ปีการศึกษา 2531

แผงวงจรควบคุมภาพ (IMAGE INTERFACE CARD)

โดย

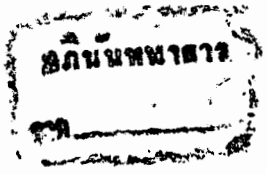
นายสุพรชัย โชติพุทธิกุล

นายสุพัฒน์ เอี่ยมวิวัฒน์

นายอนุวัฒน์ พฤกษ์พิจารณ์

อาจารย์ที่ปรึกษา

ผศ.ดร. พูศักดิ์ ชีวสุวิทย์



คำนำ

ในปัจจุบันเทคโนโลยีทางด้าน Image Processing ได้แพร่หลายในเมืองไทยมากพอสมควร และมีแนวโน้มว่าต่อไปในอนาคตจะมีการเอาเทคโนโลยีทางด้านนี้ไปประยุกต์ใช้กับงานต่างๆอีกมากมาย โครงงานนี้เป็นการศึกษาและทดลองเกี่ยวกับ Image Processing ซึ่งแบ่งออกเป็นสองส่วนคือ การศึกษาและทดลองทาง Hardware และ Software ในส่วน Hardware จะทำให้สำเร็จออกมาในรูปของ Image Interface Card และในส่วนของ Software จะศึกษาในด้านการประมวลภาพด้วยโปรแกรม

อย่างไรก็ตามการศึกษาและทดลองของคณะผู้จัดทำ ก็ยังเป็นแนวทางเบื้องต้นเท่านั้น อาจจะมีข้อผิดพลาดอยู่บ้าง จึงขออภัยมา ณ ที่นี้ด้วย และจักขอบพระคุณเป็นอย่างยิ่งหากท่านผู้อ่านได้กรุณาชี้ข้อผิดพลาดหรือบอกแนวทางแก้ไขให้กับคณะผู้จัดทำ

สุดท้ายนี้ ขอขอบพระคุณ อาจารย์ ผศ.ดร. ฟูศักดิ์ ชีวสุวิทย์ ที่ได้กรุณาให้คำแนะนำในเรื่องต่าง ๆ เป็นอย่างดี และคณะผู้จัดทำหวังว่าปริญญานิพนธ์ฉบับนี้คงจะเป็นประโยชน์แก่ท่านผู้อ่านบ้างไม่มากก็น้อย

คณะผู้จัดทำ

แผ่นวงจรควบคุมภาพ

สุพรชัย โชติพุทธิกุล

สุพัฒน์ เอี่ยมวิวัฒน์

อนุวัฒน์ พฤกษ์พิจารณ์

ผศ.ดร. ฟูศักดิ์ ชีวสุวิทย์

ปีการศึกษา 2532

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เสนอการออกแบบระบบประมวลผลภาพแบบเวลาจริง (Real time image processing system) โดยสร้างแผงวงจรควบคุมภาพ(Image interface card) ที่ประมวลผลโดย IBM PC ซึ่งมีคุณสมบัติที่สำคัญคือ สามารถเก็บภาพแบบเวลาจริง(Real time image acquisition) ที่ได้จากกล้องโทรทัศน์ , วีดีโอเทป หรืออุปกรณ์ที่สามารถผลิตสัญญาณโทรทัศน์ มาเก็บไว้ในรูปของข้อมูลที่สามารถนำมาประมวลผลโดยคอมพิวเตอร์ต่อไปได้ และยังสามารถแสดงภาพของข้อมูลที่เก็บได้นั้นออกทางจอมอนิเตอร์ โดยการแปลงกลับให้เป็นสัญญาณโทรทัศน์อีกครั้งหนึ่ง ภาพที่เก็บได้หรือภาพที่แสดงโดยแผงวงจรคุมควบภาพจะเป็นภาพขาวดำที่มีความละเอียด 256x256 จุดซึ่งต้องใช้หน่วยความจำ 64 กิโลไบต์ในการประมวลผลภาพ 1 ภาพ

Supornchai Chotputtikul

Supat Iamwiwat

Anuwat prukpichan

Asistant Professor Fusak Cheevasuvit

1989

ABSTRACT

The thesis describes the hardware design of a real-time image processing card for working on a microcomputer IBM PC and also the software design for image processing. Image data can be gotten from close circuit TV or video camera with the resolution of 256 x 256 pixels for each image.

บทนำ

บทที่ 1

บทนำ

เรื่องราวเกี่ยวกับวิธีการประมวลผลภาพได้รับความสนใจเป็นอย่างยิ่ง ทั้งนี้เนื่องมาจากจินตนาการของมนุษย์ที่สำคัญ 2 ประการ คือ

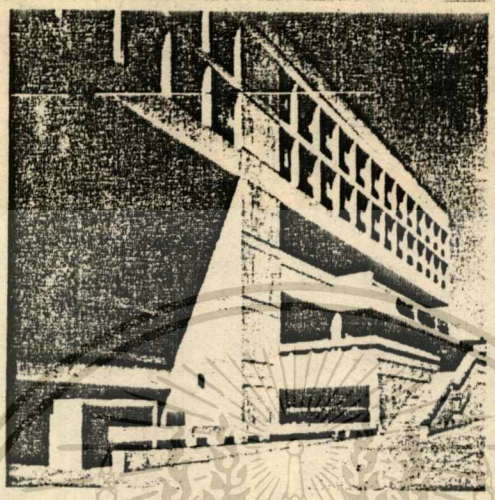
1.1 การปรับปรุงภาพให้ดีขึ้นเพื่อการตีความของมนุษย์เองดังเช่น ภาพที่ได้จากดาวเทียม LANDSAT (เป็นดาวเทียมเพื่อการสำรวจทรัพยากร) ภาพที่ส่งมาจะมีลักษณะไม่ชัดเจนยากต่อการตีความว่าพื้นที่ส่วนใด คือ พื้นที่ทางกสิกรรม พื้นที่ส่วนใด คือ ป่าไม้ และ ฯลฯ ดังนั้นมนุษย์จึงได้พยายามหาวิธีการประมวลผลภาพ ซึ่งเป็นวิธีการทางซอฟแวร์โดยผ่านขบวนการทางคณิตศาสตร์เพื่อให้ได้ภาพที่สามารถตีความได้ดียิ่งขึ้น

1.2 ความพยายามที่จะให้เครื่องจักรสามารถเข้าใจถึงภาพที่เห็นได้ ดังเช่น ความพยายามที่จะพัฒนาหุ่นยนต์ ให้มีความสามารถใกล้เคียงมนุษย์ สามารถแยกแยะวัตถุต่าง ๆ ออกจากกันได้ ซึ่งเป็นเรื่องของศาสตร์ทางด้าน pattern recognition

ในปี ค.ศ.192Ø ได้มีการทดลองส่งภาพหนังสือพิมพ์ที่ผ่านการดิจิไตซ์ (digitized) ระหว่างกรุงลอนดอนกับนิวยอร์คผ่านเคเบิลใต้น้ำข้ามมหาสมุทรเป็นครั้งแรก ในการทดลองครั้งนั้นช่วยลดระยะเวลาในการส่งข้อมูลซึ่งจากเดิมต้องใช้เวลาเป็นอาทิตย์ลงเหลือเพียง 3 ชั่วโมง โดยการนำข้อมูลภาพที่เป็นข้อมูลดิจิตอลมาเข้ารหัสแล้วส่งไปทางด้านรับ เมื่อรับสัญญาณมาก็ถอดรหัสออกแล้วพิมพ์ภาพในแบบ half tone ออกมา

ปัญหาในระยะแรกของระบบการประมวลผลภาพ คือ ระดับของความสว่าง (gray level) ซึ่งมีเพียง 5 ระดับ ทำให้ภาพที่ได้มีลักษณะหยาบแต่อย่างไรก็ตามปัญหานี้ก็ได้รับการแก้ไขมาเรื่อยๆ จนในปี ค.ศ. 1929 สามารถพัฒนาได้เป็น 15 ระดับ ดังรูป 1.1

อย่างไรก็ตามระบบการประมวลผลภาพก็ยังใช้อยู่ในวงจำกัด ทั้งนี้เพราะการประมวลผลภาพใช้เนื้อที่หน่วยความจำสูงมาก ซึ่งคอมพิวเตอร์ในสมัยก่อนยังมีขีดความสามารถจำกัดอยู่ จะมีใช้อยู่ก็ในเครื่องระดับเมนเฟรม จนมาถึงปัจจุบันเทคโนโลยีทางด้าน VLSI ได้รับการพัฒนาไปไกลมาก คอมพิวเตอร์ส่วนบุคคลยุคใหม่มีขีดความสามารถสูง จึงได้มีการพัฒนาระบบประมวลผลภาพให้เข้ากับ



รูปที่ 1.1 ภาพที่มีระดับความสว่าง 15 ระดับ

คอมพิวเตอร์ส่วนบุคคลหรือไมโครคอมพิวเตอร์กันอย่างจริงจัง ในต่างประเทศได้มีการนำระบบประมวลผลภาพไปประยุกต์ใช้กับงานด้านการออกแบบทรงผมให้เข้ากับใบหน้าของลูกค้า โดยมีการถ่ายภาพใบหน้าของลูกค้าเก็บเป็นข้อมูล แล้วนำแบบทรงผมต่าง ๆ ซึ่งเก็บอยู่ในรูปข้อมูลของคอมพิวเตอร์มาทดลองผสมเข้ากับใบหน้าของลูกค้า จนได้ทรงผมที่ถูกใจของลูกค้ามากที่สุด แล้วจึงค่อยตกแต่งทรงผม นอกจากนี้ยังมีการนำระบบการประมวลผลภาพไปประยุกต์ใช้กับงานด้านต่าง ๆ อีกมากมาย สุดแล้วแต่จินตนาการของมนุษยชาติ ท่านผู้อ่านก็เป็นอีกท่านหนึ่งที่อาจจะมีความคิดที่จะนำไปประยุกต์ใช้กับงานของท่านบ้างก็เป็นได้

ในโครงงานที่นำเสนอในปริญญานิพนธ์ฉบับนี้ ได้ทำการพัฒนาระบบประมวลผลภาพที่มีความละเอียด 256 x 256 จุดต่อภาพเข้ากับ IBM PC ซึ่งมีระดับความสว่างในการแสดงภาพ 256 ระดับ ในการดิจิไตซ์ภาพจะมีระดับความสว่าง 64 ระดับ ทั้งนี้เนื่องจากต้องการลดต้นทุนในการผลิตให้มีราคาต่ำที่สุด

ทฤษฎี

บทที่ 2

ส่วนประกอบต่าง ๆ ในระบบประมวลผลภาพ

(Element of a image processing system)

ในระบบประมวลผลภาพประกอบไปด้วยส่วนต่าง ๆ ดังนี้

2.1 ตัวประมวลผลภาพ (Image processor)

ตัวประมวลผลภาพนับได้ว่า เป็นหัวใจสำคัญของระบบประมวลผลภาพเพราะเป็นส่วนของฮาร์ดแวร์ที่มีหน้าที่ที่สำคัญอยู่ 4 ประการ คือ

2.1.1 การเก็บภาพ (Image acquisition) เป็นการแปลงสัญญาณโทรทัศน์ ซึ่งเป็นสัญญาณอนาล็อกให้เป็นข้อมูลดิจิตอลเพื่อเก็บในหน่วยความจำ โดยทั่วไปตัวประมวลผลภาพสามารถดิจิไตซ์ภาพจากสัญญาณโทรทัศน์ได้ภายในช่วงเวลาหนึ่งเฟรม และในโครงงานที่ทำอยู่ดิจิไตซ์เพียงฟิลด์เดียวในหนึ่งเฟรม

2.1.2 การบันทึก (Storage) ก็คือส่วนหน่วยความจำที่ใช้เก็บภาพที่ได้จากการดิจิไตซ์ ซึ่งจะใช้หน่วยความจำมากหรือน้อยขึ้นอยู่กับความละเอียดของจุดภาพ ในโครงงานนี้ ให้ความละเอียดของจุดภาพ 256 * 256 ซึ่งต้องใช้หน่วยความจำ 64 กิโลไบต์ เราเรียกส่วนนี้ว่าเฟรมบัฟเฟอร์ (Frame buffer)

2.1.3 การประมวลผลในระดับต่ำ (Low level processing) เป็นส่วนของฮาร์ดแวร์ที่ใช้ในการทำงานทางด้านลอจิกมักนิยมเรียกว่า ALU (Arithmetic Logic Unit) ออกแบบมาเพื่อใช้เพิ่มความเร็วของระบบ

2.1.4 การแสดงผล (Display) เป็นการอ่านข้อมูลภาพที่อยู่ในหน่วยความจำเพื่อเปลี่ยนให้เป็นสัญญาณโทรทัศน์ต่อไป

2.2 ดิจิไตเซอร์ (Digitizer)

เป็นส่วนที่แปลงภาพให้เป็นข้อมูลที่เหมาะสมทางด้าน NUMERICAL สำหรับใช้ในการประมวลผลต่อไป

2.3 คอมพิวเตอร์ (Computer)

เนื่องจากตัวประมวลผลภาพมีความสามารถเพียงแค่แปลงข้อมูลจากสัญญาณโทรทัศน์ให้มาอยู่ในรูปของข้อมูลดิจิตอลในหน่วยความจำ. แต่ในบางครั้งระบบการประมวลผลภาพจำเป็นต้องมีการคำนวณทางคณิตศาสตร์ที่ยุ่งยากหรืออาจจะมีความต้องการผสมตัวอักษรให้เข้ากับรูปภาพ แล้วบันทึกเก็บไว้และอื่นๆอีกมาก เพื่อสนองตอบต่อความต้องการเหล่านี้ เราจึงต้องทำการเชื่อมระบบของตัวประมวลผลภาพให้เข้ากับระบบคอมพิวเตอร์ทั่วไป ในโครงงานนี้เราได้ทำการเชื่อมระบบเข้ากับ IBM PC ซึ่งเป็นระบบคอมพิวเตอร์ที่ได้รับความนิยมทั่วโลก

2.4 อุปกรณ์ที่ใช้ในการเก็บข้อมูล (Storage device)

ภาพที่ได้จากตัวประมวลผลภาพมีความละเอียด 256x256 จุด แต่ละจุดมีระดับความสว่าง 256 ระดับ (8 บิท/จุด) ซึ่งต้องใช้อุปกรณ์ที่เก็บข้อมูลอย่างน้อย 64 กิโลไบต์ อุปกรณ์ที่ใช้เก็บข้อมูลที่นิยมกันได้แก่ ฟลอปปี้ดิสค์ซึ่งสามารถเก็บข้อมูลได้ 360 กิโลไบต์ , ฮาร์ดดิสค์ ซึ่งสามารถเก็บข้อมูลได้ตั้งแต่ 5 เมกกะไบต์ขึ้นไป และ แมกเนติกส์ดิสค์ เป็นต้น

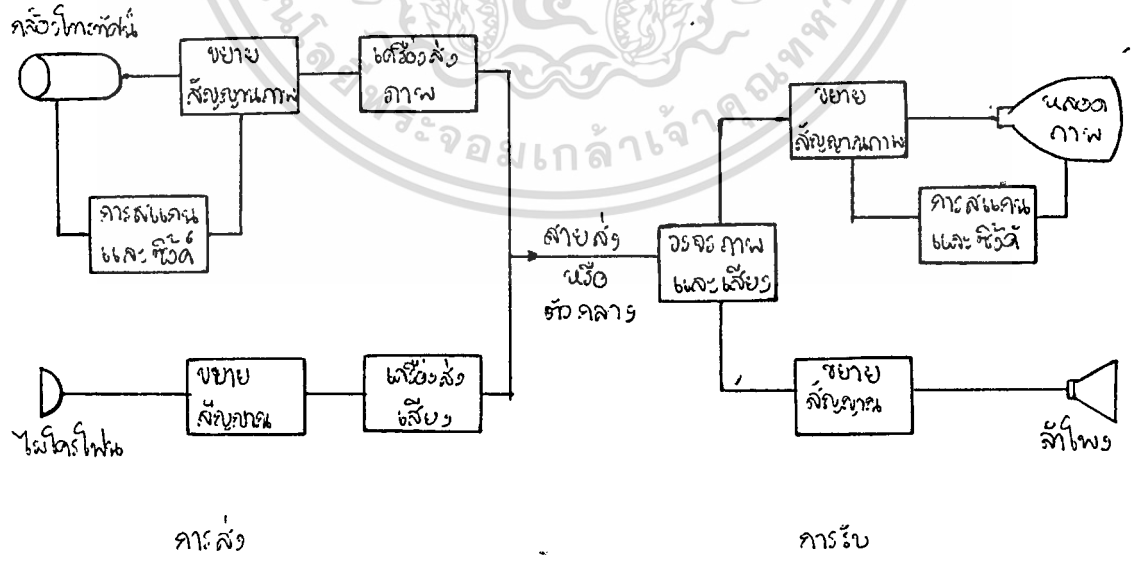
2.5 ส่วนแสดงภาพ (Display)

เป็นส่วนที่ใช้แสดงภาพที่ได้จากตัวประมวลผลภาพ, ภาพที่เก็บอยู่ในฟลอปปี้ดิสค์ และ ฯลฯ ซึ่งได้แก่ จอโมโนโครม หรือจอโทรทัศน์ ที่สามารถแสดงภาพที่มีระดับความสว่างแตกต่างกันมาก ๆ ได้ เครื่องพิมพ์ (Printer) ก็เป็นอุปกรณ์หนึ่งที่ใช้ในการแสดงภาพที่ไม่ต้องการระดับความสว่างมากนักทั้งนี้เนื่องจากในแต่ละจุดภาพของเครื่องพิมพ์จะให้ระดับความสว่างเพียง 2 ระดับเราจึงต้องมาสร้างเมตริกซ์ของจุดภาพที่มีขนาด nxn โดย n = 2,3,4,... ซึ่งแล้วแต่ว่าเราจะต้องการภาพที่มีความละเอียดแค่ไหน รวมทั้งความสามารถในการพล็อต (Plot) จุด/แถว ของเครื่องพิมพ์เองด้วย เราเรียกการแสดงภาพในลักษณะนี้ว่า half-tone picture นอกจากนี้ยังมีอุปกรณ์ที่ใช้ในการแสดงภาพได้อีก เช่น เลเซอร์พรินเตอร์ (LASER PRINTER) เป็นต้น

บทที่3

ระบบสัญญาณโทรทัศน์ (Composite signal system)

ในระบบโทรทัศน์นั้นจะต้องประกอบไปด้วย กล้องโทรทัศน์ทำหน้าที่ผลิตสัญญาณภาพ(Video Signals) และไมโครโฟนทำหน้าที่ผลิตสัญญาณเสียง (Audio Signals) โดยมีเครื่องแสดงผล (Monitor) ทำหน้าที่แสดงภาพและเสียงที่สร้างได้จากกล้องโทรทัศน์ และไมโครโฟนอีกครั้งหนึ่ง ดังรูป 3.1 และเพื่อให้การส่งและรับสัญญาณภาพได้อย่างถูกต้องจำเป็นต้องมีสัญญาณซิ้งค์ (Synchronizing Pulses) สำหรับควบคุมการทำงานให้ดำเนินไปอย่างถูกต้อง ทั้งนี้ยังมีสัญญาณอื่น ๆ ที่ช่วยให้เกิดภาพที่สมบูรณ์อีกโดยจะกล่าวละเอียดในส่วนต่อไป ในกรณีที่ต้องการส่งสัญญาณออกไปในระยะทางไกล ๆ สัญญาณเหล่านี้จะถูกมอดูเลท (Modulate) ด้วยคลื่นพาหะ (Carrier) ก่อน แล้วจึงจะทำการส่งไปได้ ในโครงงานนี้เพียงแต่นำสัญญาณโทรทัศน์ที่ได้จากอุปกรณ์กำเนิดสัญญาณโทรทัศน์ เช่น กล้องโทรทัศน์ , วีดีโอเทป ฯลฯ มาใช้ในขบวนการเก็บภาพเพียงอย่างเดียวโดยไม่มีการมอดูเลท ดังนั้นเนื้อหาที่จะกล่าวต่อไปจะเน้นเฉพาะส่วนของสัญญาณโทรทัศน์เท่านั้น



รูปที่ 3.1 ระบบการส่ง-รับโทรทัศน์

3.1 ส่วนประกอบของภาพ (Picture Elements)

ภาพขาว-ดำแต่ละภาพถ้าลองขยายให้ใหญ่ขึ้น จะพบว่าภาพประกอบด้วยจุดขาวและจุดดำมากมาย ดังเช่นภาพจากหนังสือพิมพ์ การจัดเรียงตัวของจุดขาวและจุดดำทำให้เกิดภาพขึ้นได้ จุดเหล่านี้เรียกว่า ส่วนประกอบของภาพ สำหรับในพื้นที่ที่เท่ากันแล้วภาพใดที่มีจำนวนระดับของความสว่าง (Level of Brightness) มากกว่าจะเป็นภาพที่คมชัดและกลมกลืนมากกว่า สิ่งนี้เป็นสาเหตุหนึ่งที่ทำให้เราไม่สามารถใช้ จอภาพโมโนโครมของ IBM PC/XT แสดงผลได้เพราะมีจำนวนระดับของความสว่างเพียง 2 ระดับ คือ เขียว กับ ดำ เท่านั้น

ก) หากมีจำนวนจุดดำมาก ภาพจะมองดูละเอียด

ข) หากมีจำนวนจุดดำน้อย ภาพจะมองดูหยาบ

รูปที่ 3.2 เปรียบเทียบภาพที่มีพื้นที่เท่ากัน

3.2 การสแกน (Scanning)

จากที่ได้กล่าวมาแล้วข้างต้นว่าภาพประกอบด้วยจำนวนส่วนประกอบภาพมากมาย ซึ่งแต่ละจุดภาพที่ส่งไปจะบอกว่าเป็นจุดขาวหรือดำก็แสดงโดยสัญญาณภาพ ทางด้านส่งจะส่งทีละจุดเป็นลำดับแยกกันไป ทางด้านรับก็จะนำจุดต่าง ๆ เหล่านี้มาเรียงกันใหม่ให้เป็นภาพขึ้นมา วิธีนี้เรียกว่า การสแกน เส้นที่ประกอบกันเป็นภาพในจอโทรทัศน์นี้เรียกว่า เส้นสแกน ปัจจุบันในโลกเรามีโทรทัศน์ใช้กันอยู่ 2 ระบบ คือ

3.2.1 โทรทัศน์ระบบอเมริกัน มีจำนวนเส้นสแกน 525 เส้น/ภาพ และทำการส่ง 30 ภาพ/วินาที ดังนั้นความถี่ที่ใช้ในการสแกนเท่ากับ (525)x(30) = 15,750 Hz

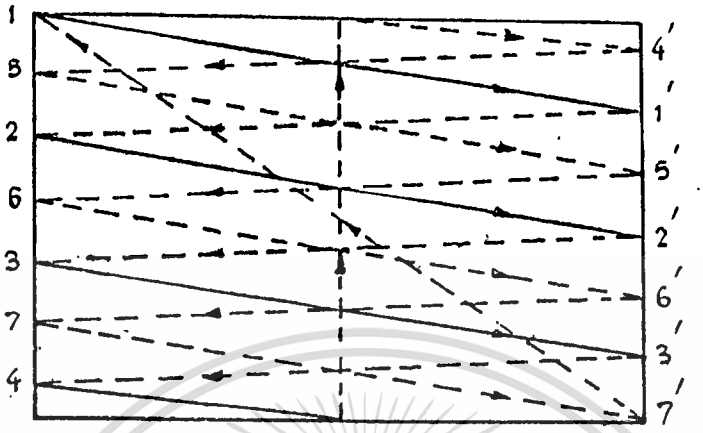
3.2.2 โทรทัศน์ระบบยุโรป มีจำนวนเส้นสแกน 625 เส้น/ภาพ และทำการส่ง 25 ภาพ/วินาที ดังนั้นความถี่ที่ใช้ในการสแกนเท่ากับ (625)x(25) = 15,625 Hz

การสแกนมีด้วยกัน 2 วิธี คือ การสแกนแบบก้าวหน้า (Progressive Scanning) และวิธีสแกนแบบสลับเส้น (Interlaced Scanning)

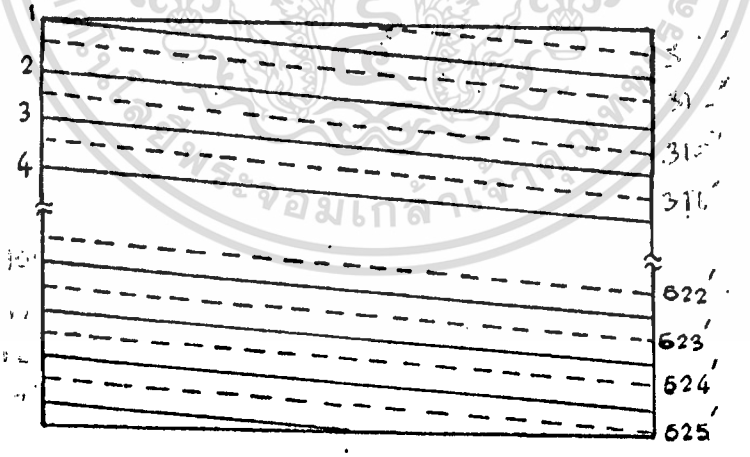
จากรูป 3.3 จะเห็นได้ว่าการสแกนเริ่มจาก a --> b , c --> d , e --> f จนถึงสุดท้าย k --> l ซึ่งเป็นการสแกนตามลำดับจากซ้ายไปขวาและข้างบนลงข้างล่างเหมือนการอ่านหนังสือหรือการพิมพ์ดีดนั่นเอง เป็นการสแกนแบบที่ใช้ในออสซิโลสโคป เรียก การสแกนแบบก้าวหน้า

