 6402                  XRL A,#02H
01D4 70E8                  JNZ SR3
01D6 C2AC                  CLR IE4
01D8 410E                  AJMP DISP_68

01DA 3091FD    SAVE_CA:    JNB P11,SAVE_CA
01DD 758968                MOV TMOD,#68H
01E0 758805                MOV TCON,#05H
01E3 758BE0                MOV TL1,#0E0H
01E6 758D00                MOV TH1,#00H
01E9 75905F                MOV P1,#5FH
01EC E4                    CLR A
01ED 75A88C                MOV IE,#8CH
01F0 30B2FD    X1:         JNB P32,X1      ; WAIT FOR INTO THAT IS VER.SYNC
01F3 20B2FD    X9:         JB P32,X9
01F6 D28E                  SETB TCN6
01F8 E4                    CLR A
01F9 60FE      X2:         JZ X2           ; WAIT FOR COUNT 16 LINES FIRST
01FB 70FE      X3:         JNZ X3          ; WAIT FOR COUNT 256 LINES THAT
01FD C28E                  CLR TCN6        ; WANT TO SAVE
01FF C2AB                  CLR IE3
0201 410E                  AJMP DISP_68


0203 3090FD    DISP CA:    JNB P10,DISP CA
```

```
0206 3095FD    LD2:        JNB P15,LD2
0209 75905F                MOV P1,#5FH    ; H,V SYNC FROM CAMARA
020C 8006                  SJMP D2
020E 3090FD    DISP_68:    JNB P10,DISP_68
0211 759053                MOV P1,#53H    ; H,V SYNC FROM 6845
0214 758968    D2:         MOV TMOD,#68H ; INITIAL FOR THE THAT COUNTER
0217 758805                MOV TCON,#05H ; MUST SAVE
021A 758BE0                MOV TL1,#0E0H
021D 758D00                MOV TH1,#00H
0220 E4                    CLR A
0221 75A88C    D3:         MOV IE,#8CH
0224 20B2FD    D4:         JB P32,D4      ; WAIT FOR INTO THAT IS VER.SYNC
0227 30B2FD    D9:         JNB P32,D9
022A D28E                  SETB TCN6
022C E4                    CLR A
022D 60FE      D5:         JZ D5          ; WAIT FOR COUNT 16 LINES FIRST
022F 70FE      D6:         JNZ D6         ; WAIT FOR COUNT 256 LINES THAT
0231 C28E                  CLR TCN6       ; WANT TO DISPLAY
0233 C2AB                  CLR IE3
0235 4114                  AJMP D2
```

3.1.6 ภาคการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

คือ ขบวนการแปลงจากสัญญาณดิจิตอลเป็นสัญญาณอนาลอก ซึ่งเป็นขบวนการย้อนกลับกับ A/D และ คำต่างๆที่ใช้ก็มีความหมายเหมือนกัน

วงจร D/A จะประกอบไปด้วย

1 วงจรตรรกะ (LOGIC CIRCUIT) เพื่อใช้รับสัญญาณดิจิตอล

2 คักดาอ้างอิง เพื่อใช้เป็นส่วนจ่ายคักดาให้แก่ เอาท์พุท ซึ่งต้องเลือกค่าว่าต้องการให้เอาท์พุทมีคักดาอยู่ช่วงใด , ขั้วบวกหรือขั้วลบ ,ค่าทิศทางกระแสของเอาท์พุท

3 อนาล็อกสวิทช์ (ANALOG SWITCH) และ วงจรขับ (DRIVE CIRCUIT)

4 วงจรความต้านทาน (RESISTER NETWORK)

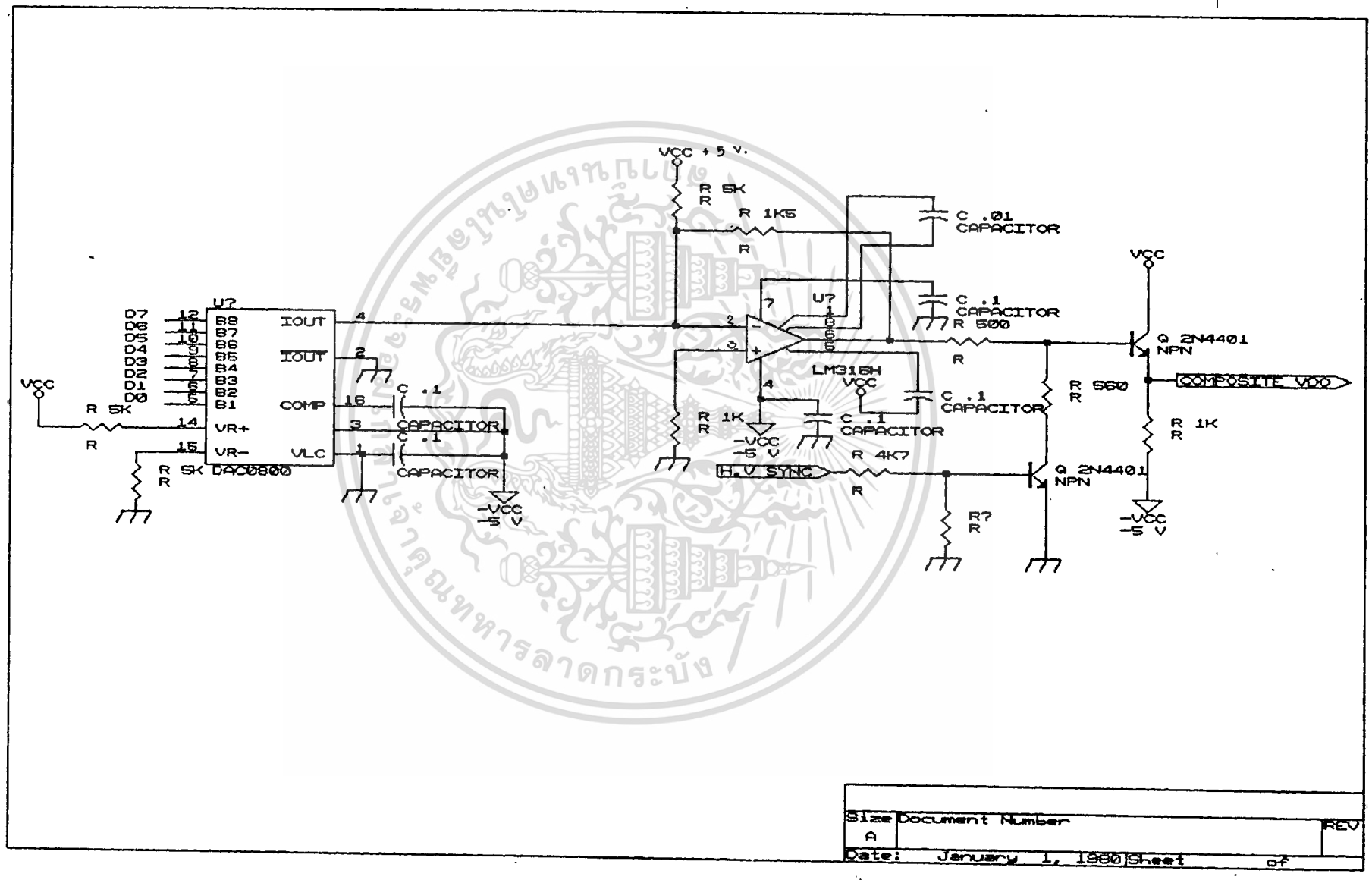
ซึ่งสามารถดูวงจรได้อย่างละเอียดจาก LINEAR DATA BOOK ซึ่งใช้ไอซีเบอร์ DAC 0800

5 วงจรเปลี่ยนกระแสเป็นโวลเตจ โดยใช้ไอซี LM 318

6 วงจรรวมสัญญาณซิงค์แนวนอนและแนวตั้ง โดยใช้ ทรานซิสเตอร์ 2N4401

หลังจากที่ข้อมูลทั้ง 8 บิท ผ่านเข้ามาใน DAC 0800 แล้วมันจะทำการเปลี่ยนสัญญาณดิจิตอลเป็นในรูปกระแสดึงเข้ามาทางขา 4 โดยถ้าค่าทางดิจิตอลมาก สมมุติเป็น 11111111 นั้นมันจะดึงกระแสเข้าตัวมันเองมากที่สุด แล้วน้อยลงตามลำดับ น้อยที่สุดก็คือ 00000000 ในส่วนของไอซี LM 318 นั้น ตัวความต้านทานขนาด 5 K นั้น ทำหน้าที่เป็นเสมือนเป็นค่ากระแสที่จ่ายคงที่ ที่นี้เรามาทำการพิจารณาค่ากระแสที่ไหลผ่านค่าความต้านทานขนาด 1K5 นั้น จะเห็นว่าค่ากระแสจะเปลี่ยนไปตามค่ากระแสที่ดึงโดยไอซี DAC 0800 ฉะนั้นกระแสที่ไหลผ่านตัวต้านทานที่ต่อคร่อมขา 2 กับ 6 ก็จะแปลงเป็นโวลเตจโดยมีความสัมพันธ์เป็นในทางตรงกันข้ามคือ ถ้ามีการดึงกระแสมากนั้น (11111111) โวลเตจที่ออกมาทางเอาท์พุทก็จะมาก ถ้ามีการดึงกระแสน้อยก็จะมีโวลเตจที่เอาท์พุทน้อย

เมื่อทำการกลับกระแสเป็นโวลเตจแล้ว ก็จะเข้าสู่วงจรรวมสัญ าณสแกนแนวนอนและสัญญาณสแกนแนวตั้ง โดยสัญญาณดังกล่าวได้มาจากสัญญาณที่ผลิตโดย ไอซีเบอร์ 6845 เมื่อรวมกับสัญญาณที่ได้มาจากไอซีเบอร์ LM 318 ก็จะเป็นสัญญาณรวม (COMPOSITE SIGNAL) ที่เข้าจอโทรทัศน์



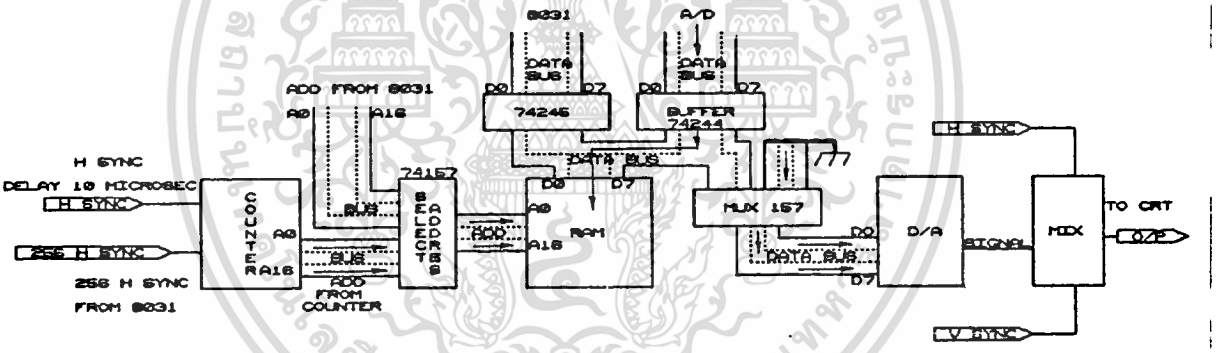
รูป แสดงวงจรการแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

3.2 อธิบายถึงการรวมวงจรย่อยเหล่านั้นเป็นวงจรที่สมบูรณ์แบบ

หลังจากที่เราได้อธิบายการทำงานเป็นส่วนย่อยแล้ว ในส่วนนี้จะทำการอธิบายเป็นส่วนใหญ่เป็นการมองภาพโดยกว้างๆเพื่อที่จะเสริมความเข้าใจแบ่งออกเป็น 4 ส่วนใหญ่ๆ

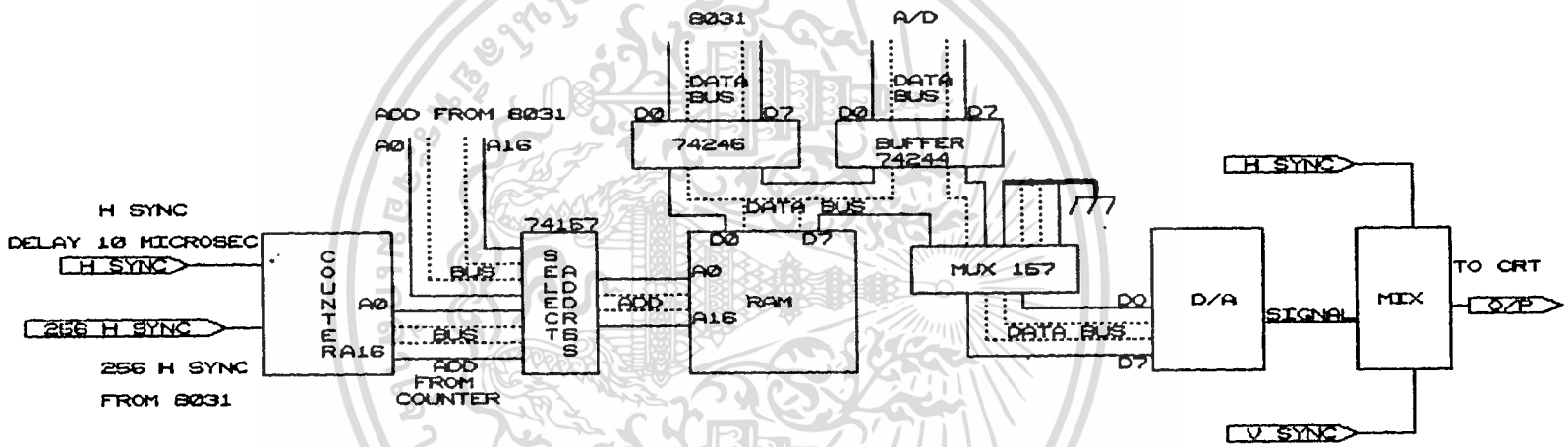
โดยมีรูปรวมดังในรูป 3.11 สำหรับในแต่ละกรรมวิธีนั้น ก็จะมีรูปของตนเอง และ แสดงทางเดินของข้อมูลตามลูกศร ในส่วนที่ไม่มีลูกศรนั้นหมายความว่าอุปกรณ์นั้นไม่มีผลต่อกรรมวิธีแต่ละวิธีที่แตกต่างกัน

การเก็บค่าข้อมูลที่เกิดจากการแซมปลิ้งในการแปลงอนาลอกเป็นดิจิตอล



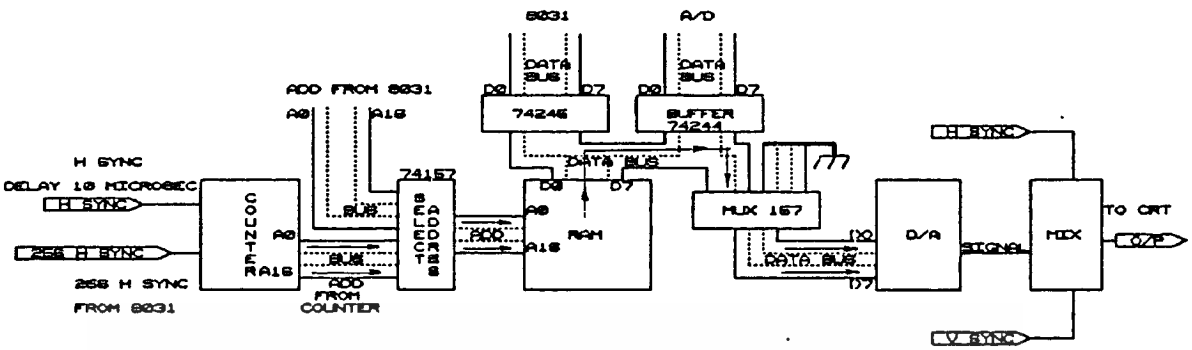
รูป 3.12 การเก็บค่าข้อมูลที่เกิดจาก A/D

การเก็บค่าข้อมูลนั้น 8031 จะทำการควบคุมให้ 74245 อยู่ในสถานะไออิมพีแดนท์ (HIGH IMPEDANCE) ให้ 74157 ทำการเลือกค่าแอดเดรสที่มาจาก วงจรนับ ให้ บัฟเฟอร์เปิดเกทให้ดาต้าออกมาจากวงจรการแปลงอนาลอกเป็นดิจิตอลได้ แล้วไหลเข้ามายังหน่วยความจำได้ นอกจากนี้ยังให้เลือก MUX 74157 ทำการเลือกดาต้า 00 ออกไปให้ วงจรการแปลงสัญญาณดิจิตอลเป็นอนาลอก นอกจากนี้ยังมีการส่งสายควบคุม B และ G จาก 8031 ซึ่งเป็นการนับ 256 เส้นสแกนในแนวนอนกับ สัญญาณสแกนในแนวนอน ที่ได้รับการหน่วงออกไปแล้ว 10 ไมโครวินาที ที่มาควบคุมในวงจรนับ



Size	Document Number	REV
A		
Date: January 1, 1980	Sheet of	

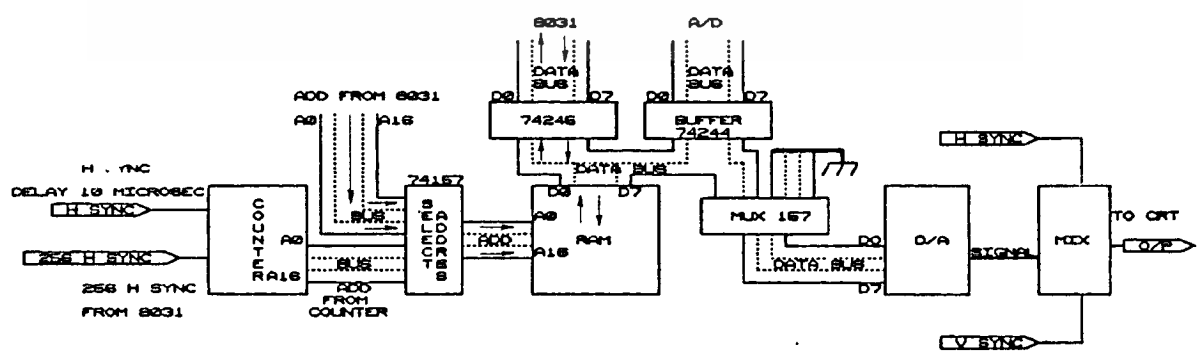
การแสดงผล (DISPLAY)



รูป 3.13 การแสดงผล

เป็นการแสดงภาพที่ได้เก็บในไว้ในหน่วยความจำที่ได้มาจากขบวนการการเก็บค่าข้อมูลจากวงจรการแปลงสัญญาณอนาลอกเป็นดิจิตอล หรือ จากขบวนการรับข้อมูลแบบอนุกรม มาโชว์ออกที่จอโมโนโครม โดยควบคุมให้ 74245 และ 74244 อยู่ในสถานะไอโอมีแดนท์ และ ให้ 74157 ทำการเลือกแอดเดรสที่มาจากวงจรนับ นอกจากนี้ยังต้องให้ MUX 74157 อยู่ในสถานะที่เลือกที่ดาต้าที่มาจากหน่วยความจำ ถ้าหากวงจรนับยังคงทำงานอยู่ แต่ถ้าหากวงจรนับ 74393 ตัวที่ 1 อยู่ในสถานะ CLEAR ก็ให้ MUX 74157 เลือกดาต้า 00 ออกไปยัง วงจรการแปลงดิจิตองเป็นอนาลอก

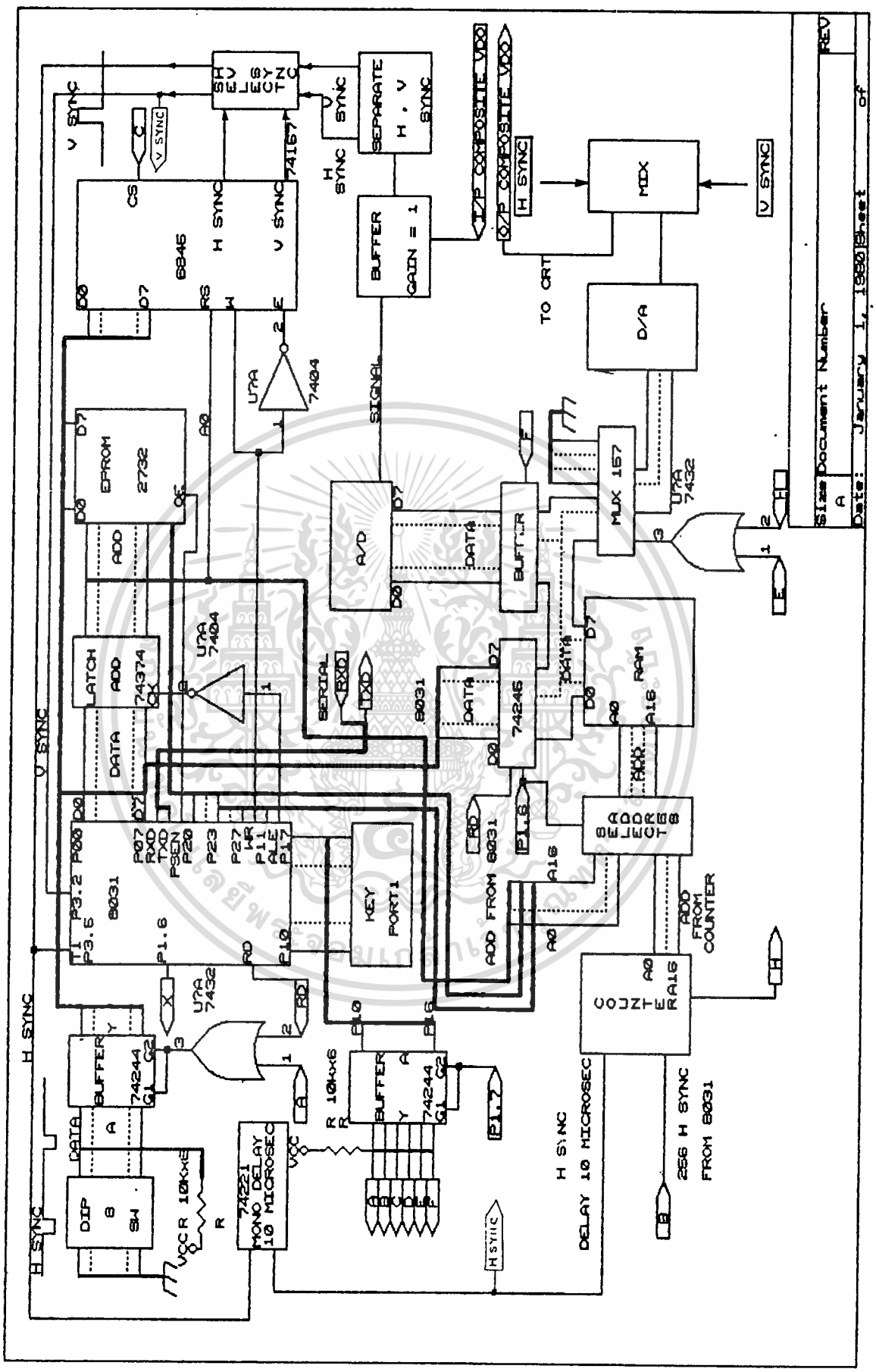
การส่งและการรับข้อมูลแบบอนุกรม

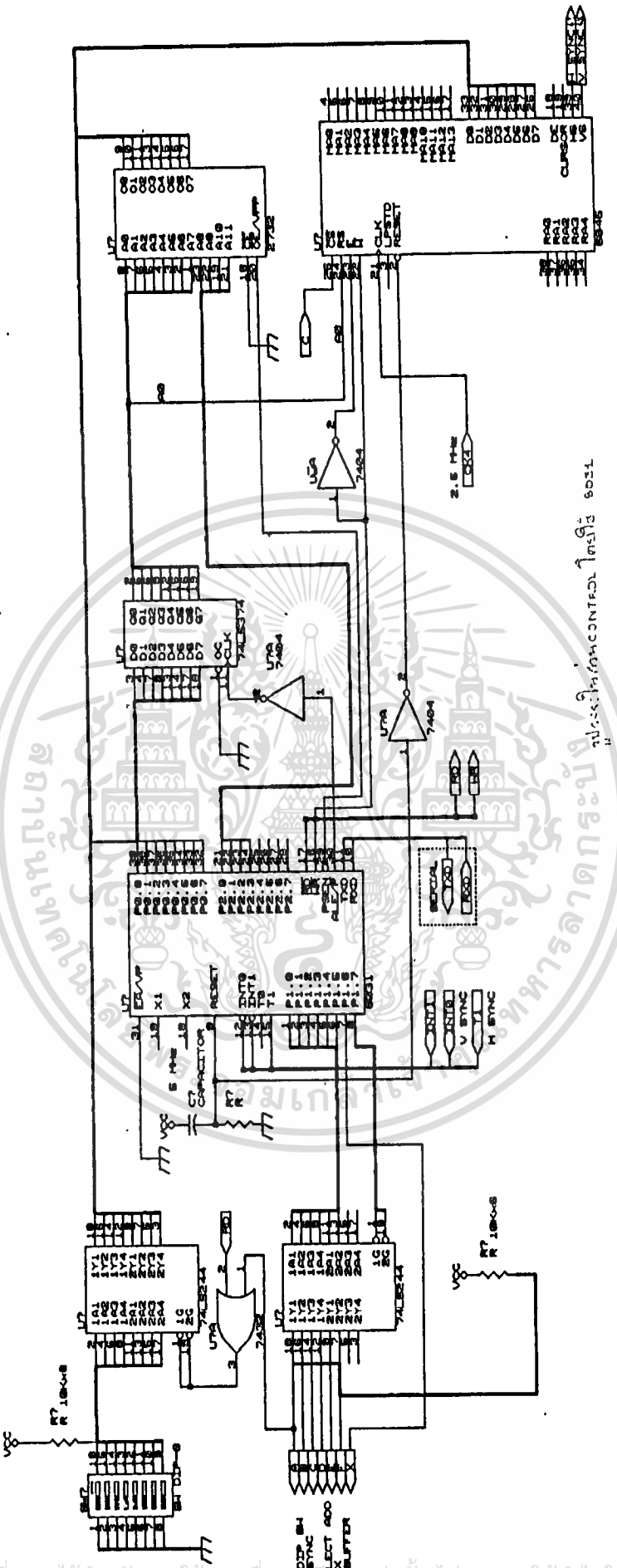


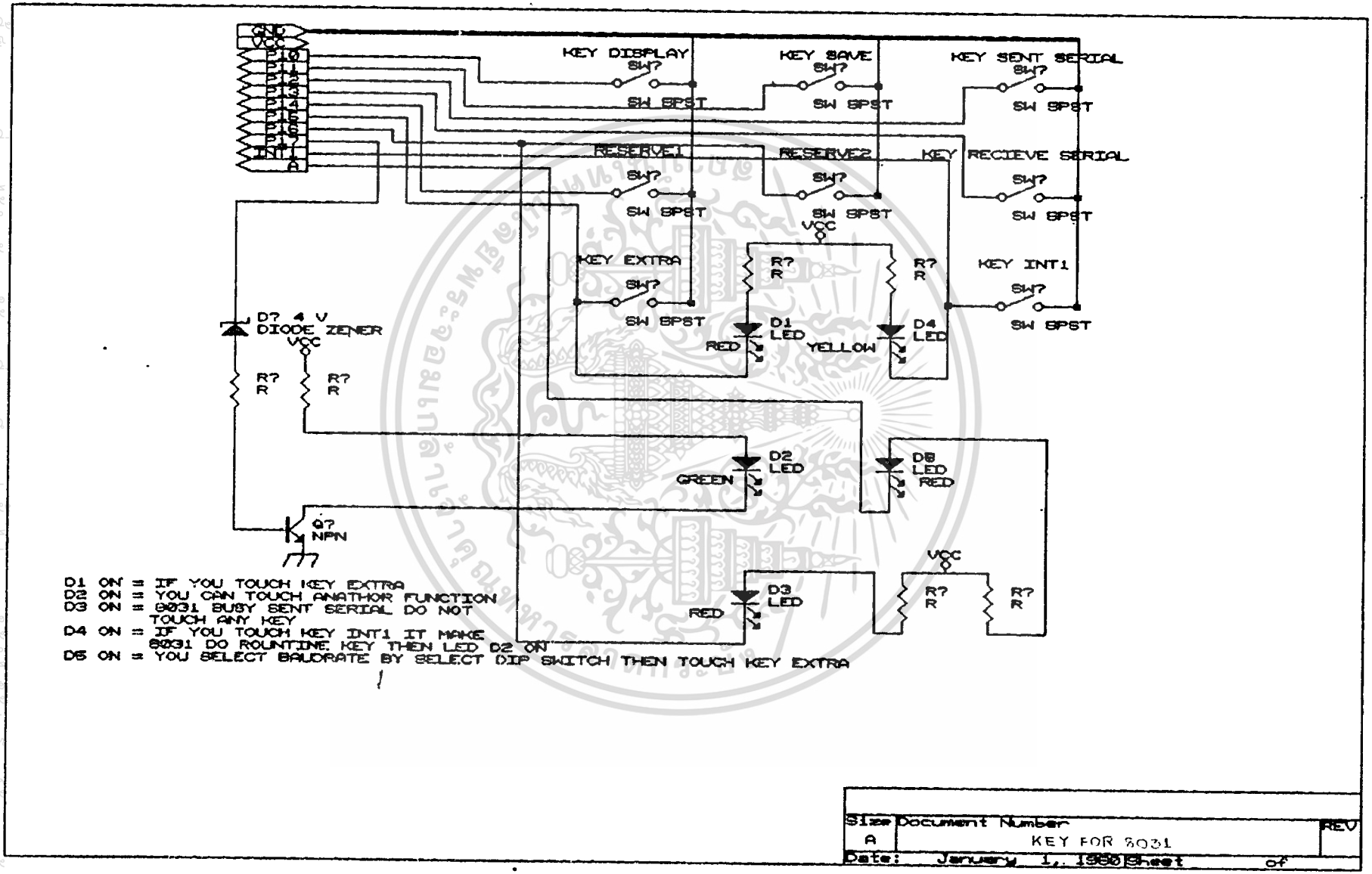
รูป 3.14 การส่งและการรับข้อมูลแบบอนุกรม

หลังจากที่ได้รับ การกดปุ่ม ให้ทำในส่วนการรับและส่งข้อมูลแบบอนุกรมแล้ว ในส่วนควบคุมของ 8031 ก็ควบคุมให้ บัฟเฟอร์ อยู่ในสถานะไอโอมพีแดนซ์ MUX 74157 เลือกดาต้า 00 ออกไปยังวงจรการแปลงดิจิตอลเป็นอนาลอก และ ให้ 74157 ทำการเลือกแอดเดรสที่มาจากแอดเดรสที่มาจาก 8031

