ปีการศึกษา 2532

เรื่อง บัฟเฟอร์ของการส่งแบบอนุกรมและ

อุปกรณ์พัฒนาโปรแกรมของ

ไมโครโปรเซสเซอร์ 8031

SERIAL BUFFER AND TWODIRECTION

RAM FOR MICROPROCESSOR 8031

โดย

นาย สมชาย วงศ์รัศมี 29.1279

อาจารย์ที่ปรึกษา

รศ.ดร.สิทธิชัย โภไคยอุดม

ปริญญานิพนธ์ ปีการศึกษา 2532

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง บัฟเฟอร์ของการรับส่งแบบอนุกรมและอุปกรณ์พัฒนาโปรแกรมของ
ไมโครโปรเซสเซอร์ 8031

SERIAL BUFFER AND TWO DIRECTION RAM FOR
MICROPROCESSOR 8031

ผู้จัดทำ นาย สมชาย วงศ์รัศมี เลขประจำตัว 29.1279

..... อาจารย์ที่ปรึกษา

รศ.ดร. สิทธิชัย โภไคยอุดม

สารบัญ

บทคัดย่อ

บัฟเฟอร์ของการส่งแบบอนุกรมและอุปกรณ์พัฒนาโปรแกรม
ของไมโครโปรเซสเซอร์ 8031

SERIAL BUFFER AND TWO DIRECTION RAM FOR
MICROPROCESSOR 8031

สมชาย วงศ์รัศมี 29.1279

อาจารย์ที่ปรึกษา: ดร.สิทธิชัย โภไคยอุดม

ปีการศึกษา 2532

ในปัจจุบันการติดต่อสื่อสารได้ถูกพัฒนาขึ้นมาก โดยเฉพาะการติดต่อสื่อสารผ่านทางสายโทรศัพท์ ซึ่งเป็นการติดต่อสื่อสารแบบอนุกรม(SERIAL) คือการส่งสัญญาณผ่านสายไปทีละบิต(BIT) ปัจจุบันมีอุปกรณ์ที่สามารถรับหรือส่งแบบอนุกรมได้หลายชิ้น เช่น - โมเด็ม(MODEM), แฟ็กซีมิลี(FAXIMILE) จึงได้มีการศึกษาเกี่ยวกับการส่งทางอนุกรมโดยการสร้างอุปกรณ์ บัฟเฟอร์ (BUFFER) คืออุปกรณ์ที่สามารถทำการเก็บข้อมูลที่ส่งเข้ามาทางอนุกรมโดยการนำเข้าไปเก็บไว้ในหน่วยความจำ ประโยชน์ของอุปกรณ์บัฟเฟอร์ของการส่งทางอนุกรมคือเพื่อช่วยในการรับหรือส่งแบบอนุกรมโดยที่ไม่จำเป็นที่จะต้อง เปิดเครื่องคอมพิวเตอร์เอาไว้ในขณะที่ต้องการที่จะรอข้อมูลและยังสามารถที่จะนำเอาคอมพิวเตอร์ไปใช้งานอื่นได้ในขณะที่มีข้อมูลส่งเข้ามาพอดี อุปกรณ์อีกตัว คือ เครื่องช่วยพัฒนาการเขียนโปรแกรมของไมโครโปรเซสเซอร์ เบอร์ 8031 มีประโยชน์ช่วยในการเขียนโปรแกรม และลดเวลาในการทดลอง เพื่อที่จะสร้างอุปกรณ์ที่ต้องใช้ 8031 โดยจะไม่จำเป็นที่จะต้องล้างและอัดอีพรอมท์ (EPROM) เนื่องจากว่าการเขียนโปรแกรมในการทดลองจำเป็นที่จะต้องมีการเปลี่ยนแปลงอยู่ตลอดเวลาจนกว่าจะได้โปรแกรมที่สมบูรณ์ ดังนั้นการสร้างอุปกรณ์พัฒนาโปรแกรมนี้จึงมีประโยชน์เป็นอย่างมากส่วนประกอบของอุปกรณ์ตัวนี้เป็น คาร์ด (CARD) ที่สามารถเสียบไว้บนเครื่องคอมพิวเตอร์และอาศัยโปรแกรมช่วยในการส่งผ่านจาก หน่วยความจำของคอมพิวเตอร์เพื่อที่จะไปเก็บไว้ในหน่วยความจำของอุปกรณ์พัฒนาโปรแกรม

บทที่ 1

บทนำ

เนื่องจากว่าในปัจจุบันการสื่อสารที่ผ่านทางสายโทรศัพท์นั้น กำลังเป็นที่นิยมกันอย่างมาก จึงทำให้ข้าพเจ้ามีความคิดในการที่จะสร้างอุปกรณ์ขึ้นมาชิ้นหนึ่งเพื่อที่จะทำการช่วยในการรับหรือส่งข้อมูลกันตามสายโทรศัพท์ ซึ่งการส่งข้อมูลผ่านทางสายโทรศัพท์นั้นก็คือการส่งข้อมูลแบบอนุกรม และการส่งข้อมูลที่เป็นที่นิยมกันวิธีหนึ่งก็คือการส่งข้อมูลผ่านทางโมเด็ม โมเด็มในแบบที่ต้องวางเอาไว้นอกเครื่องคอมพิวเตอร์ (STAND ALONE) นั้น จำเป็นที่จะต้องเปิดเครื่องคอมพิวเตอร์ทิ้งเอาไว้ หรือว่าต้องมีบุคคลใดคนหนึ่งเฝ้าดูว่า ทำให้เกิดความคิดว่า ถ้าเกิดเราสามารถสร้างอุปกรณ์ที่สามารถมาตอบรับแทนเครื่องคอมพิวเตอร์ขึ้นมาได้ก็จะเป็นประโยชน์มาก ส่วนอุปกรณ์ที่จะสามารถทำหน้าที่ติดต่อตอบรับแทนเครื่องคอมพิวเตอร์นั้น ก็ควรจะเป็นไมโครโปรเซสเซอร์ซึ่งเราเห็นว่าควรจะใช้หน่วยควบคุมไมโครโปรเซสเซอร์ แบบ MCS-51 เบอร์ 8031 เนื่องจากว่าได้มีพอร์ทอนุกรมอยู่ในตัวอยู่แล้วจะทำให้การทำงานสะดวกขึ้นและไม่สับสน

ในขณะที่ได้ดำเนินการออกแบบและคิดวงจรนั้น เราได้ทำการทดลองใช้อุปกรณ์อีพรอมท์ (EPROM) ซึ่งเป็นหน่วยความจำแบบสามารถที่จะล้างและโปรแกรมได้นั้นและเป็นอุปกรณ์ที่จำเป็นที่จะต้องใช้ร่วมกันกับ 8031 ด้วย ก็ได้เกิดความคิดขึ้นมาว่า ถ้าเกิดสร้างอุปกรณ์ช่วนในการที่จะไม่ต้องให้ต้องทำการล้างและอัดอีพรอมท์บ่อยๆ ด้วยก็จะช่วยให้การทำงานสะดวกและรวดเร็วขึ้น จึงทำการสร้างอุปกรณ์พัฒนาโปรแกรมขึ้น และเนื่องจากว่าโปรแกรมที่ทำการทำการตรวจและถอดรหัส (COMPLIER)ของไมโครโปร-เซสเซอร์เบอร์ 8031 นี้ได้มีอยู่แล้ว เราจึงเขียนโปรแกรมส่งรหัสที่ได้แปลงออกมาแล้ว นั้นผ่านเข้ามาอยู่บนตัวอุปกรณ์และต่อสายออกมาเสียบแทนที่ตัวอีพรอมท์ได้ และนำสัญญาณที่จะทำการอ่านหน่วยความจำของโปรแกรมของ 8031 มาต่อสายผ่านเข้ามาบนอุปกรณ์

พัฒนาโปรแกรมนั้นด้วย โดยจะนำสัญญาณนี้มาถอดรหัสเพื่อทำการควบคุมการอ่านหน่วยความจำได้ด้วยอุปกรณ์ตัวนี้เพียงแต่เราเขียนโปรแกรมของ 8031 แล้วนำไปผ่านการตรวจและถอดรหัสแล้ว ก็สามารถที่จะส่งโปรแกรมเข้ามาอยู่ในหน่วยความจำที่ 8031 สามารถเข้ามาอ่านได้โดยตรงโดยไม่ต้องอาศัยอีพรอมท์

บทที่2

ทฤษฎี

บัฟเฟอร์ของการส่งแบบอนุกรม

ลักษณะของการส่งแบบอนุกรมทางสายโทรศัพท์ที่เราให้ความสนใจ คือ โมเด็ม (MODEM) ในการที่เราจะสร้างบัฟเฟอร์ของโมเด็มนั้นจำเป็นที่จะต้องมีอุปกรณ์ที่ใช้ควบคุม คือ ไมโครโปรเซสเซอร์ (MICROPROCESSOR) ในที่นี้เราเลือกใช้ไมโครโปรเซสเซอร์ ตระกูล MCS-51 เป็นแบบชิงเกิ้ลชิพ (SINGLE CHIP) เบอร์ 8031 เนื่องด้วยเห็นว่ามีคุณสมบัติพิเศษหลายอย่างที่เหมาะสม เช่น มีพอร์ทอนุกรม (SERIAL PORT)ในตัวเอง และมีวงจรนับเวลาและตั้งเวลาในตัวด้วย เนื่องด้วยเราต้องการที่จะติดต่อกันกับโมเด็มโดยตรงโดยที่ไม่ต้องอาศัยคอมพิวเตอร์ (COMPUTER) ดังนั้นเราจึงจำเป็นที่จะต้องใช้ไมโครโปรเซสเซอร์ 8031 ติดต่อโดยตรงกับโมเด็มเลย

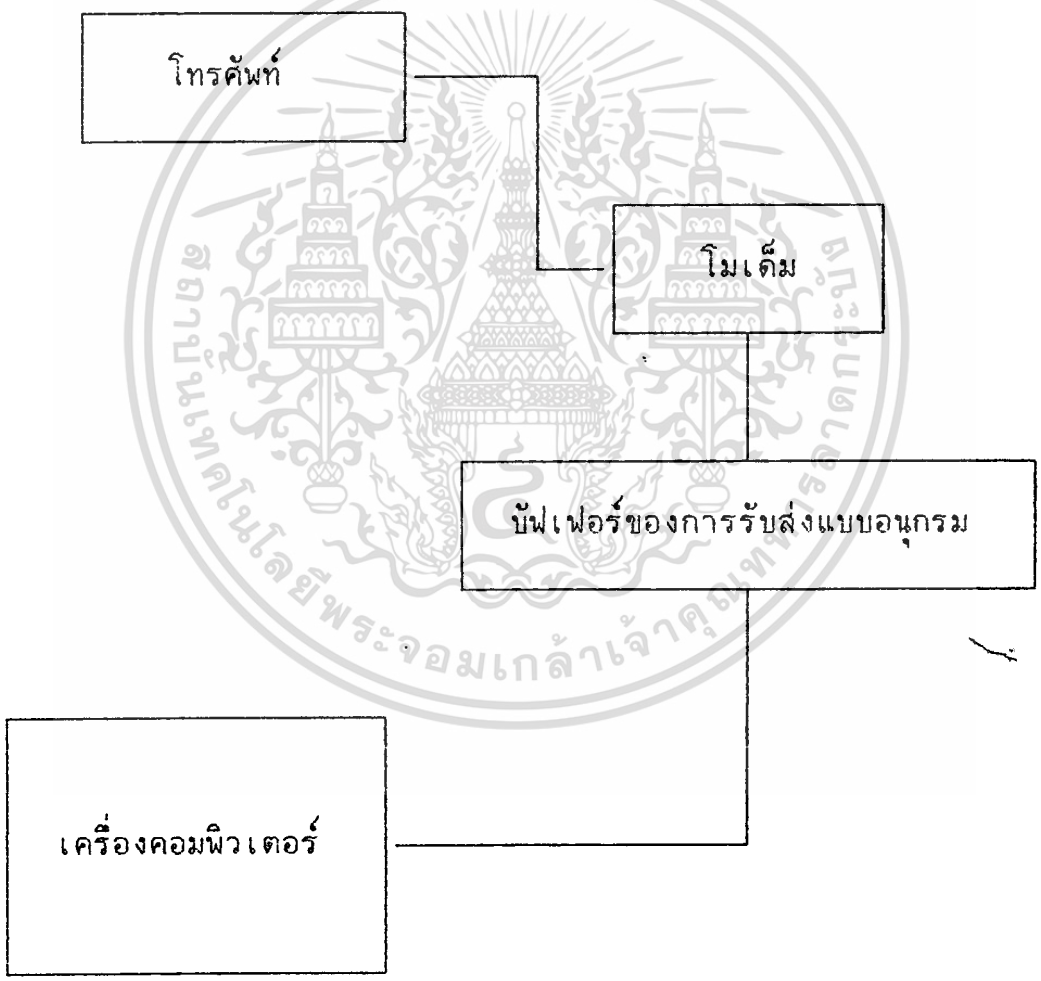
หน้าที่หลักของ 8031 คือ

1. ทำการติดต่อกับโมเด็มในขณะที่จะรับหรือส่งข้อมูลแทนคอมพิวเตอร์
2. นำเอาข้อมูลที่ได้รับจากโมเด็มหรือคอมพิวเตอร์เข้าไปเก็บไว้ในหน่วยความจำ
3. ทำการติดต่อกับคอมพิวเตอร์เพื่อส่งหรือรับข้อมูลที่อยู่ในกน่วยความจำ