รูปที่ 3.3 การสแกนแบบก้าวหน้า

จากรูป 3.4 (ก) จะเห็นว่าการสแกนเริ่มจาก 1,2,3,4 เรียกว่า ฟิลด์คี่ (Odd Field) และในระหว่างเส้นต่อเส้นก็จะเว้นช่องว่างให้พอสแกนได้อีกครั้งหนึ่ง จากนี้ก็จะเริ่ม 5,6,7 ใหม่อีกครั้งหนึ่ง เรียกว่า ฟิลด์คู่ (Even Field) เป็นการสแกนแบบเส้นเว้นเส้นซึ่งต้องใช้การสแกนในแนวตั้ง 2 ครั้ง ดังนั้นถ้าต้องการส่งภาพ 25 ภาพ/วินาที ก็ต้องส่ง 50 ครั้ง (เป็นแบบระบบยุโรป) นั่นคือเป็นการส่งภาพแบบหยาบ ๆ ไป 50 ภาพนั่นเอง วิธีนี้เราเรียกว่าวิธีการสแกนแบบสลับเส้น



(ก)



(ข)

รูปที่ 3.4 การสแกนแบบสลับเส้น

ในการสแกนแบบสลับเส้นนี้ต้องใช้การสแกนแนวตั้ง 2 ครั้ง การสแกนแนวตั้ง 1 ครั้งเรียกว่า การสแกน 1 ฟิลด์ และการสแกน 2 ฟิลด์ เรียกว่า 1 เฟรม (Frame) ในระบบ 625 เส้น จากรูป 3.4 (ข) การสแกน 1 ฟิลด์ มี 312.5 เส้น เนื่องจากตาของมนุษย์มีคุณสมบัติในการคงอยู่ของภาพ (Presistance Of Image) และระยะเวลาในการเรืองแสงของฟอสเฟอร์ที่จอภาพ ทำให้ภาพของฟิลด์ที่หนึ่งยังคงอยู่ในขณะที่ฟิลด์ที่สองสแกนเสร็จแล้ว ภาพที่มองเห็นจึงมีจำนวน 625 เส้น ข้อดีของวิธีนี้ก็คือทำให้ลดการกระพริบของภาพ (Flicker) ได้ถึงเท่าตัว

ในการสแกนทั้งแบบก้าวหน้าและแบบสลับเส้น เมื่อสแกนไปสุดของแต่ละเส้นแล้ว ต้องรีบกลับมาเริ่มเส้นใหม่ ทั้งแนวตั้งและแนวนอน ระยะเวลาในการวิ่งกลับมาเริ่มใหม่นั้นยิ่งน้อยเท่าไรก็ยิ่งดี จากรูป 3.3 คือเส้นประจาก b --> c , d --> e เส้นนี้เรียกว่าเส้นสแกนกลับ (Retrace or Flyback) เส้นนี้ไม่มีความจำเป็นในการประกอบเป็นภาพจึงมีวงจรควบคุมไม่ให้ปรากฏที่จอภาพ

ในโครงงานนี้จะทำการดิจิไตซ์ (Digitize) เพียงฟิลด์เดียว จากสัญญาณที่ส่งมาโดยวิธีการสแกนแบบสลับเส้น ดังนั้นก่อนที่จะทำการเก็บภาพ ภาพที่แสดงบนจอภาพโทรทัศน์จะเป็นวิธีการสแกนแบบสลับเส้น เมื่อทำการเก็บภาพแล้วภาพที่แสดงบนจอภาพโทรทัศน์ จะเป็นภาพของข้อมูลที่เก็บอยู่ในหน่วยความจำ และแสดงผลโดยวิธีการสแกนแบบก้าวหน้า

3.3 สัญญาณโทรทัศน์ (Composite Signal)

เพื่อจะทำให้เกิดภาพที่สมบูรณ์ สัญญาณโทรทัศน์ต้องประกอบด้วยสัญญาณต่าง ๆ ดังนี้

3.3.1 สัญญาณภาพและสัญญาณเสียง (Video And Audio Signal)

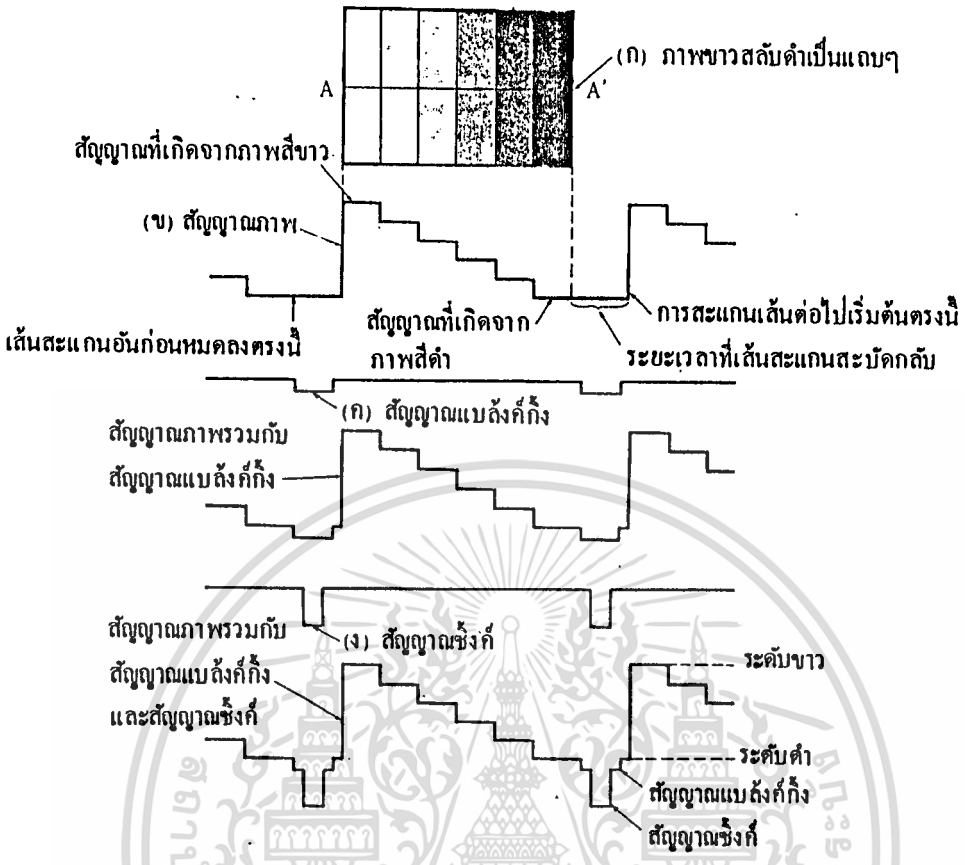
เป็นสัญญาณที่ใช้เพื่อทำให้เกิดภาพและเสียงตามต้องการ ในโครงงานนี้เราใช้เพียงสัญญาณภาพเพียงอย่างเดียว

3.3.2 สัญญาณแบล้งกิ้ง (Blanking Pulses)

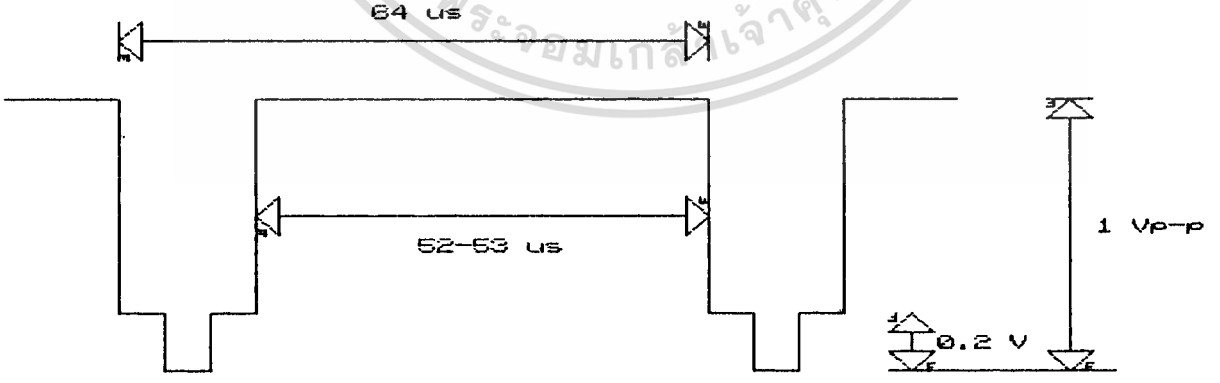
เป็นสัญญาณที่ใช้เพื่อลบเส้นสแกนสะบัดกลับทั้งในแนวนอนและในแนวตั้ง เพื่อมิให้สังเกตเห็นได้ชัดทางจอภาพ สำหรับโทรทัศน์ระบบอเมริกันสัญญาณแบล้งกิ้งระหว่างเส้นสแกน(แบล้งกิ้งทางแนวนอน) จะมีขนาดประมาณ 10 ไมโครวินาที ทำนองเดียวกันสัญญาณแบล้งกิ้งระหว่างฟิลด์ (แบล้งกิ้งทางแนวตั้ง) จะมีขนาดประมาณ 1,250 ไมโครวินาที ส่วนสัญญาณแบล้งกิ้งของโทรทัศน์ระบบยุโรปก็มีค่าประมาณนี้ (ดูรูปที่ 3.5)

3.3.3 สัญญาณซิ้งค์

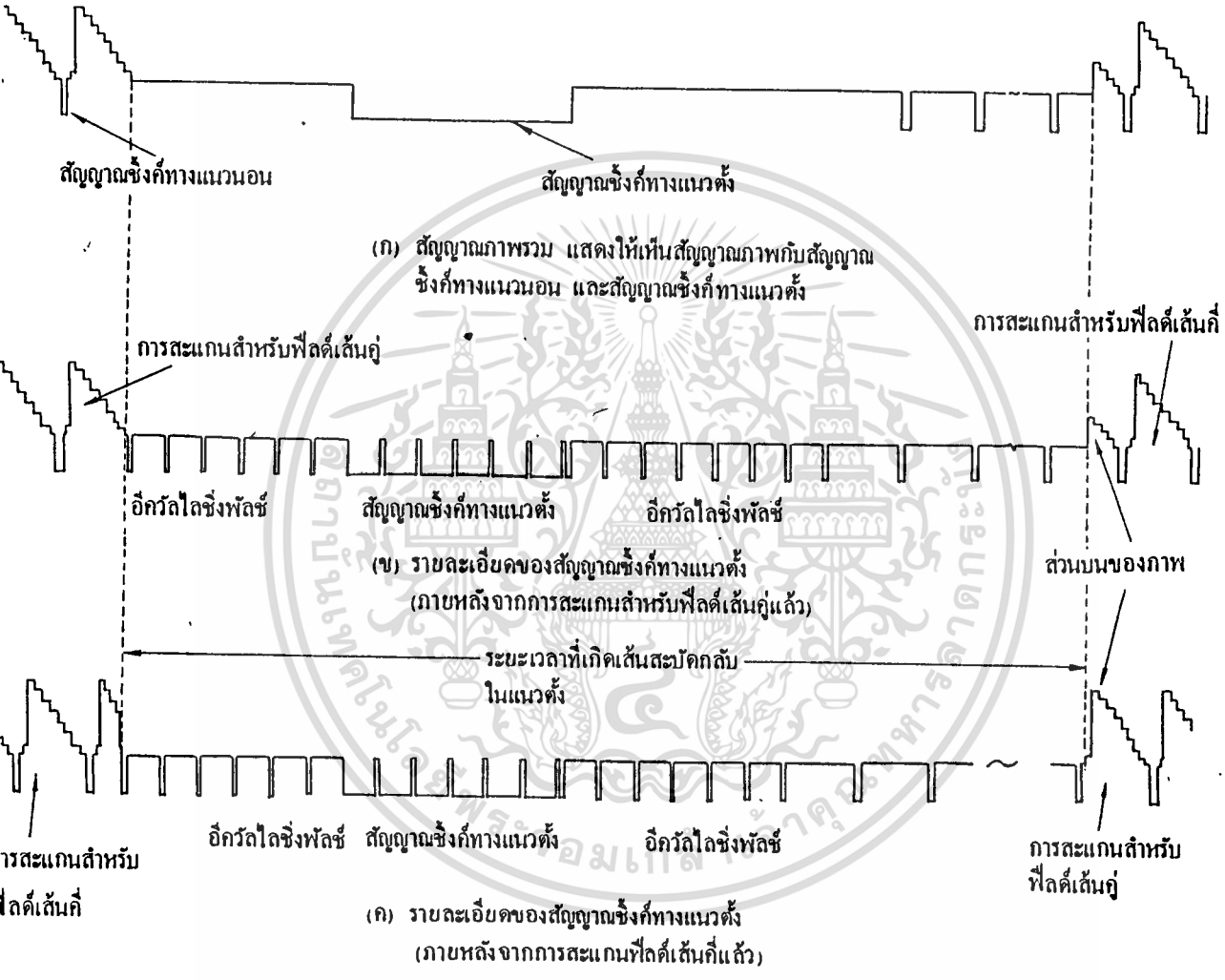
เป็นสัญญาณที่ใช้เพื่อช่วยทำให้วงจรหักเหทางแนวนอนและแนวตั้ง (เรื่องราวเกี่ยวกับวงจรหักเหสามารถค้นคว้าได้จากตำราโทรทัศน์ทั่วไป) ในเครื่องส่งและเครื่องรับโทรทัศน์มีความถี่ตรงกันตลอดเวลา ในระบบยุโรปสัญญาณซิ้งค์ทางแนวนอนมีความถี่ 15,625 Hz ซึ่งเท่ากับความถี่ของวงจรหักเหทางแนวนอน และสัญญาณซิ้งค์ทางแนวตั้งก็มีความถี่ 50 Hz ซึ่งเท่ากับความถี่ของวงจรหักเหทางแนวตั้งเหมือนกัน เนื่องจากว่าความถี่ของสัญญาณซิ้งค์มีค่าเท่ากับความถี่ของสัญญาณแบล้งกิ้งพอดี จึงจำเป็นต้องป้องกันการรบกวนที่อาจเกิดขึ้นโดยการกำหนดขนาดของซิ้งค์พัลซ์ให้น้อยกว่าขนาดของแบล้งกิ้งพัลซ์ คือทำให้ซิ้งค์พัลซ์ทางแนวนอนมีขนาดเพียง 5 ไมโครวินาที และซิ้งค์ทางแนวตั้งมีขนาดประมาณ 190 ไมโครวินาทีเท่านั้น นอกจากนี้ยังใช้วิธีส่งซิ้งค์เหล่านี้ปนไปกับแบล้งกิ้งพัลซ์อีกด้วย โดยให้ฐานของซิ้งค์พัลซ์อยู่ทับขอบบนของแบล้งกิ้งพัลซ์อีกชั้นหนึ่ง เมื่อจัดขอบเขตความต่างศักย์ให้ระดับสูงสุดของแบล้งกิ้งพัลซ์เป็นระดับดำมืดจนมองไม่เห็นแล้ว ระดับของซิ้งค์พัลซ์ที่อยู่บนยอดสูงสุดของแบล้งกิ้งพัลซ์ก็จะเป็นดำมืดสนิทไปด้วย และไม่ทำให้เกิดการรบกวนภาพที่จอภาพแต่อย่างใด (ดูรูป 3.5,3.6,3.7)



รูปที่ 3.5 รูปร่างของสัญญาณโทรทัศน์ที่เกิดจากภาพขาวสลับดำเป็นแถบ ๆ



รูปที่ 3.6 ขนาดและช่วงเวลาของสัญญาณภาพ



รูปที่ 3.7 สัญญาณภาพรวม

3.3.4 สัญญาณอีควอไลซิ่ง (Equalizing Pulses)

เป็นสัญญาณที่ใช้เพื่อช่วยให้สัญญาณซิ้งค์ทางแนวตั้ง ยังคงมีรูปร่างดีเหมือนเดิมหลังจากแยกออกมาจากสัญญาณซิ้งค์ทางแนวนอนแล้ว นอกจากนี้ยังช่วยทำให้การสแกนแบบไขว้กันเป็นไปโดยเรียบร้อยสม่ำเสมอ รวมทั้งสัญญาณซิ้งค์ทางแนวนอนก็ไม่ขาดหายไปในช่วงเวลาของสัญญาณซิ้งค์ทางแนวตั้งอีกด้วย สัญญาณอีควอไลซิ่งนี้มีความจำเป็นสำหรับการสแกนแบบสลับเส้น เพราะช่วยลดความผิดพลาดที่เกิดขึ้นเนื่องจากการสะบัดกลับผิดตำแหน่ง ในโครงงานนี้การสร้างสัญญาณโทรทัศน์ในขณะแสดงภาพของข้อมูลที่เก็บอยู่ในหน่วยความจำ (ภาพที่ดิจิไตซ์ได้) จะไม่มีสัญญาณอีควอไลซิ่งทั้งนี้เนื่องจากเป็น วิธีการสแกนแบบก้าวหน้า ซึ่งจะทำให้ระบบวงจรง่ายขึ้น และยังประหยัดค่าใช้จ่ายอีกด้วย

สัญญาณทั้งหมดที่กล่าวมานี้ เมื่อนำมารวมกันเราเรียกว่า สัญญาณโทรทัศน์ (Composite Signal)

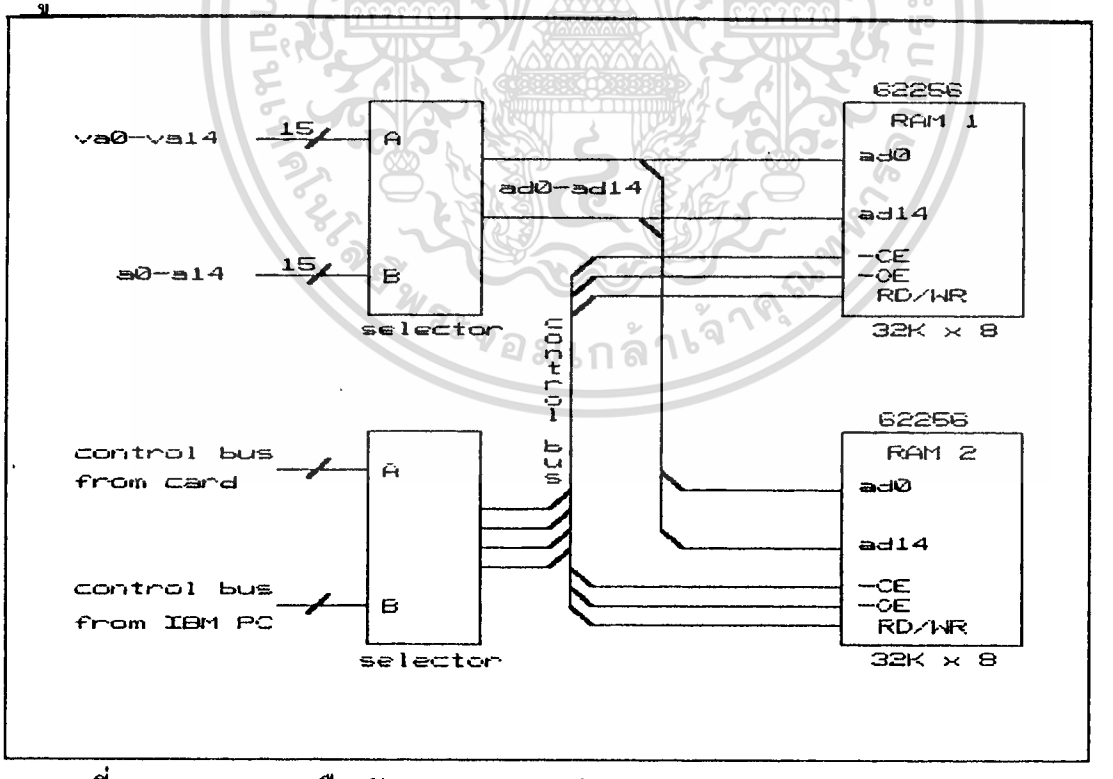
การคำนวณและการสร้าง

บทที่ 4

การจัดการหน่วยความจำ(Memory management)

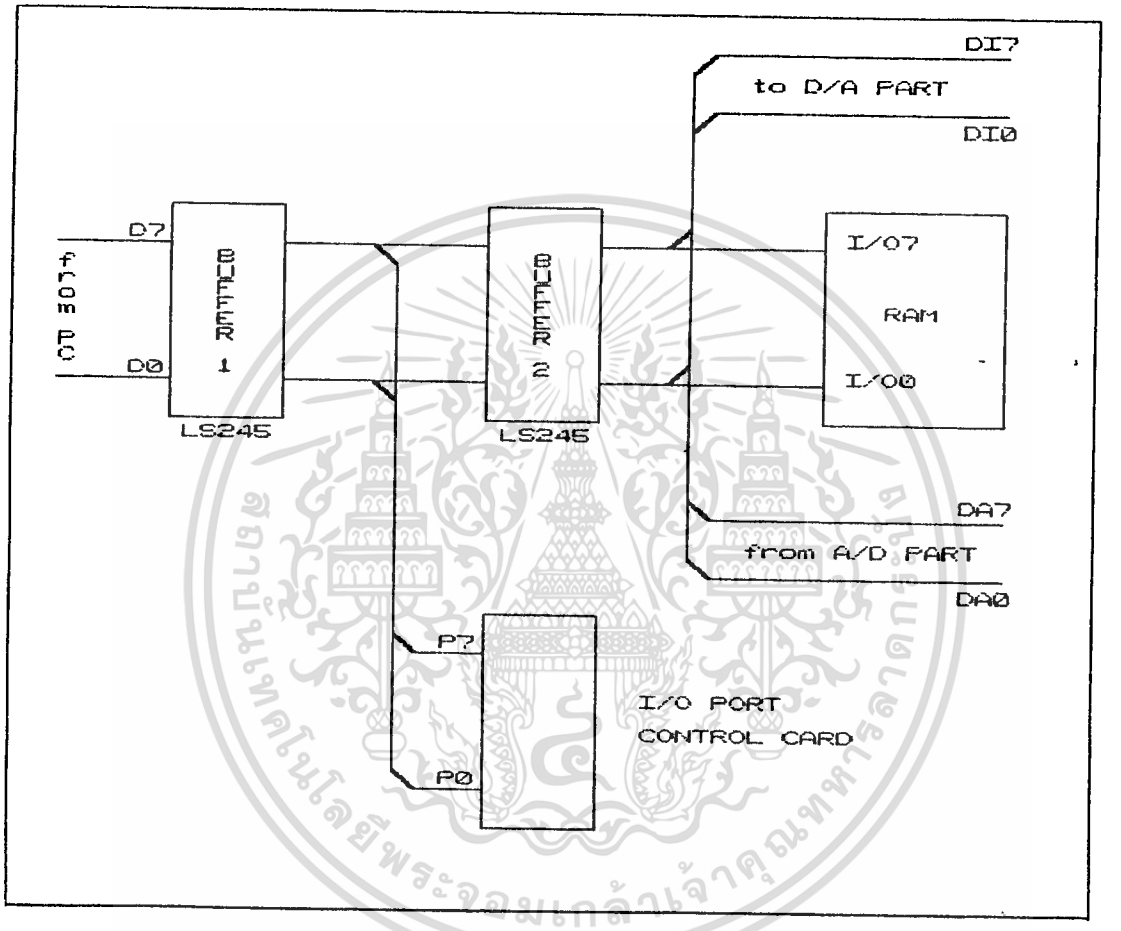
บน card จะมีหน่วยความจำ 64 kbytes แยกออกจาก mainboard ของ IBM PC/XT โดยใช้ Ram เบอร์ 62256 ซึ่งเป็น 32K*8 Static Ram จำนวน 2 ตัว เหตุที่ต้องใช้หน่วยความจำถึง 64 Kbyte เนื่องจากต้องการเก็บจุดภาพ (Pixel) 256x256x8 บิทใน 1 ฟิลด์ ในปัจจุบันราคาของ Static Ram กับราคาของ Dynamic Ram ไม่ต่างกันมากนัก และข้อดีอีกข้อหนึ่งในการเลือกใช้ Static Ram คือไม่ต้องทำการ Refresh หน่วยความจำ การอ้างตำแหน่งก็ไม่ยุ่งยาก ทำให้วงจรไม่ซับซ้อน

การอ้างตำแหน่งของหน่วยความจำจะต้องทำการ Multiplex ใน card นี้ใช้ IC เบอร์ 74LS157 และ 74LS257 ซึ่งเป็น Selector ทำการเลือกสัญญาณจาก Address Bus และสัญญาณควบคุม (Control Signals) จาก Card หรือ IBM PC/XT ออกไปที่ Ram ทั้งสอง ตามรูปที่ 4.1



รูปที่ 4.1 แสดงการเลือกสัญญาณควบคุมและสัญญาณแอดเดรส (Address Bus)

ในการต่อบัสข้อมูล (Data Bus) จะต้องต่อบัฟเฟอร์ (Buffer) IC เบอร์ 74 LS 245 เนื่องจากเราต้องการสร้างพอร์ทอินพุทและเอาท์พุท (Input And Output Port) มาควบคุมการทำงานของวงจรด้วย ดังแสดงในรูป 4.2