สำหรับรูปวงจรรวมการทำงานทั้งหมดได้แสดงไว้ในรูป 3.15






KEY DISPLAY
SW?
SW SPST
KEY SAVE
SW?
SW SPST
KEY SENT SERIAL
SW?
SW SPST
RESERVE1
SW?
SW SPST
RESERVE2
SW?
SW SPST
KEY RECIEVE SERIAL
SW?
SW SPST
VCC
KEY EXTRA
SW?
SW SPST
R?
R
R?
R
KEY INT1
SW?
SW SPST
D1
LED
RED
D4
LED
YELLOW
D7 4 V
DIODE ZENER
VCC
R?
R
R?
R
D2
LED
GREEN
D8
LED
RED
Q?
NPN
VCC
D3
LED
RED
R?
R
R?
R
D1 ON = IF YOU TOUCH KEY EXTRA
D2 ON = YOU CAN TOUCH ANATHOR FUNCTION
D3 ON = 8031 BUSY SENT SERIAL DO NOT
TOUCH ANY KEY
D4 ON = IF YOU TOUCH KEY INT1 IT MAKE
8031 DO ROUNTINE KEY THEN LED D2 ON
D5 ON = YOU SELECT BAUDRATE BY SELECT DIP SWITCH THEN TOUCH KEY EXTRA
Size
A
Document Number
KEY FOR 8031
REV
Date: January 1, 1990 Sheet of

บทที่ 4

การทดลองและผลการทดลอง

จากบทที่แล้วจะเห็นว่าเราสามารถที่จะแบ่งวงจรออกได้เป็น 6 ส่วนด้วยกัน นั้น จะขออธิบายการทดลองออกเป็นส่วนดังนี้

4.1 ภาคแยกสัญญาณสแกนในแนวนอนและสัญญาณสแกนในแนวตั้ง

สำหรับภาคนี้นับว่าง่ายที่สุดสามารถทำความเข้าใจในวงจรได้ง่ายที่สุด ไม่ค่อยมีปัญหาในการต่อ ในกรณีที่ต้องการให้สัญญาณออกมาสวยเราอาจทำการต่อ 7404 เข้าไปอีกทีหนึ่งก็ได้จะทำให้สัญญาณที่ได้สวยเป็นเหลี่ยมเป็นมุมมากขึ้นได้

4.2 ภาคการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

สำหรับภาคนี้เป็นการต่อภาคสุดท้าย เพราะเป็นภาคที่เสี่ยงมากที่สุด เนื่องมาจากไอซีที่จะทำการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลนั้นมีราคาแพงถึง 2,250 บาท ทำการต่อต้องทำด้วยความรอบคอบและพิจารณาวงจรเป็นอย่างดี โดยเรื่องการใส่ขั้วพายที่ผิดขั้วนั้นอันตรายมาก และ ในส่วนของสัญญาณเข้าจะต้องไม่เกิน 6 โวลต์ ฉะนั้นควรจะใส่ซีเนอร์ไดโอด ประมาณ 3 โวลต์ ต่อคร่อมที่ขาอินพุท และ กราวด์ ในส่วนนี้การแปลงสัญญาณออกมาเป็นภาพเป็นไปด้วยดี

สำหรับในกรณีที่ต้องตัดสัญญาณรบกวนออกนั้น เราสามารถทำการแยกไฟเลี้ยงในส่วนของดิจิตอลได้ โดยเด็ดขาด ทำให้สามารถตัดการรบกวนออกไปได้

4.3 ภาควงจรนับ

ในส่วนนี้ได้ทำการออกแบบและทำการทดลองไปได้แล้วตั้งแต่เทอมที่แล้ว สำคัญมากเรื่องเวลา คือ จะต้องทำการนับให้ได้ 512 ครั้ง ภายในเวลา 52 ไมโครวินาที จากทดลองนั้นได้ให้ฐานเวลาเข้าค่า ซึ่งถ้าน้อยกว่าเวลาในการนับ 512 ครั้งนั้น วงจรจะหยุดทำการนับก่อนครบ 512 ต่อจากนั้นเพิ่มฐานเวลาให้อีกโดยมากกว่าเวลาที่นับให้ครบ 512 ครั้ง ปรากฏว่าไม่ว่าเพิ่มฐานเวลาไปเท่าไร ความกว้างของสัญญาณที่ใช้

การนับก็เท่าเดิม สรุปได้ว่าในส่วนของวงจรนับสามารถทำงานได้

4.4 ภาคหน่วยเก็บความจำความถี่สูง

ในส่วนที่ทำในเทอมแรกนั้น ในการต่อทดลองประสบปัญหา คือ ค่าที่ทำการ แลทช์ ค่าแอดเดรสในแรมตัวที่ 2 นั้นไม่สามารถทำการ แลทช์ ค่าข้อมูล ทำการขบคิดเป็นเวลานาน ปัญหาที่เกิดนั้นเป็นเหตุมาจากเราให้ขา CK กับ 74374 ซึ่งใช้ขอบขาขึ้นในการ แลทช์ ค่าข้อมูล แล้วเราให้ CK ในขณะที่ค่าข้อมูลทางด้านอินพุทยังมีการเปลี่ยนแปลง ทำให้ 74374 ไม่ทราบว่าข้อมูลใดควรจะทำการ แลทช์ ฉะนั้นควรจะทำการหน่วงเวลาขอบขาขึ้นของสัญญาณสนาฬิกาออกไปเล็กน้อย โดยใช้ 7404 2 ตัวต่อกัน

จากปัญหาในเทอมแรกทำให้เป็นประสบการณ์ในการออกแบบวงจรในเทอมที่ 2 โดยในตอนแรกใช้ 74221 ซึ่งเป็นโมโนสเตเบิล ทำการสร้างสัญญาณเขียนไปยังแรมนั้น โดยมีเวลาประมาณ 200 นาโนวินาที แต่เนื่องจากอาจารย์ วันชัย ริ้วรุจาได้ทำการแนะนำว่า เนื่องจากไอซี 74221 นั้น ใช้ R, C ในการสร้างฐานเวลา ถ้าหากอุณหภูมิสูงประกอบกับใช้งานความถี่สูง อาจทำให้ค่าฐานเวลาเปลี่ยนแปลงได้ถึง 100 เปอร์เซนต์ จึงจำเป็นต้องทำการเปลี่ยนแปลงวงจรดังที่เห็น

สำคัญที่สุดสำหรับวงจรนี้ คือ การที่ต้องระวังสัญญาณเอาท์พุทเกิดการช็อคถึง ซึ่งอาจทำให้ แรม พังได้

4.5 ภาคควบคุมและรับคำสั่ง

นับว่าเป็นภาคที่ยากที่สุด และ ประสบปัญหามากที่สุด เนื่องผมเสียเวลาประมาณ 1 เดือนกว่าในการเตรียมเครื่องมือที่จะใช้ในการทดลองการทำงาน 8031 อาทิ เช่น แรม 2 ทาง แรมแพ็ค แรมแพ็คที่สามารถเปลี่ยนแปลงไปมาได้ระหว่าง 6116 กับ 6264 นอกจากนี้ยังต้องทำการเรียนรู้คอมไพล์เรอร์อีกหลายตัว เพื่อที่สามารถที่จะแปลงโค๊ดต่างๆให้เป็นภาษาเครื่อง เพื่อที่ใช้งานกับซิมมูเลเตอร์ เพื่อที่จะทำการก๊อปปี้โปรแกรมเข้าอีพรอม ซึ่งในส่วนที่กล่าวมานั้นทางคณะมิให้น้อยทำให้เสียเวลานานมาก

ในตอนแรกได้ทำการทดลองโปรแกรมทั้งหมดปรากฏไม่ทำงาน ทำให้ตัดสินใจตรวจสอบการทำงานเป็นโปรแกรมย่อย ก็สามารถผ่านไปทั้งหมด 4 โปรแกรม ที่ขาดอยู่ คือ โปรแกรมในการรับ KEY ซึ่งตอนนี้นับว่าเป็นปัญหาใหญ่ เพราะในส่วนนี้เป็นส่วนที่ใช้ในการควบคุม ถ้าหากไม่ออกก็ไม่สามารถแสดงผล เมื่อต่อเป็นโมดูลใหญ่ได้เลย

ซึ่งก็เท่ากับผลงานที่ได้ทำมาก็เป็นศูนย์

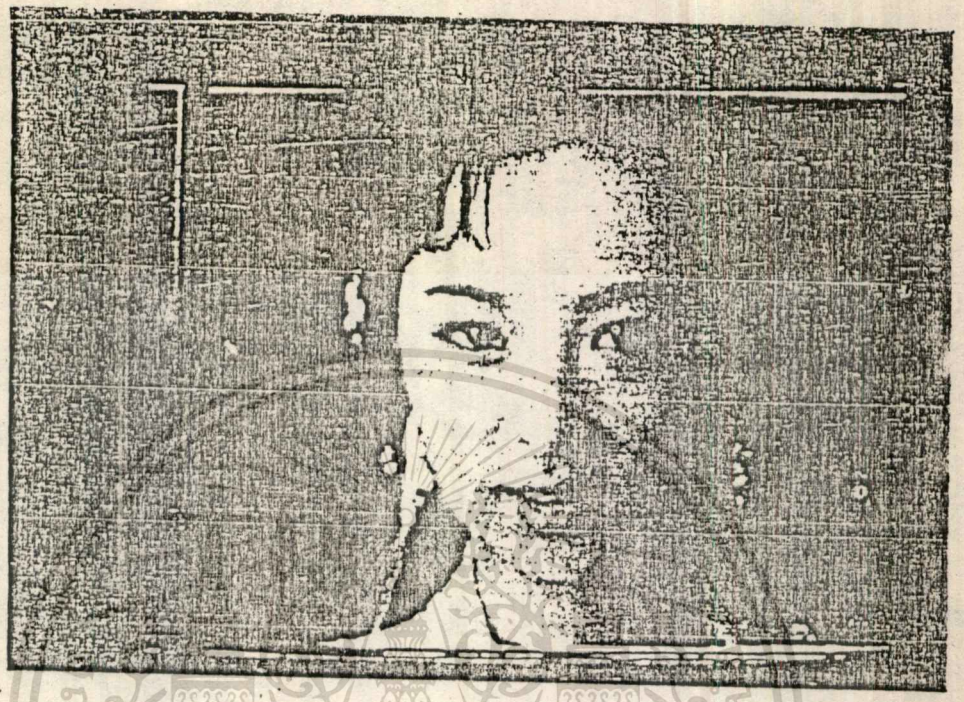
ในส่วนนี้ที่กลัวไม่ทำงานก็ คือ 6845 ในตอนแรกนั้นไม่ทำงานทำให้ต้องไปศึกษาการทำงานใหม่ก็พบว่า ขา E เราไม่ได้ต่อ โดยปกติไอซีตัวนี้เป็นไอซีในตระกูล 6800 ซึ่งมีขาให้ต่อกับขา E อยู่แล้ว ก็ได้ทำการแก้ปัญหาโดยการใช้สัญญาณเขียนจาก 8031 ต่อ 7404 เข้าขา E ปรากฏสามารถใช้งานได้ โดยให้สัญญาณนาฬิกา 2.5 M

4.6 การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก

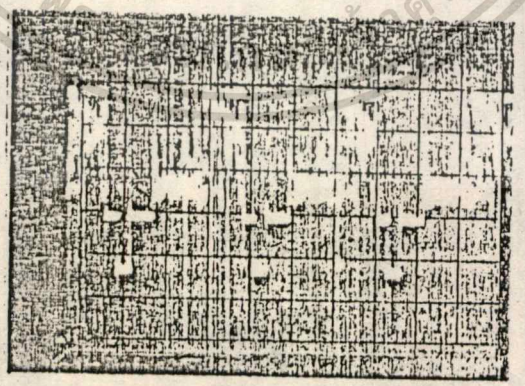
ในภาคนี้นั้นประสบปัญหาการเกิดออสซิลเลท ทำให้จำเป็นต้องใส่ C คอมเพนเซท ขนาด .01 ไมโครฟารัด แต่ก็จะมีผลเสียทำให้แบนด์วิดของ LM 318 มีค่าน้อยลง สำหรับในส่วนนี้ไม่มีปัญหาอะไรมาก คงต้องรอต่อเข้ากับระบบทั้งหมดก่อน

หลังจากที่ได้ทำการต่อวงจรทั้งหมดแล้ว มีการเปลี่ยนแปลงวงจรอีกเล็กน้อย โดยเปลี่ยนแปลงขาเอาท์พุทเอ็นอเบิ้ลของ 74245 ซึ่งเป็นบัฟเฟอร์สองทางของดาต้าที่ 8031 จะติดต่อกับไอฟิเควนซี่แรม โดยเปลี่ยนแปลงดังนี้

ลำหรับภาพที่ได้ผ่านการเก็บแล้ว ทำการแสดงออกมาทางจอคอมพิวเตอร์ ภาพที่ได้ปรากฏดังที่เห็นในภาพหน้าถัดไป



แสดงภาพที่ได้ทำการเก็บเข้าไว้ภายในแรม



แสดงสัญญาณภาพที่วัดได้จากภาพข้างบน

บทที่ 5

บทสรุปและวิจารณ์

จากการดำเนินงานมาทั้งหมดปัญหาที่ใหญ่ที่สุด ก็คือปัญหาการทำงานของ 8031 ซึ่งถ้าไม่สามารถทำงานได้ทำให้ไม่กล้าที่ทำการต่อออกเป็นระบบ เพราะจะทำให้หาที่ผิดยากทำให้การทำโครงงานนี้ล่าช้าออกไปอีก

ปัญหาในการออกแบบนั้น สำคัญมากในการเขียนตารางเวลา เพราะจะทำให้สามารถทำความเข้าใจได้ง่าย และ ควรจะมองระบบทั้งหมดให้ออกก่อนที่จะไปทำอย่างอื่นๆ สำหรับการใช้ 8031 นั้น ขอแนะนำเลยว่าควรจะเล่นเป็นอย่างมากเนื่องเหตุผลหลายอย่างที่ได้กล่าวไปแล้ว แต่จะประสบปัญหาในเรื่องอุปกรณ์สนับสนุน เพราะเป็นไอซีที่พึ่งเข้า 2-3 ปีมานี้เอง ทำให้ขาดผู้ที่รู้จริงๆ และ สามารถแนะนำอุปกรณ์สนับสนุนได้ ถึงอย่างไรก็ตามไอซีตัวนี้คงเข้ามามีบทบาทในการเรียนการสอนในอนาคตเป็นแน่

สำหรับโครงการนี้ แม้ไม่สามารถนำไปใช้งานได้มากนัก แต่ก็เป็นพื้นฐานที่ดีสำหรับผู้ที่สนใจในงานด้าน IMAGE PROCESSING และ คงมีประโยชน์ในการส่งข่าวสารที่เป็นภาพความละเอียดสูงได้

หากมีข้อผิดพลาดใดๆ ขออภัยมา ณ.ที่นี้ด้วย

ภาคผนวก

คู่มือการใช้ชิงเกิลบอร์ด 8031

สารบัญ

บทที่ 1 บทนำ

บทที่ 2 การทำงานภายใน MCS-51