การที่เราใช้ไมโครโปรเซสเซอร์ 8031 ทำให้เราจำเป็นที่จะต้องมีฐานเวลาภายนอก (CLOCK) โดยสร้างจากผลึกคริสตรอล (CRYSTAL) ขนาด 18.342 เมกกะเฮิรตซ์ (MHz) เพื่อทำการสร้างความถี่ออสซิเลเตอร์ (OSCILLATOR FREQUENCY) ให้กับชิพตยเข้าที่ขา 18 (X2) จะมีผลต่อการกำหนดอัตราการรับส่งแบบอนุกรม (BAUD RATE) ในที่นี้เราเลือกใช้การรับส่งอนุกรมในโหมดหนึ่ง เพื่อที่จะได้กำหนดอัตราการรับส่งได้ และที่ขารับและส่ง

ของตัวชิพ 8031 จำเป็นต้องมีบัฟเฟอร์ของการรับและไดรเวอร์ (DRIVER) ของการส่ง เนื่องจากว่าการส่งข้อมูลไปทางสายโทรศัพท์อาจจะมีระยะทางไกลมากก็ได้จึงต้องเพิ่มแรงดันในการส่งให้สูงขึ้น

ลักษณะการต่อบัฟเฟอร์ของการส่งแบบอนุกรมกับโมเด็ม, โทรศัพท์ และการต่อกับคอมพิวเตอร์มีลักษณะตามรูปที่ 2.1



รูปที่ 2.1 ลักษณะของการต่ออุปกรณ์บัฟเฟอร์ของการส่งแบบอนุกรม

อุปกรณ์พัฒนาโปรแกรมของไมโครโปรเซสเซอร์เบอร์ 8031

ในการทดลอง หรือการสร้างอุปกรณ์แต่ละชิ้นที่จำเป็นที่จะต้องอาศัยไมโครโปรเซสเซอร์ควบคุมนั้น ผู้สร้างจำเป็นที่จะต้องเขียนโปรแกรมเพื่อที่จะใช้เป็นโปรแกรมของตัวไมโครโปรเซสเซอร์ (PROGRAMMEMORY) โดยเขียนแล้วอัดเข้าไปเก็บไว้ในหน่วยความจำแบบอีพรอม (EPROM) การที่จะอัดอีพรอมท์นั้นทำให้ต้องเสียเวลาในการล้างและอัดอีพรอมท์ ซึ่งถ้าเป็นโปรแกรมที่จะต้องทดลองสร้างบางครั้งอาจจะต้องทำการล้างและอัดอีพรอมบ่อยมาก เราจึงได้คิดสร้างคาร์ด (CARD) ที่มีหน่วยความจำแบบแรม (RAM) เพื่อที่จะให้ไมโครโปรเซสเซอร์ได้เข้ามาอ่านได้ โดยเราได้สร้างคาร์ดนี้สำหรับไมโครโปรเซสเซอร์ ตระกูล MCS-51 เบอร์ 8031 โดยเฉพาะและคาร์ดนี้สามารถใช้ได้โดยการนำเอาไปเสียบไว้ในช่องเสียบอุปกรณ์เพิ่มเติมของคอมพิวเตอร์ของไอบีเอ็ม รุ่น พีซีเอ็กที (IBM PC/XT) และหลังคาร์ดนี้ได้มีการต่อสายออกมา 25 เส้นโดยผ่านช่องต่อสายแบบ 25 ช่อง (DB-25) เพื่อที่จะได้นำไปต่อกับอุปกรณ์ภายนอกที่จำเป็นจะต้องพัฒนาโปรแกรม โดยอุปกรณ์ภายนอกนั้นจะต้องสาย 25 เส้นนั้นเหมือนกับที่จะต่อกับอีพรอมท์

หลักการทำงานของอุปกรณ์พัฒนาโปรแกรม

เริ่มต้นด้วยการที่เรานำเอาขาแอดเดรส (ADDRESS) A0-A9 ในช่องสล๊อต (SLOT) ของไอบีเอ็ม มาทำการดีโค็ด (DECODE) โดยเลือกเอาแอดเดรสที่อนุญาติให้ผู้เขียนโปรแกรมใช้ได้ โดยให้ผลที่ได้จากการดีโค็ดไปควบคุมการทำงานของอุปกรณ์ควบคุมพอร์ทแบบขนาน (PARARELL PORT) เพื่อที่จะบอกว่าข้อมูลที่ถูกส่งมาทางพอร์ทข้อมูลของ ไอบีเอ็มนั้นเป็นอะไรบ้าง ซึ่งก็จะมีอยู่เพียงสามอย่าง คือ

1. แอดเดรสไบท์สูง (HIGH ADDRESS)

2. แอดเดรสไบท์ต่ำ (LOW ADDRESS)

3. ข้อมูล

ต่อจากนั้นเราจะต้องต่อสายข้อมูลและแอดเดรสเข้ากับอุปกรณ์ที่เป็นตัวเลือกว่าข้อมูลกรือแอดเดรสนั้นมาจากไอบีเอ็มหรือมาจากไมโครโปรเซสเซอร์ภายนอก โดยใช้การนำเอาสัญญาณอ่าน (IOR) และ สัญญาณเขียน (IOW) จากเครื่องไอบีเอ็ม และนำเอาสัญญาณอ่านโปรแกรมภายนอกของ 8031(PSEN) และสัญญาณที่ได้จากการดีโค็ดทั้งหมดนี้นำเอามาดีโค็ดอีกครั้ง เพื่อที่จะมาควบคุมทิศทางของข้อมูลและแอดเดรส

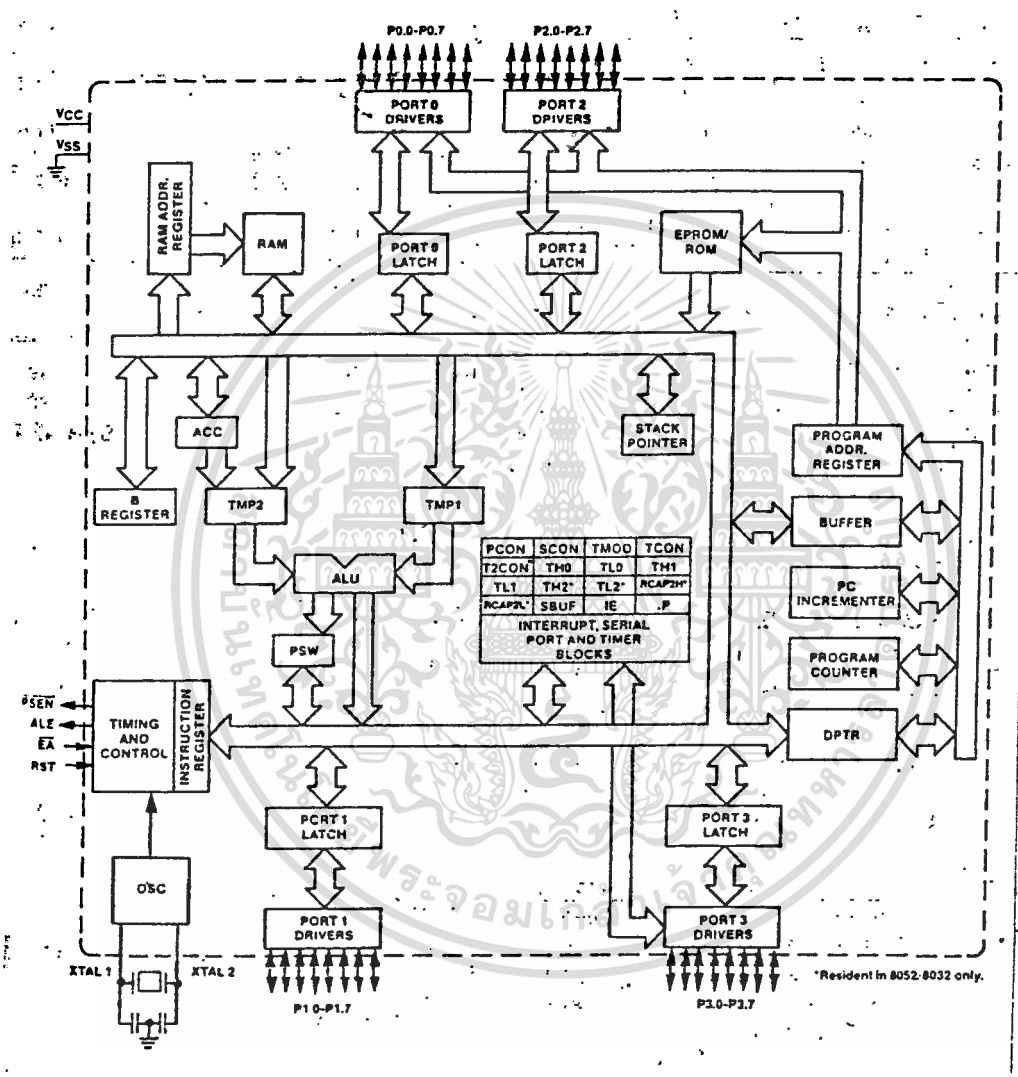
ลักษณะการทำงานภายในของไมโครโปรเซสเซอร์ตระกูล MCS- 51เบอร์ 8031

จากสองหัวข้อที่ผ่านมานั้นสังเกตุได้ว่าเกี่ยวข้องกับการทำงานของ-ไมโครโปรเซสเซอร์เบอร์8031 ทั้งสองหัวข้อจึงควรมีการอธิบายถึงการทำงานภายในของไมโครโปรเซสเซอร์ เบอร์ 8031 ดังนี้

การทำงานภายใน mcs-51

ไมโครโปรเซสเซอร์ในตระกูล MCS-51 ซึ่งเป็นไมโครโปรเซสเซอร์ขนาด 8 บิท ประกอบด้วยโมโครโปรเซสเซอร์เบอร์ต่างๆ ดังตารางที่ 1 ทุกๆ เบอร์จะมี สถาปัตยกรรมพื้นฐานเหมือนกับรูป 2.1 แต่เดิม 8051 ถูกสร้างด้วยวิธี HMOSI แต่ในปัจจุบันได้สร้างด้วยวิธี HMOS II จึงมีชื่อเป็น 8051 AH ไมโครโปรเซสเซอร์ในตระกูล 51 นั้นถึงแม้ ว่าจะมีหลายเบอร์ แต่เราก็จะเรียกว่าเป็น "8Q51" ซึ่งเราจะใช้ชื่อ 8051 นี้ โดยตลอดเมื่อหมายถึงไมโครโปรเซสเซอร์ ตระกูล 51 ส่วนเบอร์ 8032 และ 8052 มีหน่วยความจำภายในเพิ่มขึ้น และมีวงจรนับ, ตั้งเวลาขนาด 16 บิทเพิ่มขึ้น โดยวงจรนับ, ตั้งเวลาสามารถใช้เป็นวงจรนับ , วงจรตั้งเวลา และเป็นตัวกำหนดอัตราการส่งข้อมูล

ทางอนุกรม (Serial port) ข้อมูลของขาแต่ละขาของไมโครโปรเซสเซอร์อยู่ในภาคผนวกตอนท้าย