รูปที่ 4.2 การอินเตอร์เฟสบัสข้อมูลกับ IBM PC/XT

4.1 <u>การอ้างตำแหน่งแอดเดรสของ IBM PC/XT</u>

ในการอ้างตำแหน่งแอดเดรสของ IBM PC/XT จะต้องบอกค่าสองค่าคือ เซกเมนต์ และออฟเซท (Segment And Offset) โดยเขียนในรูป Segment:Offset ทั้งเซกเมนต์และออฟเซทเป็นข้อมูลขนาด 16 บิท การคำนวนตำแหน่งที่แท้จริงทำได้โดยชิฟท์ไปทางซ้าย(Shift left) 4 บิท แล้วนำมาบวกกับค่าออฟเซทจะได้ค่าแอดเดรสขนาด 20 บิท ดังตัวอย่าง

ตัวอย่าง การคำนวนตำแหน่งแอดเดรสที่แท้จริงของค่า D000:FFFF คือ

D0000

+

0FFFF

DFFFF

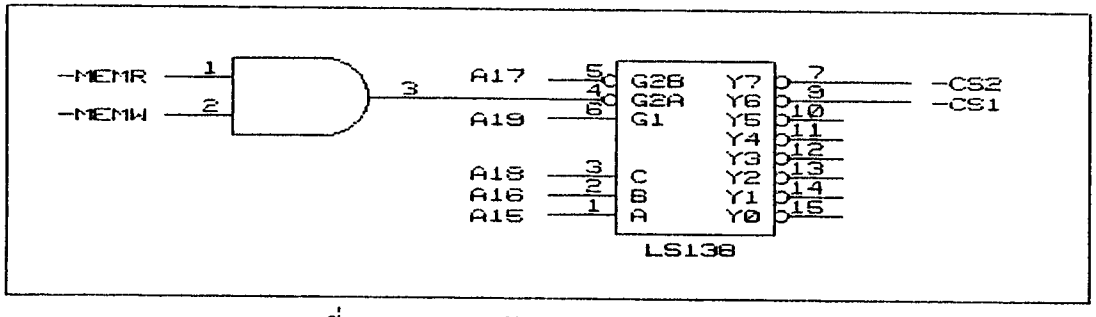
4.2 การดีโค้ด (decode) ตำแหน่งแอดเดรสของหน่วยความจำ

การดีโค้ดตำแหน่งแอดเดรสของ Card นี้ใช้เพียงค่าเซกเมนต์ เนื่องจากหน่วยความจำที่ต้องการดีโค้ดเป็น 64 Kbyte ค่าเซกเมนต์ที่ใช้คือ D000H ซึ่งเป็นเซกเมนต์ที่ว่างอยู่ (IBM PC/XT Technical Reference) ให้ Ram 1 เป็นหน่วยความจำ 32 Kbyte แรกและ Ram 2 เป็นหน่วยความจำ 32 Kbyte ถัดมา เพราะฉะนั้น Ram 1 จะมีค่าออฟเซทแอดเดรสอยู่ช่วง 0000H - 7fffH และ Ram 2 จะมีออฟเซทแอดเดรสอยู่ในช่วง 8000H - ffffH ดังนี้

Address	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Ram 1	1	1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Ram 2	1	1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x : don't care

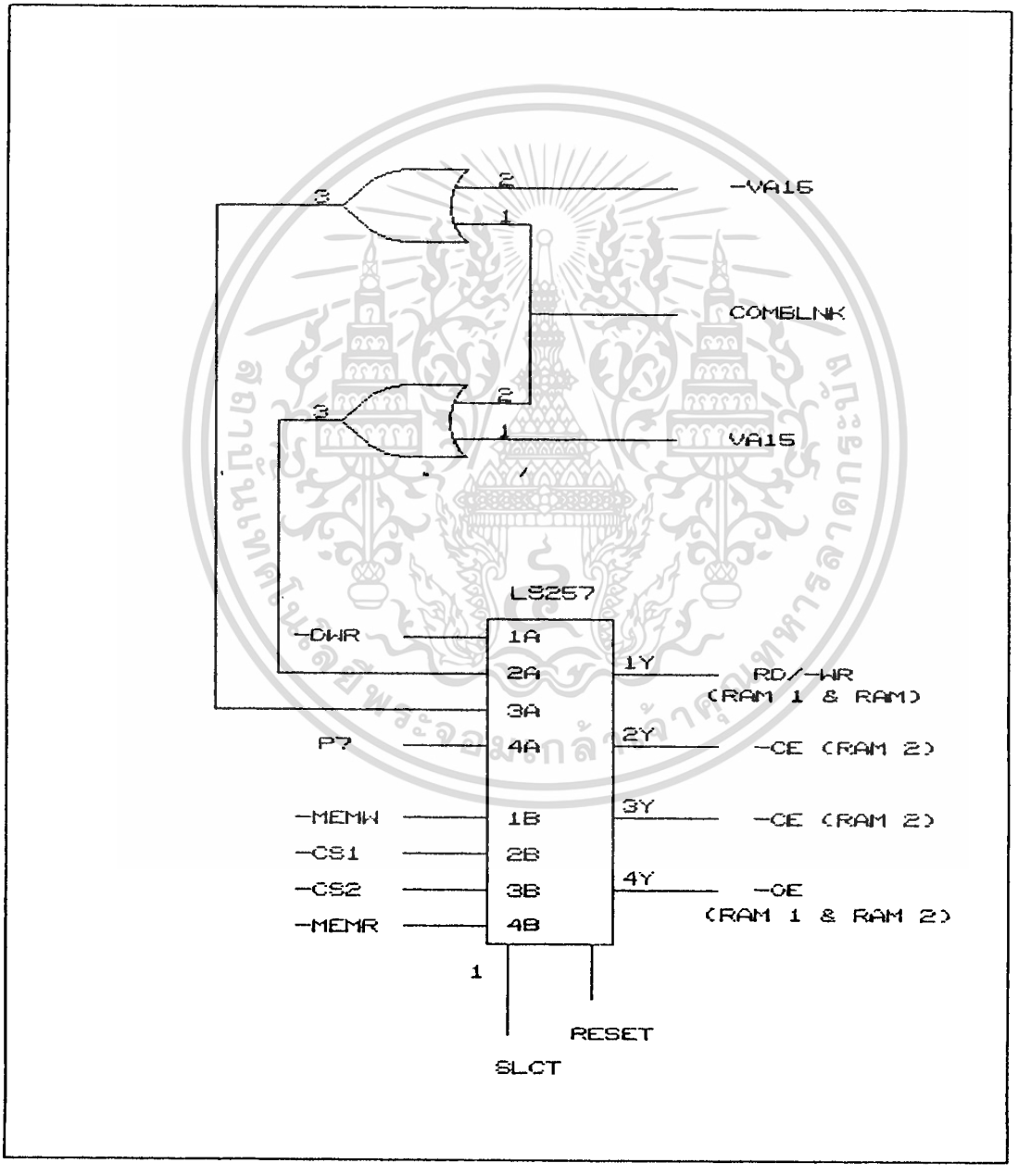
วงจรดีโค้ดแสดงดังรูปที่ 4.3 ใช้ IC 74 LS 138 เป็นตัวดีโค้ด เอาท์พุทที่ได้ (Y6, Y7) นำไปเป็นสัญญาณ Chip Enable ของ Ram แต่ละตัว



รูปที่ 4.3 การสร้างสัญญาณ Chip Select

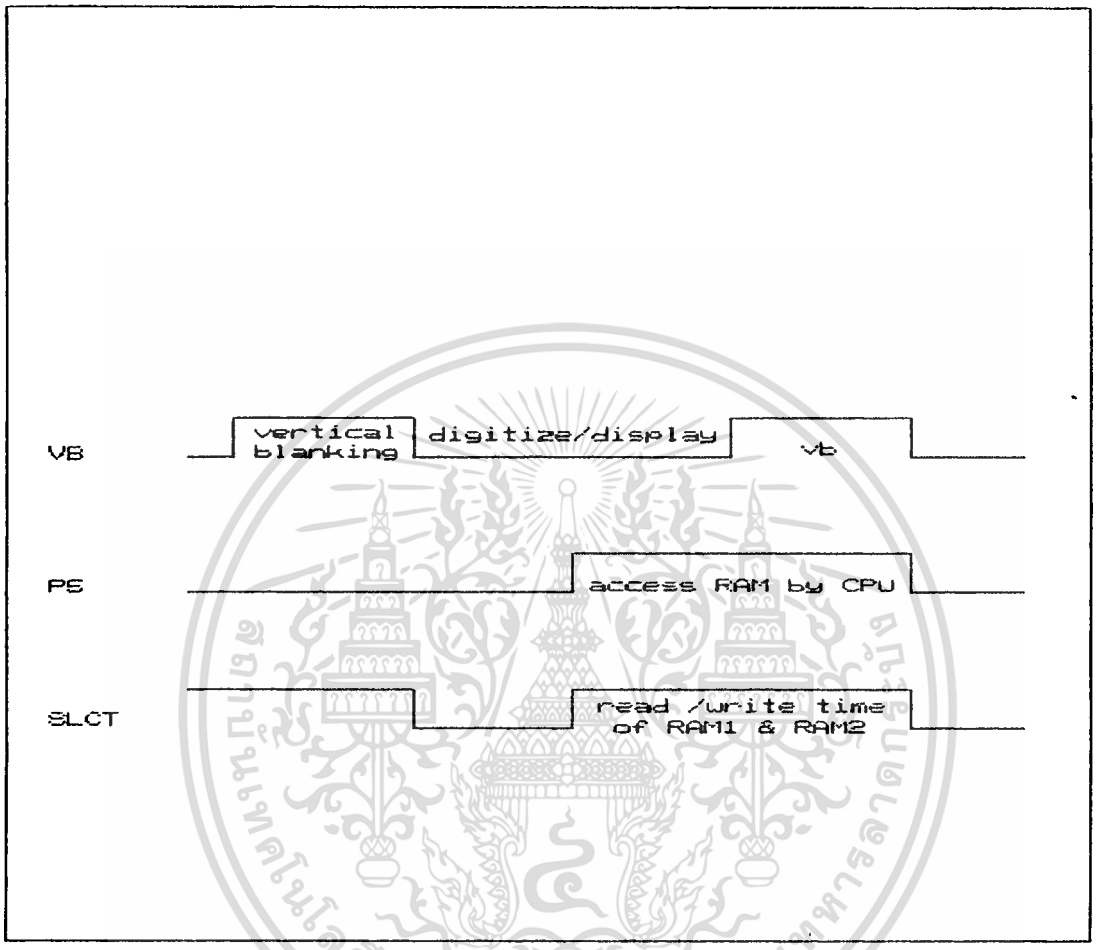
เมื่อเราต้องการติดต่อกับ Ram ตัวใดตัวหนึ่งจะต้องให้แอดเดรสและต้องทำการ Enable ขาสัญญาณควบคุม 3 ขาคือ CE (Chip Enable) ซึ่งใช้สัญญาณจากการดีโค้ดแอดเดรส , ขา OE (Output Enable) , ขา RD/WR (Read Or Write)

สัญญาณควบคุมจะทำการ Multiplex โดยใช้ IC เบอร์ 74LS257 Tri-state Selector ดังรูปที่ 4.4



รูปที่ 4.4 การ Multiplex สัญญาณควบคุม

สัญญาณ SLCT ใช้เป็นสัญญาณ Select ของ 74 LS 257 มี Timing Diagram ดังรูป 4.5



รูปที่ 4.5 Timing Diagram ในการควบคุม Ram

4.3 <u>การดีโค้ดตำแหน่งแอดเดรสของอินพุทเอาท์พุทพอร์ท</u>

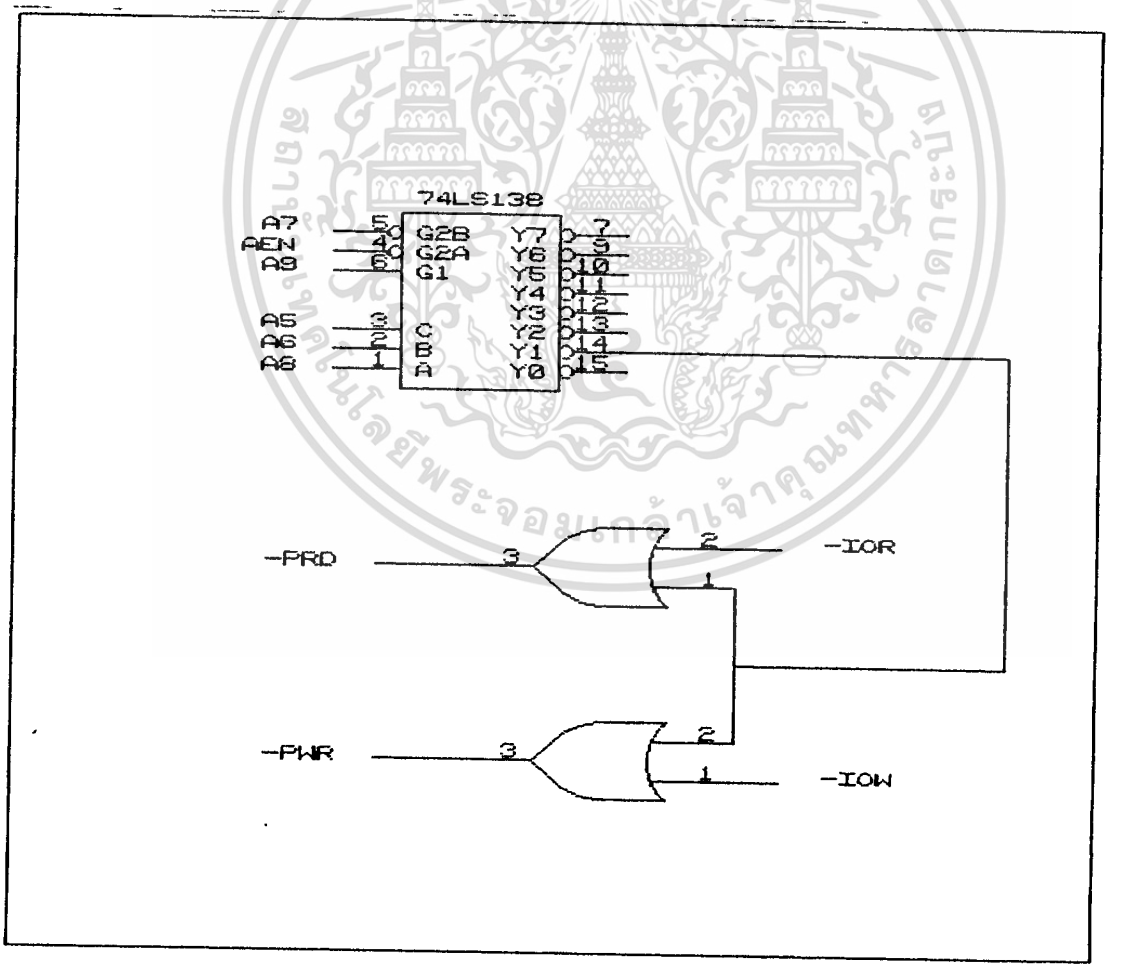
ในการอ้างตำแหน่งแอดเดรสของพอร์ทต่าง ๆ บน IBM PC/XT จะใช้บัสแอดเดรสเพียง 10 เส้นเท่านั้นคือ A0 - A9 จะได้แอดเดรสของพอร์ทสูงสุดทั้งหมด 1,024 พอร์ท ซึ่งจะแบ่งออกเป็นกลุ่มสองกลุ่มคือ

4.3.1 เมื่อบิท A9 เป็น 0 จะเป็นพอร์ทที่อยู่บน Mainboard เท่านั้น ได้แก่พอร์ท 0-1ffH (512 พอร์ท)

4.3.2 เมื่อบิท A9 เป็น 1 จะเป็นพอร์ทที่อยู่บน Card ต่างๆได้แก่พอร์ท 200H - 3ffH (512 พอร์ท เช่นกัน)

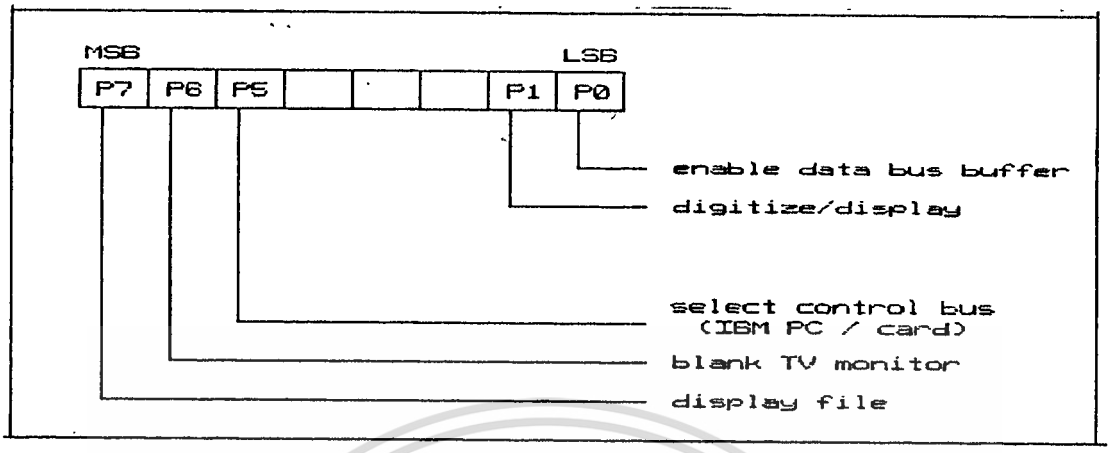
บน Card เราใช้พอร์ท 300H - 31fH ซึ่งเป็นพอร์ทที่ IBM PC/XT เตรียมไว้ให้ Card ต่าง ๆ ได้ใช้ แท้จริงแล้วต้องการใช้เพียงแอดเดรสเดียวแต่เพื่อความประหยัดอุปกรณ์ที่ใช้ในการดีโค้ดแอดเดรสจึงต้องทำเช่นนี้ ในการใช้งานร่วมกับ Card อื่นจึงควรระวังกรณีพอร์ทซ้ำกันด้วย

Address	9	8	7	6	5	4	3	2	1	0
300 H	1	1	0	0	0	0	0	0	0	0
31f H	1	1	0	0	0	1	1	1	1	1



รูปที่ 4.6 วงจรดีโค้ดพอร์ท 300H - 31fH

Byte ควบคุมพอร์ทแต่ละบิทมีความหมายดังรูปที่ 4.7



รูปที่ 4.7

บิท P0 : 1 = CPU สามารถติดต่อกับ Ram ได้

บิท P1 : 1 = ทำการดิจิไตซ์

Ø = แสดงภาพในหน่วยความจำ

บิท P5 : 1 = ต่อ Ram เข้ากับ CPU ใช้ร่วมกับบิท P0

บิท P6 : 1 = ดับจอภาพ

บิท P7 : 1 = Enable การอ่าน Ram ของ Counter

Byte แสดงสภาวะใช้เพียง 2 บิทคือ

บิท P0 : แสดงสภาวะการทำงานของวงจร

1 = กำลังทำการดิจิไตซ์

บิท P1 : แสดงสภาวะของสัญญาณ VB

1 = VB เป็น High

Ø = VB เป็น Low

การใช้ Byte ควบคุมดังตัวอย่างในโปรแกรม Display.asm ซึ่งเป็นโปรแกรมอ่านไฟล์ Image ขนาด 64 Kbyte 256 Gray Level เพื่อแสดงภาพ

บทที่ 5

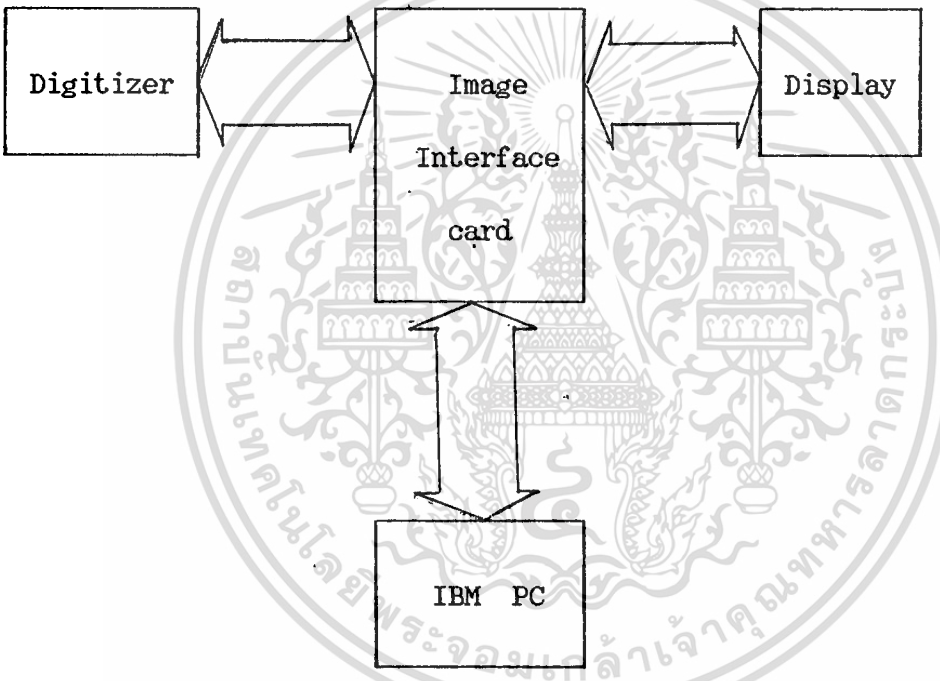
ระบบการทำงานของตัวประมวลผลภาพ

ในปริญญานิพนธ์ฉบับนี้ได้ทำการออกแบบตัวประมวลผลภาพที่เรียกว่า Image interface card ขึ้นมาซึ่งแบ่งออกได้เป็น 3 ส่วนใหญ่ ๆ คือ

5.1 ส่วน ดิจิไตเซอร์(Digitizer module)

5.2 ส่วน ควบคุม และ อินเตอร์เฟส (Control and Interface module)

5.3 ส่วน แสดงผล (Display module)

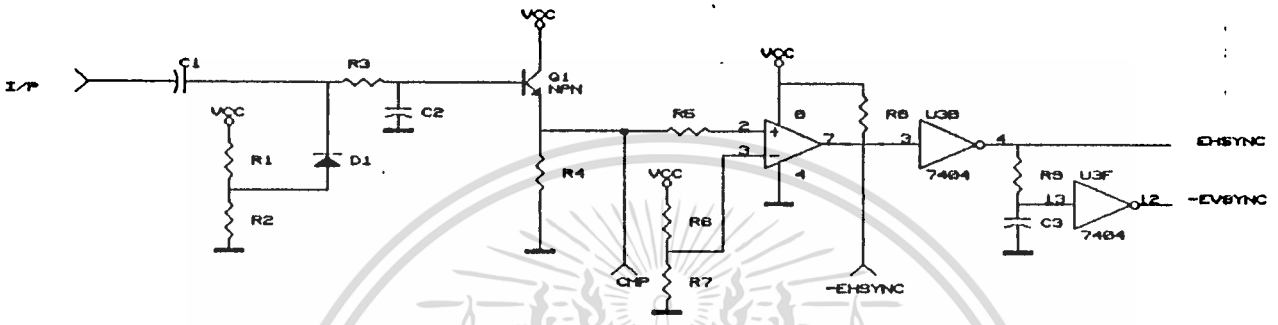


รูปที่ 5.1 ระบบการทำงานของ Image interface card

5.1 ส่วนดิจิไตเซอร์ (Digitizer)

ในส่วนนี้จะทำหน้าที่แปลงสัญญาณภาพให้กลายเป็นสัญญาณดิจิตอล 6 บิท แล้วส่งต่อให้ส่วนควบคุม และยังทำการแยกสัญญาณซิ้งค์ (Sync) ของสัญญาณภาพออกมา เพื่อนำไปใช้ในการแสดงภาพในกรณีแสดงภาพแบบเวลาจริง (real time) สัญญาณดิจิตอล 6 บิทนั้นใช้เป็นรหัสแทนภาพขาวดำที่มีระดับความเข้ม 64 ระดับ โดยทำการสุ่ม (Sampling) สัญญาณภาพด้วยความถี่ 6 MHhz ซึ่งแสดงว่า 1 จุดจะใช้เวลาในการสุ่ม Ø.166 uS และในระบบโทรทัศน์ 625 เส้น จะ

ใช้เวลาในการสแกน (scan) 1 เส้น 64 uS ดังนั้นจะสุ่มได้ 384 จุดต่อ 1 เส้น แต่ในการใช้งานจริงจะใช้เพียง 256 จุดต่อเส้น ตามมาตรฐานทั่วไปในการแสดงภาพซึ่งมีผลทำให้ได้ภาพออกมาเป็นรูปสี่เหลี่ยม



รูปที่ 5.2 วงจรตรวจจับสัญญาณซิ้งค์

5.1.1 หลักการทำงานของวงจรตรวจสอบสัญญาณซิ้งค์

เริ่มต้นจากมีสัญญาณโทรทัศน์เป็นสัญญาณอินพุทผ่าน C1 ซึ่ง C1 มีหน้าที่คัปปลิ้ง (Coupling) สัญญาณให้ปลายของสัญญาณซิ้งค์อยู่ที่ระดับกราวนด์พอดี R1,R2,D1 ทำหน้าที่ยกระดับสัญญาณดีซีของสัญญาณโทรทัศน์ให้สูงขึ้น (Clamping signal) ประมาณ Ø.6 โวลท์ทั้งนี้เนื่องจากเมื่อสัญญาณผ่าน Q1 ซึ่งเป็นบัฟเฟอร์แล้วจะทำให้ศักดาที่จุด A สูงกว่าที่จุด B ประมาณ Ø.6 โวลท์ ดังนั้นเราจึงต้องชดเชยโดยการเพิ่มสัญญาณไฟดีซีเข้าที่จุด A ออปแอมป์ LM 311N ซึ่งเป็นออปแอมป์ประเภทโวลท์เตจคอมแพเรเตอร์ (Voltage comparator) ใช้ในการเปรียบเทียบสัญญาณที่ต่ำกว่า Ø.2 โวลท์ซึ่งเป็นสัญญาณซิ้งค์ โดยเมื่อมีสัญญาณซิงค์ผ่านเข้ามาจะทำให้มีเอาท์พุทที่ขา 7 ของ LM 311N มีค่าเป็นลอจิก Ø สัญญาณซิ้งค์ที่แยกได้จะนำไปใช้ในกระบวนการดิจิไตซ์ต่อไป