บทที่ 3 โครงสร้างหน่วยความจำ, แอดเดรสซิ่งโหมดและตรรกคาลยร์

3.1 การจัดการหน่วยความจำ

3.2 แอดเดรสซิ่งโหมด

3.3 บูลีนโพรเซสเซอร์

บทที่ 4 ชุดคำสั่ง

4.1 ฟังก์ชั่นทั้งหมด

4.2 ความหมายของคำสั่ง

บทที่ 5 ฟังก์ชั่นและการใช้งานซิงเกิลบอร์ด

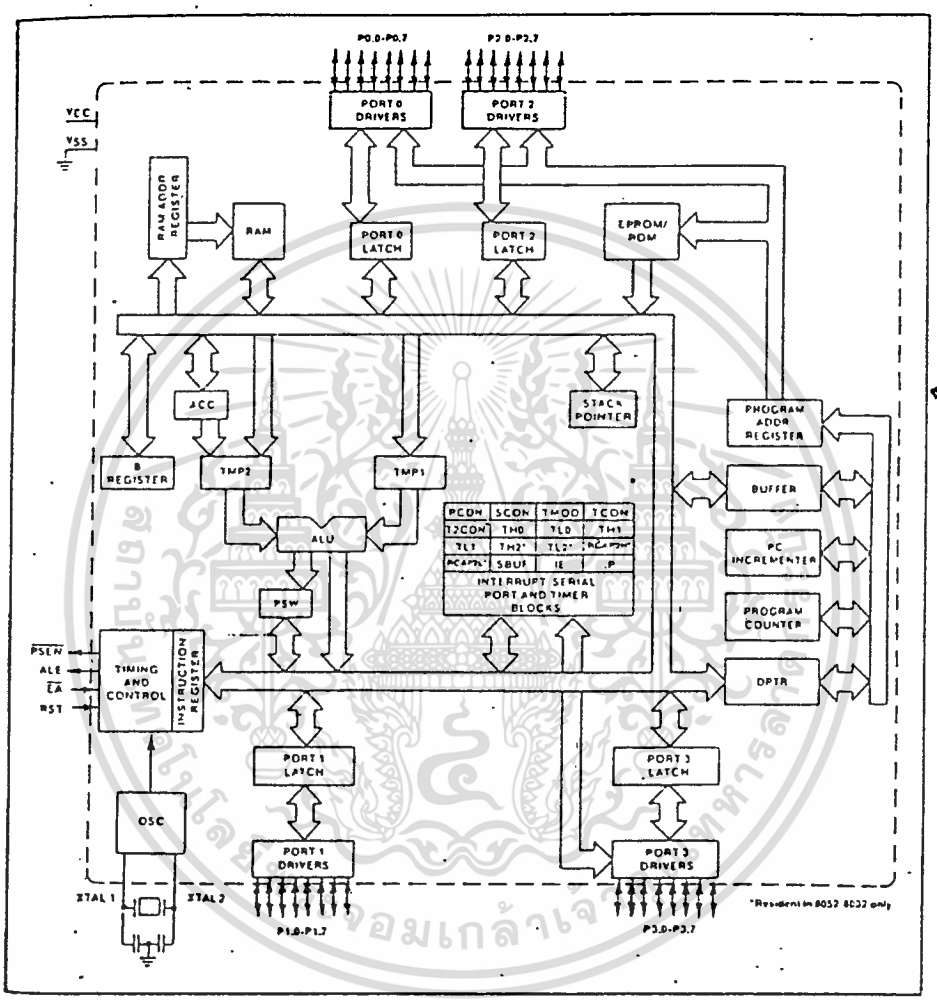
บทที่ 6 ตัวอย่างโปรแกรม

ภาคผนวก ก. ชุดคำสั่ง

บทที่ 1 บทนำ

ไมโครโปรเซสเซอร์ตระกูล MCS-51 เป็นไมโครโปรเซสเซอร์ แบบชิงเกิลชิพ เพราะภายในชิพประกอบด้วยหน่วยความจำ , ส่วนการคำนวณ , ส่วนสร้างสัญญาณนาฬิกา , พอร์ทแบบขนาน , วงจรนับเวลาและตั้งเวลา , พอร์ทแบบอนุกรมทำให้มีการใช้งานอย่างกว้างขวาง เพราะในการออกแบบระบบจะไม่ต้องใช้อุปกรณ์ภายนอกมาก จึงประหยัดทั้งพลังงานที่ใช้และการลงทุนในระบบ ไมโครคอมพิวเตอร์ตระกูลMCS-51 มีข้อดีเหนือกว่าไมโครโปรเซสเซอร์ตัวอื่นๆที่มีการใช้งานในปัจจุบันมาก เพราะมีหน่วยความจำสำหรับใช้เขียนโปรแกรมได้ถึง 64 กิโลไบท์ และมีหน่วยความจำสำหรับการใช้เก็บข้อมูลแยกออกมาอีก 64 กิโลไบท์ หน่วยความจำภายในแบบแรนดอมแอคเซส (RAM) ขนาด 256 ไบท์ จะมีบางส่วนที่โปรแกรมจะสามารถติดต่อได้ทีละบิท และแบบไบท์ พอร์ทขนานภายในสามารถใช้งานได้ทั้งแบบอินพุท และเอาท์พุท และพอร์ท 3 สามารถใช้งานในฟังก์ชั่นอื่นได้ วงจรนับเวลาและตั้งเวลาภายใน 2 ชุด ขนาด 16 บิท (TIMER 0 , TIMER 1) สามารถโปรแกรมเพื่อการใช้งานแยกจากกันได้อย่างอิสระ สามารถใช้โปรแกรมกำหนดอัตราการส่งข้อมูล (BAUD RATE) ของพอร์ทอนุกรม ซึ่งเป็นฟังก์ชั่นของพอร์ท 3 สามารถเลือกให้ทำงานจากโปรแกรมที่อยู่ภายในชิพหรือภายนอกชิพ โดยการต่อขาหนึ่งเข้ากับ ลอจิก 0 หรือ ลอจิก 1 อินเทอร์รัพท์ภายนอก 2 อินเทอร์รัพท์ และภายใน 3 อินเทอร์รัพท์ ทำให้ใช้งานควบคุมได้อย่างมีประสิทธิภาพ

ชิงเกิลบอร์ด 8051 ถูกออกแบบขึ้นให้ผู้ใช้สามารถศึกษาการทำงานของ 8051 ได้อย่างเต็มประสิทธิภาพการใช้งานของ 8051 ผู้ใช้สามารถใช้ทั้งพอร์ท และฟังก์ชั่นภายในได้ อีกทั้งสามารถนำเข้าไปใช้ในระบบควบคุมได้สะดวกกว่าชิงเกิลบอร์ดแบบอื่น ๆ



ไมโครโปรเซสเซอร์ในตระกูล MCS-51 ซึ่งเป็นไมโครโปรเซสเซอร์ขนาด 8 บิท ประกอบด้วยโมโครโปรเซสเซอร์เบอร์ต่างๆ ดังตารางที่ 1. ทุกๆ เบอร์จะมีสถาปัตยกรรมพื้นฐานเหมือนกับรุ่น 1:1 แต่เดิม 8051 ถูกสร้างด้วยวิธี HMOS I แต่ในปัจจุบันได้สร้างด้วยวิธี HMOS II จึงมีชื่อเป็น 8051 AH ไมโครโปรเซสเซอร์ในตระกูล 51 นั้นถึงแม้ ว่าจะมีหลายเบอร์ แต่เราก็จะเรียกว่าเป็น "8051" ซึ่งเราจะใช้ชื่อ-

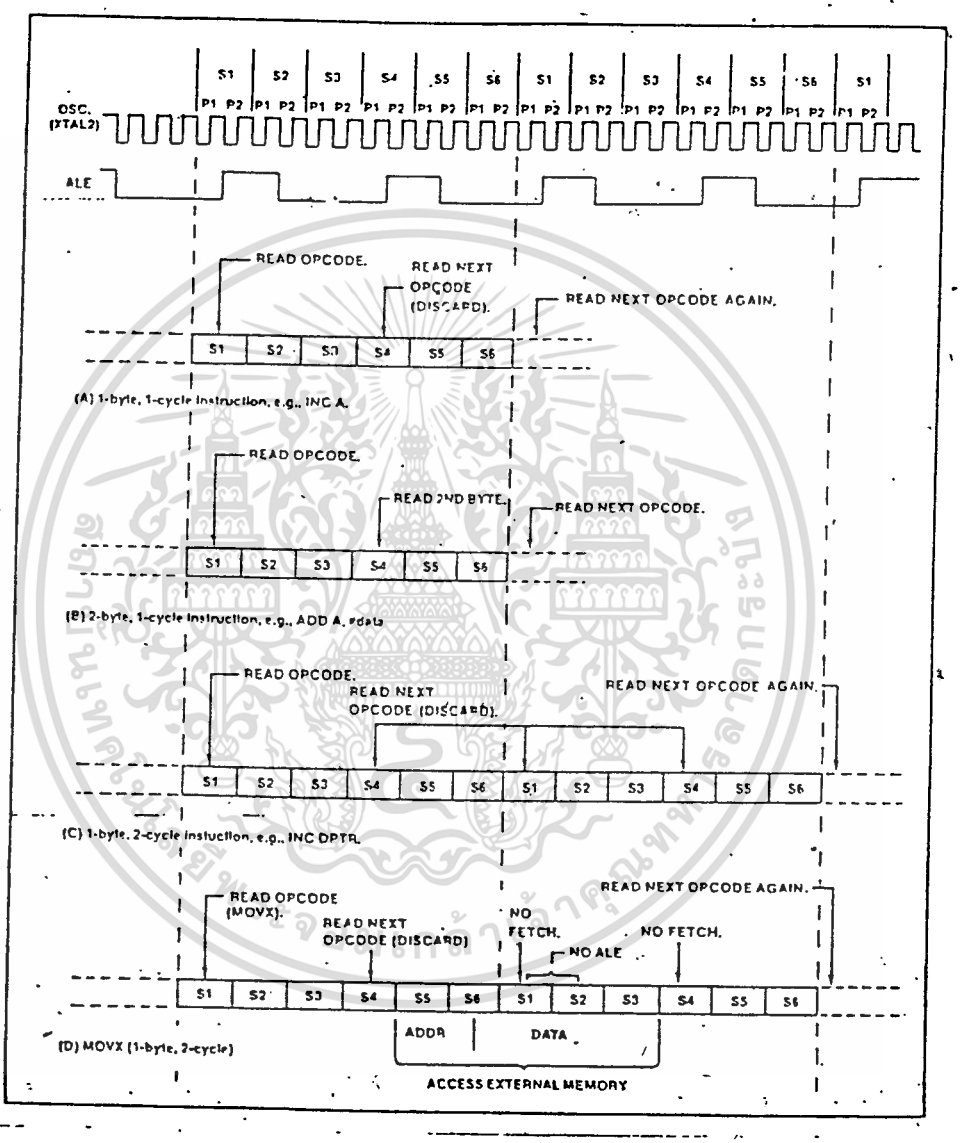
8051 นี้ โดยตลอดเมื่อหมายถึงไมโครโปรเซสเซอร์ ตระกูล 51 ส่วนเบอร์ 8032 และ 8052 มีหน่วยความจำภายในเพิ่มขึ้น และมีวงจรนับ , ตั้งเวลาขนาด 16 บิทเพิ่มขึ้น โดยวงจรนับ , ตั้งเวลาสามารถใช้เป็นวงจรนับ , วงจรตั้งเวลา และเป็นตัวกำหนดอัตราการส่งข้อมูลทางอนุกรม (Serial port) ข้อมูลของขาแต่ละขาของไมโครโปรเซสเซอร์อยู่ในภาคผนวกตอนท้าย

PART	TECHNIOLOGY	ON-CHIP PROGRAM MEMORY	ON-CHIP DATA MEMORY
8051AH	HMOS II	4K-ROM	128
8031AH	HMOS II	NONE	128
8751H	HMOS II	4K-EPROM	128
80C51	CHMOS	4K-ROM	128
80C31	CHMOS	NONE	128
8052	HMOS II	8K-ROM	256
8032	HMOS II	NONE	256

คุณสมบัติสำคัญของตระกูล 51 นี้คือ

- ไมโครโปรเซสเซอร์แบบ 8 บิท
- มีวงจรกำเนิดสัญญาณนาฬิกาภายใน
- 32 อินพุท / เอาท์พุท
- หน่วยความจำภายนอกสำหรับเก็บข้อมูล 64 กิโลไบท์ (64 kbyte)
- หน่วยความจำภายนอกสำหรับเก็บโปรแกรม 64 กิโลไบท์ (64 kbyte)
- วงจรนับ, ตั้งเวลา 16 บิท 2 ชุด (3 ชุด สำหรับ 8032/8052)
- 5 อินเทอร์รัพท์ (6 สำหรับ 8032/8052) สามารถจัดลำดับความสำคัญของการอินเทอร์รัพท์ได้ 2 ระดับ
- ส่งข้อมูลอนุกรมแบบ 2 ทิศทางได้

- สามารถประมวลผลตรรกศาสตร์
- มี RAM ภายในขนาด 128 ไบท์ สามารถอ้างได้ทั้งแบบบิทและแบบเป็นไบท์



การติดต่อกับหน่วยความจำภายนอกมี 2 แบบ ซึ่งมีการทำงานดังรูป 2.2 คือ ติดต่อกับหน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับข้อมูลการติดต่อกับ

หน่วยความจำสำหรับโปรแกรมภายนอกจะใช้สัญญาณ PSEN (Program store enable) เป็นสัญญาณสโตรป (strobe) สัญญาณ RD และ WR จะใช้เป็นสัญญาณสโตรป (strobe) สำหรับการติดต่อกับหน่วยความจำสำหรับข้อมูล การอ่านโปรแกรมจากหน่วยความจำภายนอกจะใช้แอดเดรส 16 บิทเสมอ ส่วนการติดต่อกับหน่วยความจำสำหรับข้อมูลจะใช้แอดเดรสได้ทั้งแบบ 16 บิท(เช่นคำสั่ง MOVX @DPTR) หรือใช้แอดเดรส 8 บิท (เช่นคำสั่ง MOVX @Ri) เมื่อการติดต่อกับหน่วยความจำเป็นแบบ 16 บิท ไบท์สูงของแอดเดรสจะมาจากพอร์ท 2 ซึ่งจะส่งค่าแอดเดรสออกมา ขณะทำการอ่านและเขียน ในระหว่างนี้ พอร์ท 2 LATCH (ซึ่งเป็นหนึ่งใน Special Function Register,SFR) จะไม่มีการเปลี่ยนแปลงข้อมูล เมื่อหมดคำสั่งการติดต่อกับหน่วยความจำภายนอก ค่าของ SFR พอร์ท 2 ก็จะปรากฏที่พอร์ท 2 ต่อไป ถ้าเป็นคำสั่งการอ้างอิงตำแหน่งหน่วยความจำ 2 จะไม่เปลี่ยนแปลง ในขณะที่มีการติดต่อกับหน่วยความจำภายนอกทั้งการติดต่อแบบ 8 บิท และ 16 บิท ไบท์ต่ำของแอดเดรสจะเป็นแบบมัลติเพลกซ์กับข้อมูลของพอร์ท 0 สัญญาณ ADDR/DATA จะทำให้ FET ของเอาท์พุทพอร์ท 0 ทำงาน ดังนั้นในการใช้งานพอร์ท 0 ลักษณะนี้จำไม่ต่อ PULL UP ภายนอก วงจร LATCH ภายนอกจะเก็บค่าของแอดเดรสไบท์ต่ำ โดยใช้สัญญาณ ALE เป็นตัวกำหนด โดยค่าของแอดเดรสไบท์ต่ำ จะมีค่าคงที่ขณะที่สัญญาณ ALE เปลี่ยนจาก 1 เป็น 0 ดังนั้นในการเขียนข้อมูลไปยังหน่วยความจำนั้น ข้อมูลจะถูกส่งออกไปทางพอร์ท 0 ก่อนจะมีสัญญาณ WR เป็น 0 และจะคงค่านั้นไว้จนกว่าสัญญาณ WR จะกลับเป็น 1 ในการอ่านข้อมูลจากหน่วยความจำภายนอกนั้น ข้อมูลจะต้องปรากฏที่พอร์ท 0 ก่อนสัญญาณ RD จะเป็น 0 เสมอ

ระหว่างการติดต่อกับหน่วยความจำภายนอกนั้น CPU จะส่งข้อมูล OFFH ไปยังพอร์ท 0 LATCH ทำให้ข้อมูลที่พอร์ท 0 เปลี่ยนไป

การอ่านโปรแกรมจากหน่วยความจำภายนอกจะเกิดได้ใน 2 กรณี คือ

1. EA เป็น 0

2. เมื่อ EA เป็น 1 จะอ่านโปรแกรมจากหน่วยความจำภายนอกเมื่อโปรแกรมเคาร์เตอร์ (PC) มีค่ามากกว่า OFFFH หรือ 1FFFH สำหรับ 8052

ดังนั้น ในรุ่นที่ไม่มี ROM ภายในจึงต้องต่อ EA ไว้ที่ 0 เพื่อให้มีการอ่านโปรแกรมจากหน่วยความจำภายนอกเสมอ

SERIAL INTERFACE

Serial พอร์ทเป็นแบบสองทาง (full duplex) หมายความว่าสามารถทั้งส่งและรับได้พร้อมกัน ในการรับจะมีบัฟเฟอร์ซึ่งสามารถให้รับข้อมูลในไบท์ที่สองโดยที่ไบท์แรกยังไม่ได้ถูกอ่านเข้าไปจากรีจิสเตอร์ตัวรับ แต่ถ้าเมื่อไบท์ที่สองรับเข้ามาครบแล้ว ไบท์ที่ 1 ยังไม่ได้ถูกอ่านเข้าไป แล้วข้อมูลในไบท์ที่ 1 จะหายไป รีจิสเตอร์ SBUFจะใช้เป็นบัฟเฟอร์สำหรับทั้งรับและส่งข้อมูล การเขียนข้อมูลไปยัง SBUF จะเป็นการโหลดค่าไปทำการส่งและโครงสร้างภายในแล้ว รีจิสเตอร์ SBUF ของการส่งและรับข้อมูลเป็นคนละตัวกันแต่เรียกเหมือนกัน CPU จะรู้เองว่าเรียกใช้ SBUF สำหรับการส่งหรือการรับ โดยถ้าเป็นการอ่านค่า SBUF จะเป็นการรับข้อมูล ส่วนถ้าเป็นการอ่านค่า SBUF จะเป็นการส่งข้อมูล Serial พอร์ท สามารถทำงานได้ใน 4 โหมด

โหมด 0 ข้อมูลอนุกรมจะส่งผ่านขา Rxd และ Txd โดยการเลื่อนด้วยอัตราคงที่ = 1/12 ของความถี่ออสซิเลเตอร์ ข้อมูลที่ส่งและรับเป็นแบบ 8 บิท