รูปที่ 2.2 แสดงถึงโครงสร้างของ MCS-51

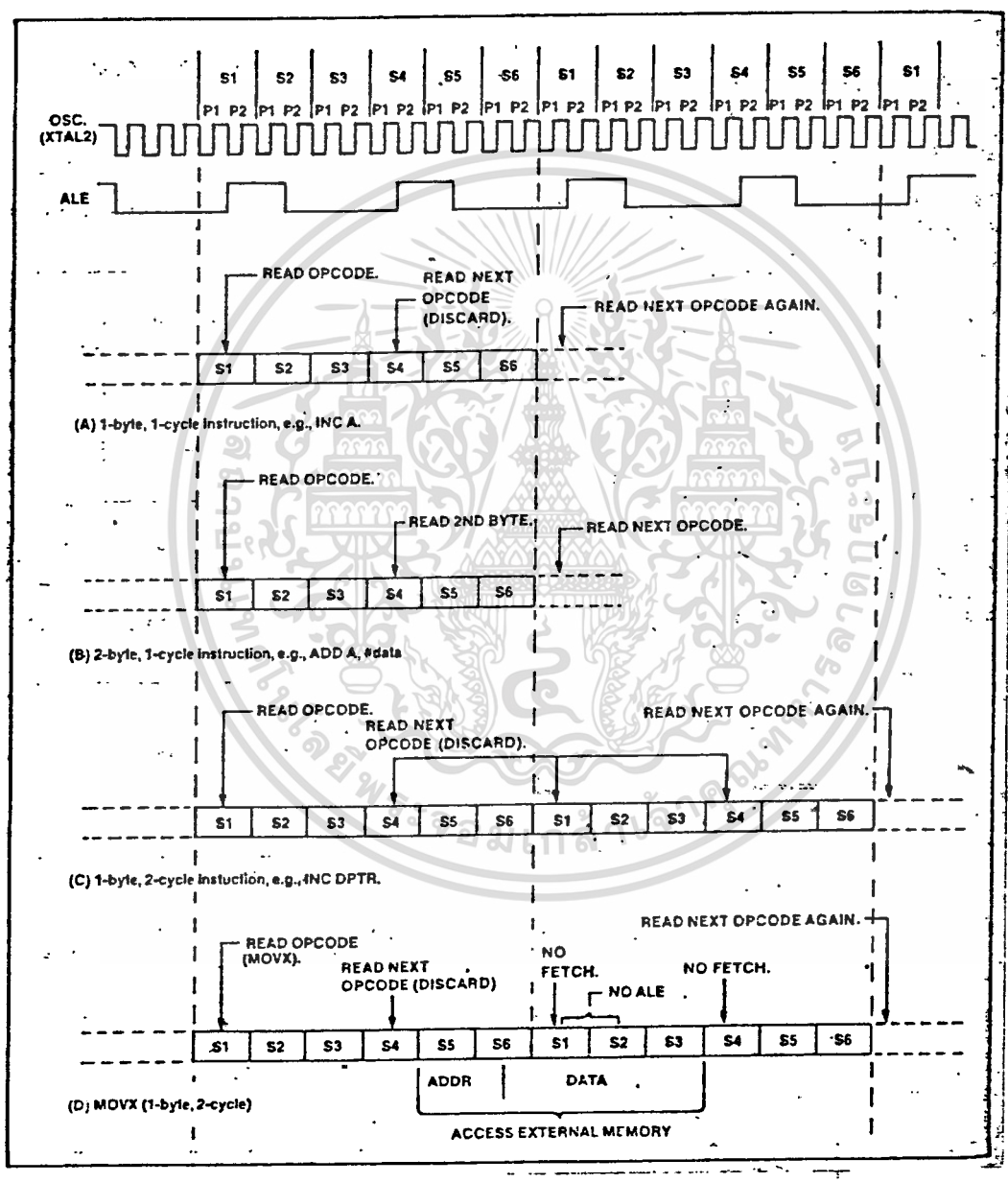
PART	TECHNIOLOGY	ON-CHIP PROGRAM MEMORY	ON-CHIP DATA MEMORY
8051AH	HMOS II	4K-ROM	128
8031AH	HMOS II	NONE	128
8751H	HMOS I	4K-EPROM	128
80C51	CHMOS	4K-ROM	128
80C31	CHMOS	NONE	128
8052	HMOS II	8K-ROM	256
8032	HMOS II	NONE	256

ตารางที่ 1

คุณสมบัติสำคัญของตระกูล 51 นี้คือ

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

- ส่งข้อมูลอนุกรมแบบ 2 ทิศทางได้
- สามารถประมวลผลตรรกศาสตร์
- มี RAM ภายในขนาด 128 ไบท์ สามารถอ้างได้ทั้งแบบบิทและแบบเป็นไบท์



รูปที่ 2.3 ภาพแสดง TIMIMG ของ MCS-51

การติดต่อกับหน่วยความจำภายนอกมี 2 แบบ ซึ่งมีการทำงานดังรูป 2.3 คือ ติดต่อกับหน่วยความจำสำหรับโปรแกรมและหน่วยความจำสำหรับข้อมูลการติดต่อกับหน่วยความจำสำหรับโปรแกรมภายนอกจะใช้สัญญาณ PSEN (Program store enable)เป็นสัญญาณสโตรป (strobe) สัญญาณ RD และ WR จะใช้เป็นสัญญาณสโตรป (strobe)สำหรับการติดต่อกับหน่วยความจำสำหรับข้อมูล การอ่านโปรแกรมจากหน่วยความจำภายนอกจะใช้แอดเดรส 16บิทเสมอ ส่วนการติดต่อกับหน่วยความจำสำหรับข้อมูลจะใช้แอดเดรสได้ทั้งแบบ 16 บิท (เช่นคำสั่ง MOVX @DPTR) หรือใช้แอดเดรส 8 บิท (เช่นคำสั่ง MOVX @Ri) เมื่อการติดต่อกับหน่วยความจำเป็นแบบ 16 บิท ไบท์ส่งของแอดเดรสจะมาจากพอร์ท 2 ซึ่งจะส่งค่าแอดเดรสออกมา ขณะทำการอ่านและเขียน ในระหว่างนี้ พอร์ท 2 LATCH (ซึ่งเป็นหนึ่งใน Special Function Register, SFR) จะไม่มีการเปลี่ยนแปลงข้อมูล เมื่อหมดคำสั่งการติดต่อกับหน่วยความจำภายนอก ค่าของ SFR พอร์ท 2 ก็จะปรากฏที่พอร์ท 2 ต่อไป ถ้าเป็นคำสั่งการอ้างอิงตำแหน่งหน่วยความจำ 2 จะไม่เปลี่ยนแปลง ในขณะที่มีการติดต่อกับหน่วยความจำภายนอกทั้งการติดต่อแบบ 8 บิท และ 16 บิท ไบท์ต่ำของแอดเดรสจะเป็นแบบมัลติเพลกซ์กับข้อมูลของพอร์ท 0 สัญญาณ ADDR/DATA จะทำให้ FET ของเอาท์พุทพอร์ท 0 ทำงาน ดังนั้นในการใช้งานพอร์ท 0 ลักษณะนี้จำไม่ต่อ PULL UP ภายนอก วงจร LATCH ภายนอกจะเก็บค่าของแอดเดรสไบท์ต่ำ โดยใช้สัญญาณ ALE เป็นตัวกำหนด โดยค่าของแอดเดรสไบท์ต่ำ จะมีค่าคงที่ขณะที่สัญญาณ ALE เปลี่ยนจาก 1 เป็น 0 ดังนั้นในการเขียนข้อมูลไปยังหน่วยความจำนั้น ข้อมูลจะถูกส่งออกไปทางพอร์ท 0 ก่อนจะมีสัญญาณ WR เป็น 0 และจะคงค่านั้นไว้จนกว่าสัญญาณ WR จะกลับเป็น 1 ในการอ่านข้อมูลจากหน่วยความจำภายนอกนั้น ข้อมูลจะต้องปรากฏที่พอร์ท 0 ก่อนสัญญาณ RD จะเป็น 0 เสมอ

ระหว่างการติดต่อกับหน่วยความจำภายนอกนั้น CPU จะส่งข้อมูล 0FFH ไปยังพอร์ท 0 LATCH ทำให้ข้อมูลที่พอร์ท 0 เปลี่ยนไป

การอ่านโปรแกรมจากหน่วยความจำภายนอกจะเกิดได้ใน 2 กรณี คือ

1. EA เป็น 0

2. เมื่อ EA เป็น 1 จะอ่านโปรแกรมจากหน่วยความจำภายนอกเมื่อโปรแกรมเคาร์เตอร์ (PC) มีค่ามากกว่า 0FFFH หรือ 1FFFH สำหรับ 8052

ดังนั้น ในรุ่นที่ไม่มีรอมภายในจึงต้องต่อ EA ไว้ที่ 0 เพื่อให้มีการอ่านโปรแกรมจากหน่วยความจำภายนอกเสมอ

<u>ลักษณะของพอร์ทอนุกรม (SERIAL INTERFACE)</u>

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

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

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

ส่งสามารถโปรแกรมได้

โหมด 2 - การส่งและรับข้อมูลเป็นแบบ 11 บิท คือ 1 start บิท (0) 8 data บิท (บิท 0-7) บิทที่ 9 สามารถกำหนดได้ และ 1 stop บิท (1). ในการส่งบิทที่ 9 (TB 8 ใน SCON) สามารถกำหนดค่าให้เป็น 1 หรือ 0 ก็ได้ เช่น เอาพาริตี้บิท (Parity, P ใน PSW) ย้ายเข้าไปที่ TB8 หรือในตอนรับบิทที่ 9 จะถูกส่งเข้าไปใน RB8 ของ SCON ขณะที่ stop บิทจะไม่ถูกเก็บไว้ อัตราการส่งสามารถโปรแกรมเป็น 1/32 หรือ 1/64 ของความถี่ออสซิเลเตอร์

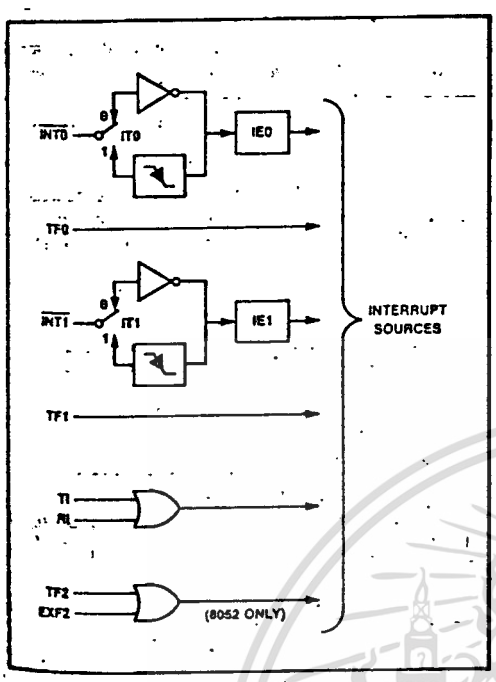
โหมด 3 การทำงานของโหมดนี้จะเหมือนกับโหมด 2 ยกเว้นเราสามารถโปรแกรมอัตราการส่งได้

ในทุกโหมด การส่งข้อมูลจะเริ่มจากคำสั่งใดก็ได้ที่มีการส่งข้อมูลไปยัง SBUF ในการรับข้อมูลจะต่างกันเล็กน้อย คือในโหมด 0 การรับข้อมูลจะเริ่มโดย RI = 0 และ REN = 1 ส่วนในโหมดอื่นๆ จะเริ่มการรับข้อมูลเข้ามาโดย REN = 1 เมื่อมี start บิทเข้ามา

อินเทอรัพท์ (INTRRUPT)

8051 สามารถอินเทอรัพท์ได้ด้วย 5 วิธี ดังรูป 2.4

สัญญานอินเทอรัพท์จากภายนอก INT0 และ INT1 จะสามารถทำงานได้ทั้งแบบระดับ (LEVEL) และแบบการเปลี่ยนระดับ (TRANSITION) ขึ้นกับ IT0 และ IT1 ใน TCON แฟลกที่ควบคุมสัญญาณอินเทอรัพท์ นี้คือ IE0 และ IE1 ใน TCON เมื่อเกิดการอินเทอรัพท์จากภายนอก แฟลกที่สร้างอินเทอรัพท์จะถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อมีตัวชี้ตำแหน่งของโปรแกรมตอบสนองอินเทอรัพท์ ถ้าเป็นการอินเทอรัพท์แบบเปลี่ยนสภาวะเท่านั้น อินเทอรัพท์ของ TIMER 0 และ TIMER 1 จะสร้างโดย TF0 และ TF1 ซึ่งถูกเซ็ท โดยการนับถึง 255 ของรีจิสเตอร์ตัวนั้น ๆ เมื่อเกิดอินเทอรัพท์ของ TIMER แฟลกที่เกิดขึ้นจะถูกทำ



(MSB)							(LSB)
EA	X	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved
ET2	IE.5	enables or disables the Timer 2 overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External interrupt 1. If EX1 = 0, External interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External interrupt 0. If EX0 = 0, External interrupt 0 is disabled.