5.1.2 วงจรส่วนแปลงสัญญาณอนาล็อกเป็นดิจิตอล (A/D)

A/D เป็นส่วนแปลงสัญญาณอนาล็อก (มีทั้งค่า ศักดา,กระแส,ความถี่ ฯลฯ) ให้เป็นสัญญาณดิจิตอล ในการออกแบบที่จะใช้ A/D นี้ควรจะทราบถึงความหมายของคำต่างๆดังนี้

รีโซลูชั่น หมายถึงจำนวนชั้นที่แบ่งจากช่วงระหว่างระดับสัญญาณต่ำสุดถึงสูงสุด ซึ่งมันจะเป็นตัวกำหนดความกว้างของระดับสัญญาณในแต่ละชั้นอีกด้วย การบอกรีโซลูชั่นจะบอกเป็นจำนวนบิทของเอาท์พุท เช่น ถ้ามีรีโซลูชั่น 6 บิท ก็จะมีจำนวนชั้น 64 ชั้น

LSB (Least Significant Bit) เป็นบิทที่มีความสำคัญน้อยที่สุด (ถูกถ่วงน้อยที่สุด) คือเป็นบิทที่จะเปลี่ยนแปลงได้ด้วยค่าศักดาอินพุทเพียงเล็กน้อยคือเท่ากับช่วงของศักดาที่วัดจากระดับเปรียบเทียบ (voltage reference,v-) ต่ำสุดจนถึงระดับแรกที่แบ่งไว้ เช่น (ØØØØØØ-ØØØØØ1)

MSB (Most Significant Bit) เป็นบิทที่มีความสำคัญมากที่สุด (ถูกถ่วงมากที่สุด) คือเป็นบิทที่จะเปลี่ยนแปลงได้ด้วยค่าศักดาอินพุทมากคือเท่ากับช่วงที่แบ่งไว้ ซึ่งจะเท่ากับครึ่งหนึ่ง / ของจำนวนช่วงทั้งหมด เช่น (ØØØØØØ-1ØØØØØ)

ความแม่นยำ (Accuracy) เป็นผลรวมของ ความผิดพลาด (Error) ทั้งหมดที่เกิดขึ้น คือจาก ความไม่เป็นเชิงเส้น (Non-linearity) ,ความผิดพลาดระดับต่ำ (Zero scale error), ความผิดพลาดระดับสูง (Full scale error), ความผิดพลาดจากอุณหภูมิ (Temperature drift)

เวลาในการแปลง (Convertion time) เป็นเวลาที่ใช้ในการแปลงสัญญาณจากอินพุทจนได้เป็นเอาท์พุท

ศักดาเปรียบเทียบ (Reference voltage) มี ด้านต่ำ (V-) และด้านสูง (V+) คือเป็นช่วงขอบเขตศักดาที่ต้องการแบ่งให้เป็นชั้นโดย

ด้านต่ำ จะเป็นระดับศักดาที่จะให้ค่าเอาท์พุทเป็น Ø หมด

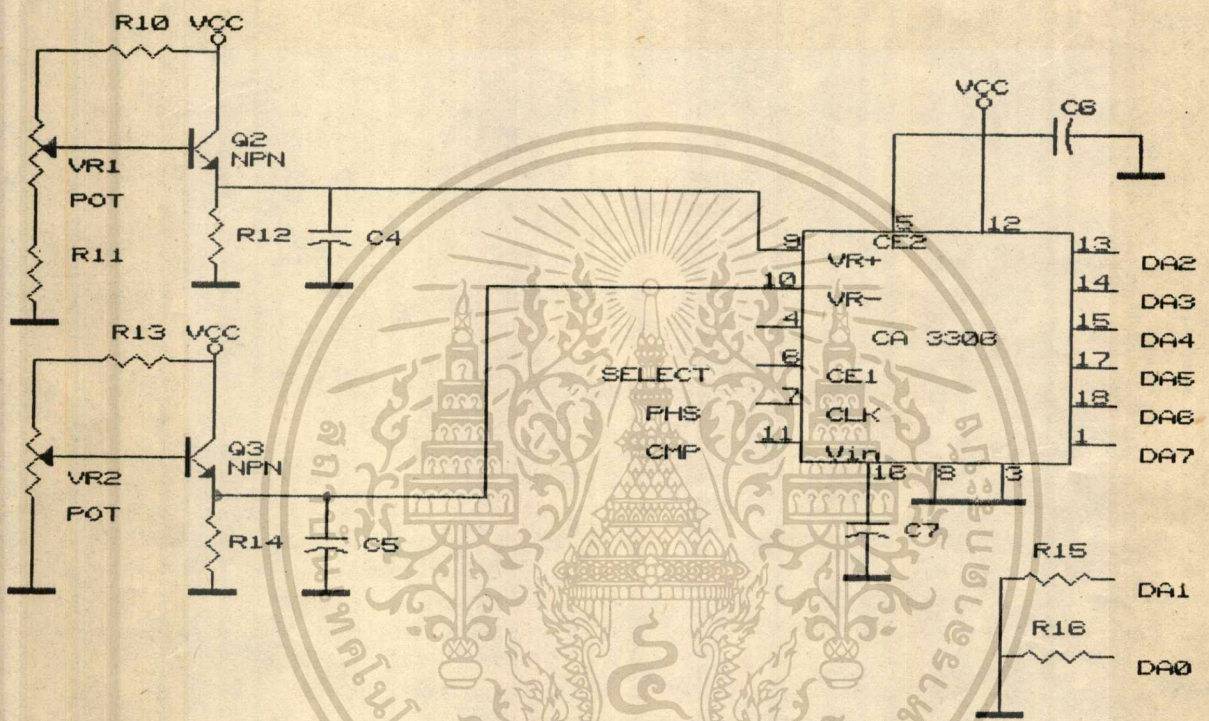
ด้านสูง จะเป็นระดับศักดาที่จะให้ค่าเอาท์พุทเป็น 1 หมด

สำหรับในโครงงานนี้ใช้ A/D ที่เป็นไอซีสำเร็จรูป เบอร์ CA33Ø6

V- เป็นระดับสัญญาณภาพ มืดที่สุด

V+ เป็นระดับสัญญาณภาพ สว่างที่สุด

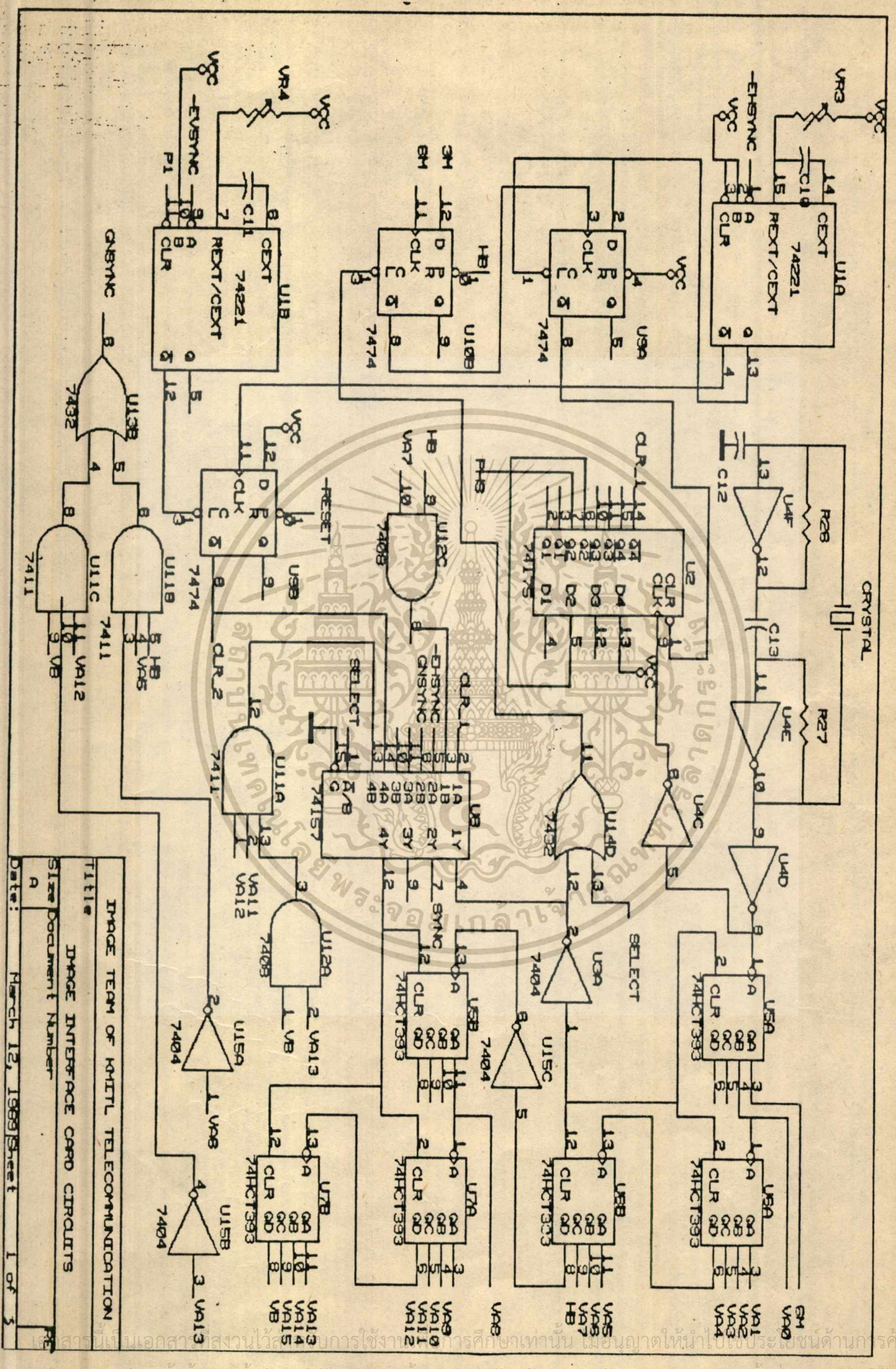
มีรีโซลูชั่น 6 บิท ความแม่นยำ ~1/2 บิท สุ่มด้วยความเร็ว 6 M hz



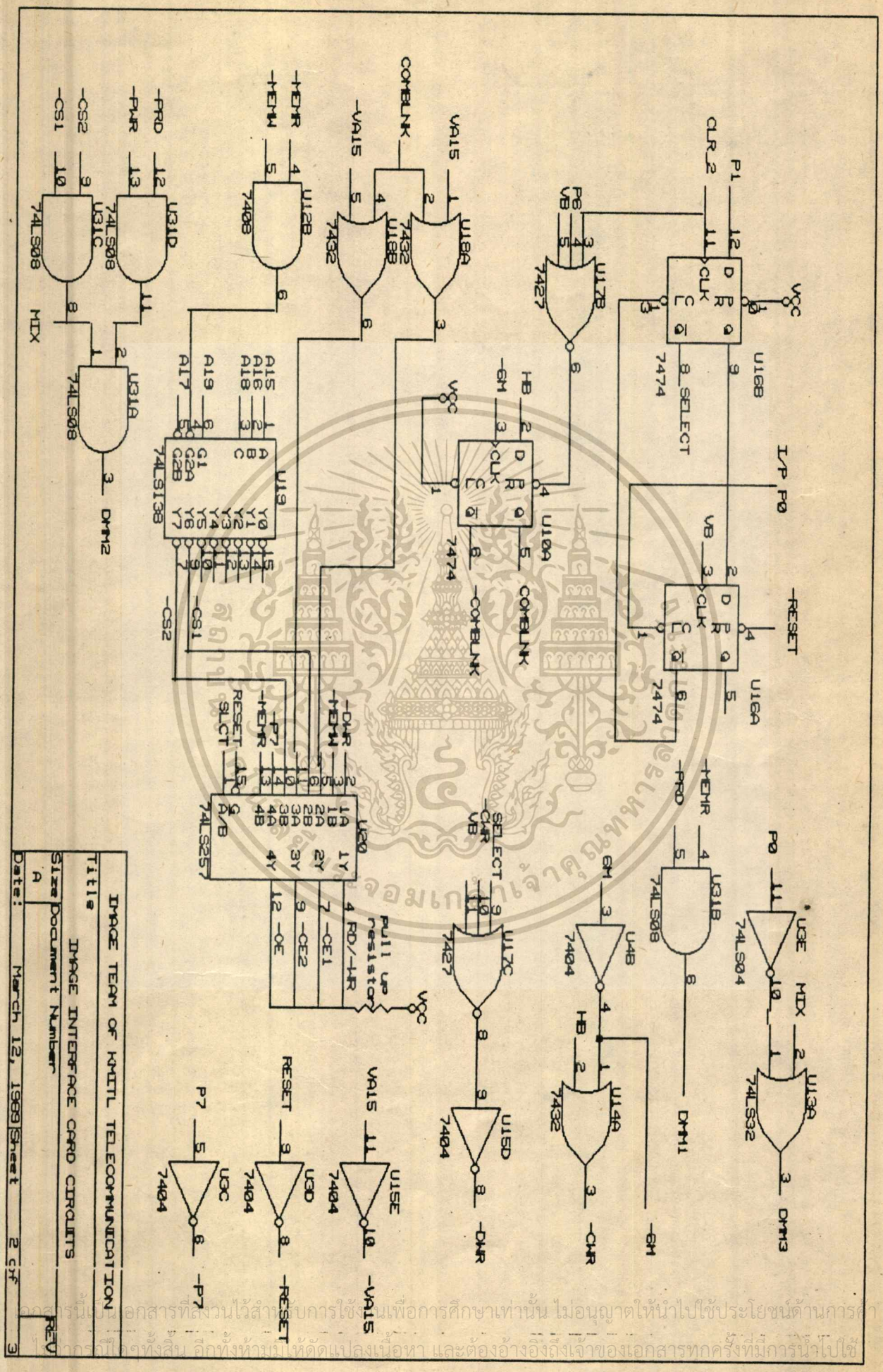
รูปที่ 5.3 วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล

ส่วนควบคุมและอินเตอร์เฟส(Control and Interface module)

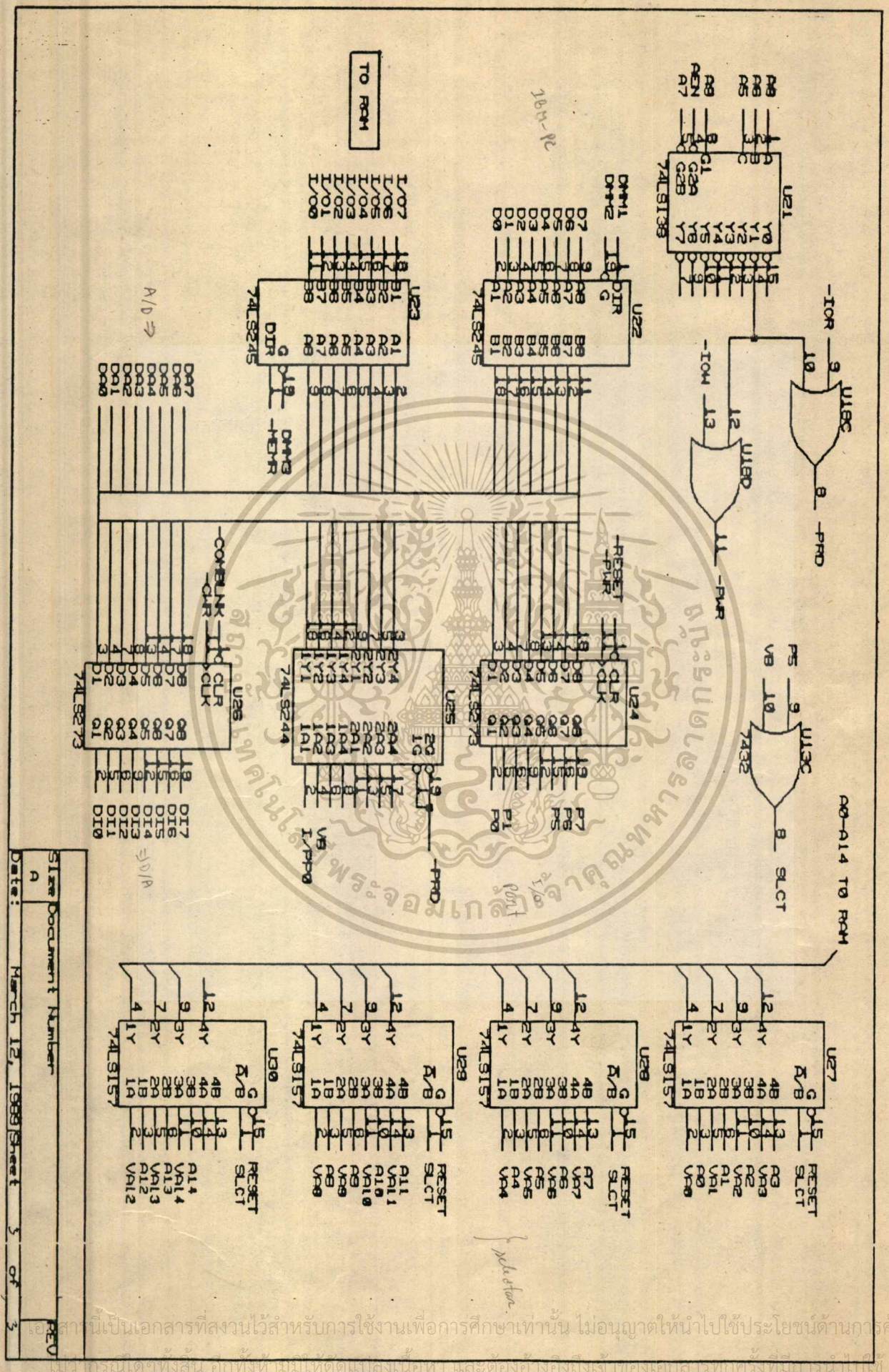
ส่วนควบคุมทำหน้าที่ควบคุมการทำงานส่วนต่างๆในวงจรให้ทำงานสอดคล้องกันได้ อีกทั้งยังน้าที่ในการติดต่อกับหน่วยความจำอีกด้วย วงจรโดยรวมแสดงดังรูปที่ 5.4 , 5.5 , 5.6 เป็นส่วนย่อย ๆ คือ



รูปที่ 5.4



รูปที่ 5



รูปที่ 5.6

5.2.1 ส่วนควบคุมและอินเตอร์เฟสภายใน (Internal control and interface module)

แบ่งเป็นส่วนผลิตสัญญาณนาฬิกา (Clock Generater) มีหน้าที่ผลิตสัญญาณนาฬิกาป้อนให้กับส่วนอื่นๆที่ต้องการใช้ โดยจะทำการสร้างสัญญาณนาฬิกาที่มีความถี่ 12 MHZ

ส่วนวงจรนับ (Counter) มีหน้าที่ชี้ตำแหน่งแอดเดรส (Address) ของหน่วยความจำเพื่อใช้อ่านข้อมูลหรือเขียนข้อมูลแบบเวลาจริง เข้าสู่หน่วยความจำด้วยความถี่ 6 MHZ

ส่วนหน่วยความจำ (Memory) เป็นส่วนเก็บข้อมูลที่มาจาก ส่วนดิจิไตซ์หรือจากเครื่องคอมพิวเตอร์ ประกอบด้วย Static Ram เบอร์ 62256 จำนวน 2 ตัว

5.2.2 ส่วนควบคุมและอินเตอร์เฟสกับคอมพิวเตอร์

เป็นส่วนที่จะเชื่อมโยงเครื่องคอมพิวเตอร์ ให้เข้ากับอุปกรณ์ ต่าง ๆ ในวงจร เพื่อให้คอมพิวเตอร์สามารถควบคุมการทำงานได้

5.3 ส่วนแสดงภาพ

ส่วนนี้ประกอบด้วย ตัวแปลงสัญญาณดิจิตอลเป็นอนาล็อก (Digital to analog converte) และส่วนเพิ่มสัญญาณซิงค์

ตัวแปลงสัญญาณดิจิตอลเป็นอนาล็อก จะแปลงสัญญาณดิจิตอลที่รับมาจากหน่วยความจำให้เป็นสัญญาณอนาล็อกซึ่งเป็นสัญญาณภาพ จากนั้นก็ส่งต่อไปยังส่วนเพิ่มสัญญาณซิงค์ ซึ่งจะได้เอาท์พุทเป็นสัญญาณคอมโพสิต (Composite) ออกทางจอทีวีเป็นภาพขาวดำที่มีระดับความเข้ม 256 ระดับ และมีความละเอียด (Resolution) เท่ากับ 256x256 จุด

5.1.3 วงจรส่วนแปลงสัญญาณดิจิตอลเป็นอนาล็อก (D/A)

คือขบวนการแปลงจากดิจิตอลเป็นอนาล็อก ซึ่งเป็นขบวนการย้อนกลับกับ A/D และคำต่าง ๆ ที่ใช้ก็มีความหมายเหมือนกัน

วงจร D/A จะประกอบด้วย -วงจรตรรกะ (Logic circuit) เพื่อใช้รับสัญญาณดิจิตอล

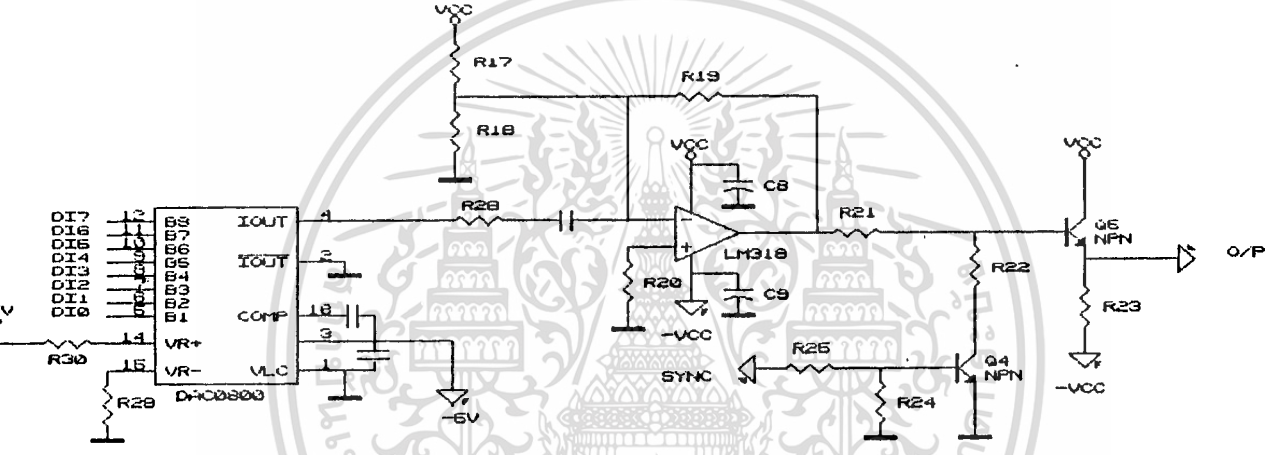
-ศักดาอ้างอิง เพื่อใช้เป็นส่วนจ่ายศักดาให้แก่ เอาท์พุท ซึ่งต้องเลือกค่าที่ต้องการว่าต้องการให้เอาท์พุทมีศักดาอยู่ในช่วงใด,

ขั้วบวกหรือลบ, ค่าและทิศทางของกระแสเอาท์พุท

-อนาล็อกสวิทช์ (Analog switch) และวงจรขับ (Drive circuit)

-วงจรความต้านทาน (Resister network)

ในโครงงานนี้ใช้ไอซีเบอร์ Ø8ØØ ดูรายละเอียดได้จาก Data book



รูปที่ 5.4 วงจรแปลงสัญญาณดิจิตอลเป็นอนาลอก

หลังจากที่ข้อมูลทั้ง 8 บิท ผ่านเข้ามาใน D/A แล้ว D/A ก็จะเปลี่ยนสัญญาณให้ออกมาในรูปของอนาลอกที่เป็นขั้วลบ ดังนั้นจึงจำเป็นต้องกลับขั้วสัญญาณให้เป็นบวก โดยอาศัยออปแอมป์เข้าช่วย ลักษณะการต่อเข้าออปแอมป์เป็นแบบการขยายแบบกลับขั้วสัญญาณ (INVERTING AMP.) ที่มีอัตราขยายเป็น 1

เมื่อสัญญาณกลับขั้วเป็นบวกแล้วก็จะเข้าสู่วงจรรวมสัญญาณซิงค์ ซึ่งเป็นแบบสัญญาณซิงค์รวม (COMPOSITE SYNC.) ที่มีทั้งสัญญาณซิงค์แนวตั้ง (VERTICAL SYNC.) และสัญญาณซิงค์

แนวนอน (HORIZONTAL SYNC.) ซึ่งสัญญาณซิงค์ที่เข้ามานี้มาจาก 2 ทางด้วยกัน คือ ในช่วงการดิจิไตซ์ ใช้สัญญาณซิงค์ที่แยกได้จากสัญญาณภาพรวม (COMPOSITE SIGNAL) และในช่วงการแสดงภาพจากไฟล์ จะใช้สัญญาณที่ผลิตขึ้นมาโดยวงจรนับ