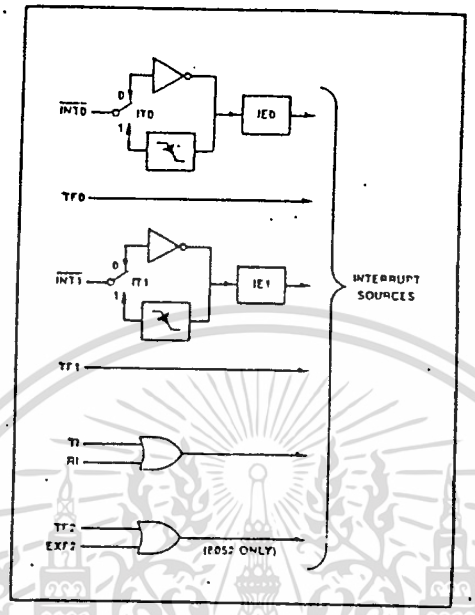
โหมด 1 การส่งข้อมูลผ่าน RXD และ TXD เป็นแบบ 10 บิท ประกอบด้วย 1 start บิท (0) 8 data บิท (บิทต่ำสุดก่อน) และ 1 stop bit (1) ในตอนรับบิทท์ 10 จะไปอยู่ใน RB8 ใน SFR ชื่อ SCON อัตราการส่งสามารถโปรแกรมได้

โหมด 2 การส่งและรับข้อมูลเป็นแบบ 11 บิท คือ 1 start บิท (0) 8 data บิท (บิท 0-7) บิทที่ 9 สามารถกำหนดได้ และ 1 stop บิท (1) ในการส่งบิทที่ 9 (TB 8 ใน SCON) สามารถกำหนดค่าให้เป็น 1 หรือ 0 ก็ได้ เช่น เอาพาริตี้บิท (Parity, P ใน PSW) ย้ายเข้าไปที่ TB8 หรือในตอนรับบิทที่ 9 จะถูกส่งเข้าไปในRB8 ของ SCON ขณะที่ stop บิทจะไม่ถูกเก็บไว้ อัตราการส่งสามารถโปรแกรมเป็น 1/32 หรือ 1/64 ของความถี่ออสซิเลเตอร์

โหมด 3 การทำงานของโหมดนี้จะเหมือนกับโหมด 2 ยกเว้นเราสามารถโปรแกรมอัตราการส่งได้

ในทุกโหมด การส่งข้อมูลจะเริ่มจากคำสั่งใดก็ได้ที่มีการส่งข้อมูลไปยัง SBUF ในการรับข้อมูลจะต่างกันเล็กน้อย คือในโหมด 0 การรับข้อมูลจะเริ่มโดย RI = 0 และ REN = 1 ส่วนในโหมดอื่น จะเริ่มการ รับข้อมูลเข้ามาโดย REN = 1 เมื่อมี start บิทเข้ามา

INTERRUPT (อินเทอรัพท์)



8051 สามารถอินเทอรัพท์ได้ด้วย 5 วิธี ดังรูป 2.3

สัญญาณอินเทอรัพท์จากภายนอก INT0 และ INT1 จะสามารถทำงานได้ทั้งแบบระดับ (LEVEL) และแบบการเปลี่ยนระดับ (TRANSITION) ขึ้นกับ IT0 และ IT1 ใน TCON แฟลกที่ควบคุมสัญญาณอินเทอรัพท์ นี้คือ IE0 และ IE1 ใน TCON เมื่อเกิดการอินเทอรัพท์จากภายนอก แฟลกที่สร้างอินเทอรัพท์จะถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อมีตัวชี้ตำแหน่งโปรแกรมตอบสนองอินเทอรัพท์ ถ้าเป็นการอินเทอรัพท์แบบเปลี่ยนสภาวะเท่านั้น อินเทอรัพท์ของ TIMER 0 และ TIMER 1 จะสร้างโดย TF0 และ TF1 ซึ่งถูกเซทโดยการนับถึง 255 ของรีจิสเตอร์ตัวนั้น ๆ เมื่อเกิดอินเทอรัพท์ของ TIMER แฟลกที่เกิดขึ้นจะถูกเคลียร์โดยฮาร์ดแวร์ภายใน เมื่อมีการชี้ไปยังตำแหน่งของโปรแกรมตอบสนองอินเทอรัพท์

อินเทอรัพท์ของ Serial พอร์ท จะสร้างโดยการ OR ของ RI และ TI แฟลกทั้ง 2 ตัวนี้จะไม่ถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อมีการชี้ยังตำแหน่งโปรแกรมตอบสนองอินเทอรัพท์ในโปรแกรมตอบสนองอินเทอรัพท์จะต้องรู้ว่าเป็น RI หรือ TI แล้วจึงเคลียร์ด้วยซอฟแวร์

ทุกบิทที่สร้างอินเทอรัพท์สามารถเซทหรือเคลียร์ได้ด้วยซอฟแวร์ เหมือนกับทำด้วยฮาร์ดแวร์ ดังนั้นอินเทอรัพท์สามารถสร้าง หรือยกเลิกได้โดยกำหนดในซอฟแวร์ แต่ละอินเทอรัพท์สามารถอินาเบิลหรือดิสเอเบิล แยกจากกันโดยอิสระด้วยการกำหนดใน SFR ชื่อ IE (รูป 2.4) ใน IE จะมีบิทหนึ่งคือ EA ซึ่งจะดิสเอเบิลทั้งหมดพร้อมกันได้

(MSB)							(LSB)
X	X	PT2	PS	PT1	PX1	PT0	PX0

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	defines the Timer 2 interrupt priority level. PT2 = 1 programs it to the higher priority level.
PS	IP.4	defines the Serial Port interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	IP.3	defines the Timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	IP.2	defines the External interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
PT0	IP.1	defines the Timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	IP.0	defines the External interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

บทที่ 3 โครงสร้างหน่วยความจำ, แอดเดรสซิงโหมดและตรรกศาสตร์

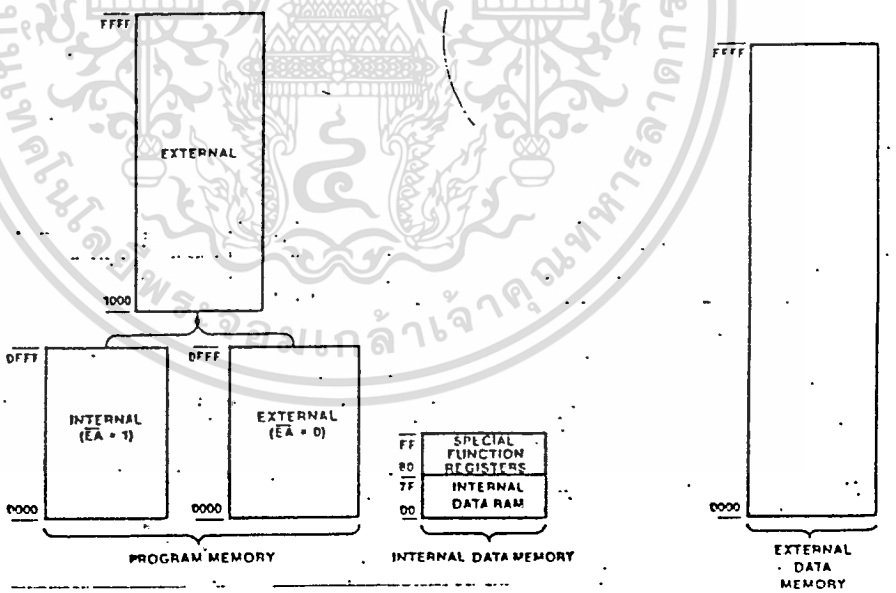
โครงสร้างของตระกูล 51 สามารถใช้งานหน่วยความจำภายในและภายนอกได้อย่างดี วิธีการแอดเดรส ต่าง ๆ จะทำให้คำสั่งมีประสิทธิภาพ

3.1 การจัดการหน่วยความจำ

8051 มีหน่วยความจำพื้นฐานได้ดังนี้

- หน่วยความจำสำหรับโปรแกรม 64 กิโลไบท์
- หน่วยความจำสำหรับข้อมูลอยู่ภายนอก 64 กิโลไบท์
- หน่วยความจำภายในแบบ RAM 256 ไบท์ (ใน 8032/8052 มีถึง 380 ไบท์)

รูป 3-1 แสดง memory map ของ ตระกูล 51



Program Memory Address Space

หน่วยความจำ 64 กิโลไบท์ สำหรับโปรแกรมจะมีทั้งภายในและภายนอก ถ้าขา EA ต่อไว้ที่ High 8051 จะทำงานจากโปรแกรมภายใน นอกเสียจากจะมีการเรียกโปรแกรมจากตำแหน่งมากกว่า OFFFH (1FFFH ใน 8052) ตำแหน่ง 1000H ถึง OFFFFH (2000H ถึง OFFFFH ใน 8052) จะอยู่ภายนอกเสมอ ถ้าขา EA ต่อไว้ที่ LOW 8051 จะอ่านโปรแกรมจากภายนอกเท่านั้น ทั้งสองแบบจะใช้ PC ขนาด 16 บิท

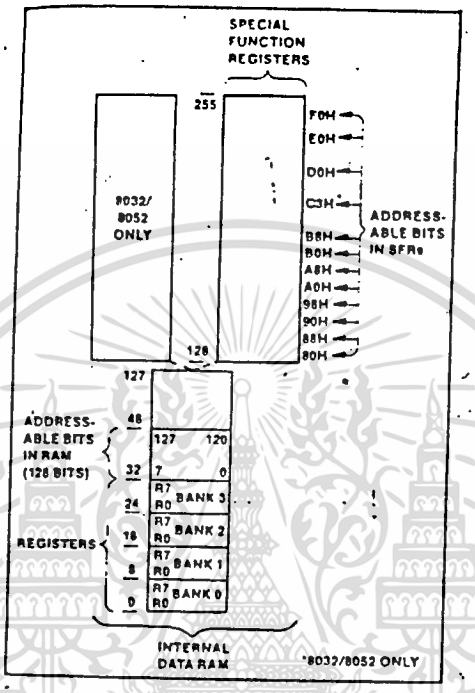
ตำแหน่ง 00 ถึง 23 H (00 ถึง 2B H ใน 8032/8052) ในหน่วยความจำสำหรับโปรแกรมจะถูกใช้สำหรับเก็บโปรแกรมตอบสนองการอินเทอรัพท์ ดังตารางที่ 2

Source	Address
External Interrupt 0	0003H
Timer 0 Overflow	000BH
External Interrupt 1	0013H
Timer 1 Overflow	001BH
Serial Port	0023H
Timer 2 Overflow/TZEX Negative Transition	002BH

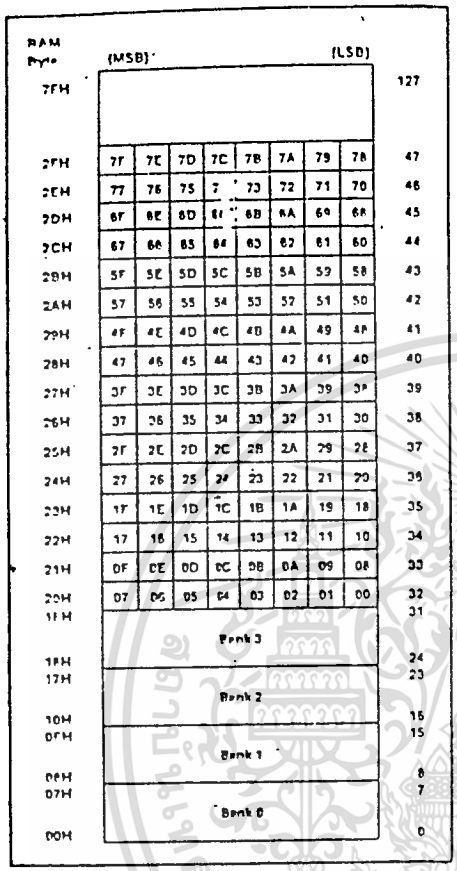
Data Memory Address Space

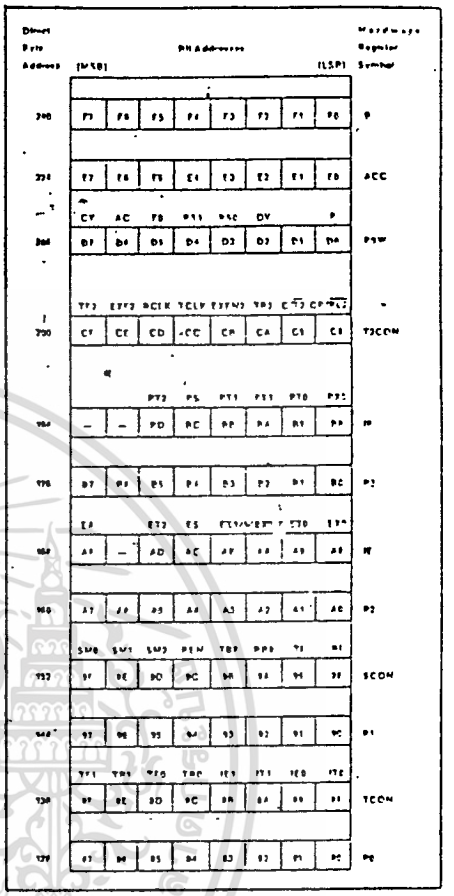
Data Memory Address Space ประกอบด้วยหน่วยความจำภายในและภายนอกการติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง MOVX เท่านั้น

หน่วยความจำ(RAM) ภายในจะแบ่งออกเป็น 3 ส่วน 128 ไบท์ต่ำของ RAM. (128 ไบท์สูงของ RAM จะติดต่อได้เฉพาะใน 8032/8052 เท่านั้น) และ 128 ไบท์ของ SFR 128 ไบท์ของ RAM จะใช้ตำแหน่งแอดเดรสเดียวกันกับ SFR แต่การติดต่อจะใช้แอดเดรสโหมดต่างกัน ซึ่งจะกล่าวถึงต่อไป



รูป 3.2 แสดง address mapping ของหน่วยความจำสำหรับข้อมูลภายใน มีรีจิสเตอร์อยู่ 4 ชุด (BANK) ชุดละ 8 รีวิสเตอร์ ในตำแหน่ง 0 ถึง 31 ใน RAM ชุดล่าง การใช้งาน แต่จะขณะจะใช้ได้เพียงชุดเดียวเท่านั้น (กำหนดใน PSW) 16 ไบท์ต่อมา (ตำแหน่ง 32 ถึง 47) จะใช้สำหรับ 128 บิทแอดเดรสเอเบิล เป็นช่วงของหน่วยความจำซึ่งสามารถอ้างอิงได้ถึงทีละบิท รูป 3.3 แสดงแอดเดรส ของ RAM บิท SFR ก็สามารถใช้แบบบิท-แอดเดรสได้ ดังรูป 3.4





การอ่านข้อมูลจากตำหน่งหน่วยความจำภายใน ซึ่งไม่ได้ใช้งานจะได้ข้อมูลไม่แน่นอน

3.2 แอดเดรสซิงโหมด

8051 มี 5 แอดเดรสซิงโหมด

- รีจิสเตอร์ (Register addressing)
- ไดเรค (Direct addressing)
- รีจิสเตอร์ อินไดเรค (Indirect Register addressing)
- อิมมีเดียท (Immediate addressing)

- เบส-รีจิสเตอร์ ร่วมกับ อินเดค-รีจิสเตอร์ อินไดเรค

(Base-Register + Index-Register Indirect addressing)

ตารางสองสรุป หน่วยความจำที่สามารถติดต่อได้ด้วยแต่ละวิธีแอดเดรสซิงโหมด

<u>รีจิสเตอร์-แอดเดรสซิง</u>

รีจิสเตอร์ แอดเดรสซิงจะทำงานผ่านชุดของรีจิสเตอร์ (R0-R7) ที่กำลังใช้งานอยู่ 3 บิทล่างของชุดคำสั่งจะบอกว่าเป็นการใช้แอดเดรสใด ACC , B, DPTR และ CY Boolean Processor Accumulator สามารถใช้งานโดยการแอดเดรสเหมือนเป็นรีจิสเตอร์

<u>ไดเรคแอดเดรสซิง</u>

ไดเรคแอดเดรสซิงเป็นวิธีเดียวที่จะใช้ติดต่อกับ SFR 128 ไบท์ต่ำของ RAM ภายใน

<u>รีจิสเตอร์-อินไดเรค แอดเดรสซิง</u>

รีจิสเตอร์ อินไดเรค แอดเดรสซิง จะใช้ค่าของ R0 หรือ R1 (ของชุดที่กำลังใช้งาน) เป็นตัวชี้ไปยังตำแหน่งใน 256 ไบท์ คือ 128 ไบท์ล่างของ RAM ภายใน, 128 ไบท์บน ของ RAM ภายใน (ใน 8032/8052 เท่านั้น) หรือใน 256 ไบท์ของหน่วยความจำสำหรับข้อมูลภายนอก SFR จะไม่สามารถติดต่อได้ด้วยวิธีนี้ และการติดต่อหน่วยความจำภายนอกถึง 64 กิโลไบท์ จะต้องใช้ Data Pointer แบบ 16 บิทเท่านั้น