รูปที่ 2.4 ภาพแสดงการอินเทอรัพท์และรีจิสเตอร์ IE

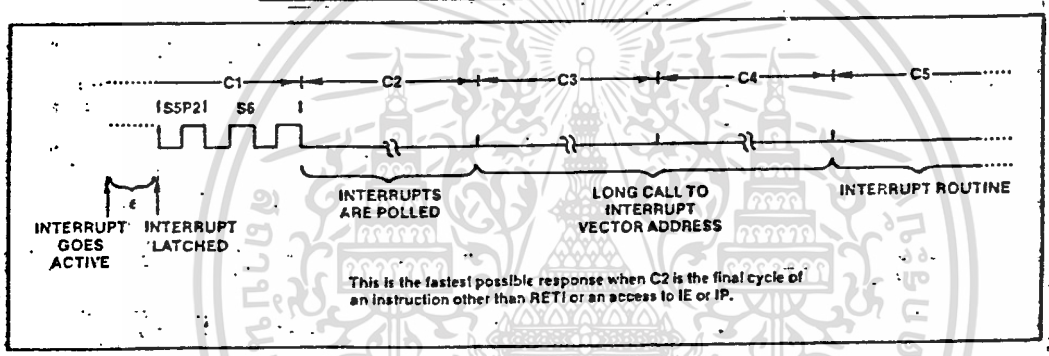
การเคลียร์โดยฮาร์ดแวร์ภายในเมื่อมีการชี้ไปยังตำแหน่งของโปรแกรมตอบสนองอินเทอรัพท์

อินเทอรัพท์ของพอร์ทอนุกรม จะสร้างโดยการ OR ของ RI และ TI แฟลกทั้ง 2 ตัวนี้จะไม่ถูกเคลียร์ด้วยฮาร์ดแวร์ เมื่อมีการชี้ยังตำแหน่งโปรแกรมตอบสนองอินเทอรัพท์ในโปรแกรมตอบสนองอินเทอรัพท์จะต้องรู้ว่าเป็น RI หรือ TI แล้วจึงเคลียร์ด้วยซอฟแวร์

ทุกบิทที่สร้างอินเทอรัพท์สามารถทำการเซทหรือเคลียร์บิทได้ด้วยซอฟแวร์เหมือนกับทำด้วยฮาร์ดแวร์ ดังนั้นอินเทอรัพท์สามารถสร้าง หรือยกเลิกได้โดยกำหนดในซอฟแวร์ แต่ละอินเทอรัพท์สามารถอีนาเบิลหรือดิสเอเบิล แยกจากกันโดยอิสระด้วยการกำหนดใน SFR ชื่อ IE (รูป 2.5) ใน IE จะมีบิทหนึ่งคือ EA ซึ่งจะดิสเอเบิลทั้งหมดพร้อมกันได้

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

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



รูปที่ 2.5 แสดง TIMING ของการตอบสนองการอินเทอรัพท์

โครงสร้างหน่วยความจำ, แอดเดรสซิงโหมดและตรรกศาสตร์

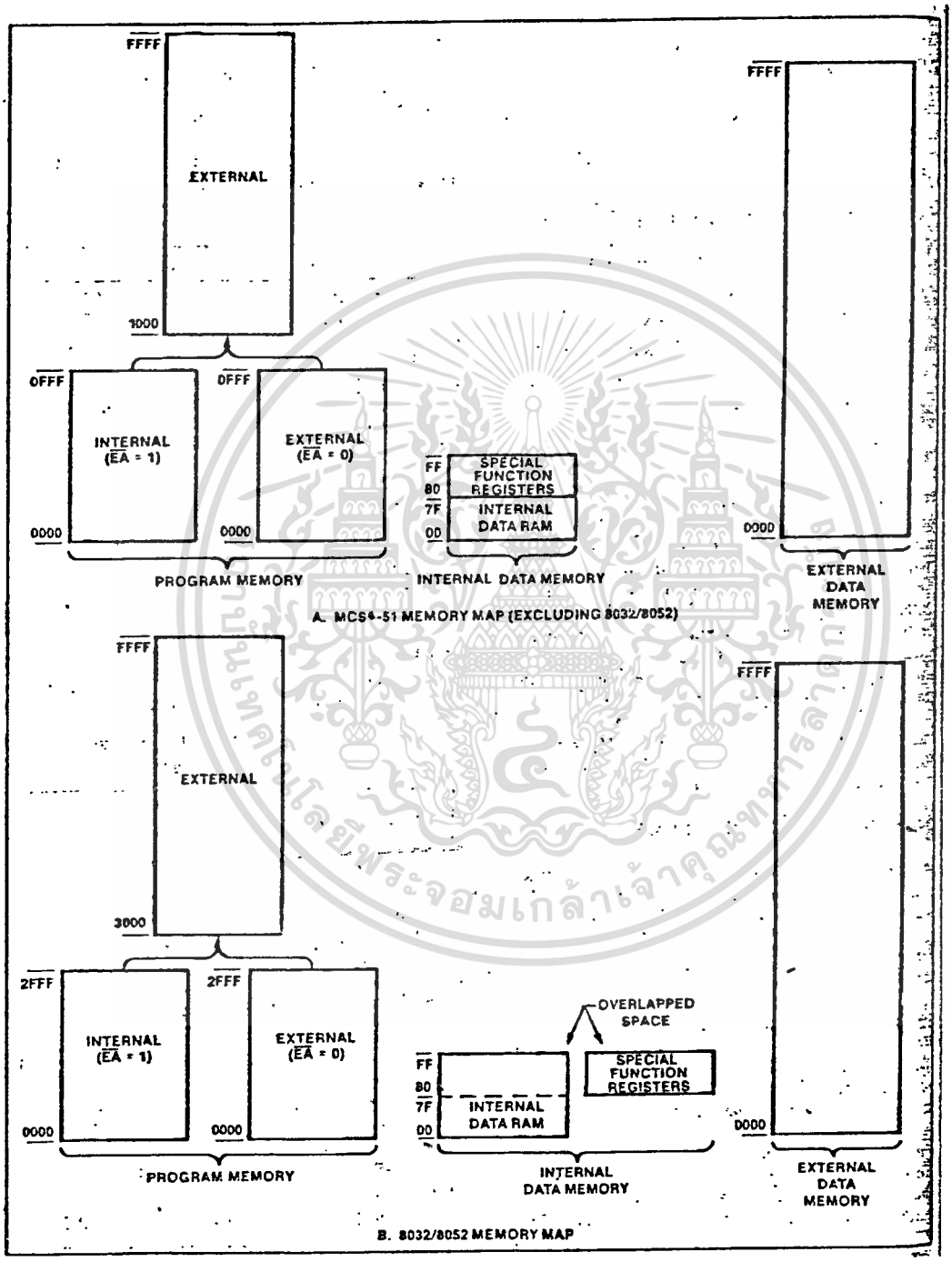
โครงสร้างของตระกูล 51สามารถใช้งานหน่วยความจำภายในและภายนอกได้อย่างดี วิธีการแอดเดรส ต่าง ๆ จะทำให้คำสั่งมีประสิทธิภาพ

การจัดการหน่วยความจำ

8051 มีหน่วยความจำพื้นฐานได้ดังนี้

- หน่วยความจำสำหรับโปรแกรม 64 กิโลไบท์
- หน่วยความจำสำหรับข้อมูลอยู่ภายนอก 64 กิโลไบท์

- หน่วยความจำภายในแบบ RAM 256 ไบท์ (ใน 8032/8052 มี ถึง 380 ไบท์)



รูปที่ 2.6 แสดงแผนผังของหน่วยความจำ

หน่วยความจำเก็บโปรแกรม (Program Memory Address Space)

หน่วยความจำ 64 กิโลไบท์ สำหรับโปรแกรมจะมีทั้งภายในและภายนอก ถ้าขา EA ต่อไว้ที่ High 8051 จะทำงานจากโปรแกรมภายใน นอกเสียจากจะมีการเรียกโปรแกรมจากตำแหน่งมากกว่า OFFFH (1FFFH ใน 8052) ตำแแหน่ง 1000H ถึง OFFFFH (2000H ถึง OFFFFH ใน 8052) จะอยู่ภายนอกเสมอ ถ้าขา EA ต่อไว้ที่ LOW 8051 จะอ่านโปรแกรมจากภายนอกเท่านั้น ทั้งสองแบบจะใช้ PC ขนาด 18 บิท

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

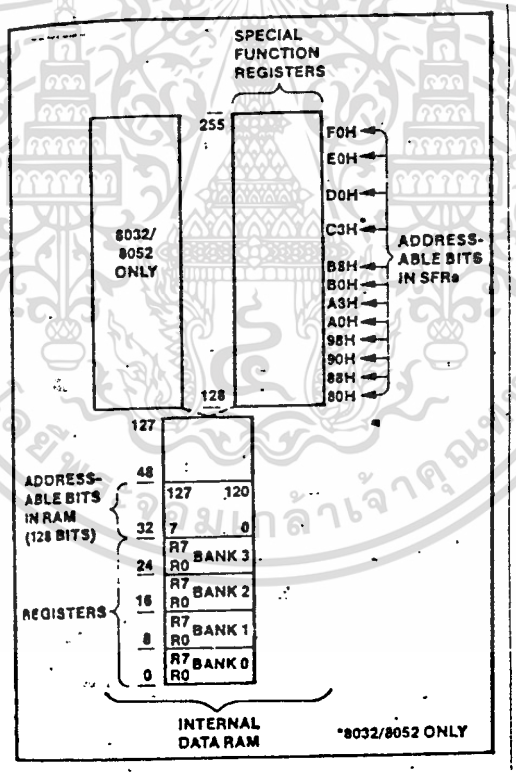
Source	Address
External Interrupt 0	0003H
Timer 0 Overflow	000BH
External Interrupt 1	0013H
Timer 1 Overflow	001BH
Serial Port	0023H
Timer 2 Overflow/TZEX Negative Transition	002BH

ตารางที่ 2

ที่ว่างสำหรับเก็บข้อมูล (Data Memory Address Space)

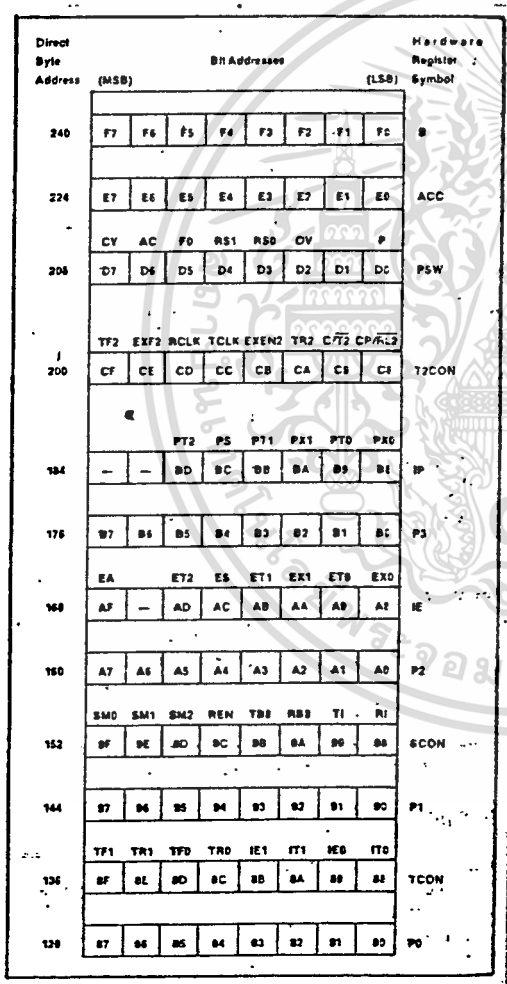
Data Memory Address Space ประกอบด้วยหน่วยความจำภายในและภายนอกการติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง MOVX เท่านั้น