บทที่6

ซอฟท์แวร์ (SOFTWARE)

ในส่วนของซอฟแวร์ ประกอบไปด้วยโปรแกรมที่เขียนขึ้นมาเพื่อทดลองใช้งานกับการ์ดนี้มีทั้งหมด 5 โปรแกรม ซึ่งทั้งหมดเขียนด้วย ภาษาซี (เทอร์โบซี) ดังมีรายละเอียดต่อไปนี้

6.1 โปรแกรมแสดงผลภาพ (โปรแกรม 1) เป็นโปรแกรมแสดงภาพที่เก็บเป็นไฟล์จากแผ่นดิสก์ซึ่งเป็นไฟล์ขนาด 256x256 จุด ภาพมีระดับความสว่างไม่เกิน 256 ระดับ ในการที่จะเข้าไปเขียนหน่วยความจำในการ์ดนั้นจะต้องส่งค่า Ø61H (61 ฐานสิบหก) ออกไปที่พอร์ท Ø3ØØH เสียก่อน แล้วจึงย้ายข้อมูลเข้าไปได้ (ส่วนการอ้างตำแหน่งบนจอทีวี ดูรายละเอียดได้ในตอนต้นของปริญญานิพนธ์นี้) เมื่อย้ายข้อมูลจนครบภาพ (64 Kbytes) แล้วก็จะทำการส่งค่า Ø8ØH ออกพอร์ท Ø3ØØH เพื่อแสดงภาพในหน่วยความจำที่เขียนเข้าไป

6.2 โปรแกรมถ่ายภาพ (โปรแกรม 2) เป็นโปรแกรมเก็บภาพจากกล้อง สำหรับในโครงงานนี้ใช้กล้องโทรทัศน์วงจรปิด หลักการเก็บภาพคือส่งค่า Ø2H ออกมาที่พอร์ท Ø3ØØH เพื่อแสดงภาพที่ดิจิไตซ์ได้โปรแกรมจะคอยตรวจสอบว่ามีการกดคีย์บอร์ดหรือไม่ ถ้ามีจะส่งค่า Ø8ØH ออกมาที่พอร์ทเดิม ทำให้เกิดเป็นภาพนิ่งบนจอทีวีและเมื่อต้องการเก็บภาพก็ส่งค่า Ø61H ออกมาที่พอร์ท Ø3ØØH แล้วจึงทำการอ่านข้อมูลเพื่อเก็บลงไฟล์ต่อไป

6.3 โปรแกรมย่อภาพ (โปรแกรม 3) เป็นโปรแกรมใช้ย่อภาพที่แสดงบนจอทีวีหรือภาพจากไฟล์ มีหลักการคือ นำค่าระดับความสว่างของจุดที่อยู่รอบๆจุดที่สนใจมาหาค่าเฉลี่ย ค่าที่ได้คือข้อมูลของภาพที่ย่อแล้วจากนั้นก็นำกลับไปใส่ในตำแหน่งบนจอทีวี

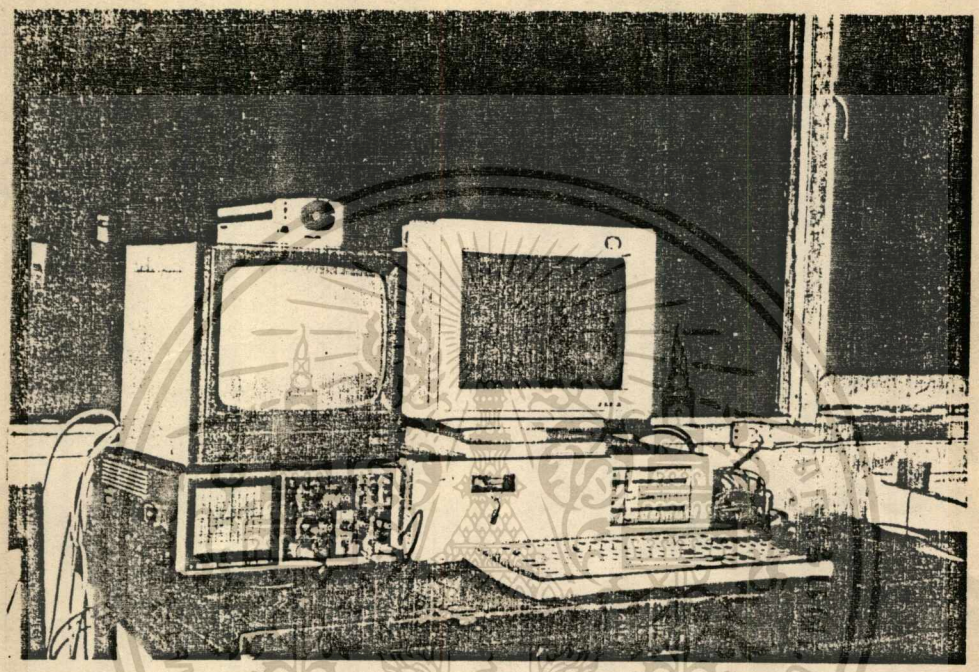
6.4 โปรแกรมพิมพ์ภาพ (โปรแกรม 4) ใช้พิมพ์ภาพที่มีระดับความสว่างออกทางเครื่องพิมพ์ โดยมีหลักการคือ ในหนึ่งจุดภาพมีระดับความสว่าง 256 ระดับ ทำการกดลงมาให้เหลือ 64 ระดับ โดยสร้างตารางค้นหา (LOOK UP TABLE) แล้วขยายหนึ่งจุดภาพนี้เป็น 8x8 โดยสร้างเมตริกซ์ขนาด 8x8 ใช้แทนระดับความสว่างได้ 64 ระดับ รูปแบบของเมตริกซ์มีมากมายแล้วแต่ว่าจะใช้รูปแบบใดที่จะให้ความเข้มดีกว่าจากการลองพิมพ์ภาพถ่ายดาวเทียม ปรากฏว่ารายละเอียดของภาพเสียไป แต่สามารถนำมาใช้พิมพ์ภาพคนหรือสิ่งของได้ เพราะว่าภาพเหล่านี้

ไม่ต้องการความละเอียดมากนัก

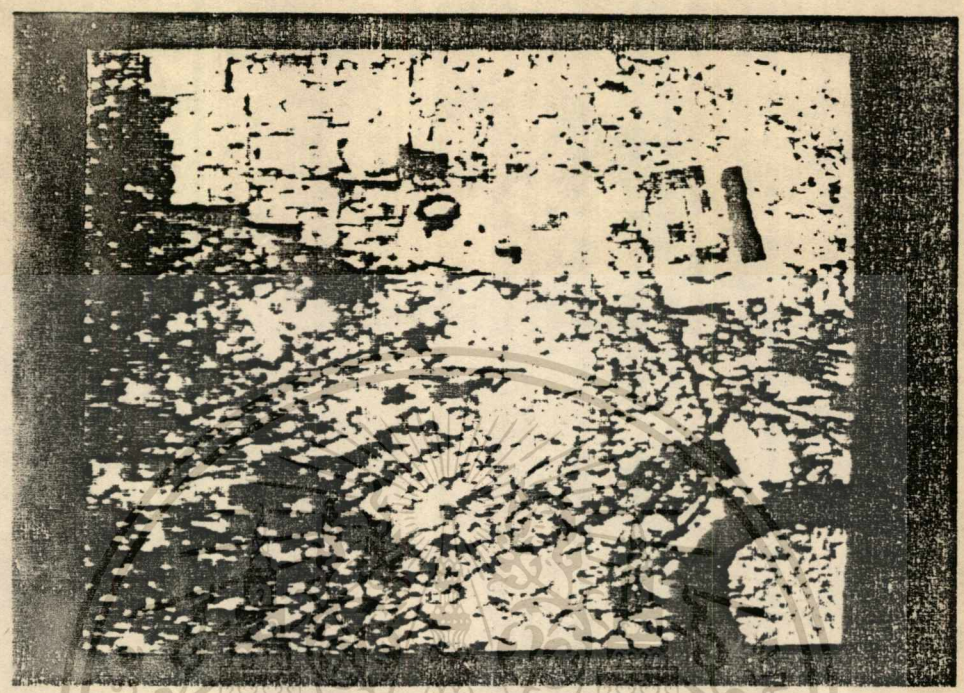
6.5 <u>โปรแกรมแสดงภาพบนจอโมโนโครม</u> (โปรแกรม 5) เป็นโปรแกรมที่นำภาพที่มีระดับความสว่าง 256 ระดับ มาแสดงบนจอโมโนโครม โดยใช้หลักการเดียวกับโปรแกรม 4 แต่เนื่องจากว่าจอโมโนโครมมีจำนวนจุดน้อยจึงไม่ต้องทำการขยายขนาด โดยเราจะแสดงเพียง 256x256 จุดในกราฟฟิกโหมด เมื่อนำมาแสดงภาพถ่ายดาวเทียม ปรากฏว่ารายละเอียดของภาพเสียไปมากจนดูภาพไม่ออก

การทดลองและผลการทดลอง

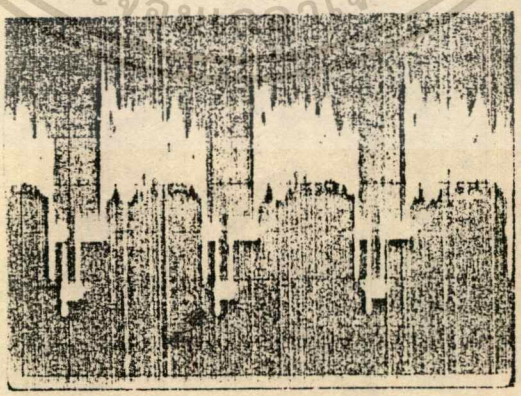
ผลการทดลอง



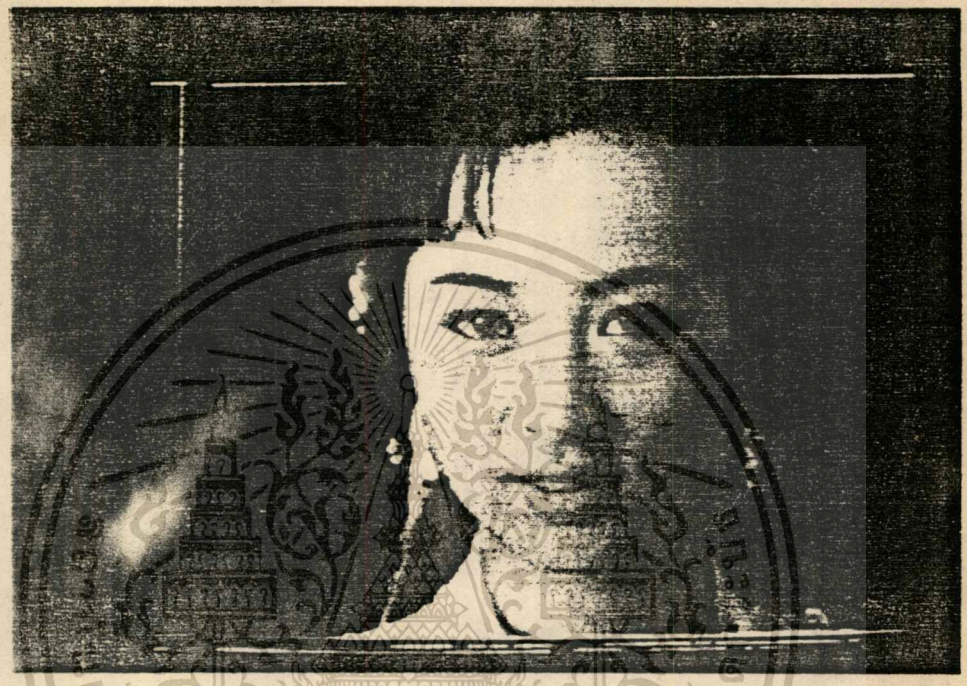
รูป ก. ระบบรวมทั้งหมดของโครงงาน



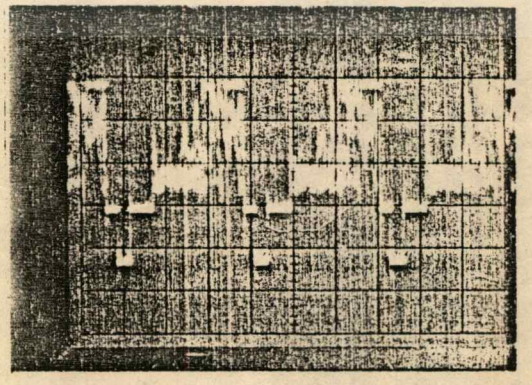
รูป ข. ภาพที่แสดงจากไฟล์ มีความละเอียด 256x256 จุด ระดับความสว่าง 256 ระดับ



รูป ค. สัญญาณภาพจาก SCOPE ที่วัดได้จาก รูป ข.



รูป ง. แสดงภาพที่ผ่านขบวนการดิจิไตซ์



รูป จ. สัญญาณภาพที่วัดได้จาก รูป ง.

บทสรุปและวิจารณ์

บทสรุปและวิจารณ์

การดำเนินงานสร้าง แผงวงจรควบคุมภาพนี้ ได้เริ่มดำเนินงานมาตั้งแต่ภาคเรียนที่ 1 ในปีการศึกษา 2531 โดยเริ่มต้นจากการทดลองวงจรนับและวงจรอินเตอร์เฟส ระหว่าง RAM ในการ์ดกับเครื่องไมโครคอมพิวเตอร์ ซึ่งการทดลองก็พบปัญหาต่าง ๆ มากพอสมควรและได้มีการเปลี่ยนแปลงแก้ไขวงจรจนกระทั่งได้ผลงานออกมาเป็นที่น่าพอใจ คือสามารถแสดงภาพจากไฟล์ได้ และยังสามารถทำการดิจิไตซ์ได้ด้วย แต่วงจรทั้งหมดนี้ก็ยังเป็นเพียงชุดการทดลองบนโปรโตบอร์ดเท่านั้น ซึ่งก็ทำให้เกิดปัญหาเรื่องสัญญาณรบกวนเข้ามารบกวนภาพ ขณะแสดงภาพ และขณะดิจิไตซ์

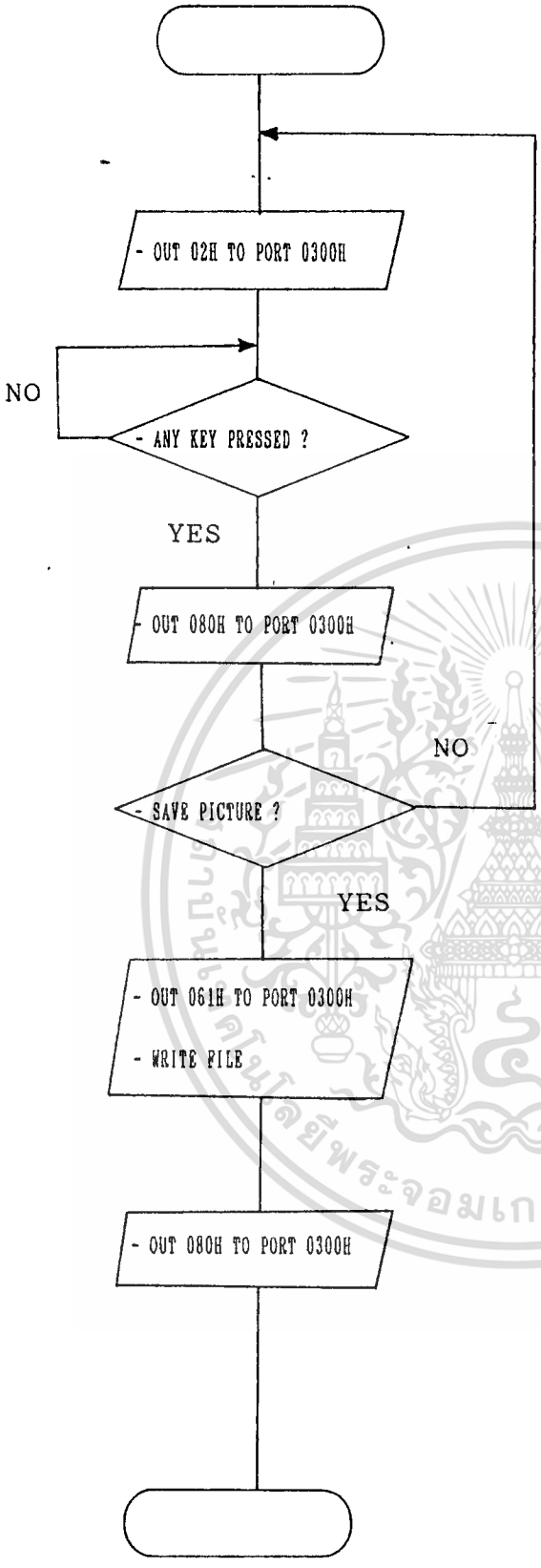
ในภาคเรียนที่ 2 ได้นำวงจรที่ทดลองได้ผลแล้วนั้นมาทำการบัดกรีลงปริ๊นท์เอนกประสงค์ซึ่งใช้การ WIRE WRAP ระหว่างขาอุปกรณ์ต่าง ๆ ผลปรากฏว่าวงจรสามารถทำงานได้อย่างดีรวมทั้งปัญหาเรื่องสัญญาณรบกวนที่เข้ามารบกวนภาพก็หมดไปด้วย แต่ว่าการ์ดที่ทำจากปริ๊นท์เอนกประสงค์ยังเป็นการ์ดที่ไม่อยู่ในลักษณะใช้งานได้ทั่ว ๆ ไป (IMPRACTICAL) ดังนั้นจึงได้ทำการออกแบบลายปริ๊นท์ ซึ่งนับว่าเป็นปัญหามากพอสมควรเนื่องจากมีอุปกรณ์มากจึงมีลายเส้นต่าง ๆ ที่ต้องเชื่อมถึงกันมาก แม้ว่าจะแก้ปัญหาด้วยการทำเป็นลายปริ๊นท์ 2 หน้า แต่ก็ทำให้เกิดปัญหาเรื่องการ THROUGH HOLES ขึ้นมาอีก นอกจากนี้ยังถูกจำกัดขนาดของการ์ดด้วย คือจะต้องไม่กว้างเกิน 4.3 นิ้ว เพราะว่าเมื่อเสียบสล็อตแล้วจะปิดฝาครอบไม่ลง

สำหรับโครงงานนี้ แม้ว่าจะยังไม่สามารถนำมาใช้งานต่าง ๆ ได้มากนัก แต่คณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นพื้นฐานสำหรับผู้ที่สนใจในด้าน IMAGE PROCESSING และเป็นประโยชน์ในการพัฒนาต่อไป ซึ่งจะเป็นหนทางหนึ่งในการช่วยพัฒนาเทคโนโลยีที่ผลิตขึ้นภายในประเทศได้ต่อไปในอนาคต

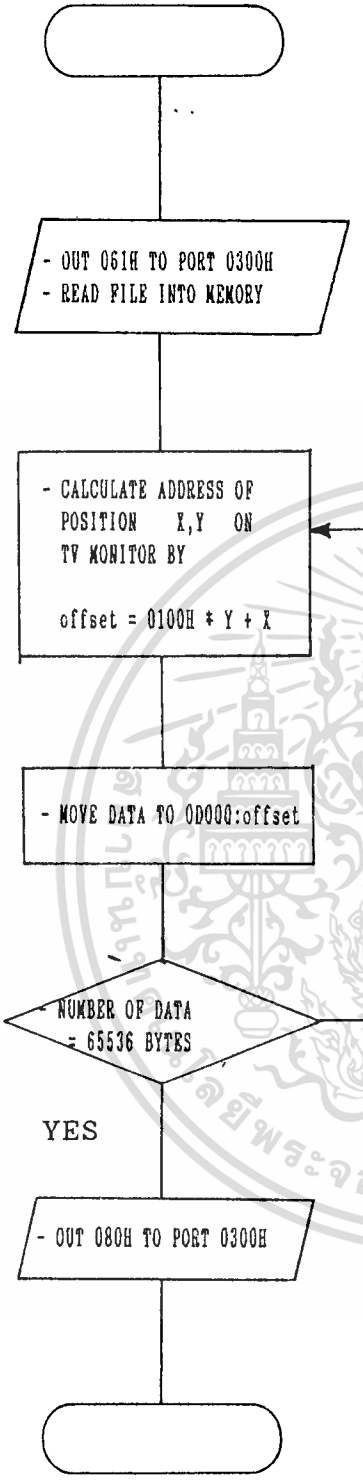
ภาคผนวก

COMPONENT	TYPE
DAC0800	8 BIT D/A CONVERTER
CA 3306	6 BIT FLASH A/D
62256	32 KBYTE STATIC RAM

COMPONENT	RESISTOR (K)	CAPACITOR (uF)
R1,R3,R4,R5,R9,R11	1	
R18,R20,R23,R26,R27	1	
R2,R7	0.22	
R6,R25,R29,R30	4.7	
R8,R15,R16	10	
R10	5.1	
R12,R21	0.5	
R13	3.3	
R14	0.1	
R17	5	
R19,R28	1.5	
R22	0.56	
R24	2.2	
VR1,VR2	2.2	
VR3,VR4	4.7	
C1,C3,C4,C5,C6,C7,C8,C9,C13		0.1
C2,C12		0.0001
C10		0.01
C11		3.3



FLOW CHART FOR DISPLAY IMAGE FILE .



FLOW CHART FOR DIGITIZE IMAGE .

โปรแกรม 1

```
/*============= DISPLAY IMAGE 256 x 256 ================*/
/*=======================================================*/

#include  <stdio.h>
#include  <alloc.h>
#include  <fcntl.h>
#include  <float.h>
#include  <dos.h>

          int   Glev   = 256          ;
          int   flag   = 1310         ;

extern    int   _fmode                ;

typedef   struct data {
                        unsigned char far *pxy[256] ;
                      } dataxy              ;
typedef   struct  dataxy  *dpointer         ;
typedef   unsigned int      table[256]      ;

/*=======================================================*/
main()
 {
     table      htb         ;
     table      heq         ;

     _fmode  =  O_BINARY    ;

     printf("Gray level of IMAGE ") ;
     scanf("%d",&Glev) ;
     printf("%10d\n",Glev) ;
     begin(htb,heq)     ;
     printf("End.") ;
   }

/*=======================================================*/
begin(table htb,table heq)
 {

  unsigned  char      lut[256]   ;
            char      fname[60]  ;
            dataxy    dst        ;
            int       c          ;
            int       read = 1   ;
            int       i          ;

 char       *errmsg[]  = {"\n Not enough memory",
                          "\n Read error ! "    ,
                          "\n Write error !"    ,
                          "\n Do not REPLACE any disk during calculation !"
                          } ;

   if((c = imalloc(&dst)) != flag) {
      puts(errmsg[0])   ;
      ifree(&dst,c)     ;
      return(1)         ;
     }
b1: getname(fname,read) ;           /*for loading file*/
    printf("\nLoading..") ;
```

```
    if(iload(&dst,fname) != flag) {
      puts(errmsg[1]) ;
      getch() ;
      goto b1 ;
     }
    printf("\nDisplay SOURCE image..") ;
    disp(&dst)          ;
    ifree(&dst,256)     ;
    s()                 ;
  }
/**********************************/
s()
  {
     sound(800) ;
     delay(200) ;
     nosound()  ;
   }
/**********************************/
imalloc(dpointer dptr)
 {
 int   i ;

     for(i = 0 ; i < 256 ; i++) {
       if((dptr->pxy[i] = (unsigned char *) farmalloc(256)) == NULL)
         return(i) ;
       }
    return(flag) ;
   }
/**********************************/
ifree(dpointer dptr,int num)
 {
 int   i ;

    for(i = 0 ; i < num ; i++) {
      farfree(dptr->pxy[i]) ;
     }
  }
/**********************************/
iload(dpointer dp ,char *fname)
 {
         int   handle    ;
         int   i         ;
         int   mode_rd   ;
         int   ok = flag ;


   mode_rd = O_RDONLY | O_BINARY                ;

   if((handle = _open(fname,mode_rd)) == EOF) {
     fprintf(stderr,"Can't open file %s",fname) ;
      return(NULL) ;
     }
   for(i = 0 ; i < 256 ; i++) {
     if( _read(handle,dp->pxy[i],256) != 256) {
       ok = EOF    ;
       break       ;
        }
      }
   close(handle) ;
   return(ok)    ;
 }
/**********************************/
```

```
disp(dpointer dp )
 {
   unsigned int    offset  = 0 ;
            int    i ,j        ;
   unsigned char   huge *d    ;

    outportb(0x300,0x61) ;
    for(j = 0 ; j < 256 ; j++) {
      d = dp->pxy[j] ;
      for(i = 0 ; i < 256 ; i++)
        pokeb(0xd000,offset+(j*256)+i,*(d+i)) ;
     }
    outportb(0x300,0x81) ;
  }
/**************************************/
getname(char *fname,int mode)
 {
   int    lp = 0 ;

   do {
       clrscr() ;
       printf("Enter file name <drive:file.ext> ") ;
       scanf("%s",fname) ;
       lp = !file_exist(fname,mode) ;
       if(mode) {
         if(lp)  {
           printf("Permission denied OR file not found.\n\n<press any key> ") ;
           bioskey(0) ;
           }
         }
       else {
         if(!lp) {
           printf("File already exist overwrite  Y/N ? ") ;
            switch(getche()) {
               case 'Y' : lp = 0 ; break ;
               case 'y' : lp = 0 ; break ;
               default  : lp = 1 ; break ;
                }
              }
        else
          lp = 0 ;
         }
     } while(lp) ;
 }
/**************************************/
file_exist(char *fname , int mode)   /* rd mode = 1 <> wr mode = 0 */
 {
   int    acs  ;

    acs = (!mode) * 06  ;
    return(access(fname,acs) == 0) ;
 }
/**************************************/
```