การทำงานของคำสั่ง PUSH และ POP จะเป็นแบบ รีจิสเตอร์-อินไดเรค แอดเดรสซิง Stack Pointer อาจชี้ตำแหน่งใดก็ได้ในหน่วยความจำภายใน

<u>อิมมีเดียท แอดเดรสซิง</u>

อิมมีเดียทแอดเดรสซิง จะมีค่าคงที่ปรากฏอยู่ในรหัสคำสั่ง (OP Code) ในโปรแกรม

เบส-รีจิสเตอร์ ร่วมกับ อินเดกรีจิสเตอร์ อินไดเรคแอดเดรสซิง จะเป็นการแอดเดรสข้อมูลจากหน่วยความจำของโปรแกรม โดยการใช้แอดเดรสจากผลรวมของเบสรีจิสเตอร์ (DPTR) และอินเดกรีจิสเตอร์ Acc จะใช้งานเป็นแบบเปิดตาราง

3.3 บูลีนโพรเซสเซอร์ (Boolean Processer)

บูลีนโพรเซสเซอร์ เป็นอินทิเกรตบิทโพรเซสเซอร์ ใน 8051 จะมีชุดคำสั่งเฉพาะ . แอคคิวมูเลเตอร์ (Carry Flag) และบิทแอดเดรสเอเบิลใน RAM และอินพุท /เอาท์พุท สามารถใช้ตรวจเงื่อนไขเพื่อให้มีการกระโดดหรือไม่กระโดด เมื่อมีการเซทหรือไม่ถูกเซท , ข้ามถ้ามีการเซทแล้วเคลียร์, MOVX เข้า Carry แอดเดรสเอเบิลบิท หรือค่าคอมพลีเมนท์สามารถ AND หรือ OR กับค่าของ carry Flag ได้ ผลลัพท์จะเก็บใน Carry รีจิสเตอร์

บทที่ 4 ชุดคำสั่ง

ใน MCS-51 จะมีชุดคำสั่งอยู่ 111 คำสั่ง 49 คำสั่งเป็นแบบไบท์เดียว, 45 คำสั่งเป็นแบบ 2 ไบท์ และ 17 คำสั่งเป็นแบบ 3 ไบท์ รูปแบบรหัสคำสั่งประกอบด้วย มิวโมนิค (MNEMONIC) ตามด้วย "Destination Source" (Operand Field) ซึ่งในโอเปอแรนด์ฟิลด์นี้ จะมีชนิดของข้อมูลและวิธีแอดเดรสซึ่งอยู่ด้วย

4.1 ฟังก์ชั่นทั้งหมด (Functional Overview)

ชุดคำสั่ง MCS 51 จะแบ่งออกเป็น 4 กลุ่ม ตามการทำงาน ดังนี้

- เคลื่อนย้ายข้อมูล (Data Transfer)
- คำนวณ (Arithematic)
- ลอจิก (Logic)
- คอนโทรลทรานสเฟอร์ (Control Tranfer)

4.1.1 เคลื่อนย้ายข้อมูล (Data Transfer)

การเคลื่อนย้ายข้อมูล แบ่งออกเป็น 3 แบบ

- การย้ายข้อมูลทั่วไป
- กำหนดแอคคิวมูเลเตอร์
- แอดเดรส ออบเจค

การทำงานข้างบนไม่มีผลต่อแฟลกใน PSW ยกเว้น POP หรือ MOV โดยตรงกับ PSW

การย้ายข้อมูลทั่วไป

* MOV จะสามารถย้ายข้อมูลได้ทั้งแบบบิทหรือไบท์ จาก Source ในโอเปอแรนด์ ไปยัง Destination ในโอเปอแรนด์
* PUSH จะเพิ่มค่าของริจิสเตอร์ SP แล้วย้ายข้อมูลจาก Source ในโอเปอแรนด์ ไปยังตำแหน่งที่ชี้โดย SP
* POP จะย้ายข้อมูลจากสแตกตำแหน่งที่ชี้โดย SP ไปยัง Destination ในโอเปอแรนด์ แล้วลดค่า SP

กำหนดแอคคิวมูเลเตอร์

. XCH จะแลกข้อมูลจาก Source Operand กับ Accumulator

. XCHD จะแลกข้อมูลเฉพาะ 4 บิทล่างของไบท์ กับ 4 บิทล่างของ Accumulator

. MOVX จะย้ายข้อมูลระหว่างหน่วยความจำภายนอก และแอคคิวมูเลเตอร์ ตำแหน่งหน่วยความจำภายนอกจะอ้างอิงโดยรีจิสเตอร์ DPTR (16 บิท) หรือรีจิสเตอร์ R1,R0 (8 บิท)

. MOVC ย้ายข้อมูลจากในหน่วยความจำของโปรแกรมไปยังแอคคิวมูเลเตอร์ โอเปอแรนด์ใน A จะถูกใช้เป็นอินเด็กซ์ ไปยัง 256 ไบท์ โดยใช้ร่วมกับ DPTR หรือ PC ค่าจากหน่วยความจำจะถูกอ่านมายังแอคคิวมูเลเตอร์

แอดเดรส-ออบเจค ทรานเฟอร์

MOV DPTR , #data จะกำหนดค่าจากโอเปอแรนด์ให้กับรีจิสเตอร์ DPH และ DPL โดยตรง

4.1.2 คำนวณ (Arithematic)

8051 สามารถทำการคำนวณได้ 4 แบบ และเป็นแบบ 8 บิท ไม่มีเครื่องหมาย มีแฟลกบอกการโอเวอร์โฟลว์ อย่างไรก็ตามในการบวกและลบ จะสามารถทำได้ทั้งแบบมีกครื่องหมายและ ไม่มี การคำนวณสามารถให้ผลออกมาในรูปของ ฐานสิบ (BCD) ได้โดยตรง

การบวก

. INC (Increment) บวก 1 เข้ากับ Source ใน Operand และเก็บผลลัพธ์ใน Operand

. ADD บวก A เข้ากับ Source ใน Operand และเก็บผลลัพธ์ใน A

. ADDC (Add with Carry) บวก A เข้ากับ Source ในโอเปอแรนด์ และบวกด้วย 1 ถ้า CY ถูกเซท เก็บผลลัพธ์ใน A

. DAA (Decimal-Add-Adjust for BCD Addition) จะปรับค่าของการบวกซึ่งเกิดจากการบวกเลขฐานสิบเข้าด้วยกัน ให้ได้ผลลัพธ์เป็นเลข

ฐานสิบ ค่าที่ได้จะเก็บใน A จะมี CY ถูกเซท ถ้าค่าที่ได้เกิน 99 นอกนั้น CY จะถูกเคลียร์

การลบ

. SUBB (Substract with Borrow) ลบ Source ที่สองในโอเปอแรนด์ออกจาก Source แรก ในโอเปอแรนด์(แอคคิวมูเลเตอร์) และลบด้วย 1 ถ้า CY ถูกเซท เก็บผลลัพธ์ใน A

. DEC (Decrement) ลบ 1 ออกจาก Source ในโอเปอแรนด์ และเก็บผลลัพธ์ในโอเปอแรนด์

การคูณ (Multiplication)

. MUL จะทำการคูณค่าจากรีจิสเตอร์ A กับรีจิสเตอร์ B แบบไม่คิดเครื่องหมายเข้าด้วยกัน และผลลัพธ์จะเป็น 2 ไบท์ ไบท์ต่ำเก็บใน A และไบท์สูงเก็บใน B โอเวอร์โฟล์วจะถูกเคลียร์ถ้าไบท์บนเป็น 0 และจะถูกเซ็ทถ้าไม่เป็น 0 Cy จะถูกเคลียร์ Acไม่มีการเปลี่ยนแปลง

การหาร

. DIV จะทำการหารแบบไม่คิดเครื่องหมายของรีจิสเตอร์ A ด้วยรีจิสเตอร์ B และเก็บผลหารใน A เก็บเศษในรีจิสเตอร์ B การหารด้วย 0 จะให้ผลลัพธ์ไม่แน่นอนใน A และ B และเกิด OV ถูกเซท นอกนั้น OV จะถูกเคลียร์ CY จะถูกเคลียร์ AC จะไม่เปลี่ยนแปลง

นอกจากที่กล่าวมาข้างบนแล้ว แฟลกใน PSW จะมีการเปลี่ยนแปลง ดังนี้

. CY จะถูกเซทถ้าการคำนาณเกิดตัวทดที่บิทสูงสุดของผลลัพธ์ นอกนั้น CY จะถูกเคลียร์

. AC จะถูกเซทถ้าผลลัพธ์เกิดตัวทดจาก 4 บิทล่างของผลลัพธ์ (ในระหว่างการบวก) หรือเกิดตัวทดที่บิทสูงสุด(ในระหว่างการลบ) นอกนั้น AC จะถูกเคลียร์

. OV จะถูกเซทถ้าการคำนวณเกิดตัวทดขึ้นในบิทสูงสุดของผลลัพธ์ นอกนั้น OV

จะถูกเคลียร์ OV จะถูกใช้ในการทำงานแบบ 2's Complement เพราะ OV จะถูกเซทเมื่อผลลัพธ์ไม่สามารถเก็บในค่า 8 บิท

P จะถูกเซทเป็น Modulo 2 ผลบวกของ 8 บิท ในแอคคิวมูเลเตอร์เป็น 1 (Odd Parity) นอกนั้น P จะถูกเคลียร์ (Even Parity) เมื่อค่านี้ถูกเก็บใน PSW บิท P จะไม่เปลี่ยนแปลงและจะแสดงพาร์ริตี้ของ A เสมอ

4.1.3 ลอจิก (Logic)

8051 สามารถทำงานเกี่ยวกับลอจิกได้ทั้งแบบบิทและไบท์โอเปอแรนด์

การทำงานโอเปอแรนด์เดียว

CLR ให้ค่าใน A หรือ ไดเรคแอดเดรสเอเบิล มีค่าเป็น 0

SETB ให้ค่าในไดเรคเอเบิลบิท เป็น 1

CPL ใช้ทำการคอมพลีเมนต์ค่าของรีจิสเตอร์ A โดยไม่มีผลต่อแฟลก หรือไดเรคแอดเดรสเอเบิลใดๆ

RL,RLC,RR,RRC,SWAP เป็น 5 การทำงานสำหรับการวนข้อมูล โดยสามารถทำงานกับ Acc . RL เป็นการ-วนข้อมูลไปทางซ้าย,RR เป็นการวนข้อมูลไปทางขวา, RLC เป็นการ-วนข้อมูลไปทางซ้ายผ่าน C , RRC เป็นการ-วนข้อมูลไปทางขวาผ่าน C และ SWAP เป็นการ-วนข้อมูลไปทางซ้าย 4 ตำแหน่ง สำหรับ RLC และ RRC ค่าของแฟลก CY จะเท่ากับบิทที่วนเข้ามายัง C SWAP จะวนข้อมูลไปทางซ้าย 4 ตำแหน่ง โดยจะเป็นการสลับข้อมูลบิท 3 ถึง 0 กับ บิท 7 ถึง 4

การทำงานแบบ 2 โอเปอแรนด์

ANL จำทำการ AND บิทของ 2 Source ในโอเปอแรนด์ (หรือทั้งบิทและไบท์โอเปอแรนด์) แล้วเก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

ORL จะทำการ OR บิทของ 2 Source ในโอเปอแรนด์ (หรือทั้งบิทและไบท์โอเปอแรนด์) แล้วเก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

XRL ทำการ Exclusive OR บิทของ Source ในโอเปอแรนด์เก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

4.1.4 คอนโทรล ทรานสเฟอร์

มี 3 วิธีสำหรับการคอนโทรลทรานสเฟอร์ การ CALL แบบไม่มีเงื่อนไข, RETURN และ JUMP, การ JUMP แบบมีเงื่อนไข และอินเทอรัพท์ การคอนโทรลทราน