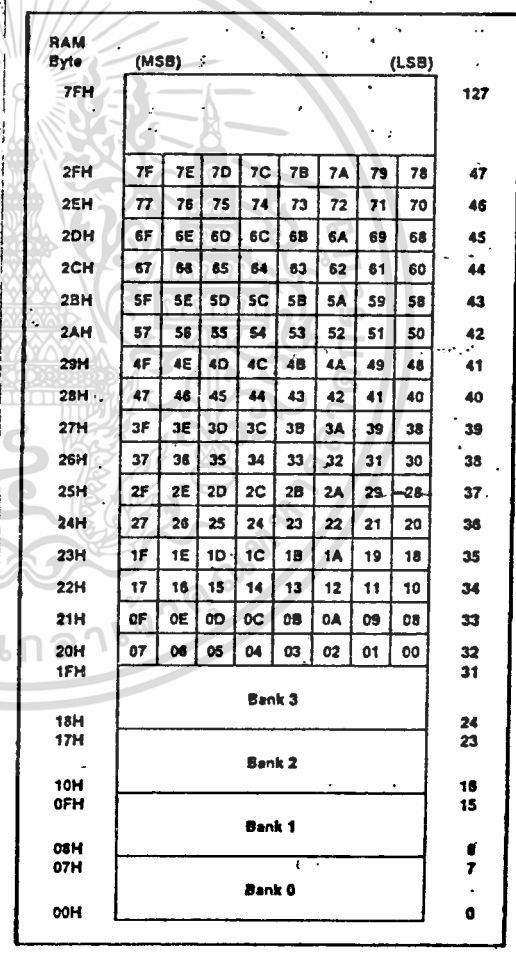
หน่วยความจำ(RAM) ภายในจะแบ่งออกเป็น3 ส่วน 128 ไบท์ต่ำของ RAM,(128 ไบท์สูงของRAM จะติดต่อได้เฉพาะใน 8032/8052 เท่านั้น) และ 128 ไบท์ของ SFR 128 ไบท์ของ RAM จะใช้ตำแหน่งแอดเดรสเดียวกันกับ SFR แต่การติดต่อจะใช้แอดเดรสโหมดต่างกัน ซึ่งจะกล่าวถึงต่อไป



รูปที่ 2.7 ภาพแสดงถึงหน่วยความจำภายใน

รูป 2.7 แสดง address mapping ของหน่วยความจำสำหรับข้อมูลภายในมีรีจิสเตอร์อยู่ 4 ชุด (BANK) ชุดละ 8 รีวิสเตอร์ ในตำแหน่ง 0 ถึง 31ใน RAM ชุดล่าง การใช้งานแต่ละขณะจะใช้ได้เพียงชุดเดียวเท่านั้น (กำหนดใน PSW) 16 ไบท์ต่อมา (ตำแหน่ง 32 ถึง 47) จะใช้สำหรับ 128 บิท แอดเดรสเอเบิล เป็นช่วงของหน่วยความจำซึ่งสามารถอ้างอิงได้ถึงทีละบิท รูป 2.8 แสดงแอดเดรส ของ RAM บิท SFR ก็สามารถใช้แบบบิท-แอดเดรสได้ ดังรูป 2.9





รูปที่ 2.8 แสดงตำแหน่งของ SFR

รูปที่ 2.9 SFR รีจิสเตอร์

การอ่านข้อมูลจากตำหน่งหน่วยความจำภายใน ซึ่งไม่ได้ใช้งานจะได้ข้อมูลไม่แน่นอน

แอดเดรสชิงโหมด

8051 มี 5 แอดเดรสชิงโหมด

- รีจิสเตอร์ (Register addressing)
- ไดเรค (Direct addressing)
- รีจิสเตอร์ อินไดเรค (Indirect Register addressing)
- อิมมีเดียท (Immediate addressing)
- เบส-รีจิสเตอร์ ร่วมกับ อินเดค-รีจิสเตอร์ อินไดเรค (Base-Register+Index-RegisterIndirect addressing)

ตาราง 2 สรุป หน่วยความจำที่สามารถทำการติดต่อได้ด้วยแต่ละวิธีแอดเดรสชิงโหมด

รีจิสเตอร์-แอดเดรสชิง

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

ไดเรคแอดเดรสชิง

ไดเรคแอดเดรสชิงเป็นวิธีเดียวที่จะใช้ติดต่อกับ SFR 128 ไบท์ต่ำของ RAM ภายใน

รีจิสเตอร์-อินไดเรค แอดเดรสซิง

รีจิสเตอร์ อินไดเรค แอดเดรสซิง จะใช้ค่าของ R0 หรือ R1 (ของชุดที่กำลังใช้งาน) เป็นตัวชี้ไปยังตำแหน่งใน 258 ไบท์ คือ 128 ไบท์ล่าง ของ RAM ภายใน, 128 ไบท์บน ของ RAM ภายใน (ใน 8032/8052 เท่านั้น) หรือใน 256 ไบท์ของหน่วยความจำสำหรับข้อมูลภายนอก SFR จะไม่สามารถติดต่อได้ด้วยวิธีนี้ และการติดต่อหน่วยความจำภายนอกถึง 64 กิโลไบท์ จะต้องใช้ Data Pointer แบบ 16 บิทเท่านั้น

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

อิมมีเดียท แอดเดรสซิง

อิมมีเดียทแอดเดรสซิง จะมีค่าคงที่ปรากฏอยู่ในรหัสคำสั่ง (OP Code) ในโปรแกรม

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

บูลีนโพรเซสเซอร์ (Boolean Processer)

บูลีนโพรเซสเซอร์ เป็นอินทิเกรตบิทโพรเซสเซอร์ ใน 8051 จะมีชุดคำสั่งเฉพาะ , แอคคิวมูเลเตอร์ (Carry Flag) และบิทแอดเดรสเอเบิลใน RAM และอินพุท /เอาท์พุท สามารถใช้ตรวจเงื่อนไขเพื่อให้มีการกระโดดหรือไม่กระโดดเมื่อมีการเซทหรือไม่ถูกเซท , ข้ามถ้ามีการเซทแล้วเคลียร์, MOVX เข้า Carry แอดเดรสเอเบิลบิท หรือค่าคอมพลีเมนท์สามารถ

AND หรือ OR กับค่าของ carry Flag ได้ ผลลัพท์จะเก็บใน ตัวทด (Carry) รีจิสเตอร์

ชุดคำสั่ง

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

ฟังก์ชั่นทั้งหมด (Functional Overview)

ชุดคำสั่ง MCS 51 จะแบ่งออกเป็น 4 กลุ่ม ตามการทำงาน ดังนี้

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

เคลื่อนย้ายข้อมูล (Data Transfer)

การเคลื่อนย้ายข้อมูล แบ่งออกเป็น 3 แบบ

- การย้ายข้อมูลทั่วไป
- กำหนดแอคคิวมูเลเตอร์
- แอดเดรส ออบเจค

การทำงานข้างบนไม่มีผลต่อแฟลกในPSW ยกเว้น POP หรือ MOV โดยตรงกับ PSW

การย้ายข้อมูลทั่วไป

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

<u>กำหนดแอคคิวมูเลเตอร์</u>

. XCH จะแลกข้อมูลจาก Source Operand กับ Accumulator
. XCHD จะแลกข้อมูลเฉพาะ 4 บิทล่างของไบท์กับ 4 บิทล่างของ Accumulator
. MOVX จะย้ายข้อมูลระหว่างหน่วยความจำภายนอก และแอคคิวมูเลเตอร์ ตำแหน่งหน่วยความจำภายนอก จะอ้างอิงโดย รีจิสเตอร์ DPTR (16 บิท) หรือรีจิสเตอร์ R1,R0 (8 บิท)
. MOVC ย้ายข้อมูลจากในหน่วยความจำของโปรแกรมไปยังแอคคิวมูเลเตอร์ โอเปอแรนด์ใน A จะถูกใช้เป็นอินเด็กซ์ไปยัง 256 ไบท์ โดยใช้ร่วมกับ DPTR หรือ PC ค่าจากหน่วยความจำจะถูกอ่านมายังแอคคิวมูเลเตอร์

<u>แอดเดรส-ออเจค ทรานเฟอร์</u>

MOV DPTR, #data จะกำหนดค่าจากโอเปอแรนด์ให้กับคู่รีจิสเตอร์ DPH และ DPL โดยตรง

การคำนวณ (Arithmatics)

8051 สามารถทำภารคำนวณได้ 4 แบบ และเป็นแบบ 8 บิท ไม่มีเครื่องหมาย มีแฟลกบอกการโอเวอร์โฟลว์ อย่างไรก็ตามในการบวกและลบจะสามารถทำได้ทั้งแบบมีกครื่องหมายและ ไม่มี การคำนวณสามารถให้ผลออกมาในรูปของ ฐานสิบ (BCD) ได้โดยตรง

การบวก

- INC (Increment) บวก 1 เข้ากับ Source ใน Operand และเก็บผลลัพธ์ใน Operand
- ADD บวก A เข้ากับ Source ใน Operand และจะทำการเก็บผลลัพธ์ใน A
- ADDC (Add with Carry) บวก A เข้ากับ Source ในโอเปอแรนด์ และบวกด้วย 1 ถ้า CY ถูกเซท เก็บผลลัพธ์ใน A
- DAA (Decimal-Add-Adjust for BCD Addition) จะปรับค่าของการบวกซึ่งเกิดจาก การบวกเลขฐานสิบเข้าด้วยกัน ให้ได้ผลลัพธ์เป็นเลขฐานสิบ ค่าที่ได้จะเก็บใน A จะมี CY ถูกเซท ถ้าค่าที่ได้เกิน 99 นอกนั้น CY จะถูกเคลียร์

การลบ

- SUBB (Substract with Borrow) ลบ Source ที่สองในโอเปอแรนด์ออกจาก Source แรก ในโอเปอแรนด (แอคคิว-มูเลเตอร์) และลบด้วย 1 ถ้า CY ถูกเซทเก็บผลลัพธ์ใน A
- DEC (Decrement) ลบ 1 ออกจาก Source ในโอเปอแรนด์ และเก็บผลลัพธ์ในโอเปอแรนด์

การคูณ (Multiplication)

. MUL จะทำการคูณค่าจากรีจิสเตอร์ A กับรีจิสเตอร์ B แบบไม่คิดเครื่องหมายเข้าด้วยกันและผลลัพธ์จะเป็น 2 ไบท์ ไบท์ต่ำเก็บใน A และไบท์สูงจะเก็บใน B โอเวอร์โฟล์วจะถูกเคลียร์ถ้าไบท์บนเป็น 0 และจะถูกเซ็ทถ้าไม่เป็น 0 Cy จะถูกเคลียร์ Accไม่มีการเปลี่ยนแปลง

การหาร

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

นอกจากที่กล่าวมาข้างบนแล้ว แฟลกใน PSW จะมีการเปลี่ยนแปลง ดังนี้

. CY จะถูกเซท ถ้าการคำนาณเกิด ตัวทดที่บิทสูงสุดของผลลัพธ์ นอกนั้น CY จะถูกเคลียร์
. AC จะถูกเซทถ้าผลลัพธ์เกิดตัวทดจาก 4 บิทล่างของผลลัพธ์ (ในระหว่างการบวก) หรือ เกิดตัวทดที่บิทสูงสุด(ในระหว่างการลบ) นอกนั้น AC จะถูกเคลียร์
. OV จะถูกเซทถ้าการคำนวณเกิดตัวทดขึ้นในบิทสูงสุดของผลลัพธ์ นอกนั้น OV จะถูกเคลียร์ OV จะถูกใช้ในการทำงานแบบ 2's Complement เพราะ OV จะถูกเซทเมื่อผลลัพธ์ไม่สามารถเก็บในค่า 8 บิท

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

<u>ลอจิก (Logic)</u>

* 8051 สามารถทำงานเกี่ยวกับลอจิกได้ ทั้งแบบบิทและ ไบท์ โอเปอแรนด์ การทำงานโอเปอแรนด์เดียว
* CLR ให้ค่าใน A หรือ ไดเรคแอดเดรสเอเบิล มีค่าเป็น 0
* SETB ให้ค่าในไดเรคเอเบิลบิท เป็น 1
* CPL ใช้ทำการคอมพลีเมนต์ค่าของรีจิสเตอร์ A โดยไม่มีผลต่อแฟลก หรือไดเรคแอดเดรสเอเบิลใดๆ
* RL,RLC,RR,RRC,SWAP เป็น 5 การทำงานสำหรับการวนข้อมูล โดยสามารถทำงานกับ Acc , RL เป็นการ-วนข้อมูลไปทางซ้าย,RR เป็นการวนข้อมูลไปทางขวา, RLC เป็นการวนข้อมูลไปทางซ้ายผ่าน C , RRC เป็นการวนข้อมูลไปทางขวาผ่าน C และ SWAP เป็นการวนข้อมูลไปทางซ้าย 4 ตำแหน่ง สำหรับ RLC และ RRC ค่าของแฟลก CY จะเท่ากับบิทที่วนเข้ามายัง C SWAP จะวนข้อมูลไปทางซ้าย 4 ตำแหน่ง โดยจะเป็นการสลับข้อมูลบิท 3 ถึง 0 กับ บิท 7 ถึง 4

การทำงานแบบ 2 โอเปอแรนด์

* ANL จำทำการ AND บิทของ 2 Source ในโอเปอแรนด์ (หรือทั้งบิทและไบท์โอเปอแรนด์) แล้วเก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