โปรแกรม 2

```
/*==================== DIGITIZE.EXE ====================*/
/*======================================================*/

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

#define    port      0x300
#define    fcmd      0x61
#define    dcmd      0x80
#define    digi      0x02

unsigned char  fname[40]  ;
        char  f[MAXFILE]  ;
        char  d[MAXDRIVE] ;
        char  dir[MAXDIR] ;
        char  ex[MAXEXT]  ;

main()
 {
   int   spf = 12 ;
   int   lop = 0  ;

  digitize()    ;
  do{
    getname(&spf)  ;
    if(spf == 1234) {
      printf("\nInsert another new DISK and press any key.") ;
      getch()       ;
      lop   = 1     ;
       }
    if(spf == 12) {
      printf("\nSAVING       %s..",fname) ;
      if(wrfile(fname))
        printf("\nWRITE ERROR") ;
      lop  = 0      ;
       }
    }while(lop) ;
  outportb(port,digi) ;
  puts("\nEnd program") ;
 }
/*======================================================*/
getname(int *sp_fl)
 {
 struct ffblk  ffblk  ;
 struct dfree  dfreep ;
 long   sp            ;

 int    i             ;
 int    flag          ;
 int    drive         ;
 int    fl = 0        ;
 int    wr = 0        ;
 int    pres          ;
 char   tf[60]        ;

clrscr()  ;
*sp_fl = 12 ;
```

```
do
  {
  printf("Enter file name..") ;
  scanf("%s",fname) ;
  flag  = fnsplit(fname,d,dir,f,ex) ;
   if(flag & DRIVE) {
      if(d[0] >= 'a' &&  d[0] <= 'z')
        drive = d[0] - 'a'+ 1 ;
      else if(d[0] >= 'A' && d[0] <= 'Z')
        drive = d[0] - 'A'+ 1 ;
      else {
        puts("Illegal drive name") ;
        fl = NULL  ;
            }
         }
  else {
      drive = 0  ;
      strcpy(d,"X:") ;
      d[0]  = 'A' + getdisk() ;
      strcpy(tf,d) ;
      strcpy(tf+2,"\\")  ;
      strncat(tf+3,fname,strlen(fname))  ;
      strncpy(fname,tf,strlen(fname)+3)  ;
       }
  pres = findfirst(fname,&ffblk,0) ;
  if(!pres) {
     printf("%s   ALREADY EXIST.. OVERWRITE  Y/N ? ",fname) ;
      switch(getche()) {
         case('y') :  wr = 1 ;  break ;
         case('Y') :  wr = 1 ;  break ;
         default   :  wr = 0 ;  break ;
                    }
         printf("\n") ;
            }
   } while(!pres && !wr) ;
  getdfree(drive,&dfreep) ;
  sp = (long)dfreep.df_avail * (long)dfreep.df_bsec * (long)dfreep.df_sclus ;
  sp += (long) (wr * ffblk.ff_fsize) ;
  printf("\nDRIVE %s %10ld bytes available.",d,sp) ;
  if(sp < 0x10000) {
    printf("\nWARNING  Insufficient disk space for IMAGE file. ") ;
    *sp_fl  =  1234 ;
               }
  return(fl) ;
  }
/*=================================================*/
digitize()
 {
  int    dig ;
  int    lp  ;

    printf("Do you want to DIGITIZE  Y/N ? ") ;
    switch(getche()) {
       case('N')   :   dig = 0 ; break ;
       case('n')   :   dig = 0 ; break ;
       default     :   dig = 1 ; break ;
        }
    if(dig) {
       do{
         clrscr() ;
```

```
        outportb(port,digi) ;
        printf("\nPress any key to get PICTURE ") ;
        getch() ;
        outportb(port,dcmd) ;
        printf("\nSave Y/N ? ") ;
        switch(getche()){
          case('Y') : lp = 0 ; break ;
          case('y') : lp = 0 ; break ;
          default   : lp = 1 ; break ;
            }
         }while(lp) ;
       }
     else
       outportb(port,dcmd) ;
  }
/*=================================================*/
wrfile(char *file)
{
  struct   SREGS sregs    ;
         union    REGS  regs     ;
  unsigned  int   byte     ;

          int   ret = 0   ;
          int   handle    ;
          int   i         ;

                regs.h.ah   = 0x3c          ;
                regs.x.cx   = 0             ;
                sregs.ds    = _DS           ;
                regs.x.dx   = (unsigned) file ;

    handle      = intdosx(&regs,&regs,&sregs) ;
    if(regs.x.cflag) {
        printf("\nDirectory full or file has READ ONLY attribute.") ;
        return(NULL) ;
        }
    outportb(port,fcmd) ;
   for (i = 0 ; i < 2 ; i++)  {
    regs.h.ah   = 0x40          ;
    regs.x.bx   = handle        ;
    regs.x.cx   = 0x8000        ;
    regs.x.dx   = i*0x8000      ;
    sregs.ds    = 0xd000        ;

    byte        = intdosx (&regs,&regs,&sregs) ;
    if (regs.x.cflag) {
                    ret = 345 ;  break ;
      }
    else if (byte != 0x8000) {
                                ret = 345 ;  break ;
      }
    else ;
    }
     regs.h.ah  = 0x3e          ;
     regs.x.bx  = handle        ;

     intdosx(&regs,&regs,&sregs)  ;
          if(regs.x.cflag)
                              ret = 345 ;
                return(ret) ;
        }
/*=================================================*/
```

โปรแกรม 3

```
/*-------------------------------------------------------*/
/*      This is the source program of reduce.exe         */
/*-------------------------------------------------------*/
#include <stdio.h>
#include <bios.h>
#include <errno.h>
#include <alloc.h>
#include <float.h>

#define   port     0x300
#define   vbcmd    0x81
#define   wrcmd    0x61
#define   dpcmd    0x80
#define   segment  0xd000

char      *rdblock()          ;

extern    int      errno      ;
extern    char     *sys_errlist[] ;

int       RATIO               ;
/*=======================================================*/
main()
  {
    int   x1,x2,y1,y2      ;
    int   x0,y0            ;
    int   s       = 1      ;
    int   size    = 2      ;

      menu() ;

      do{
      clrscr()    ;
      printf("\nEnter coordinate (x1 y1 x2 y2)..") ;
      scanf("%d %d %d %d",&x1,&y1,&x2,&y2) ;
      printf("Choose ratio to zoom  ? : 1\nratio = ") ;
      scanf("%d",&size) ;
      printf("Source of Image to zoom.\n") ;
      printf("          M  from monitor\n") ;
      printf("          F  from file\n")    ;
      printf("Where ? ") ;
      switch(getche()) {
         case  'm' :  s = 1 ; break ;
         case  'M' :  s = 1 ; break ;
         default   :  s = 0 ; break ;
        }
      printf("\nEnter coordinate of UPPER LEFT corner to display result.\n") ;
      printf("(x y).. ") ;
      scanf("%d %d",&x0,&y0) ;
      para(x1,y1,x2,y2,size,s,x0,y0) ;
      printf("\nPress any key to cancel or <G> to GO  ") ;
         } while(getch() != 'g') ;
      RATIO = size ;
      reduction(x1,y1,x2,y2,s,x0,y0) ;
      puts("End main") ;
   }
/*=======================================================*/
menu()
 {
  clrscr() ;
```

```
    printf("<READ ME>\n") ;
    printf("\nPlease read carefully !\n") ;
    printf("\nParameter to use in this program .\n") ;
    printf("    (1) coordinate of area to zoom x1 y1 x2 y2\n") ;
    printf("            : each value must in the range 0 - 255.\n") ;
    printf("    (2) ratio of zooming\n") ;
    printf("            : posible value -1,2,3,...\n") ;
    printf("    (3) source of data to zoom\n") ;
    printf("            : M  zoom data on TV monitor.\n") ;
    printf("            : F  zoom data from file.\n") ;
    printf("    (4) location to display result x0 y0\n") ;
    printf("            : value in range 0 - 255.\n") ;
    printf("            : MAKE SURE THERE'S ENOUGH AREA TO DISPLAY.\n\n") ;
    printf("Press any key ") ;
    getch() ;
  }
/*=================================================*/
para(int x1,int y1,int x2,int y2,int size,int s,int x0,int y0)
 {
  clrscr() ;
  printf("Working area..\n") ;
  printf("UPPER LEFT              X = %5d , Y = %5d\n",x1,y1) ;
  printf("LOWER RIGHT             X = %5d , Y = %5d\n",x2,y2) ;
  printf("RATIO                  %d : 1\n",size) ;
  if(s)
    printf("GRAB data on  TV monitor.\n") ;
  else
    printf("READ disk file.\n") ;
  printf("Display area..\nUPPER LEFT                ") ;
  printf("X = %5d , Y = %5d\n",x0,y0) ;
  }
/*=================================================*/
reduction(int x1,int y1,int x2,int y2,int source,int x0,int y0)
  {
    FILE    *fp          ;
            int  hi , wi  ;
    unsigned char *zbuf    ;
    unsigned char *data    ;
            char file[60] ;
            char *data1   ;
    hi = y2-y1+1 ;
                wi = x2-x1+1 ;

    if((zbuf = (unsigned char *) malloc(hi/RATIO*wi/RATIO)) == NULL)
      {  perror("Memory allocation") ;  return(NULL) ;
       }

    if((data  = (unsigned char *) malloc(hi*wi)) == NULL)
      {  puts(sys_errlist[errno]) ;     return(NULL) ;
       }

    if(source)
        getblock(x1,y1,x2,y2,data) ;
    else
       {
        clrscr() ;
        printf("Enter file name..") ;
                                    scanf("%s",file) ;
        if((fp = fopen(file,"rb")) == NULL)
          return(NULL) ;
```

```
        if((data = rdblock(x1,y1,x2,y2,fp)) == NULL)
           return(NULL) ;
        fclose(fp) ;
         }
    puts("\nPress any key")   ;
    getch()                  ;
                  if(RATIO == 1)
                     dispblock(x0,y0,x0+wi-1,y0+hi-1,data) ;
    else {
        zoom_s(data,zbuf,hi,wi)   ;
        dispblock(x0,y0,x0+(wi/RATIO)-1,y0+(hi/RATIO)-1,zbuf) ;
                                    }
                  free(data) ;
    return(0) ;
   }
/*=================================================*/
/*                  Small                          */
zoom_s(unsigned char *data,unsigned char *zdata,int hi,int wi)
{
          int   i ,j ,k ,l  ;
  unsigned char  *dptr       ;
          float d = 0.0     ;
  printf("Working..") ;
  for(j = 0 ; j < hi/RATIO ; j++)
   {
    for(i = 0 ; i < wi/RATIO ; i++)
     {
      dptr = data + j*wi*RATIO + i*RATIO  ;
      for(k = 0 ; k < RATIO ; k++)
       {
        for(l = 0 ; l < RATIO ; l++)
          d  += *(dptr + l + k*wi) ;
        }
            d   /= (float) (RATIO*RATIO)  ;
        *zdata++  = (unsigned char) d      ;
            d   -= d                       ;
      }
     printf(".") ;
    }
  printf("\n") ;
}
/*=================================================*/
dispblock(int x1,int y1,int x2,int y2,char *data)
{
   int  i,j ;

  if(x1 <= x2 && y1 <= y2 && x2 <= 255 && y2 <= 255)
   {
     outportb(port,wrcmd) ;
     for(j = y1 ; j < y2+1 ; j++)
      {
        for(i = x1 ; i < x2+1 ; i++)
          pokeb(segment,j*256+i,*data++) ;
        }
      outportb(port,dpcmd) ;
     }
   else
     puts("Limits are out of range.");
  }
/*=================================================*/
```

```
getblock(int x1,int y1,int x2,int y2,char *data)
 {
   int  i,j ;

   if(x1 <= x2 && y1 <= y2 && x2 <= 255 && y2 <= 255)  {
     outportb(port,wrcmd) ;
     for(j = y1 ; j < y2+1 ; j++)  {  . .
       for(i = x1 ; i < x2+1 ; i++)
         *data++ = peekb(segment,j*256+i) ;
       }
      outportb(port,dpcmd) ;
     }
    else
     puts("Limits are out of range.") ;
   }
/*=======================================================*/
int rdfile(fp,buffer,pos,size)
         FILE   *fp ;
         char   *buffer ;
 unsigned int    pos ,size ;
  {
    if(!(fseek(fp,pos,SEEK_SET)))
      {
       fread(buffer,size,1,fp) ;
       return(0) ;
       }
    else
       return(1) ;
   }

/*=======================================================*/
char *rdblock(int x1,int y1,int x2,int y2,FILE *fp)
 {
    unsigned int    bsize ;
             int    wd ;
             char   *data ;
             int    i,j ;

  if(x1 <= x2 && y1 <= y2 && x2 <= 255 && y2 <= 255)
    {
     bsize = (x2-x1+1)*(y2-y1+1) ;
     wd    = (x2-x1+1) ;

     if((data = (char *) malloc(bsize)) == NULL)
       {  perror("in rdblock()") ;   return(NULL) ; }
      printf("Reading") ;
      for(i = y1,j = 0 ; i <= y2+1 ; i++,j++)
        {
         if( rdfile(fp,data+j*wd,i*256+x1,wd) )
           return(NULL) ;
         printf(".") ;
         }
       return((char *) data);
      }
   else
     {
      puts("Limits are out of range.") ;
      return(NULL) ;
      }
   }
/*=======================================================*/
/*====================== end prog =======================*/
```

โปรแกรม 4

```
/*=================================================*/
/*                                                 */
/* pi()          print gray level of pattern used. */
/* pfile()       read file and print picture.       */
/* lutable()     set up look up table.             */
/* rdfile()      read file at given position.(for any size) */
/* setdambx()    form a pattern of given data.     */
/* prn()         send data format to print in graphics mode */
/* init_prn()    initial printer ,set line feed,on line,etc */
/*                                                 */
/*=================================================*/
#include  <stdio.h>
#include  <bios.h>
#include  <errno.h>
#include  <alloc.h>
#include  <conio.h>

extern   int  errno ;
extern   char *sys_errlist[] ;
struct [
         int    prnpix   ;
         int    datpix   ;
         char   mode[3]  ;
         int    fool     ;
         int    frow     ;
        } pinfo ;
/*=================================================*/
main()
  [
     int     pattern  = 3   ;
     char    level    = 64  ;
     int     flag           ;
     int     f_pi           ;
     char    ch             ;
     int     loop     = 1   ;
     int     rv       = 1   ;

      setvalprn() ;
      clrscr()    ;
      printf(" <Read me>\n") ;
      printf(" Print Image  Version 0.0 \n") ;
      printf("       by     IMAGE  TEAM \n") ;
      printf("\n\n This program develop on LQ1050.\n") ;
      printf(" Picture size  29 x 29 cm.\n\n") ;
      printf(" Press any key   < Q to quit > ") ;
      ch = getch()     ;   clrscr()  ;
        if( ch == 'Q' || ch == 'q')
           exit(1)  ;
      puts(" <Parameter>") ;
      puts(" Font pattern (1) snoud use Gray level 32 :") ;
      puts("                  (2)                      48 :") ;
      puts("                  (3)                      64 :") ;
      puts("                  (4)                      64 :") ;
      puts("                  (5)                      64 :") ;
      puts("                  (6)                      64 :") ;
      puts("                  (7)                      64 :") ;
      printf("\n Press any key ")  ;
      getch()      ;   clrscr() ;

      printf(" Do you want to print gray pattern (y or n) ? ") ;
```

```
        switch(getch()) {
          case('n') : f_pi = 0 ; break ;
          case('N') : f_pi = 0 ; break ;
          default   : f_pi = 1 ; break ;
          }
        do{
          clrscr() ;
          printf(" Gray level used (0 to 64).. ") ;
          scanf("%d",&level)  ;
          while(loop) {
                    printf(" Choose Font pattern used (1,2,..,7).. ") ;
                    scanf("%d",&pattern) ;
                    if(pattern > 7) {
                      printf(" Illegal value..enter a new one.\n") ;
                      loop = 1 ;
                     }
                    else
                      loop = 0 ;
                    }
         printf(" Select mode of printing.\n") ;
         printf("          - N    Negative \n") ;
         printf("          - P    Positive \n") ;
         printf(" mode ? ") ;
          switch(getch()) {
             case('N') :  rv = 0  ;  break ;
             case('n') :  rv = 0  ;  break ;
             default   :  rv = 1  ;  break ;
                         }
        puts("\n Press any key to <GO> or C to cancel.") ;
          ch   = getch()     ;
          flag  = (ch  == 'c') || (ch == 'C') ;
        if(f_pi)
          pi(level,pattern-1,rv)     ;
        loop   =  1 ;
       } while(flag) ;
      clrscr()  ;
      pfile(level,pattern-1,rv)       ;

  }
/*=================================================*/
setvalprn()
   {
      pinfo.prnpix  = 14*180     ;   /* size of paper * DPI        */
      pinfo.datpix  = 256*8*3    ;   /* data pixel = 256*font_col  */
      pinfo.mode[0] = 27         ;   /*  Esc * m                   */
      pinfo.mode[1] = 42         ;   /*  m = 39                    */
      pinfo.mode[2] = 39         ;   /*  use 24 pins of print head */
      pinfo.fcol    = 8          ;  /* column of font metrix      */
      pinfo.frow    = 8          ;   /* row of font metrix         */
   }

/*=================================================*/
pfile(char level,int pattern,int rv)
 {
   FILE  *fp ;
   char  file[40] ;
   int   i,n ;
   int   size = 3*256 ;
   int   item = 85  ;              /* file size = item * size   */
   char  tabl[256] ;
```

```
    unsigned char *buffer  ;
    unsigned char *pdata   ;

    printf(" Enter file name..") ;
    scanf("%s",file)             ;

    if((fp = fopen(file,"rb")) == NULL) {. .
       puts(sys_errlist[errno]) ;  exit(1) ;
      }
    if(buffer = (unsigned char *) malloc(size)) == NULL)
     {
       perror("Memory allocation") ;  exit(2) ;
      }
    if(pdata  = (unsigned char *) malloc(pinfo.datpix)) == NULL)
     {
       puts(sys_errlist[errno]) ;  exit(3) ;
      }

    lutable(tabl,level) ;
    printf(" Use font style %d \n Gray level %d \n",pattern+1,level) ;
     if(rv)  printf(" Normal mode.\n") ;

     else
            printf(" Reverse mode.\n") ;
    init_prn() ;
    for(n = 0 ; n < item ; n++)
     {
       if(rdfile(fp,buffer,n*size,size))
         {  puts("read error")  ;  exit(4) ;  }
       printf(".") ;
       setdamtx(buffer,pdata,tabl,1,pattern,rv) ;
       prn(pinfo.datpix,pdata) ;
      }
    rdfile(fp,buffer,85*size,(int) size/3) ;
    setdamtx(buffer,pdata,tabl,0,pattern,rv) ;
    prn(pinfo.datpix,pdata) ;
    fclose(fp) ;
    free(buffer) ;
    free(pdata) ;
    puts("EOF") ;
  }
/*=================================================*/
pi(char level,int pt,int rv)        /* print font of gray level 16 used. */
 {
   unsigned  char  test[256*3]  ;
             char  table[256]   ;
   unsigned  char  *data        ;
             int   i            ;

   for( i = 0 ; i < 256; i++)
    {
      test[i]     = (char) i ;
      test[i+256] = (char) i ;
      test[i+512] = (char) i ;
    }
   lutable(table,level) ;
   if((data = (unsigned char*) malloc(pinfo.datpix)) == NULL) {
     perror("Allocation") ;    exit(2) ;
   }
   setdamtx(test,data,table,1,pt,rv) ;
```

```
    init_prn() ;
    prn(pinfo.datpix,data) ;
    prn(pinfo.datpix,data) ;
    prn(pinfo.datpix,data) ;
    setdamtx(test,data,table,0,pt,rv) ;
    prn(pinfo.datpix,data) ;
    puts("end pi()") ;
   }
/*=================================================*/
lutable(char *table,char newgrayh)
 {
   int  gmax = 256 , gmin = 0 ;
   char ngl  = 0 ;
   int  c,k,gray ;

   c = newgrayh - ngl  ;
   k = gmax - gmin     ;

   for(gray = 0 ; gray < 256 ; gray++)
     *(table+gray) = (char) ((gray - gmin)*c/k) + (char) ngl ;
 }
/*=================================================*/
init_prn()
 {
   int i               ;
   int cmd[3] = {0,1,2} ;
   int initial[] = {7,27,51,23,27,85,1} ;
   int port = 0 ;

   biosprint(cmd[1],0,port) ;
   for(i=0 ; i<7 ; i++)
     biosprint(cmd[0],initial[i],port) ;
 }
/*=================================================*/
prn(int nbyte,char *data)
 {
   unsigned char  *prnbuf  ;
   unsigned char  *pdata   ;
          int   n ;
          int   MAXPOINT ;
          int   n1,n2 ;
          int   port  = 0 ;
   unsigned char  backg = 00;

   MAXPOINT  = 3*pinfo.prnpix  ;

   if( nbyte < MAXPOINT )
     {
     if((prnbuf = (unsigned char*) malloc(MAXPOINT)) == NULL)
       { perror("in prn") ; exit(1) ; }

      n1 =((MAXPOINT-5)/3) %256 ;
      n2 =((MAXPOINT-5)/3) /256 ;

      *(prnbuf)             = pinfo.mode[0]  ;
      *(prnbuf+1)           = pinfo.mode[1]  ;
      *(prnbuf+2)           = pinfo.mode[2]  ;
      *(prnbuf+3)           = n1   ;
      *(prnbuf+4)           = n2   ;
      *(prnbuf+MAXPOINT-2)  = 0xd  ;
      *(prnbuf+MAXPOINT-1)  = 0xa  ;
```

```
            for(n = 5 ; n < MAXPOINT-2 ; n++)
               *(prnbuf+n) = backg   ;
            pdata   = data ;
            for(n = 701 ; n < 701+nbyte ; n++)
               *(prnbuf+n) = *(pdata++) ;
            for(n = 0 ; n < MAXPOINT ; n++)
               biosprint(0,*(prnbuf+n),port) ;
            free(prnbuf) ;
           }
    }
/*=================================================*/
int rdfile(fp,buffer,pos,size)
          FILE   *fp ;
          char   *buffer ;
 unsigned int    pos ,size ;
    {
      if(!(fseek(fp,pos,SEEK_SET))) {
         fread(buffer,size,1,fp) ;
         return(0) ;
         }
      else
         return(1) ;
    }

/*=================================================*/
setdamtx(unsigned char *buf,unsigned char *prndat,char *table,int mode,int pt,int rv)
  /*  buf    = data to set :                 */
  /*  prndat = data to printer               */
  /*  table  = address of look up table      */
  /*  mode   = use when print file  1 = data 3 lines , 0 = 1 line */
  /*  pt     = matrix of font used           */
  /*  rv     = mode of printing 1 = normal , 0 = reverse          */

{
 char pattern[][8][8] ={ { { 0,16, 4,20, 1,17, 5,21  },
                          {24, 8,28,12,25, 9,29,13   },
                          { 6,22, 2,18, 7,23, 3,19   },
                          {30,14,26,10,31,15,27,11   },
                          { 1,17, 5,21, 0,16, 4,20   },
                          {25, 9,29,13,24, 8,28,12   },
                          { 7,23, 3,19, 6,22, 2,18   },
                          {31,15,27,11,30,14,26,10   } } ,

                        { { 38,46,0,24,20,0,41,37    },
                          { 42,34,0,12,16,0,33,45    },
                          { 0,0,29,4,8,28,0,0        },
                          { 21,17,9,0,3,7,15,23      },
                          { 25,13,5,1,2,11,19,27     },
                          { 0,0,30,10,6,31,0,0       },
                          { 43,35,0,18,14,0,32,40    },
                          { 39,47,0,26,22,0,44,36    } } ,

                        { {0,32,8,40,2,34,10,42      },
                          {48,16,56,24,50,18,58,26   },
                          {12,44,4,36,14,46,6,38     },
                          {60,28,52,20,62,30,54,22   },
                          {3,35,11,43,1,33,9,41      },
                          {51,19,59,27,49,17,57,25   },
                          {15,47,7,39,13,45,5,37     },
                          {63,31,55,23,61,29,53,21   } } ,
```

```
                    { {0,8,16,24,32,40,48,56    },
                      {1,9,17,25,33,41,49,57    },
                      {2,10,18,26,34,42,50,58   },
                      {3,11,19,27,35,43,51,59   },
                      {4,12,20,28,36,44,52,60   },
                      {5,13,21,29,37,45,53,61   },
                      {6,14,22,30,38,46,54,62   },
                      {7,15,23,31,39,47,55,63   }  } ,

                    { {0,1,2,3,4,5,6,7          },
                      {8,9,10,11,12,13,14,15    },
                      {16,17,18,19,20,21,22,23  },
                      {24,25,26,27,28,29,30,31  },
                      {32,33,34,35,36,37,38,39  },
                      {40,41,42,43,44,45,46,47  },
                      {48,49,50,51,52,53,54,55  },
                      {56,57,58,59,60,61,62,63  }  },

                    { {0,1,2,3,4,5,6,7          },
                      {27,28,29,30,31,32,33,8   },
                      {26,47,48,49,50,51,34,9   },
                      {25,46,59,60,61,52,35,10  },
                      {24,45,58,63,62,53,36,11  },
                      {23,44,57,56,55,54,37,12  },
                      {22,43,42,41,40,39,38,13  },
                      {21,20,19,18,17,16,15,14  }  },

                    { {28,36,43,49,54,58,61,63  },
                      {21,29,37,44,50,55,59,62  },
                      {15,22,30,38,45,51,56,60  },
                      {10,16,23,31,39,46,52,57  },
                      {6,11,17,24,32,40,47,53   },
                      {3,7,12,18,25,33,41,48    },
                      {1,4,8,13,19,26,34,42     },
                      {0,2,5,9,14,20,27,35      }  } };

  unsigned char   *row1,*row2,*row3 ;
           int    c,r,j ;
  unsigned char   n1,n2,n3,b1=0,b2=0,b3=0  ;

if(mode) {
           row1 = buf.          ;
           row2 = buf + 0x100 ;
           row3 = buf + 0x200 ;
  for(j = 0 ; j < 256 ; j++) {
     n1 =  *(table+(*(row1+j))) ;
     n2 =  *(table+(*(row2+j))) ;
     n3 =  *(table+(*(row3+j))) ;
    for(c = 0 ; c < pinfo.fcol ; c++)  {
      for(r = 0 ; r < pinfo.frow ; r++) {
        b1 <<= 1 ;   b2 <<= 1 ;   b3 <<= 1 ;
        b1 |= n1 <  pattern[pt][r][c] ? rv : !rv ;
        b2 |= n2 <  pattern[pt][r][c] ? rv : !rv ;
        b3 |= n3 <  pattern[pt][r][c] ? rv : !rv ;
        }
       *prndat++ = b1 ;
       *prndat++ = b2 ;
       *prndat++ = b3 ;
    }
  }
}
```

```
  else {
     row1 = buf ;
     for(j = 0 ; j < 256 ; j++) {
       n1 = *(table+(*(row1+j))) ;
       for(c = 0 ; c < pinfo.fcol ; c++) {
         for(r = 0 ; r < pinfo.frow ; r++) {
            b1 <<= 1 ;
            b1 |= n1 < pattern[pt][r][c] ? rv : !rv ;
          }
          *prndat++ = b1 ;
          *prndat++ = 0  ;
          *prndat++ = 01 ;
        }
       }
    }
  }
/*=================== end prog ===================*/
```