เปอร์สะทำให้การทำงานของโปรแกรมไม่เรียงแอดเดรสต่อเนื่อง

UNCONDITIONAL CALL, RETURNS, และ JUMPS

จะทำให้การทำงานจากโปรแกรมเคาร์เตอร์เดิมแ ไปยังแอดเดรสตำแหน่งใหม่ ทั้งแบบโดยตรงและโดยอ้อม

. ACALL และ LCALL จะเก็บค่าของแอดเดรสของคำสั่งต่อไป ไว้ในสแตกแล้วข้ามไปทำงานยังแอดเดรสที่ต้องการ ACALL เป็นคำสั่งแบบ 2 ไบท์ ถูกใช้เมื่อตำแหน่งที่จะไปอยู่ภายในแต่ละช่วง 2 K โดยเริ่มนับจาก PC ที่ต่อจากคำสั่ง ACALL , LCALL เป็นคำสั่งแบบ 3 ไบท์ ซึ่งจะอ้างถึง ตำแหน่งหน่วยความจำได้ ถึง 64 กิโลไบท์ ใน ACALL ข้อมูล 11 บิทจากโอเปอแรนด์จะรวมกับ 5 บิทบนของ PC ชี้ไปยังตำแหน่งหน่วยความจำที่ต้องการ ถ้า ACALL ชี้ในตำแหน่ง 2 ไบท์สุดท้ายของหน้า 2 K การ์ CALL จะทำงานในหน้าต่อไปโดยค่าของ PC จะเพิ่มไปเพื่อชี้ตำแหน่งที่จะทำงาน

.RET จะกระโดดข้ามการทำงานไปยังตำแหน่งแอดเดรสที่ถูกเก็บอยู่ในสแตก โดยการ CALL ที่ผ่านมา และลดค่าของ SP ลง 2 เพื่อให้ Sp จะได้ POP แอดเดรสต่อไปได้ถูกต้อง

.AJMP, LJMP และ SJMP เป็นคำสั่งข้ามการทำงานไปยังตำแหน่งในโอเปอแรนด์ การทำงานของ AJMP และ LJMP จะเหมือนกับ ACALL และ LCALL คำสั่ง SJMP ให้การข้ามการทำงานภายในช่วง 128 ไบท์ จากแอดเดรสของคำสั่งต่อไป (-128 ถึง 127) โดยมีแอดเดรสนั้นเป็นกึ่งกลาง

. JMP @A + DPTR คำสั่งข้ามการทำงานแบบอ้างอิงกับ DPTR โอเปอแรนด์ใน A จะใช้สำหรับเป็นออฟเซท (0-255) ยังแอดเดรสในรีจิสเตอร์ DPTR ดังนั้นจะสามารถข้ามไปยังตำแหน่งใดๆ ในหน่วยความจำ

การข้ามแบบมีเงื่อนไข (Conditional Jumps)

การข้ามแบบมีเงื่อนไข จะเป็นการข้ามตามสภาวะที่กำหนด ปลายทางจะอยู่ใน +128 ไบท์ จากกึ่งกลาง คือแอดเดรสเริ่มต้นของคำสั่งต่อไป (-128 -> 127)

JZ เป็นคำสั่งการข้ามถ้าแอคคิวมูเลเตอร์ เป็น 0

JNZ เป็นคำสั่งให้ข้ามถ้าแอคคิวมูเลเตอร์ ไม่เป็น 0

JC คำสั่งให้ข้ามถ้าแฟลกตัวทด (Carry Flag) ถูกเซท

JNC. คำสั่งให้ข้ามถ้าแฟลกตัวทด (Carry Flag) ถูกเคลียร์

JB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ถูกเซท

JNB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ไม่ถูกเซท

CJNE จะเปรียบค่าใน 2 โอเปอแรนด์และทำการภระโดด ถ้าโอเปอแรนด์ตัวแรกมีค่าน้อยกว่าโอเปอแรนด์ที่ 2 CY จะถูกเซท นอกนั้นแล้ว จะถูกเคลียร์ การเปรียบเทียบอาจเป็นระหว่าง A โดยตรงกับแอดเดรสเอเบิลไบท์ ในหน่วยความจำ ข้อมูลภายในหรือระหว่างค่าคงที่กับริจิสเตอร์ A , ริจิสเตอร์ในชุดริจิสเตอร์ที่กำลังใช้งาน หรือกับริจิสเตอร์-อินไดเรคแอดเดรสไบท์ ของ RAM ภายใน

DJNZ ลดค่าของ Source ในโอเปอแรนด์ และเก็บค่าผลลัพธ์ในโอเปอแรนด์ การข้ามจะเกิดขึ้นถ้าผลลัพธ์ไม่เป็นศูนย์ Source ในโอเปอแรนด์ ในคำสั่ง DJNZ อาจเป็นไบท์ใดในหน่วยความจำข้อมูลภายในก็ได้ หรือริจิสเตอร์แอดเดรสซึ่ง ก็จะใช้เป็นแอดเดรสใน Source ในโอเปอแรนด์ได้

INTERRUPT RETURN

RETI จะเป็นคำสั่งการข้ามคล้ายกับ RET จะเพิ่มการทำงานในการอินาเบิลอินเทอร์รัพท์ที่มีลำดับความสำคัญเท่ากัน เพื่อให้มีการอินเทอร์รัพท์ต่อไปได้

4.2 ความหมายของคำสั่ง (Instruction Definition)

ใน MCS-51ชุดคำสั่งการทำงานของตระกูล MCS-51 เรียงตามลำดับตัวอักษรของนิโมนิค (Mnemonic)ซึ่งแสดงการทำงานดังในภาคผนวก ก. ตัวอย่างสั้นๆที่แสดงไว้ถึงวิธีการใช้งานของคำสั่ง อาจบอกถึงผลที่เกิดขึ้นในแฟลก PSW และยังบอกถึงจำนวนไบท์ และจำนวนวงรอบการทำงาน. คำสั่งในรูปเลขฐานสอง และคำอธิบายสัญญลักษณ์ ก็จะปรากฏอยู่

ข้อสำคัญ จะมีการกล่าวถึงเฉพาะ Carry , Auxillary-Carry และ Overflow แฟลกเท่านั้น พาร์ริตี้บิทจะถูกคำนวณทุกครั้งหลังจากคำสั่งซึ่งมีการเปลี่ยนแปลงค่าของแอคคิวมูเลเตอร์ ในทำนองเดียวกับคำสั่งที่มีการชี้ริจิสเตอร์โดยตรง ก็จะมีผลต่อสภาวะแฟลก ถ้าคำสั่งนั้นเกี่ยวข้องกับ PSW แฟลกแสดงสภาวะสามารถเปลี่ยนแปลง

ใช้โดยวิธี Bit Manipulation

ตารางที่ 3 การทำงานของคำสั่งในตระกูล 8051

Interrupt Response Time: To finish execution of current instruction, respond to the interrupt request, push the PC and to vector to the first instruction of the interrupt service program requires 3 to 8 oscillator periods (3 to 7μs @ 12 MHz).

INSTRUCTIONS THAT AFFECT FLAG SETTINGS*

INSTRUCTION	C	OV	AC	INSTRUCTION	C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

*Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Notes on instruction set and addressing modes:

Rn — Register R7-R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or a SFR [i.e., I/O port, control register, status register, etc. (128-255)].

@Ri — 8-bit internal data RAM location (0-255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64k-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2k-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

— New operation not provided by 8048AH/8049AH.

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Oscillator Period
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12

ARITHMETIC OPERATIONS Cont.

Mnemonic		Description	Byte	Oscillator Period
SUBB	A,@Ri	Subtract indirect RAM from Acc with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment indirect RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

LOGICAL OPERATIONS

Mnemonic		Description	Byte	Oscillator Period
ANL	A,Rn	AND register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12

LOGICAL OPERATIONS Cont.

Mnemonic		Description	Byte	Oscillator Period
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12

DATA TRANSFER

Mnemonic		Description	Byte	Oscillator Period
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12

DATA TRANSFER Cont.

Mnemonic		Description	Byte	Oscillator Period
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Oscillator Period
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to Carry	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct bit is set	3	24
JNB	bit,rel	Jump if direct bit is Not set	3	24
JBC	bit,rel	Jump if direct bit is set & clear bit	3	24

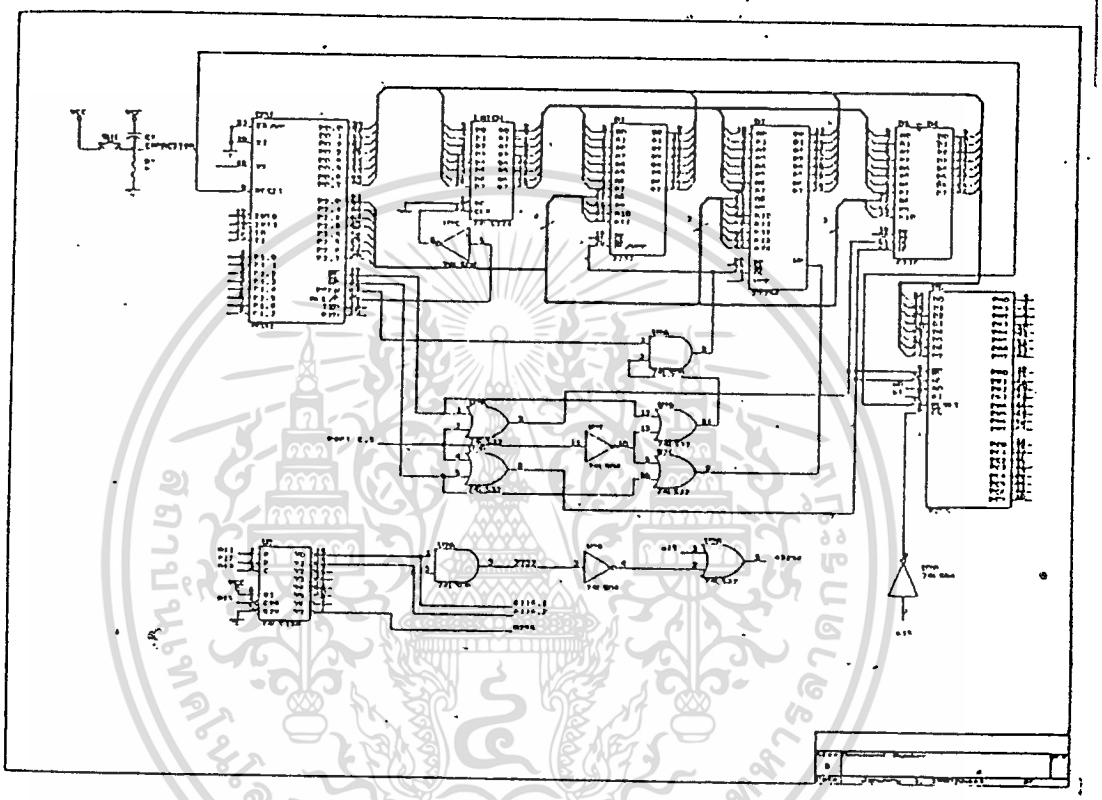
PROGRAM BRANCHING

Mnemonic		Description	Byte	Oscillator Period
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24

PROGRAM BRANCHING Cont.

Mnemonic		Description	Byte	Oscillator Period
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	3	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

ซิงเกิลบอร์ด MCS-51 มีวงจรตามรูป



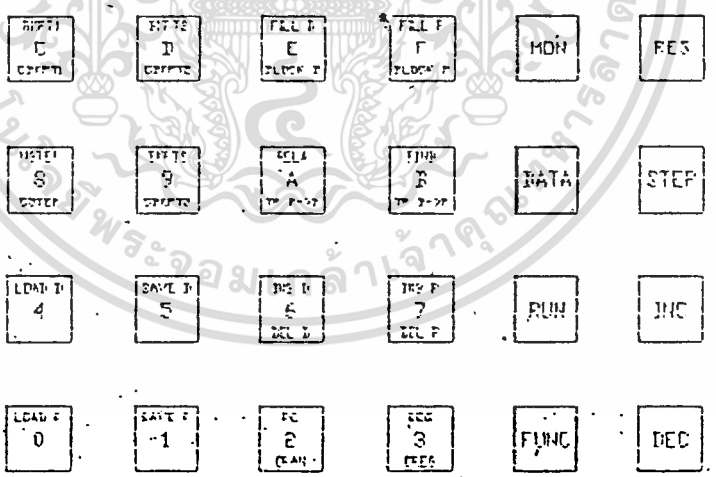
RAM D2 มีขนาด 32 กิโลไบท์ ใช้สำหรับเก็บโปรแกรม เนื่องจากหน่วยความจำตำแหน่ง 0000H ถึง 0FFFH ได้ถูกใช้งานสำหรับมอนิเตอร์โปรแกรม RAM D2 จึงใช้สำหรับเก็บโปรแกรมตำแหน่ง 1000H ถึง 7FFFH

RAM D3,D4 มีขนาดตัวละ 2 กิโลไบท์ ใช้สำหรับเก็บข้อมูล D3 จะใช้สำหรับตำแหน่ง 1000H ถึง 17FFH ,D4 ใช้สำหรับตำแหน่ง 1800 ถึง 1FFFH