* ORL จะทำการ OR บิทของ 2 Source ในโอเปอแรนด์ (หรือทั้งบิทและไบท์โอเปอแรนด์) แล้วเก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก
* XRL ทำการ Exclusive OR บิทของ Source ในโอเปอแรนด์ เก็บผลลัพธ์ไว้ในโอเปอแรนด์แรก

<u>คอนโทรล ทรานสเฟอร์</u>

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

UNCONDITIONAL CALL, RETURNS, และ JUMPS

จะทำให้การทำงานจากโปรแกรมเคาร์เตอร์เดิมแ ไปยังแอดเดรสตำแหน่งใหม่ ทั้งแบบโดยตรงและโดยอ้อม

. ACALL และ LCALL จะเก็บค่าของแอดเดรสของคำสั่งต่อไปไว้ในสแตกแล้วข้ามไปทำงานยังแอดเดรสที่ต้องการ ACALL เป็นคำสั่งแบบ 2 ไบท์ ถูกใช้เมื่อ ตำแหน่งที่จะไปอยู่ภายในแต่ละช่วง 2 K โดยเริ่มนับจาก PC ที่ต่อจากคำสั่ง ACALL , LCALL เป็นคำสั่งแบบ 3 ไบท์ ซึ่งจะอ้างถึง ตำแหน่งหน่วยความจำได้ถึง 64 กิโลไบท์ ใน ACALL ข้อมูล 11 บิทจากโอเปอแรนด์จะรวมกับ 5 บิทบนของ PC ชี้ไปยังตำแหน่งหน่วยความจำที่ต้องการ ถ้า ACALL ชี้ในตำแหน่ง 2 ไบท์สุดท้ายของหน้า 2 K การ CALL จะทำงานในหน้าต่อไป โดยค่าของ PC จะเพิ่มไปเพื่อชี้ตำแหน่งที่จะทำงาน

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

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

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

การข้ามแบบมีเงื่อนไข (Conditional Jumps)

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

- JZ เป็นคำสั่งการข้ามถ้าแอคคิวมูเลเตอร์ เป็น 0
- JNZ เป็นคำสั่งให้ข้ามถ้าแอคคิวมูเลเตอร์ ไม่เป็น 0
- JC คำสั่งให้ข้ามถ้าแฟลกตัวทด (Carry Flag) ถูกเซท
- JNC คำสั่งให้ข้ามถ้าแฟลกตัวทด (Carry Flag) ถูกเคลียร์
- JB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ถูกเซท
- JNB คำสั่งให้ข้ามถ้าไดเรคแอดเดรสบิท ไม่ถูกเซท
- CJNE จะเปรียบค่าใน 2 โอเปอแรนด์และ ทำการกระโดด ถ้า

โอเปอแรนด์ตัวแรกมีค่าน้อยกว่า โอเปอแรนด์ที่ 2 CY จะถูกเซท นอกนั้นแล้ว จะถูกเคลียร์ การเปรียบเทียบอาจเป็นระหว่าง A โดยตรงกับแอดเดรสเอเบิลไบท์ ในหน่วยความจำ ข้อมูลภายในหรือระหว่างค่าคงที่กับรีจิสเตอร์ A ,รีจิสเตอร์ในชุดรีจิสเตอร์ที่กำลังใช้งาน หรือกับ รีจิสเตอร์-อินไดเรคแอดเดรสไบท์ ของ RAM ภายใน DJNZ ลดค่าของ Source ในโอเปอแรนด์ และเก็บค่าผลลัพธ์ในโอเปอแรนด์การข้ามจะเกิดขึ้นถ้าผลลัพธ์ไม่เป็นศูนย์ Source ในโอเปอแรนด์ ในคำสั่ง DJNZ อาจเป็นไบท์ใดในหน่วยความจำข้อมูลภายในก็ได้ หรือรีจิสเตอร์แอดเดรสซึ่ง ก็จะใช้เป็นแอดเดรสใน Source ในโอเปอแรนด์ได้

การวนกลับของอินเทอร์รัพท์(INTERRUPT RETURN)

RETI จะเป็นคำสั่งการข้ามคล้ายกับ RET จะเพิ่มการทำงานในการอินาเบิลอินเทอร์รัพท์ที่มีลำดับความสำคัญเท่ากัน เพื่อให้มีการอินเทอร์รัพท์ต่อไปได้

ความหมายของคำสั่ง (Instruction Definition)

ใน MCS-51ชุดคำสั่งการทำงานของตระกูล MCS-51 เรียงตามลำดับตัวอักษรของนีโมนิค (Mnemonic) ซึ่งแสดงการทำงานของอุปกรณ์ ตัวอย่างสั้นๆที่แสดงไว้ถึงวิธีการใช้งานของคำสั่ง อาจบอกถึงผลที่เกิดขึ้นในแฟลก PSW และยังบอกถึงจำนวนไบท์ และจำนวนวงรอบการทำงาน, คำสั่งในรูปเลขฐานสอง และคำอธิบายสัญญลักษณ์ ก็จะปรากฏอยู่

ข้อสำคัญ จะมีการกล่าวถึงเฉพาะ Carry , Auxillary-Carry และ Overflow แฟลกเท่านั้น พาร์ริตี้บิทจะถูกคำนวณทุกครั้งหลังจากคำสั่งซึ่งมีการเปลี่ยนแปลงค่าของแอคคิวมูเลเตอร์ ในทำนองเดียวกับคำสั่งที่มีการชี้

รีจิสเตอร์โดยตรง ก็จะมีผลต่อสภาวะแฟลก ถ้าคำสั่งนั้นเกี่ยวข้องกับ PSW แฟลกแสดงสภาวะสามารถเปลี่ยนแปลงได้โดยวิธี Bit Manipulation

บทที่ 3

การคำนวณและการทดลอง

3.1 บัฟเฟอร์ของการส่งแบบอนุกรม

การคำนวณของอุปกรณ์ชิ้นนี้มีดังต่อไปนี้

3.1.1 การออกแบบวงจร

3.1.2 การคำนวณเกี่ยวกับอัตราการส่งแบบอนุกรม

3.1.3 การเขียนโปรแกรมต่างๆ แบ่งเป็น

- การเขียนโปรแกรมสำหรับไมโครโปรเซสเซอร์ 8031
- การเขียนโปรแกรมสำหรับคอมพิวเตอร์เพื่อรับข้อมูล

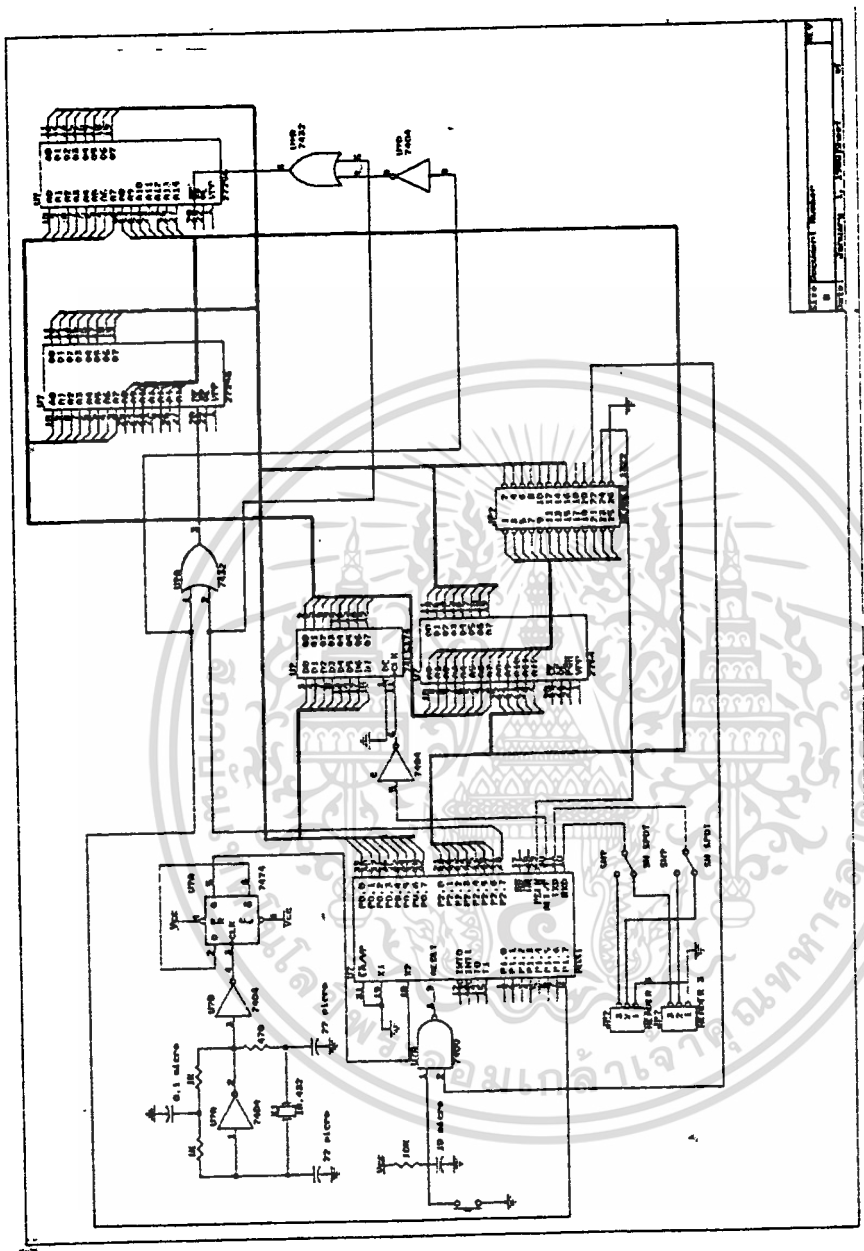
3.1.1 การออกแบบวงจร

วงจรที่ได้ออกแบบได้ถูกแสดงไว้ในรูปที่ 3.1

อธิบายวงจร

ในรูปที่ 3.1 เป็นวงจรบัฟเฟอร์ของการรับส่งแบบอนุกรมที่ทำงานเป็นบัฟเฟอร์ของ โมเด็ม โดยต่อผ่านพอร์ท RS-232 ของคอมพิวเตอร์และ พอร์ทอนุกรมของโมเด็ม โดยผ่านสวิทช์ที่ทำหน้าที่ เป็นตัวเลือกว่าจะติดต่อผ่านไปทางคอมพิวเตอร์ หรือทางโมเด็ม

ส่วนทางด้านการสร้างสัญญาณนาฬิกา เพื่อที่จะเป็นฐานเวลาให้กับ 8031 นั้น ได้ถูกสร้างขึ้นโดยผลึกคริสตรอล ขนาด 18.432 เมกกะเฮิรตซ์ เมื่อได้สัญญาณนาฬิกาความถี่ 18.432 เมกกะเฮิรตซ์แล้วก็นำเอามาหารด้วยสอง



รูปที่ 3.1 ภาพแสดงวงจรบัฟเฟอร์ของการรับส่งแบบอนุกรม

โดยใช้ไอซี เบอร์ 7474 คือ ไอซี ดี ฟลิบ-ฟลอป (D FLIP-FLOP) เพื่อที่จะได้สัญญาณนาฬิกาความถี่เท่ากับ 9.216 เมกกะเฮิรตซ์ เพื่อที่จะป้อนให้กับ 8031 ที่ขา 18

ทางด้านตัวไมโครโปรเซสเซอร์ 8031 ซึ่งมีพอร์ทแบบขนานอยู่ 3พอร์ทด้วยกัน คือ พอร์ท 0,พอร์ท 1 และพอร์ท 2 โดยกำหนดให้ใช้ พอร์ท 0 เป็นพอร์ทที่ส่งผ่านทั้งตำแหน่งข้อมูลไบท์ต่ำ (LOW ADDRESS) และส่งผ่านข้อมูลด้วย ส่วน พอร์ท 2 ได้ถูกใช้ให้เป็นพอร์ทที่ส่งผ่านตำแหน่งข้อมูลไบท์สูง (HIGH ADDRESS) เนื่องจากว่าเราได้ใช้ขาของพอร์ททั้งหมด 16 ขา เพื่อที่จะไปติดต่อกับแหล่งเก็บข้อมูลขนาด 64 กิโลไบท์ และได้ใช้ขาที่ 8 ของพอร์ท 1 และขาที่ 8 ของพอร์ท 2 เป็นตัวที่จะกำหนดว่าจะติดต่อกับหน่วยความจำ หรือไม่และเป็นตัวกำหนดว่าจะติดต่อกับหน่วยความจำ 32 กิโลไบท์ล่างหรือ 32 กิโลไบท์บน เนื่องจากว่า หน่วยความจำได้ถูกแบ่งออกเป็นสองส่วน คือ บนและล่าง