โปรแกรม 5

```
/*=================================================*/
#include   <stdio.h>
#include   <alloc.h>
#include   <fcntl.h>
#include   <float.h>
#include   <dos.h>
#include   <graphics.h>
#include   <bios.h>
#include   <errno.h>

           int    Glev    = 256          ;
           int    flag    = 1310         ;
           int    FROW    = 8            ;
           int    FCOL    = 8            ;

typedef  unsigned char  *dataptr[256]     ;

extern           int   _fmode             ;
extern           int   errno              ;
extern           char  *sys_errlist[]     ;

/*=================================================*/
main()
 {

     _fmode  =  O_BINARY   ;

      if(begin() == flag) {
        s()                ;
        outtextxy(50,320,"Press any key to TEXT mode. ") ;
        getch()            ;
        closegraph()       ;
        printf("End.")     ;
       }
      else {
        s()   ;   s() ;  s() ;
       }
   }
/*=================================================*/
begin()
 {
   unsigned   char       lut[256]       ;
   unsigned   char       *mnset[8][256] ;
              char       fname[60]      ;
              dataptr    dst            ;
              int        c              ;
              int        read = 1       ;
              int        i              ;

 char       *errmsg[]  = {"\n Not enough memory",
                          "\n Read error ! "    ,
                          "\n Write error !"      } ;

    if((c = imalloc(dst)) != flag) {
       puts(errmsg[0])   ;
       ifree(dst,c)      ;
       return(1)         ;
      }
b1: getname(fname,read) ;          /*for loading file*/
    printf("\nLoading..") ;
```

```
    if(iload(dst,fname) != flag) {
       puts(errmsg[1]) ;
       getch() ;
       goto b1 ;
     }
    lutable(lut,64)      ;
    monosetdata(mnset)   ;
    if(tograph()) {
      process(dst,lut,mnset)  ;
      ifree(dst,256)          ;
      return(flag)            ;
      s()                     ;
     }
    else
      return(NULL)            ;
  }
/*================================================*/
lutable(char *table,char ngh)
  {
    int  gmax = 256 , gmin = 0 ;
    char ngl  = 0 ;
    int c ,k ;
    int gray  ;

     c = ngh - ngl    ;
     k = gmax - gmin  ;

     for(gray = 0 ; gray < 256 ; gray++)
       *(table+gray) = (char) ((gray - gmin)*c/k) + (char) ngl ;
   }
/*================================================*/
tograph()
 {
    int      graphdriver = DETECT ;
    int      gmode  ;
    int      gerror ;

    initgraph(&graphdriver,&gmode,"") ;
    gerror = graphresult() ;
    if(gerror < 0) {
      printf("initgraph error: %s.\n",grapherrormsg(gerror)) ;
      closegraph() ;
      return(NULL) ;
      }
    return(1) ;
  }
/*================================================*/
monosetdata(unsigned char monotable[][256])
 {
  char   pattern[2][8][8] ={  { {0,32,8,40,2,34,10,42      },
                                {48,16,56,24,50,18,58,26   },
                                {12,44,4,36,14,46,6,38     },
                                {60,28,52,20,62,30,54,22   },
                                {3,35,11,43,1,33,9,41      },
                                {51,19,59,27,49,17,57,25   },
                                {15,47,7,39,13,45,5,37     },
                                {63,31,55,23,61,29,53,21   }
                              },
```

```
        {  {0,12,2,15,0,12,2,15      },
           {8,5,10,6,8,5,10,6        },
           {3,14,1,13,3,14,1,13      },
           {11,7,9,4,11,7,9,4        },
           {0,12,2,15,0,12,2,15      },
           {8,5,10,6,8,5,10,6        },
           {3,14,1,13,3,14,1,13      },
           {11,7,9,4,11,7,9,4        }
           }
      } ;

        int    row,col ;
        int    ncol    ;
   for(col = 0 ; col < 256 ; col++)  {
     for(row = 0 ; row < 8 ; row++)  {
       ncol = col % 8   ;
       monotable[row][col] = pattern[0][row][ncol]  ;
      }
    }
  }
/*********************************/
s()
  {
     sound(800) ;
     delay(200) ;
     nosound()  ;
  }
/*********************************/
imalloc(dataptr dptr)
 {
  int  i ;

     for(i = 0 ; i < 256 ; i++) {
       if((dptr[i] = (unsigned char *) farmalloc(256)) == NULL)
         return(i) ;
      }
    return(flag) ;
  }
/*********************************/
ifree(dataptr dptr,int num)
 {
 int  i ;

    for(i = 0 ; i < num ; i++) {
      farfree(dptr[i]) ;
     }
  }
/*********************************/
process(unsigned char *dat[],unsigned char ltb[],unsigned char monotb[][256])
 {
      int     i,j,k          ;
 unsigned char    d,ng           ;
 unsigned char   *data           ;

  for(i = 0 ; i < 256 ; i++) {
    data  = dat[i]  ;
    for(j = 0 ; j < 256 ; j++) {
      d   = *data++  ;
      ng  = ltb[d]   ;
      k   = i % 8    ;
```

```
      if(ng <= monotb[k][j] )
        putpixel(50+j,50+i,0) ;
      else
        putpixel(50+j,50+i,1) ;
      }
    }
 }
/*****************************************/
iload(dataptr dp,char *fname)
 {
         int   handle    ;
         int   i         ;
         int   mode_rd   ;
         int   ok = flag ;

    mode_rd = O_RDONLY | O_BINARY  ;

    if((handle = _open(fname,mode_rd)) == EOF) {
      fprintf(stderr,"Can't open file %s",fname) ;
       return(NULL) ;
      }
    for(i = 0 ; i < 256 ; j++) {
      if( _read(handle,dp[i],256) != 256) {
        ok = EOF    ;
        break       ;
         }
        }
    close(handle)  ;
    return(ok)     ;
  }
/*****************************************/
getname(char *fname,int mode)
 {
   int    lp = 0 ;

   do {
        clrscr() ;
        printf("Enter file name <drive:file.ext> ") ;
        scanf("%s",fname) ;
        lp = !file_exist(fname,mode) ;
        if(mode) {
          if(lp)  {
            printf("Permission denied OR file not found.\n\n<press any key> ") ;
            bioskey(0) ;
            }
          }
        else {
          if(!lp) {
            printf("File already exist overwrite  Y/N ? ") ;
             switch(getche()) {
                case 'Y' : lp = 0 ; break ;
                case 'y' : lp = 0 ; break ;
                default  : lp = 1 ; break ;
                 }
               }
          else
            lp = 0 ;
           }
      } while(lp) ;
 }
/*****************************************/
```

```
file_exist(char *fname , int mode)   /* rd mode = 1 <> wr mode = 0 */
 {
   int    acs ;

     acs = (!mode) * 06  ;
     return(access(fname,acs) == 0) ;
  }
/*********************************/
```

CA3300

CMOS Video Speed 6-Bit Flash Analog-to-Digital Converter

For Use in Low-Power Consumption, High-Speed Digitization Applications

Features:

- *CMOS low power with speed*
- *Parallel conversion technique*
- *15-MHz sampling rate (66-ns conversion time)*
- *6-bit latched 3-state output with overflow bit*
- *±½ LSB accuracy*
- *Single supply voltage (3 to 10 V)*
- *2 units in series allow 7-bit output*
- *2 units in parallel allow 30-MHz sampling rate*
- *Internal V_{REF} with ext V_{REF} option*
- *Available with EVP processing for improved reliability*

The RCA-CA3300 types are CMOS 50-mW parallel (FLASH) analog-to-digital converters designed for applications demanding both low-power consumption and high-speed digitization.

The CA3300 types operate over a wide full-scale input-voltage range of 2.4 volts up to the dc supply voltage with maximum power consumption as low as 50 to 200 mW, depending upon the clock frequency selected. When operated from a 5-volt supply at a clock frequency of 11 MHz, the power consumption of the CA3300 is less than 50 mW. When operated from an 8-volt supply at a frequency of 15 MHz the power consumption is less than 150 mW.

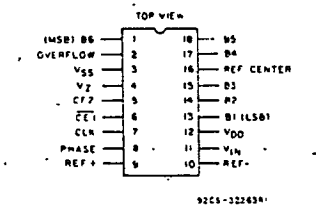
The intrinsic high conversion rate makes the CA3300 types ideally suited for digitizing high-speed signals. The overflow bit makes possible the connection of two or more CA3300's in series to increase the resolution of the conversion system. A series connection of two CA3300's may be used to produce a 7-bit high-speed converter. Operation of two CA3300's in parallel doubles the conversion speed (i.e., increases the sampling rate from 15 to 30 MHz). CA3300's in parallel may be combined with a high-speed 6-bit D/A converter, a binary adder, control logic, and an op amp to form a very-high-speed A/D converter.

Sixty-four paralleled auto-balanced voltage comparators measure the input voltage with respect to a known reference to produce the parallel-bit outputs in the CA3300. Sixty-three comparators are required to quantize all input voltage levels in this 6-bit converter, and the additional comparator is required for the overflow bit.

The CA3300 types are available as follows: Types CA3300D and CA3300DX in an 18-lead dual-in-line ceramic package (D suffix), types CA3300E and CA3300CE in an 18-lead dual-in-line plastic package (E suffix), or in chip form (H suffix). The CA3300DX offers the additional advantage of improved reliability as a result of EVP (Extra Value Program) processing. For further information on EVP, see RCA publication EVP-300B or contact your RCA representative.

Applications:

- *The CA3300 types are especially suited for high-speed conversion applications where low power is also important*
- *TV video digitizing (industrial/security)*
- *High-speed A/D conversion*
- *Ultrasound signature analysis*
- *Transient signal analysis*
- *High-energy physics research*
- *High-speed oscilloscope storage/display*
- *General-purpose hybrid ADC's*
- *Optical character recognition*
- *Radar pulse analysis*
- *Motion signature analysis*



TERMINAL ASSIGNMENT

CA3300

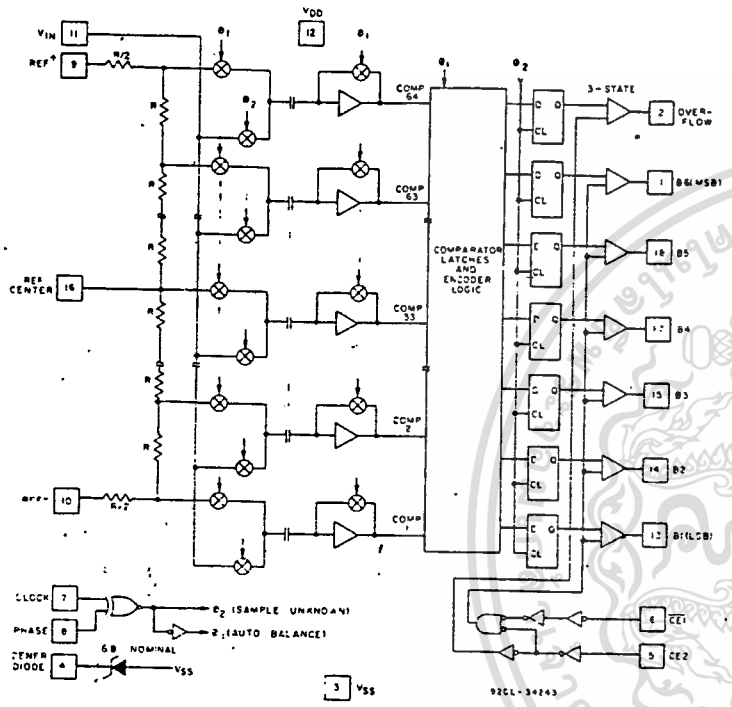


Fig. 1 - Block diagram for the CA3300.

MAXIMUM RATINGS, *Absolute-Maximum Values:*

DC SUPPLY VOLTAGE RANGE (V_{DD})
(VOLTAGE REFERENCED TO V_{SS} TERMINAL) ..... -0.5 to 10 V
INPUT VOLTAGE RANGE
ALL INPUTS EXCEPT ZENER (PIN 4) ..... -0.5 to V_{DD} +0.5 V
DC INPUT CURRENT
CLK, PH, $\overline{CE1}$, CE2, V_{IN} ..... ± 10 mA
POWER DISSIPATION PER PACKAGE (P_D)
FOR T_A = -55 to +55°C ..... 315 mW
FOR T_A = +55°C to +125°C ..... Derate linearly at 3.3 mW/°C
TEMPERATURE RANGE
OPERATING (CA3300DX, Refer to Fig. 3) ..... -55 to +125°C
OPERATING (CA3300D, E, CE) ..... -40 to +85°C
STORAGE ..... -65 to +150°C
LEAD TEMPERATURE (DURING SOLDERING)
At distance 1/16 ± 1/32 in. (1.59 ± 0.79 mm) from case for 10 s max ..... +265°C

CA3300

ELECTRICAL CHARACTERISTICS

CHARACTERISTIC	TEST CONDITIONS @ 25°C	LIMITS CA3300D, DX, E MIN.	LIMITS CA3300D, DX, E TYP.	LIMITS CA3300D, DX, E MAX.	UNITS
Resolution		—	—	6	Bits
Linearity Error	V_{DD}=8 V, V_{REF}=7.68 V, CLK=15 MHz, gain adjusted	—	±0.5	±0.8	LSB
Differential Linearity Error	V_{DD}=8 V, V_{REF}=7.68 V, CLK=15 MHz	—	±0.5	±0.8	LSB
Quantizing Error		-½	—	½	LSB
Analog Input:	V_{DD}=8 V, CLK=15 MHz				
Full Scale Range		2.4	—	V_{DD}+0.5	V
Input Capacitance		—	50	—	pF
Input Current		—	600	1000	µA
Gain Temperature Coefficient	V_{DD}=8 V, CLK=15 MHz	—	0.016	—	LSB/°C
Maximum Conversion Speed	V_{DD}=5 V	—	12M	—	SPS
	V_{DD}=8 V	15M	19M	—	SPS
Device Current (Excludes I_{REF}, I_Z)	V_{DD}=5 V (CLK=11 MHz)	—	7	—	mA
	V_{DD}=8 V (CLK=15 MHz)	—	22	—	mA
	V_{DD}=5 V (Auto Balance State)	—	6.4	16	mA
	V_{DD}=8 V (Auto Balance State)	—	24	40	mA
Ladder Impedance		1000	1400	1800	Ω
Digital Inputs:					
Low Voltage	V_{DD}=5 V	—	—	1.5	V
	V_{DD}=8 V	—	—	2.5	V
High Voltage	V_{DD}=5 V	3.5	—	—	V
	V_{DD}=8 V	5.5	—	—	V
Input Current	V_{DD}=8 V	—	±1	—	µA
Digital Outputs:					
Output Low (Sink) Current	V_{DD}=5 V, V_O=0.4 V	1.6	10	—	mA
	V_{DD}=8 V, V_O=0.5 V	3.2	15	—	mA
Output High (Source) Current	V_{DD}=5 V, V_O=4.6 V	-0.8	6	—	mA
	V_{DD}=8 V, V_O=7.5 V	-1.6	9	—	mA
Zener Voltage	I_Z=10 mA	6.2	6.8	7.4	V
Zener Dynamic Impedance	I_Z=10 mA	—	10	30	Ω
Zener Temperature Coefficient		—	0.5	—	mV/°C
Digital Output Delay, t_d	V_{DD}=8 V	—	20	—	ns
Aperture Time	V_{DD}=8 V	—	25	—	ns

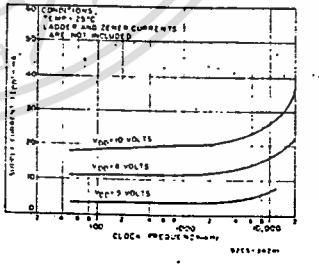


Fig. 2 - Typical current drain versus sampling rate as a function of supply voltage.

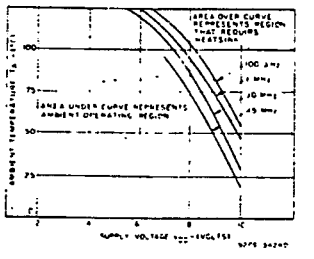


Fig. 3 - Maximum ambient temperature versus supply voltage (Above curve includes ladder dissipation but not the zener dissipation.)

CA3300

ELECTRICAL CHARACTERISTICS

CHARACTERISTIC	TEST CONDITIONS @ 25°C	LIMITS CA3300CE MIN.	LIMITS CA3300CE TYP.	LIMITS CA3300CE MAX.	UNITS
Resolution		—	—	6	Bits
Linearity Error	V_{DD}=8 V, V_{REF}=7.68 V, CLK=9 MHz, gain adjusted	—	±0.5	±0.8	LSB
Differential Linearity Error	V_{DD}=8 V, V_{REF}=7.68 V, CLK=9 MHz	—	±0.5	±0.8	LSB
Quantizing Error		-½	—	½	LSB
Analog Input:	V_{DD}=8 V, CLK=9 MHz				
Full Scale Range		2.4	—	V_{DD}+0.5	V
Input Capacitance		—	50	—	pF
Input Current		—	450	1000	µA
Gain Temperature Coefficient	V_{DD}=8 V, CLK=9 MHz	—	0.016	—	LSB/°C
Maximum Conversion Speed	V_{DD}=5 V	6M	—	—	SPS
	V_{DD}=8 V	9M	19M	—	SPS
Device Current (Excludes I_{REF}, I_Z)	V_{DD}=5 V (CLK=7 MHz)	—	4	—	mA
	V_{DD}=8 V (CLK=9 MHz)	—	12	—	mA
	V_{DD}=5 V (Auto Balance State)	—	6.4	16	mA
	V_{DD}=8 V (Auto Balance State)	—	24	40	mA
Ladder Impedance		1000	1400	1800	Ω
Digital Inputs:					
Low Voltage	V_{DD}=5 V	—	—	1.5	V
	V_{DD}=8 V	—	—	2.5	V
High Voltage	V_{DD}=5 V	3.5	—	—	V
	V_{DD}=6 V	5.5	—	—	V
Input Current	V_{DD}=8 V	—	±1	—	µA
Digital Outputs:					
Output Low (Sink) Current	V_{DD}=5 V, V_O=0.4 V	1.6	10	—	mA
	V_{DD}=8 V, V_O=0.5 V	3.2	15	—	mA
Output High (Source) Current	V_{DD}=5 V, V_O=4.6 V	-0.8	6	—	mA
	V_{DD}=8 V, V_O=7.5 V	-1.6	9	—	mA
Zener Voltage	I_Z=10 mA	6.2	6.8	7.4	V
Zener Dynamic Impedance	I_Z=10 mA	—	10	30	Ω
Zener Temperature Coefficient		—	0.5	—	mV/°C
Digital Output Delay, t_d	V_{DD}=8 V	—	20	—	ns
Aperture Time	V_{DD}=8 V	—	25	—	ns

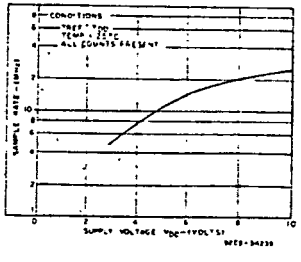


Fig. 4 - Typical maximum sample rate versus supply voltage

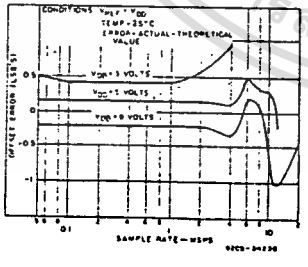


Fig. 5 - Typical offset error versus sample rate as a function of supply voltage (See literature for offset trim.)

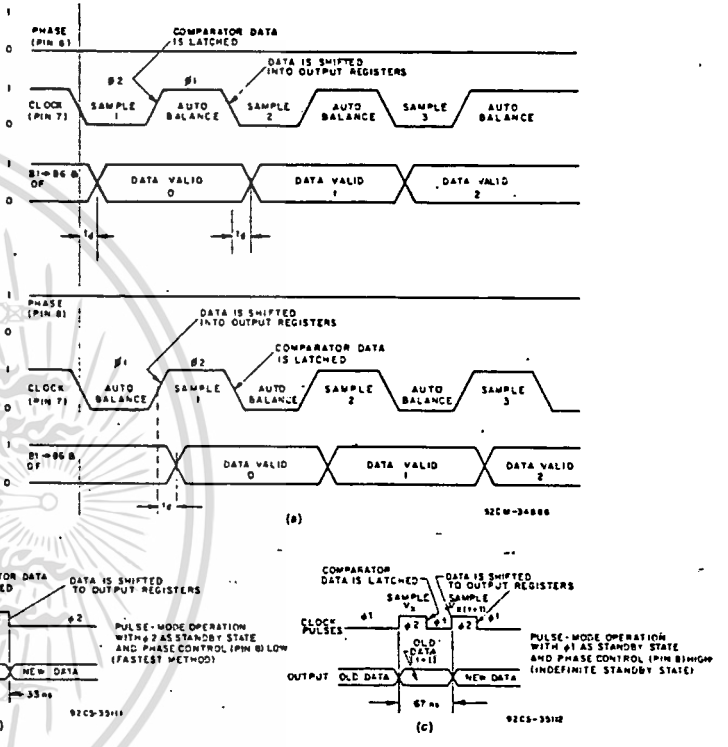


Fig. 6 - Timing diagrams for the CA3300.

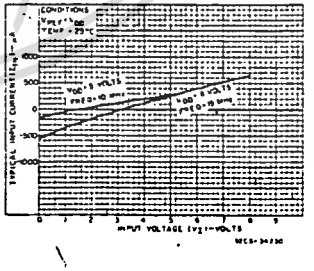


Fig. 7 - Typical input current versus input voltage as a function of supply voltage

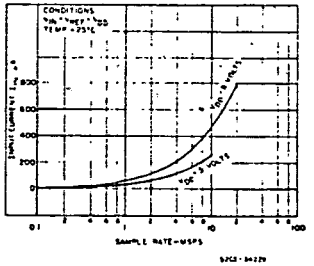


Fig. 8 - Typical input current versus sample rate as a function of supply voltage.

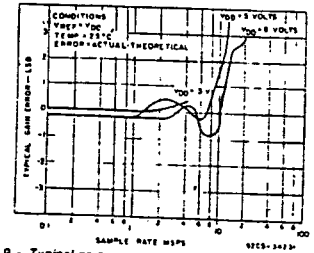


Fig. 9 - Typical gain error versus sample rate as a function of supply voltage. (See literature for gain trim.)

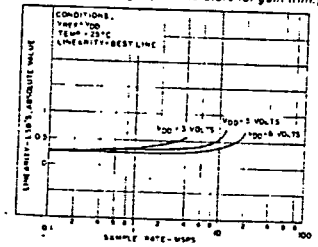


Fig. 10 - Typical linearity versus sample rate as a function of supply voltage

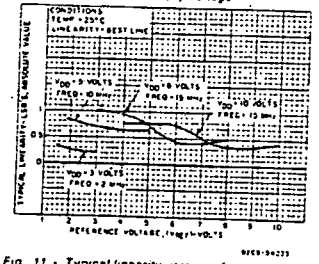


Fig. 11 - Typical linearity versus reference voltage as a function of supply voltage.

Device Operation

A sequential parallel technique is used by the CA3300 converter to obtain its high-speed operation. The sequence consists of the "Auto Balance" phase $\phi 1$ and the "Sample Unknown" phase $\phi 2$. (Refer to the circuit diagram.) Each conversion takes one clock cycle.* With the phase control (pin 8) low, the "Auto Balance" ($\phi 1$) occurs during the High period of the clock cycle, and the "Sample Unknown" ($\phi 2$) occurs during the low period of the clock cycle.