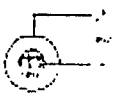
8255 D5 เป็นพอร์ทแบบขนาน ซึ่งใช้ในระบบ เพื่อจะได้ไม่ต้องใช้พอร์ทบนซิงเกิลชิพเอง ให้ผู้ใช้สามารถใช้งานได้ 8255 เป็นอินพุท/เอาท์พุทพอร์ท แบบโปรแกรมได้ ในซิงเกิลบอร์ด ใช้ 8255 สำหรับอ่านการกดคีย์บอร์ด, แสดงผลและต่อกับลำโพง 8255 ใช้เนื้อที่ของหน่วยความจำ สำหรับเก็บข้อมูลในตำแหน่ง 00-03

5.1 ฟังชั่นและการใช้งานซิงเกิลบอร์ด

การใช้งานซิงเกิลบอร์ด



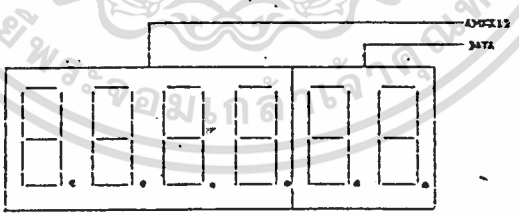
1. ไฟเลี้ยง เป็นแจ็คสำหรับเสียบไฟเข้าสู่ซิงเกิลบอร์ด ที่จุดนี้จะป้อนไฟ DC เข้า 0-15 โวลท์ โดยมีการต่อแจ็ค ดังรูป



หรือใช้อะแดปเตอร์ ซึ่งให้มากับเครื่อง

2. แสดงผล เป็นส่วนแสดงผล แบบ 7 Segment 6 หลัก ดังรูป

4 หลักแรก ใช้สำหรับแสดงค่าแอดเดรสของหน่วยความจำ และ 2 หลักหลัง แสดงข้อมูลในหน่วยความจำตำแหน่งที่แสดงค่าไว้ การแสดงผลของ 7-Segment จะแสดงค่าในรูปเลขฐาน 16



Segment เดียวกันของทุก Digit จะต่อเข้าด้วยกับ พอร์ท A ของ 8255 โดยมีการต่อ ดังนี้

Segment	a	ต่อกับบิท	0
	b	"	1
	c	"	2
	d	"	3
	e	"	4
	f	ต่อกับบิท	5
	g	"	6
	dt	"	7

คอมมอนของทั้ง 6 Digit ต่อกับเอาท์พุทของไอซีเบอร์ 74LS145 ถูกต่อไว้สำหรับดีโค้ดบิท 0=>2 ของพอร์ท C ของ 8255

3. คีย์บอร์ด คีย์บอร์ดในชิงเกิลบอร์ด MCS-51 ต่อไว้แบบเมตริก โดยมีการส่งสัญญาณสแกนคีย์บอร์ดทางเอาท์พุทของ 74LS145 และอ่านการกดคีย์บอร์ดเข้าทางบิท 4-7 ของพอร์ท C ในการสแกนคีย์บอร์ด จะทำการสแกนทีละ ROW จากบิท 0 ถึงบิท 2

คีย์บอร์ดจะแบ่งเป็น 2 กลุ่มตามการทำงาน

1. กลุ่มตัวเลขมีจำนวน 16 คีย์ สำหรับป้อนตัวเลขฐาน 16 (0-F) เพื่อใช้งานร่วมกับกลุ่มคีย์บอร์ดที่สองในการป้อนแอดเดรส, ข้อมูล, กลุ่มของคีย์บอร์ดนี้อยู่ในพื้นที่ A จากรูป

2. กลุ่มฟังชั่น เป็นกลุ่มที่ใช้เพื่อการเลือกการทำงานของชิงเกิลบอร์ด กลุ่มฟังก์ชั่นนี้ ใช้ร่วมกับกลุ่มตัวเลข ในบางคำสั่ง

2.1 RESET เป็นปุ่มรีเซทฮาร์ดแวร์ของชิงเกิลบอร์ด โดยปุ่มนี้จะต่อขา RST ของ CPU เข้ากับไฟเลี้ยง เพื่อทำการรีเซทระบบ เมื่อกดรีเซท หน่วยความจำทั้งหมดจะไม่มีการเปลี่ยนแปลง ยกเว้นในรีจิสเตอร์บางค่าเท่านั้น ดังต่อไปนี้

PC	=	0000	TH0	=	00
ACC	=	00	TL0	=	00
B	=	00	TH1	=	00
PSW	=	00	TL1	=	00
SP	=	07	SCON	=	00
DPTR	=	0000	SBUF	=	ไม่แน่นอน
P0-P3	=	0FFH	PCON	=	0XXX0000
IP	=	XX000000			
IE	=	0X000000			
TMOD	=	00			
TCON	=	00			

2.2 MON (Monitor) เป็นปุ่มที่ใช้สำหรับการกลับเข้าสู่ Prompt เพื่อการเรียกการทำงานของฟังก์ชั่นต่างๆ ต่อมา เมื่อเข้าสู่ Prompt จะแสดงเครื่องหมาย '-' บน Display ดังตัวอย่าง การกดคีย์ Monitor ตามด้วยการอ่านข้อมูลจากตำแหน่ง 2000 ซึ่งสมมุติให้มีค่า 12

กดคีย์	DISPLAY
MON	-
2 0 0 0	2000
DATA	2000 12

2.3 DATA เป็นปุ่มเพื่อแสดงข้อมูลของหน่วยความจำ ในตำแหน่งที่แสดง 4 หลัก ทางซ้ายของ Display การกดปุ่ม DATA ครั้งแรกจะเป็นการแสดงข้อมูลในหน่วยความจำ กดปุ่ม Data ครั้งที่ 2 เป็นการแสดงข้อมูลในส่วนของ Data area จะมีการแสดงจุดทศนิยมระหว่างตัวเลขของแอดเดรสกับข้อมูล(หลักที่ 4 และหลักที่ 5) เพื่อเป็นการบอกว่า เป็นค่าจากหน่วยความจำของข้อมูล เช่น ตัวอย่างการอ่านค่าจากหน่วยความจำตำแหน่ง 2000 ซึ่งกำหนดให้ในหน่วยความจำ สำหรับโปรแกรมมีค่า FAH และในหน่วยความจำตำแหน่งเดียวกันในหน่วยความจำ สำหรับข้อมูลมีค่า 12 การกดคีย์ ตามลำดับจะให้ผล ดังนี้

กดคีย์	DISPLAY
MON	-
2 0 0 0 DATA	2000 FA
DATA	2000.12

เมื่อกดปุ่ม DATA เพื่อเข้าสู่การติดต่อกับหน่วยความจำ สำหรับโปรแกรมหรือข้อมูลตามต้องการแล้ว การกดปุ่มกลุ่มตัวเลขฐาน 16 จะเปลี่ยนแปลงค่าในหน่วยความจำตำแหน่งนั้น โดยค่าที่ใส่เข้าไปใหม่ จะปรากฏที่หลักที่ 6 ของ Display และค่าในหลักที่ 6 เดิม จะถูกเลื่อนไปอยู่หลักที่ 5 ข้อมูลในหลักที่ 5 เดิมจะถูกทิ้งไป หมายความว่า ข้อมูลในตำแหน่งเดิมจะถูกเลื่อนไปทางซ้าย 4 บิท และใส่ข้อมูลใหม่ 4 บิทเข้าไปในตำแหน่งบิท 0 ถึงบิท 3 การป้อนข้อมูลใหม่เข้าไป 2 Digit ก็จะเลื่อนไปทางซ้ายทุกครั้ง และเมื่อครบ 2 Digit ค่าของ Address จะเพิ่มขึ้นไป 1 ดังตัวอย่างการแก้ข้อมูลในหน่วยความจำ ตำแหน่ง 2000 ของหน่วยความจำข้อมูล จากเดิม 12 เป็น 34 จะทำการกดคีย์ต่อจากตัวอย่างข้างบน ดังนี้

กดคีย์	DISPLAY
	2000.12
3	2000.23
4	2000.34

และชั่วขณะหนึ่ง DISPLAY จะเปลี่ยนเป็น

2001.00

สมมุติให้ แอดเดรส 2001 มีข้อมูลเป็น 00

2.4 ปุ่ม INC และ DEC เป็นปุ่มที่ใช้สำหรับเพิ่มหรือลดค่าของแอดเดรสที่แสดงใน 4 หลักทางซ้ายของ DISPLAY ไป 1 และจะแสดงค่าของหน่วยความจำในตำแหน่งนั้น โดยขึ้นกับว่ากำลังอยู่ในหน่วยความจำสำหรับโปรแกรม หรือข้อมูล ดังตัวอย่างเป็นการลดค่าของแอดเดรสจาก 2001 เป็น 2000 และเพิ่มเป็น 2001 ตามเดิม ต่อจากตัวอย่างในข้อ 2.3

	DISPLAY
DEC	2000.34
INC	2001.00

2.5 STEP ใช้เพื่อการทดสอบการทำงานของโปรแกรมทีละคำสั่ง เพื่อการตรวจสอบค่าของรีจิสเตอร์ภายใน , โปรแกรมเคาร์เตอร์, แฟลก ต่อไป ตัวอย่างสั่งให้ทำงานทีละSTEP จากแอดเดรส 2000 เป็นการเปลี่ยนค่าของแอคคิวมูเลเตอร์ (A) ให้เท่ากับ 00 โดยให้ข้อมูลในหน่วยความจำของโปรแกรม ตำแหน่ง 2000.2001 และ 2002 มีค่า 75 , 00 , 00 ตามลำดับ เป็นคำสั่ง MOV A,# 0 ตามด้วยคำสั่ง NOP

กดคีย์	DISPLAY
MON	_
2 0 0 0	2000
STEP	2002

2.6 REG Register ใช้สำหรับแสดงค่าของรีจิสเตอร์ต่างๆ ปุ่ม REG จะใช้งานร่วมกับปุ่มINC และ DEC เป็นการเปลี่ยนรีจิสเตอร์ที่แสดงผล ขณะที่ชื่อของรีจิสเตอร์ปรากฏทางซ้าย ค่าของรีจิสเตอร์จะปรากฏทางขวามือดังตัวอย่าง เป็นการแสดงค่าของรีจิสเตอร์ A ซึ่งมีค่าเท่ากับ 00 และ A เป็นรีจิสเตอร์แรกที่จะแสดงผล

	DISPLAY
FUNC	FUNC 1
REG	A 00

การกดปุ่ม INC หรือ DEC เพื่อดูค่าในรีจิสเตอร์จนหมด จะทำให้กลับสู่มอนิเตอร์โดยอัตโนมัติ การแสดงค่าของรีจิสเตอร์ จะเรียงตามลำดับ ดังนี้

A	หมายถึง	Accumulator
A0	=	R0
A1	=	R1
A2	=	R2
A3	=	R3
A4	=	R4
A5	=	R5
A6	=	R6
A7	=	R7

ขณะที่แสดงค่ารีจิสเตอร์อยู่ ถ้ามีการกดปุ่มในกลุ่มตัวเลข จะทำให้ค่าของรีจิสเตอร์นั้นเปลี่ยนไป โดยเปลี่ยนใน 4 บิทล่างของรีจิสเตอร์นั้นก่อน โดยเลื่อน 4 บิทล่างเดิมไปไว้ที่ 4 บิทบน การกดปุ่มข้อมูล 2 ครั้ง จะทำให้เปลี่ยนรีจิสเตอร์แสดงผลไปโดยอัตโนมัติ หากมีการกดปุ่ม INC, DEC เกินช่วงของ A, R0 - R7 โปรแกรมจะกลับสู่ Prompt โดยอัตโนมัติ

2.7 RUN เป็นปุ่มสำหรับสั่งให้มีการทำงานนับจากตำแหน่งที่แสดงอยู่ทางซ้ายของ DISPLAY โดยให้เริ่มจากการกดปุ่ม MON เพื่อจะทำการป้อนแอดเดรสที่ต้องการ RUN เช่นต้องการ RUN โปรแกรมตั้งแต่ตำแหน่ง 2000 .

MON	
2 0 0 0	2000
RUN	ไปทำงานตำแหน่ง 2000

ซิงเกิลบอร์ดจะเริ่มทำงานจากตำแหน่ง 2000 เป็นต้นไป

2.8 Breakpoint ใช้เพื่อทดสอบการทำงานของโปรแกรม โดยเมื่อ CPU fetch คำสั่งถึงตำแหน่ง Breakpoint แล้วจะกลับเข้าสู่โปรแกรมมอนิเตอร์เพื่อให้ผู้ใช้สามารถตรวจหรือเปลี่ยนแปลงค่าในหน่วยความจำ, รีจิสเตอร์ และอื่น ๆ ในการตั้ง Break นี้ทำไว้ให้สามารถตั้ง Break ได้ถึง 3 จุด สามารถตั้ง, ยกเลิก จุดใดก็ได้ ต.ย. เช่น มีโปรแกรม ดังนี้