ส่วนทางด้านวงจรดีโค๊ดเพื่อที่จะนำไปใช้งานกับ แอลอีดี (LED) เพื่อที่จะแสดงผลต่างๆ ได้ใช้ ไอซี เบอร์ 74138 เพื่อทำการดีโค๊ด โดยให้ขาที่ 8ของ พอร์ท 2เป็นตัวกำหนดว่าขณะนี้ไม่ได้ติดต่อกับหน่วยความจำแต่ติดต่อกับวงจรดีโค๊ดและส่วนทางด้านพอร์ท 1 ได้ถูกนำไปต่อกับคีย์บอร์ด (KEYBOARD) โดยต่อแค่ 7 ขา เพื่อที่จะติดต่อกับคีย์ 12 คีย์

3.1.2 <u>การคำนวณอัตราการส่งแบบอนุกรม</u>

การส่งแบบอนุกรมในวงจรนี้ ใช้การส่งในโหมด 1 ของการส่งแบบอนุกรมของไมโครโปรเซสเซอร์ 8031 ซึ่งเป็นโหมดที่สามารถที่จะตั้งอัตราการส่งแบบอนุกรมได้ โดยกำหนดค่าต่างๆ ในสัญญาณนาฬิกา 1 (TIMER1) โดยมีสูตรการคำนวณดังนี้

อัตราการส่ง = $(2^{SMOD}/32)*[$ความถี่ออสซิเลเตอร์$/12*(256-TH1)]$

ในวงจรนี้เราได้ใช้ความถี่ออสซิเลเตอร์ เท่ากับ 9.216 เมกกะเฮิรท์ และทำการกำหนดให้ รีจิสเตอร์ PCON บิทที่ 7 คือ SMOD เท่ากับ 0 ส่วน TH1 คือค่าในไบท์สูงของสัญญาณนาฬิกา 1 จะมีค่าเท่าไรแล้วแต่การกำหนดว่าต้องการอัตราการส่งข้อมูลให้เป็นเท่าไร เช่น ถ้าต้องการอัตราการส่งแบบอนุกราเท่ากับ 9600 บิทต่อวินาที ให้กำหนดค่า TH1 เท่ากับ 251 ในฐานสิบ

3.1.3 <u>การเขียนโปรแกรมต่างๆ</u>

-การเขียนโปรแกรมสำหรับ 8031

โปรแกรมของตัวไมโครโปรเซสเซอร์ 8031 นั้นต้องเขียนให้อยู่ในรูปของภาษาแอสเซมบลี้ (ASSEMBLY) ซึ่งต้องเป็นภาษาแอสเซมบลี้ของ MCS 51 โดยเฉพาะ

ตัวโปรแกรมนั้นสามารถแยกออกเป็นส่วนต่างๆ สำคัญได้ดังนี้

-ส่วนแรก คือ โปรแกรมเพื่อที่จะทำการตรวจเช็คคีย์บอร์ดใช้ตรวจเช็คว่ามีใครกดคีย์หรือไม่ ถ้ามีก็จะเช็คต่อไปว่าเป็นคีย์ไหน ถ้าตรงกับคีย์ที่ได้ตั้งไว้ให้เป็น คีย์รับก็ให้โปรแกรมกระโดดไปส่วนของโปรแกรมที่ทำการรับข้อมูลจากภายนอกเข้ามาเก็บเอาไว้ในหน่วยความจำ

-ส่วนที่สอง คือโปรแกรมการรับข้อมูลจากภายนอก ในที่นี้คือการรับข้อมูลที่ถูกส่งมาจากโมเด็ม ในโปรแกรมส่วนนี้ได้มีการส่งและรับโค็ดโต้ตอบกันระหว่างตัวบัฟเฟอร์และโมเด็ม เพื่อที่จะให้แน่นอนว่าเป็นสัญญาณจากโมเด็ม หลังจากนั้นทำการรับข้อมูลที่ถูกส่งผ่านมาทางโมเด็มนั้นไปเก็บไว้ที่หน่วยความจำ

-ส่วนที่สาม คือ โปรแกรมการส่งข้อมูลไปให้กับเครื่องคอมพิวเตอร์ ถ้าเกิดมีการกดคีย์ให้ทำการส่ง โปรแกรมก็จะกระโดดมาที่ส่วนของโปรแกรมส่ง เพื่อที่จะทำการส่งข้อมูลที่ได้เก็บเอาไว้นั้นไปให้ กับเครื่อง

คอมพิวเตอร์ ตัวโปรแกรมต้องมีการกำหนดอัตราการส่งให้เป็น 9600 บิทต่อวินาที

โปรแกรมการรับและส่งข้อมูลของบัฟเฟอร์ของการส่งข้อมูลแบบอนุกรม

```
P1    EQU  H90
TIMLO EQU  0
TIMHI EQU  0
KEYB1 EQU  50
KEYB2 EQU  51
KEYTM EQU  52
KEYBF EQU  53
CORX  EQU  54
CORY  EQU  55
SCLE  EQU  56
DPH   EQU  H83
DPL   EQU  H82
PCON  EQU  H87
SBUF  EQU  H99
TI    EQU  H99
RI    EQU  H98
TMOD  EQU  H89
TCON  EQU  HC8
SCON  EQU  H98
TR1   EQU  H8E
```

```
          TH1    EQU  H8D
          PO     EQU  H80
          P      EQU  H90
          P2     EQU  HA0
          P3     EQU  HB0
          PO     EQU  H80
          P      EQU  H90
          P2     EQU  HA0
          P3     EQU  HB0
          ACC    EQU  HE0
          P3.2   EQU  HB2
          P3.3   EQU  HB3
          P3.4   EQU  HB4
          P3.5   EQU  HB5
          P1.7   EQU  H97
          P1.6   EQU  H96
          P1.5   EQU  H95
          P1.1   EQU  H91
          CPR    EQU  118
          ORG
          MOV    RO,#0
DELAY:    DJNZ   RO,DELAY
          MOV    TMOD,#B00101100
          MOV    SCON,#B01010010
          MOV    TCON,#B01101001
```

```
            MOV   PCON,#B10000000
            MOV   TH1,#251
            SETB  TR1
            MOV   A,SBUF
INIT:       MOV   R0,#12
            MOV   R1,#0
LOOP1:      MOV   DPTR,#H1000
            MOV   A,R1
            INC   R1
            MOVX  @DPTR,A
            MOV   DPTR,#CRT
            MOVC  A,@A+DPTR
            MOV   DPTR,#H1001
            MOVX  @DPTR,A
            DJNZ  R0,LOOP1
;TEST PATTERN
            MOV   R0,#0
            FUCK:    DJNZ  R0,FUCK
            MOV   A,#B00001001
            MOV   DPTR,#H3000
            MOVX  @DPTR,A
            MOV   R0,#HFF
            MOV   R2,#0
            MOV   R1,#0
WRO:        INC   R0
```

```
        MOV   A,R0
        MOV   DPTR,#H2000
        MOVX  @DPTR,A
        MOV   A,#0
        MOV   DPTR,#HFF00
WR1:    MOVX  @DPTR,A
        INC   DPTR
        DJNZ  224,WR1
        DJNZ  R2,WR0
        MOV   A,#B10000000
        MOV   DPTR,#H3000
        MOVX  @DPTR,A
        MOV   R7,#0
;CHECK ROUTINE
NUE1:   JNB   RI,WAKE
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#27,NUE1    ;FOR ESCAPE
NUE2:   JNB   RI,NUE2
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#H2A,NUE2          ;FOR '*'
NUE3:   JNB   RI,NUE3
        CLR   RI
        MOV   A,SBUF
```

```
        CJNE   A,#H53,NUE4      ;FOR SEND DATA 'S'
NUE31:  JNB    RI,NUE31         ;RECEIVE SCALE
        CLR    RI
        MOV    A,SBUF
        MOV    RO,#SCLE
        MOV    @RO,A
        LJMP   TX
NUE4:   CJNE   A,#H43,NUE5      ;FOR COORDINATE 'X''Y'
  XX:   JNB    RI,XX
        CLR    RI
        MOV    A,SBUF
        MOV    RO,#CORX
        MOV    @RO,A            ;FOR 'X'
        INC    RO
YY:     JNB    RI,YY
        CLR    RI
        MOV    A,SBUF
        MOV    @RO,A            ;FOR 'Y'
       LJMP   NUE1
NUE5:   JNB    RI,NUE5
        CLR    RI
        MOV    A,SBUF
        CJNE   A,#H52,NUE5      ;FOR RECEIVE DATA
NUE6:   JNB    RI,NUE6
```

```
        CLR   RI
        MOV   A,SBUF
        MOV   R0,A
        MOV   A,#HFF
NUE8:   DJNZ  R0,NUE7
        MOV   R4,A          ;INDICATE SIZE OF IMAGE
        LJMP  RX
NUE7:   CLR   C
        RRC   A
        AJMP  NUE8
WAKE:   MOV   P1,#B01111000
        MOV   A,P1
        ORL   A,#B10000111
        CPL   A
        JZ    NUE1
INTDL:  MOV   KEYB1,#TIMLO
        MOV   KEYB2,#TIMHI
        KSCAN:  NOP
        MOV   R2,#B01111011
KEYCO:  MOV   P1,R2
        MOV   A,P1
        ORL   A,#B10000111
        MOV   KEYTM,A
        CPL   A
        JZ    NOPRS
```

```
        LJMP   PRESS
NOPRS:  MOV    A,R2
        ORL    A,#B11111000
        RRC    A
        ANL    A,#B01111111
        MOV     R2,A
        JC     KEYCO
        MOV    A,KEYB1
        DEC    A
        MOV    KEYB1,A
        JNZ    KSCAN
        MOV    A,KEYB2
        DEC    A
        MOV    KEYB2,A
        JNZ    KSCAN
        LJMP   WAKE
PRESS:  MOV    R5,#0
BAUCE:  DJNZ   R5,BAUCE
        MOV    A,P1
        ORL    A,#B10000111
        CJNE   A,KEYTM,LKSCA
        LJMP   NSHIX
LKSCA:  LJMP   KSCAN
NSHIX:  MOV    R5,#5
NSHIF:  RLC    A
```

```
        DEC     R5
        JC      NSHIF
        MOV     A,R5
        ADD     A,R5
        ADD     A,R5
        MOV     R5,A
        MOV     A,R2
NSH2:   INC     R5
        RRC     A
        JC      NSH2
        CJNE    R5,#10,NSH3
        LJMP    TX
NSH3:   CJNE    R5,#12,NSH4
        LJMP    RX
JATE:   LJMP    RX1
NSH4:   NOP
RELEA:  MOV     P1,#B01111000
        MOV     A,P1
        ORL     A,#B10000111
        CJNE    A,#HFF,RELEA
        MOV     R0,#0
        MOV     R1,#0
        MOV     A,#B11001110
        MOV     DPTR,#H3000
        MOVX    @DPTR,A
```

```
CSYNC:  JB      P3.3,CSYNC
S2:     JNB     P3.3,S2
        JB      P3.2,CSYNC
S3:     JB      P3.3,S3
S4:     JNB     P3.3,S4
        JNB     P3.2,S3
        MOV     R4,#30
S5:     JB      P3.3,S5
S6:     JNB     P3.3,S6
        DJNZ    R4,S5
        MOV     DPTR,#H2000
NEXT:   MOV     A,R1
EOC:    JB      P3.3,EOC
S7:     JNB     P3.3,S7
        MOVX    @DPTR,A
        INC     R1
        DJNZ    R0,NEXT
;DISPLAY
        MOV     A,#B10000000
        MOV     DPTR,#H3000
        MOVX    @DPTR,A
        LJMP    WAKE
TX:     NOP
        MOV     R0,#0
        MOV     R1,#0
```

```
NOK:    DJNZ   R1,NOK
        DJNZ   R0,NOK
        MOV    R0,#SCLE
        MOV    DPTR,#H3000
        MOV    A,#H81
        MOVX   @DPTR,A
        MOV    A,#00
LABEL2: MOV    R2,A
        MOV    DPTR,#H2000
        MOVX   @DPTR,A
        MOV    DPH,#HFF
        MOV    A,#00
LABEL1: MOV    R1,A
        MOV    DPL,R1
        MOVX   A,@DPTR
SER51:  JNB    TI,SER51
        CLR    TI
        MOV    SBUF,A
        MOV    A,R1
        ADD    A,@R0
        JNC    LABEL1
        CLR    C
        MOV    A,R2
        ADD    A,@R0
        JNC    LABEL2
```

```
        CLR   C
        MOV   A,#H80
        MOV   DPTR,#H3000
        MOVX  @DPTR,A
        LJMP  WAKE
RX:     MOV   R0,#0
NOK1:   DJNZ  R0,NOK1
        MOV   R1,#CORY
        MOV   DPTR,#H3000
        MOV   A,#H09
        MOVX  @DPTR,A
        MOV   A,@R1
        DEC   A
        MOV   R2,A
        MOV   A,R4
        MOV   R6,A
SER62:  INC   R2
        MOV   A,R2
        MOV   DPTR,#H2000
        MOVX  @DPTR,A
        MOV   DPH,#HFF
        MOV   R3,CORX
        DEC   R3
        MOV   A,R4
        MOV   R5,A
```

```
SER61:  INC   R3
        MOV   DPL,R3
SER77:  JNB   R1,SER77
        CLR   R1
        MOV   A,SBUF
        MOVX  @DPTR,A
        DJNZ  R5,SER61
        DJNZ  R6,SER62
        MOV   A,#B10000000
        MOV   DPTR,#H3000
        MOVX  @DPTR,A
        LJMP  WAKE
RX1:    MOV   R0,#0
        MOV   R1,#0
NOK2:   DJNZ  R1,NOK2
        DJNZ  R0,NOK2
        MOV   DPTR,#H3000
        MOV   A,#H09
        MOVX  @DPTR,A
        MOV   R2,#H70
SER71:  INC   R2
        MOV   A,R2
        MOV   DPTR,#H2000
        MOVX  @DPTR,A
        MOV   DPH,#HFF
```

```
        MOV   R3,#H65
SER72:  INC   R3
        MOV   DPL,R3
SER73:  JNB   RI,SER73
        CLR   RI
        MOV   A,SBUF
        MOVX  @DPTR,A
        CJNE  R3,#HE5,SER72
        CJNE  R2,#HFO,SER71
        MOV   A,#B10000000
        MOV   DPTR,#H8000
        MOVX  @DPTR,A
        LJMP  WAKE
HALT:   SJMP  HALT
CRT:    DB    159
        DB    128
        DB    137
        DB    14
        DB    77
        DB    0
        DB    64
        DB    68
        DB    0
        DB    3
        DB    0
```

```
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0
        DB      0
        MOV     R0,#0
DELAY:  DJNZ    R0,DELAY
        MOV     TMOD,#B00101100
        MOV     SCON,#B01010010
        MOV     TCON,#B01101001
        MOV     TH1,#236
        SETB    TR1
        MOV     A,SBUF
        MOV     R0,#0
        MOV     R1,#0
JJ:     DJNZ    R1,JJ
        DJNZ    R0,JJ
HERE:   MOV     DPTR,#COMMX
        LCALL   RESMD
        MOV     DPTR,#AAMOD
        LCALL   RESMD
        LJMP    NOP
RESMD:  CLR     A
```

```
        MOVC    A,@A+DPTR
        CJNE    A,#H24,RESM2
        LJMP    HERE1
RESM2:  JNB     TI,RESM2
        CLR     TI
        MOV     SBUF,A
        INC     DPTR
        LJMP    RESMD
HERE1:  JNB     RI,HERE1
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#H4F,HERE1
HERE2:  JNB     RI,HERE2
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#H4B,HERE2
HERE5:  JNB     RI,HERE5
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#H0A,HERE5
        RET
NOP:    NOP
        MOV     R2,#2
DELY:   MOV     R0,#0
        MOV     R1,#0
```

```
JF:     DJNZ    R1,JF
        DJNZ    R0,JF
        DJNZ    R2,DELY
        MOV     DPTR,#STNAM
RESM3:  CLR     A
        MOVC    A,@A+DPTR
        CJNE    A,#H24,RESM4
        LJMP    CON
RESM4:  JNB     TI,RESM4
        CLR     TI
        MOV     SBUF,A
        INC     DPTR
        LJMP    RESM3
CON:
HALT:   AJMP    HALT
COMMX:  DB      'A'
        DB      'T'
        DB      ' '
        DB      'Z'
        DB      H0D
        DB      '$'
AAMOD:  DB      'A'
        DB      'T'
        DB      ' '
        DB      'S'
```

```
            DB      'O'
            DB      '='
            DB      '3'
            DB      HOD
            DB      '$'
COMM1:      DB      'A'
            DB      'T'
            DB      ' '
            DB      'V'
            DB      'O'
            DB      HOD
            DB      '$'
STNAM:      DB      'A'
            DB      'T'
            DB      ' '
            DB      'D'
            DB      'T'
            DB      '5'
            DB      ','
            DB      '2'
            DB      HOD
            DB      '$'
```