During the "Auto Balance" phase, a transmission switch is used to connect each of 64 commutating capacitors to their associated ladder reference tap. Those tap voltages will be as follows:

$$V_{tap}(N) = [(V_{REF}/64) \times N] - [V_{REF}/(2 \times 64)]$$
$$= V_{REF}[(2N - 1)/128]$$

Where: $V_{tap}(n)$ = reference ladder tap voltage at point n

V_{REF} = voltage across R^+ to R^-

N = tap number (1 through 64)

The other side of the capacitor is connected to a single stage amplifier whose output is shorted to its input by a switch. This biases the amplifier at its intrinsic trip point, which is approximately, $(V_{DD}-V_{SS})/2$. The capacitors now charge to their associated tap voltages, priming the circuit for the next phase.

In the "Sample Unknown" phase, all ladder tap switches are opened, the comparator amplifiers are no longer shorted, and V_{IN} is switched to all 64 capacitors. Since the other end of the capacitor is now looking into an effectively open circuit, any voltage that differs from the previous tap voltage will appear as a voltage shift at the comparator amplifiers. All comparators with tap voltages greater than V_{IN} will drive the comparator outputs to a "low" state, all comparators with tap voltage lower than V_{IN} will drive the comparator outputs to a "high" state.

The status of all these comparator amplifiers are stored at the end of this phase ($\phi 2$), by a secondary latching amplifier stage. Once latched, the status of the 64 comparators is decoded by a 64-to 7-bit decode array and the results are clocked into a storage register at the rising edge of the next $\phi 2$.

A 3-state buffer is used at the output of the 7 storage registers which are controlled by two chip-enable signals. $\overline{CE1}$ will independently disable B1 through B6 when it is in a high state. CE2 will independently disable B1 through B6 and the OE buffers when it is in the low state.

To facilitate usage of this device a phase-control input is provided which can effectively complement the clock as it enters the chip. Also, an on-board zener is provided for use as a reference voltage

Continuous Clock Operation

One complete conversion cycle can be traced through the CA3300 via the following steps. (Refer to timing diagram Fig. 6a.) With the phase control in a 'High' state, the rising edge of the clock input will start a "sample" phase. During this entire 'High' state of the clock, the 64 comparators will track the input voltage and the 64 latches will track the comparator outputs. At the falling edge of the clock, all 64 comparator outputs are captured by the 64 latches. This ends the "sample" phase and starts the "auto balance" phase for the comparators. During this 'Low' state of the clock the output of the latches propagates through the decode array and a 7-bit code appears at the D inputs of the output registers. On the next rising edge of the clock, this 7-bit code is shifted into the output registers and appears with time delay t_a as valid data at the output of the 3-state drivers. This also marks the start of a new "sample" phase, thereby repeating the conversion process for this next cycle.

Pulse Mode Operation

For sampling high-speed nonrecurrent or transient data, the converter may be operated in a pulse mode in one of two ways. The fastest method is to keep the converter in the Sample Unknown phase, $\phi 2$, during the standby state. The device can now be pulsed through the Auto Balance phase with as little as 33 ns. The analog value is captured on the leading edge of $\phi 1$ and is transferred into the output registers on the trailing edge of $\phi 1$. We are now back in the standby state, $\phi 2$, and another conversion can be started within 33 ns, but not later than 5 μs due to the eventual droop of the commutating capacitors. Another advantage of this method is that it has the potential of having the lowest power drain. The larger the time ratio between $\phi 2$ and $\phi 1$, the lower the power consumption. (See timing diagram Fig. 6b.)

The second method uses the Auto Balance phase, $\phi 1$, as the standby state. In this state the converter can stay indefinitely waiting to start a conversion. A conversion is performed by strobing the clock input with two $\phi 2$ pulses. The first pulse starts a Sample Unknown phase and captures the analog value in the comparator latches on the trailing edge. A second $\phi 2$ pulse is needed to transfer the data into the output registers. This occurs on the leading edge of the second pulse. The conversion now takes place in 67 ns, but the repetition rate may be as slow as desired. The disadvantage to this method is the higher device dissipation due to the low ratio of $\phi 2$ to $\phi 1$. (See timing diagram Fig. 6c.)

Increased Accuracy

In most cases the accuracy of the CA3300 should be sufficient without any adjustments. In applications where accuracy is of utmost importance, three adjustments can be made to obtain better accuracy; i.e., offset trim, gain trim, and midpoint trim.

Offset Trim

In general offset correction can be done in the preamp circuitry by introducing a dc shift to V_{IN} or by the offset trim of the op amp. When this is not possible the R^- (pin 10) input can be adjusted to produce an offset trim. The theoretical input voltage to produce the first transition is ½ LSB. The equation is as follows:

$$V_{IN}\ (0 \text{ to } 1 \text{ transition}) = \tfrac{1}{2}\ LSB = \tfrac{1}{2}(V_{REF}/64)$$
$$= V_{REF}/128$$

If V_{IN} for the first transition is less than the theoretical, then a single-turn 50-ohm pot connected between R^- and ground will accomplish the adjustment. Set V_{IN} to ½ LSB and trim the pot until the 0 to 1 transition occurs.

If V_{IN} for the first transition is greater than the theoretical, then the 50-ohm pot should be connected between R^+ and a negative voltage of about 2 LSB's. The trim procedure is as stated previously.

Gain Trim

In general the gain trim can also be done in the preamp circuitry by introducing a gain adjustment for the op amp. When this is not possible, then a gain adjustment circuit should be made to adjust the reference voltage. To perform this trim, V_{IN} should be set to the 63 to overflow transition. That voltage is ½ LSB less than V_{REF} and is calculated as follows:

$$V_{IN}\ (63 \text{ to } 64 \text{ transition}) = V_{REF} - V_{REF}/128$$
$$= V_{REF}\ (127/128)$$

To perform the gain trim, first do the offset trim and then apply the required V_{IN} for the 63 to overflow transition. Now adjust V_{REF} until that transition occurs on the outputs.

Midpoint Trim

The reference center (RC) pin 16, is available to the user as the approximate midpoint of the resistor ladder. The actual count that is brought out is count 33. To trim the midpoint, the offset and gain trims should be done first. The theoretical transition from count 32 to 33 occurs at 32½ LSB's. That voltage is as follows:

$$V_{IN}\ (32 \text{ to } 33 \text{ transition}) = 32.5\ (V_{REF}/64)$$

An adjustable voltage follower can be connected to the RC pin or a 2-K pot can be connected between R^+ and R^- with the wiper connected to RC. Set V_{IN} to the 32 to 33 transition voltage, then adjust the voltage follower or the pot until the transition occurs on the output bits.

The Reference Center point can also be used to create some unique transfer functions. For example, if R^- is grounded, RC is connected to 3.25 volts, and R^+ is connected to 4.8 volts then the lower order counts, 1 through 33, will have an LSB value of 100 mV while the upper order counts, 34 through Overflow, will have an LSB value of 50 mV. This effectively provides twice the sensitivity in the upper counts as compared to the lower counts.

7-Bit Resolution

To obtain 7-bit resolution, two CA3300's can be wired together. Necessary ingredients include an open-ended ladder network, an overflow indicator, three-state outputs, and chip-enable controls—all of which are available on the CA3300.

The first step for connecting a 7-bit circuit is to totem-pole the ladder networks, as illustrated in Fig. 13. Since the absolute resistance value of each ladder may vary, external trim of the mid-reference voltage may be required.

The overflow output of the lower device now becomes the seventh bit. When it goes high, all counts must come from the upper device. When it goes low, all counts must come from the lower device. This is done simply by connecting the lower overflow signal to the $\overline{CE1}$ control of the lower A/D converter and the CE2 control of the upper A/D converter. The three-state outputs of the two devices (bits 1 through 6) are now connected in parallel to complete the circuitry. The complete circuit for a 7-bit A/D converter is shown in Fig. 14.

8-Bit to 12-Bit Conversion Techniques

To obtain 8-to 12-bit resolution and accuracy, use a feed-forward conversion technique. Two A/D converters will be needed to convert up to 11 bits; three A/D converters to convert 12 bits. The high speed of the CA3300 allows 12-bit conversions in the 500 to 900-ns range.

The circuit diagram of a high-speed 12-bit A/D converter is shown in Fig. 15. In the feed-forward conversion method two sequential conversions are made. Converter A first does a coarse conversion to 6 bits. The output is applied to a 6-bit D/A converter whose accuracy level is good to 12 bits. The D/A converter output is then subtracted from the input voltage, multiplied by 32, and then converted by a second flash A/D converter, which is connected in a 7-bit configuration. The answers from the first and second conversions, are added together with bit 1 of the first conversion overlapping bit 7 of the second conversion.

When using this method, take care that:

- The linearity of the first converter is better than ½ LSB.
- An offset bias of 1 LSB (1/64) is subtracted from the first conversion since the second converter is unipolar.
- The D/A converter and its reference are accurate to the total number of bits desired for the final conversion (the A/D converter need only be accurate to 6 bits).

The first converter can be offset-biased by adding a 20-Ω resistor at the bottom of the ladder and increasing the reference voltage by 1 LSB. If a 6.40-voltage reference is used in the system, for example, then the first CA3300 will require a 6.5-V reference.

*This device requires only a single phase clock. The terminology of $\phi 1$ and $\phi 2$ refers to the High and Low periods of the same clock.

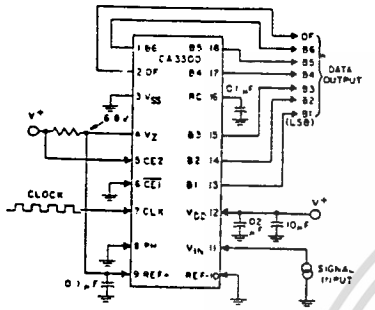


Fig. 12 - Typical CA3300 6-bit configuration 15-MHz sampling rate.

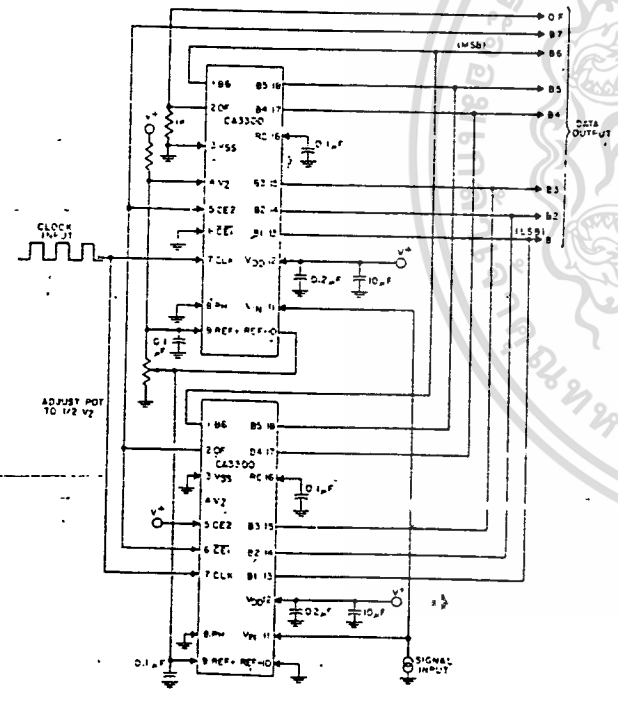


Fig. 13 - Typical CA3300 7-bit resolution configuration 15-MHz sampling rate

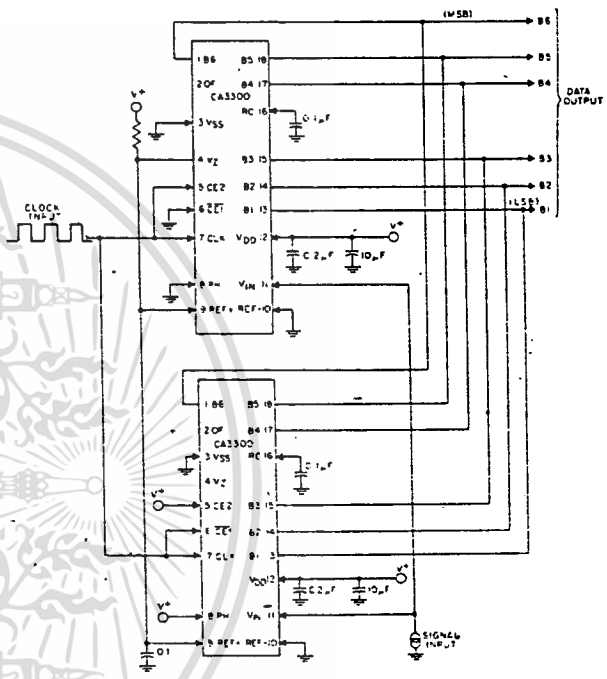


Fig. 14 - Typical CA3300 6-bit resolution configuration 30-MHz sampling rate.

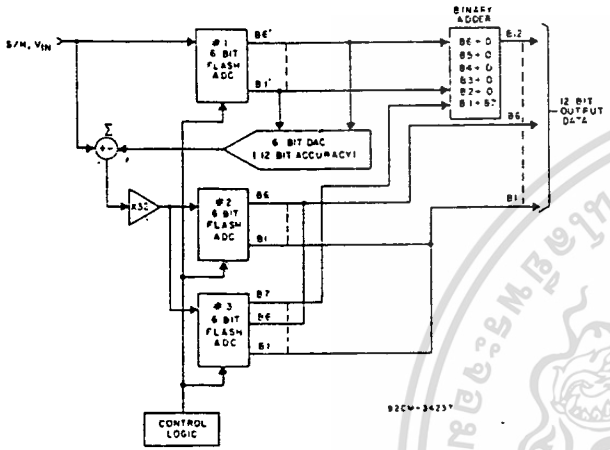


Fig. 15 - Typical CA3300 800-ns 12-bit ADC system.

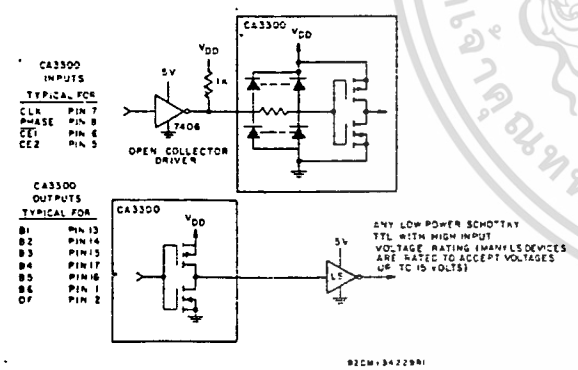
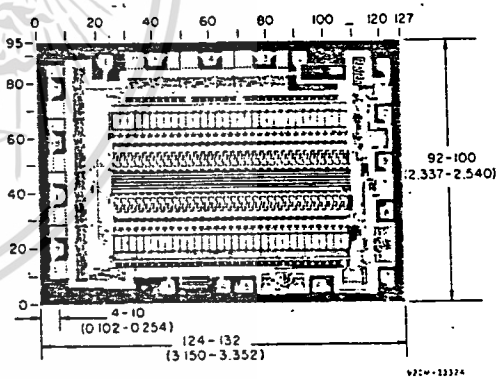


Fig. 16 - TTL interface circuit for $V_{DD} > 5.5$ volts

OUTPUT CODE TABLE

CODE DESCRIPTION	INPUT VOLTAGE* V_{REF} 7.68 (V)	V_{REF} 6.40 (V)	V_{REF} 5.12 (V)	V_{REF} 3.20 (V)	BINARY OUTPUT CODE (LSB) OF	B6	B5	B4	B3	B2	B1	DECIMAL COUNT
Zero	0.00	0.00	0.00	0.00	0	0	0	0	0	0	0	0
1 LSB	0.12	0.10	0.08	0.05	0	0	0	0	0	0	1	1
2 LSB	0.24	0.20	0.16	0.10	0	0	0	0	0	1	0	2
"	"	"	"	"				"				"
"	"	"	"	"				"				"
"	"	"	"	"				"				"
½ Full Scale — 1 LSB	3.72	3.10	2.48	1.55	0	0	1	1	1	1	1	31
½ Full Scale	3.84	3.20	2.56	1.60	0	1	0	0	0	0	0	32
½ Full Scale +1 LSB	3.96	3.30	2.64	1.65	0	1	0	0	0	0	1	33
"	"	"	"	"				"				"
"	"	"	"	"				"				"
"	"	"	"	"				"				"
Full Scale — 1 LSB	7.44	6.20	4.96	3.10	0	1	1	1	1	1	0	62
Full Scale	7.56	6.30	5.04	3.15	0	1	1	1	1	1	1	63
Overflow	7.68	6.40	5.12	3.20	1	1	1	1	1	1	1	127

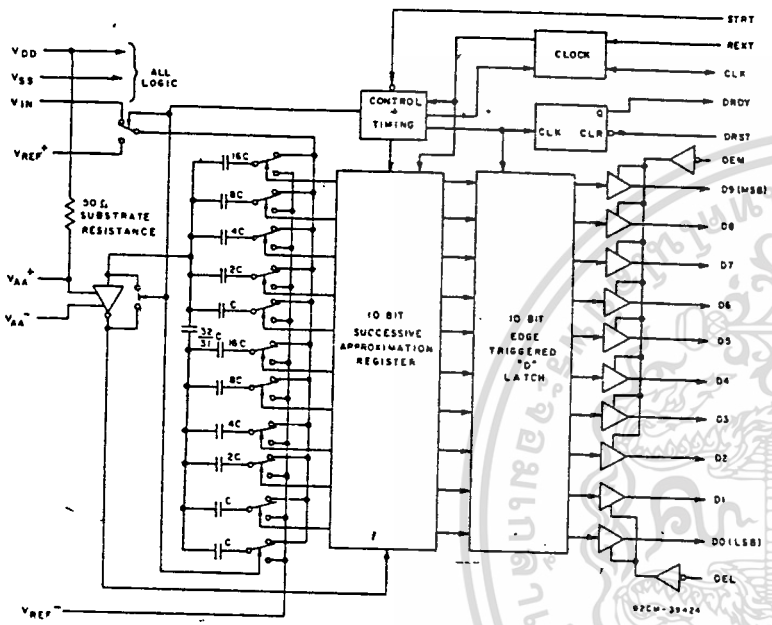
*The voltages listed below are the ideal centers of each output code shown as a function of its associated reference voltage



Dimensions and pad layout for CA3300H

Dimensions in parentheses are in millimeters and are derived from the basic inch dimensions as indicated. Grid graduations are in mils (10^{-3} inch)

The photographs and dimensions of each CMOS chip represent a chip when it is part of the wafer. When the wafer is cut into chips, the cleavage angles are 57° instead of 90° with respect to the face of the chip. Therefore the isolated chip is actually 7 mils (0.17 mm) larger in both dimensions.



Block Diagram of the CA3310

Product Preview

CMOS High-Speed 8-Bit Flash A/D Converter

Features:

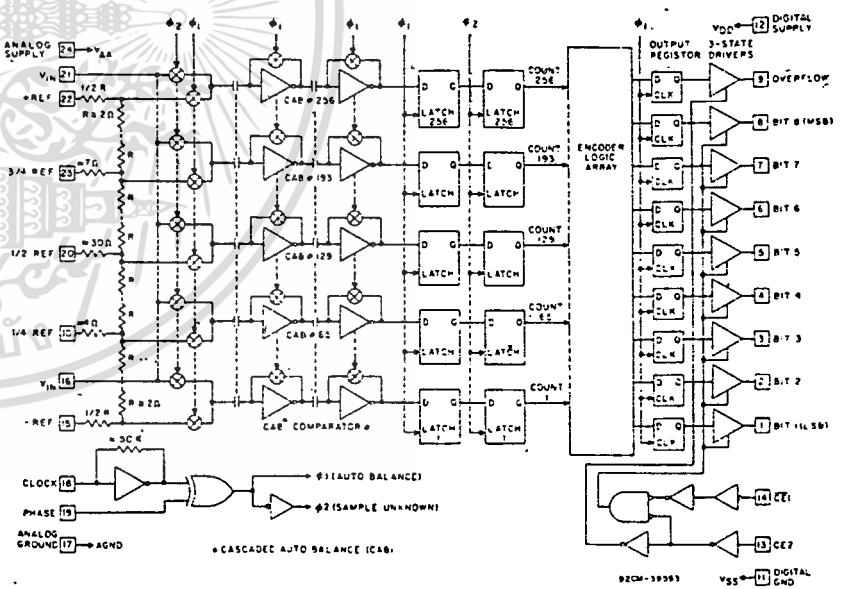
- *Pin compatible with 41051/CA3308*
- *CMOS/SOS low power*
- *Flash (Parallel) conversion technique*
- *15 MSPS conversion rate at 5 V (CA3318C)*
- *20 MSPS conversion rate at 5 V (CA3318)*
- *1 LSB differential linearity*
- *1.5 LSB integral linearity*
- *Single 4 to 6.5 V supply*
- *6 latched bit outputs plus overflow*
- *May be stacked for higher resolution*
- *May be paralleled for double speed*

Applications:

- *Especially suited for high-speed conversion applications where low power is also important*
- *TV video digitizing (industrial/security)*
- *Ultrasound signature analysis*
- *Transient signal analysis*
- *General-purpose hybrid ADC's*
- *Optical character recognition*
- *Radar pulse analysis*
- *Motion signature analysis*

The RCA CA3318 and CA3318C are pin compatible retrofits for the 41051/CA3308, but with the output data changing 1/2 clock cycle later. They have features similar to the CA3300 (File No. 1316), such as the control inputs and outputs necessary to allow stacking or paralleling for higher resolution or doubled speed. Separate analog and digital ground pins are available to allow analog to digital isolation. The reference resistor string is available at both + and - ends, and at the 1/4, 1/2, and 3/4 points, thus allowing the tailoring of non-linear transfer functions. In addition, the + reference (positive full scale) may be used above the analog + supply.

The CA3318 and CA3318C are available in a 24-lead dual-in-line plastic package (E suffix) and in a 24-lead dual-in-line ceramic package (D suffix).



Block diagram of the CA3318 and CA3318C

Preview Data only

CA3338

CMOS High-Speed R-2R D/A Converter

Product Preview

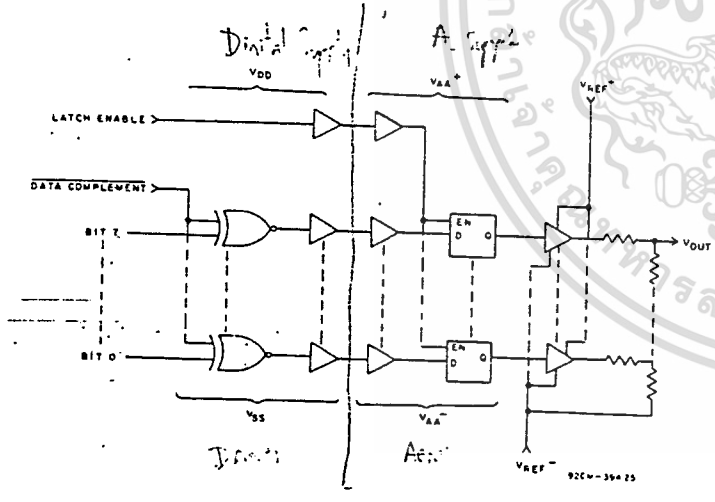
Features:

- *CMOS/SOS low power*
- *300 ohm R-2R ladder output*
- *20 MHz update rate*
- *1/2-LSB accuracy*
- *Single 3 to 8 V supply*
- *Input latch*
- *Data complement input*
- *Low glitch energy*
- *Level shifteres for bipolar output*

The RCA CA3338 is a CMOS 8-bit R-2R output D/A converter. It has separate analog and digital supply pins for isolation, and, due to level shifters on the data input lines, may operate with the output below digital ground. A data complement control inverts the output waveform, and a latch control allows the D/A to follow or hold the input data.

Careful attention has been paid to reducing output "glitch" energy and allowing clean video signals.

The CA3338 is available in a 16-lead dual-in-line plastic package (E suffix) and in a 16-lead dual-in-line ceramic package D suffix).



Block Diagram of the CA3338

Preview Data only

Product Preview

CA3999

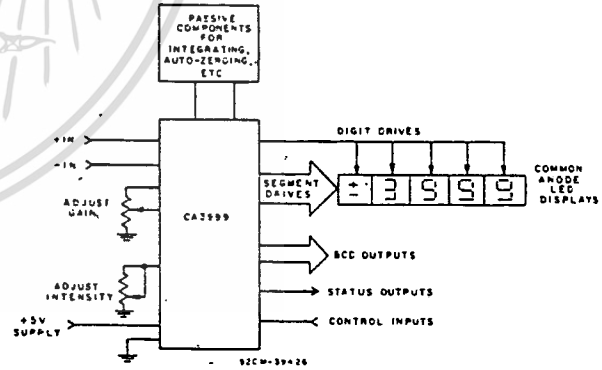
CMOS Complete 3-3/4 Digit DPM

Features:

- *Full ± 3999 count range*
- *True differential inputs*
- *Single +5 V supply*
- *Direct multiplexed LED drive*
- *Accurate on-chip reference*
- *BCD outputs available*
- *Good 50/60 cycle rejection*
- *LED brightness control*
- *Under-and over-voltage outputs*

The RCA CA3999 is a true differential input ± 3999 count DPM with direct LED drive capability. It operates from a single + 5 V source and generates a negative supply (needed for negative input signals) on chip. A stable on-chip referenced is auto-zeroed for low-drift, as is the input integrator and comparator. The dual-slope converter integrates for an integral number of 50 or 60 cycles, thus providing good AC rejection. BCD data is output as well as multiplexed 7 segment LED drive. No external components are needed to interface to LEDs, the addition of an external pot will control brightness, however.

The CA3999 is available in a 40-lead dual-in-line plastic package (E suffix).



Block Diagram of the CA3999

Preview Data only