```
2000    MOV    A,#0
2002    MOV    R0,#3
2004    MOV    P1,#4
2006    MOV    @R0,#A
```

และต้องการให้โปรแกรมหยุดที่ตำแหน่ง 2006 จะเห็นว่าตำแหน่งนี้จะต้องเป็นจุดสุดท้ายต่อจากคำสั่งใด ๆ หรือต้นคำสั่งต่อไปนั่นเอง มีวิธีการใช้ดังนี้

```
FUNC                FUNC1
BKPT1   2006        2006
DATA                2006
MON                 -
2000                2000
RUN                 2006
```

หากต้องการยกเลิกก็สามารถทำได้ดังนี้

```
FUNC    FUNC        FUNC2
CBKPT1  MON         -
```

2.9 NSTE ใช้เพื่อกำหนดจำนวนคำสั่งที่ถูก fetch ต่อการกด step หนึ่งครั้ง เช่นตั้ง NSTEP = 2 จะทำให้ fetch ทีละ 2 คำสั่ง เช่นมี program ดังตัวอย่างของการตั้ง Break point

```
FUNC                FUNC1
NSTEP   02          02
MON                 -
```

2000 2000

STEP 2004

หากต้องการให้ทำงาน Step ละ 1 ครั้ง ก็จะต้องทำการ Clear step

FUNC FUNC1

CSTEP -

โดยกดปุ่มตามตัวอย่าง

2.10 PC เป็นการขอดูค่า address ที่จะ fetch ต่อไปใน USER PROGRAM จะใช้ในกรณีที่ USER ต้องการทราบว่า add ต่อไปที่จะ fetch คือ add ใด เช่น ขณะนี้ monitor ค่าของ Register อยู่ และ Program ได้ fetch ถึงตำแหน่ง 2002 แล้วสามารถขอดู PC ได้ดังตัวอย่าง

MON -

FUNC FUNC

PC 2002

2.11 CLREG เป็นการใส่ค่า REG A, RO->R7 ให้มีค่า = 0 วิธีการใช้งานดังตัวอย่าง

FUNC FUNC FUNC 2

CLREG MON -

2.12 CLRAM เป็นคีย์ที่ใช้เพื่อ Clear ค่าใน Program area แอดเดรส 2000 7FFF และ Data area แอดเดรส 1000-2000 ให้มีค่าเป็น 0 ดังตัวอย่าง

FUNC FUNC

CLRAM MON -

2.13 BLOCK คำสั่งนี้ใช้ในสองลักษณะคือ ลักษณะแรกใช้เพื่อผ่านข้อมูลจาก add1 : -> add2 ไปไว้ที่ add3 ตัวอย่างเช่น ต้องการย้ายข้อมูลจาก 2000 ถึง 2050 ไปไว้ที่ 2500 แต่ต้องระวังไม่ให้พื้นที่ที่อยู่เดิมคือ 2000-2050 ไม่ให้ทับกับพื้นที่เป้าหมาย คือ 2500 โดย Block p จะทำกับ ส่วนโปรแกรม Block d ทำกับ Data area

FUNC FUNC 2

BLOCK FUNC

```
2000    DATA        2000
2050    DATA        2050
2500    DATA        2500
MON                 -
```

ส่วนอีกลักษณะหนึ่งจะใช้เพื่อกำหนด add 2 ค่า สำหรับคำสั่งอื่น ๆ ต่อไป โดยต่อไปจะอ้างถึงเป็น add1 และ add2 การกดปุ่มต่างกันตรงที่ไม่ต้องใส่ add3 ที่จะย้ายไปดังแบบแรกให้กด MON แทนซ้ำไปเลย คำสั่งตั้ง add 1 , add 2 นี้สามารถใช้ Block d หรือ Block p ได้โดยมีผลเหมือนกัน คำสั่งต่อไปจึงจะชี้ว่าจะทำกับ Data หรือ Program ดังตัวอย่าง ต้องการให้ add1 = 2013 add2 = 2030 ต้องกดดังนี้

```
FUNC                FUNC
BLOCK               FUNC
2013    DATA        2013
2030    DATA        2030
MON                 -
```

2.14 FILL เป็นการใส่ค่าหนึ่ง ๆ จาก add1 ไปจนถึง add2 โดย Fill d หมายถึง Fill Data area Fill p หมายถึง Fill Program area ตัวอย่างเช่นต้องการใส่ Data HFE ตั้งแต่ H2013 ถึง H2030 โดยกำหนด add1, add2 ด้วยคำสั่ง Block ดังตัวอย่างข้างต้น แล้วกดปุ่มดังนี้

```
FUNC                FUNC
FILL  F    E        FE
MON                 -
```

2.15 RELA ใช้สำหรับหาค่า relative jump เช่นมี program ดังนี้

```
2000            CJNE  R0, # 7 , NEXT
2003            MOV   R0, # 2
2005    NEXT:   MOVX  @DPTR,A
```

จะต้องป้อน add1, add2 ด้วยคำสั่ง Block โดยให้ add1 = 2003 คือ จบลิ้นค่าที่มีการกระโดดแบบ Relative Jump และ add2 = 2005 คือตำแหน่งที่จะกระ

โดดไป จากนั้น กดปุ่ม RELA จะได้ค่า Relative Jump ออกมาดังนี้

```
FUNC                    FUNC
RELA                    XX
```

2.16 TRANSFER ใช้เมื่อย้าย Data ไปมาระหว่าง Program area <-> data area ใช้ประโยชน์ เช่น ในกรณีที่ต้องการ Run Program ของ USER จาก EPROM ที่ใส่ใน Data area โดยจะย้ายจาก add1 -> add2 ไปยังจุดที่กำหนด เช่น ต้องการย้ายข้อมูลจาก Data -> Program จาก 2000 -> 2050 โดยใช้คำสั่ง Block ให้ add1 = 2000 add2 = 2050 และจะนำไปไว้ที่ตำแหน่ง 2200 ให้กดปุ่มดังนี้

```
FUNC                    FUNC
TR  D->P   2200         2200
DATA                    2200
MON
```

2.17 INS. , DEL เป็นคำสั่งที่จะลบหรือแทรกข้อมูล โดยจะต้องกำหนด add1 และ add2 ใน Block Commmond เพื่อให้ monitor รู้ว่าจะเลื่อน add สุดท้ายที่ใด เช่นการแทรกจะต้องมีการเลื่อนต่อกันไปถึง add2 ดังตัวอย่าง ต้องการแทรกข้อมูลที่ add 2030 ไปอีก 3 ค่า โดยทำในช่วง 2000 ถึง 2500 ให้ตั้ง add1 = 2000 add2 = 2500 ให้กดปุ่มดังนี้

```
FUNC                    FUNC
INS   03                .03
DATA  MON
```

2.18 LOAD, SAVE ใช้เพื่อบันทึกหรืออ่านข้อมูลจาก Tape หรือ Serial โดยต้องกำหนด add1 เป็นตำแหน่งเริ่มต้น และ add2 ใช้เฉพาะการ Save เป็น add สุดท้าย หากเป็นการ Load ให้ใส่ add2 เป็นค่าใดก็ได้ เช่นต้องการ Load เริ่มต้นที่ add 2000 เป็นต้นไปให้ ass1 = 2000 ส่วน add2 = 2000 ด้วย แล้วกดปุ่มดังนี้แล้วจึงเปิด Tape

```
FUNC                    FUNC
LOAD
```

หากเป็นการ Save ต้องใส่ทั้ง addl และ add2 แล้วกดดังนี้ โดยต้องเริ่มอัด Tape ก่อน

FUNC FUNC

SAVE

สักครู่จะแสดงผลดังนี้

การใช้ Interrupt

Interrupt routine จะมีการกำหนดเอาไว้ใน ROM เนื่องจากได้มีการนำเอา INTO ไปใช้ในการทำ Single step แล้วจึงเหลือ Interrupt ที่ผู้ใช้งานสามารถใช้ได้อีก 4 ตัว และได้กำหนดจุดเริ่มต้นใหม่ให้อยู่ใน RAM โดยมีตำแหน่งต่างๆดังนี้

Timer 0 overflow	H1200
External interrupt 1	H1300
Timer 1 overflow	H1400
Serial port	H1500

Buffer ของ RAM ภายใน

ในระบบการทำงานของโปรแกรมมอนิเตอร์นั้นต้องมีการใช้ RAM บางส่วนเพื่อใช้เป็น Buffer RAM ภายในโดยผู้ใช้งานสามารถที่จะดู RAM ภายในได้จาก Program area ตำแหน่ง H1A00 - H1A7F และขณะที่มีการเปลี่ยนการทำงานระหว่าง Monitor program กับ User program จะมีการแลก RAM ที่ใช้กับ Monitor ด้วย ดังนั้นผู้ใช้งานจึงไม่สามารถใช้ RAM ตำแหน่งดังกล่าว

กิตติกรรมประกาศ

โครงการปริญญานิพนธ์นี้คงไม่สามารถสำเร็จลงได้ถ้าหากขาดบุคคลเหล่านี้ ที่ให้แนวคิด ให้ความช่วยเหลือ เอื้อเฟื้ออุปกรณ์ในการทดลอง และ ที่สำคัญให้ความมั่นใจ ซึ่งสิ่งเหล่านี้นับว่าสำคัญมาก และ ขอขอบพระคุณมา ณ. ที่นี้ด้วย

1 ดร.วัลลภ สุระกำพลธร

2 รศ.ดร.สิทธิชัย โภไคยอุดม

3 อ.วันชัย ริ้วรุจา

4 คุณ สุเจตน์ จันทรังษ์

5 คุณ ธำรงศักดิ์ สุกใส

6 คุณ จิตเกษม งามนิล

7 คุณ กาญจน์ กาญจนรัตน์

...ที่ให้ความมั่นใจ

8 คุณ ปริญดา เมฆรา

...ที่เป็นกำลังใจ

เอกสารอ้างอิง

ก) เอกสารอ้าวอิงที่เป็นภาษาไทย

1 ประกิจ ต้งติสานนท์
ทฤษฎีโทรทัศน์ หลักการ และ เครื่องรับโทรทัศน์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

2 ศุภชัย เกียรติชนะบำรุง
INTRODUCTION TO IMAGE PROCESSING
ปริญญานิพนธ์ , 55 หน้า , 2528

3 สุพรชัย โชติพุทธิกุล, สุพัฒน์ เอี่ยมวิวัฒน์ และ อนุวัฒน์ พฤกษ์พิจารย์
แผ่นวงจรควบคุมภาพ
ปริญญานิพนธ์ , 37 หน้า , 2531

ข) เอกสารอ้างอิงภาษาต่างประเทศ

1 STEVE CIARCIA
BUILD A GRAY-SCALE VIDEO DIGITIZER
PART 1: DISPLAY/RECEIVER
MAY , 1987 . BYTE PAGE 95
PART 2: DIGITIZER/TRANSMITTER
JUNE , 1987 . BYTE PAGE 129

2 GERRY KANE
CRT CONTROLLER HANDBOOK