-การเขียนโปรแกรมสำหรับเครื่องคอมพิวเตอร์เพื่อรับข้อมูล

โปรแกรมสำหรับเครื่องคอมพิวเตอร์เพื่อที่จะทำการรับข้อมูลที่ส่งผ่านเข้ามาที่พอร์ทอนุกรมของเครื่องคอมพิวเตอร์ (RS-232) เพื่อที่จะนำข้อมูลที่ได้รับเข้ามานั้นไปเก็บไว้ในหน่วยความจำของเครื่องคอมพิวเตอร์ เพื่อที่จะนำไปแสดงผลต่อไป

ตัวโปรแกรมนี้เขียนด้วยภาษาเทอร์โบปาสคาล (TURBO PASCAL) ตัวโปรแกรมแบ่งออกเป็นโปรแกรมย่อยๆ และมีโปรแกรมหลักเป็นโปรแกรมเพื่อที่จะเลือกตัวโปรแกรมย่อย ในตัวโปรแกรมย่อยมีทั้งการรับมาเก็บแบบธรรมดาและเก็บแบบเป็นแฟ้มข้อมูล และยังมีการนำข้อมูลที่เก็บได้มาแสดงผลออกทางจอภาพได้ด้วย

******โปรแกรมสำหรับเครื่องคอมพิวเตอร์ที่จะรับข้อมูลทางอนุกรม******

```
Program Recieve SERIAL;
uses
    Dos,Crt,Printer;
label
    Loop,exit;
var
    key             :char;
    data            :array [0..127,0..255] of byte;
{-------------------------------------------------------------}
Procedure Program_8250;
begin
        Port[$3FB] := $80;
```

```
        Port[$3F8] := $0C;
        Port[$3F9] := $00;
        Port[$3FB] := $03;
        Port[$3FC] := $03;
        Port[$3F9] := $00;
end;
{-------------------------------------------------------}
Procedure ShowMenu;

label
    InKey;
begin
        ClrScr;
        GotoXY (30,2);
         writeln('MAIN  MENU');
        GotoXY (20,4);
         writeln('1 >  Input  code  from  com.1');
        GotoXY (20,6);
         writeln('2 >  Output  code  to  com.1');
        GotoXY (20,8);
         writeln('3 >  Save  code  to  data  file');
        GotoXY (20,10);
         writeln('4 >  Load  code  from  data  file');
        GotoXY (20,14);
         writeln('6 >  Exit  program');
```

```
          GotoXY (20,16);
           writeln('7 >   Show DATA  ');
          GotoXY (15,18);
           write('Select (1 - 7) ----->  ');
InKey:    key := ReadKey;
          if (key < '1') or (key > '7') then
          begin
               write(chr(7));
               goto InKey;
          end;
end;
{-------------------------------------------------------------}
Procedure Receive_Data;

label
    wait,wait1;
var
    t               :byte;
    i,j             :byte;
begin
          ClrScr;
          writeln('WAITING  FOR  INPUT  DATA');
wait:     t := Port[$3FD];
            if (t and $01) = 0 then goto wait;
          for i := 0 to 127 do
```

```
          begin
               for j := 0 to 255 do
               begin
wait1:              t := Port[$3FD];
                    if (t and $01) = 0 then goto wait1;
                    data[i,j] := Port[$3F8];
               end;
          end;
end;
{-------------------------------------------------------}
Procedure Show_DATA;
var
     i,j  : byte;
begin
      Clrscr;
      for i:= 0 to 127 do
      begin
           for j:= 0 to 255 do
           writeln ( i, j ,data[i,j]);
      end;
end;
{-------------------------------------------------------}
Procedure Send_Data;

label
```

```
    wait1;
var
    i,j             :byte;
    t               :byte;
begin
        ClrScr;
        writeln('SENDING  DATA  BACK');

        for i := 0 to 127 do
        begin
            for j := 0 to 127 do
            begin
wait1:          t := Port[$3FD];
                if (t and $20) = 0 then goto wait1;
                Port[$3F8] := data[i,j];
            end;
        end;
end;
{-------------------------------------------------------}
Procedure Write_File;

var
    name            :string;
    OutFile         :file;
begin
```

```
        ClrScr;
        write('      Output Filename --->  ');
        readln(name);
        assign (OutFile,name);

        ReWrite (OutFile);
        BlockWrite (OutFile,data[0,0],128);
        close (OutFile);
end;
{-------------------------------------------------------}
Procedure Read_File;

var
    name            :string;
    InFile          :file;
begin
        ClrScr;
        write('      Input Filename --->  ');
        readln(name);
        assign (InFile,name);

        ReSet (InFile);
        BlockRead (InFile,data[0,0],128);
        close (InFile);
end;
```

```
{*****************************************************}
{                    MAIN  PROGRAM                    }
{*****************************************************}
begin
          ClrScr;
          Program_8250;
loop:     ShowMenu;
          case key of
            '1':Receive_Data;
            '2':Send_Data;
            '3':Write_File;
            '4':Read_File;
            '6':GoTo exit;
            '7':Show_DATA;
          end;
          goto loop;
exit:
end.
```

3.2 การคำนวณของอุปกรณ์พัฒนาโปรแกรมของไมโครโปรเซสเซอร์ 8031

การคำนวณของอุปกรณ์ชิ้นนี้ประกอบด้วย

3.2.1 การออกแบบวงจร

3.2.2 การเขียนโปรแกรมสำหรับเครื่องคอมพิวเตอร์เพื่อทำการส่งข้อมูลให้มาเก็บไว้ในหน่วยความจำของตัวอุปกรณ์พัฒนาโปรแกรม

3.2.1 การออกแบบวงจร

วงจรที่ได้ถูกออกแบบไว้เป็นดังรูปที่ 3.2

อธิบายวงจร

วงจรของตัวอุปกรณ์พัฒนาโปรแกรมนั้น ส่วนประกอบสำคัญที่สุดของอุปกรณ์ชิ้นนี้คือ ไอซี เบอร์ 8255 ซึ่งทำหน้าที่เป็น อุปกรณ์รับส่งข้อมูลอินพุทและเอาท์พุทพอร์ทแบบขนาน (PARALLEL PORT) ซึ่งจะเป็นตัวกำหนดให้ว่า ข้อมูลที่ได้รับมาจากเครื่องคอมพิวเตอร์ทางพอร์ทข้อมูลนั้นเป็น แอดเดรสไบท์สูง หรือแอดเดรสไบท์สูงหรือว่าเป็นข้อมูล เพื่อที่จะได้ทำการแบ่งแยกและนำไปเก็บหรือนำไปชี้ตำแหน่งของข้อมูล ส่วนวงจรที่สำคัญมากอีกส่วนหนึ่งก็ คือ วงจรดีโค๊ด เพื่อถอดรหัสนำมาป้อนให้ที่ขาควบคุมของ 8255 โดยการนำเอาแอดเดรสของเครื่องคอมพิวเตอร์มาดีโค๊ด ในตัวอุปกรณ์นี้ ได้นำเอาแอดเดรสของเครื่องคอมพิวเตอร์ไอบีเอ็ม แบบ เอ็กที ตั้งแต่แอดเดรสที่ A0 ถึง A15 มาทำการดีโค๊ดด้วย อซี เบอร์ 74138 จะนำสัญญาณที่ได้มาจากการดีโค๊ดมาใช้เพียง 2 ค่า ค่าแรกนำเข้าไปต่อกับขาเลือกใช้ชิพนี้ (CHIP SELECT) ของ 8255 ค่าที่สองนำไปใช้ในการเลือกทิศทางของข้อมูลจริง ที่จะนำเข้าไปเก็บไว้ในหน่วยความจำ และสามารถนำเอาไปดีโค๊ดเพื่อที่จะแบ่งแยกได้ว่าค่าไหนเป็นแอดเดรสหรือว่าเป็นข้อมูลเนื่องจากว่าค่าที่สองนี้จะปรากฏออกมาก็ต่อเมื่อ เครื่อง

คอมพิวเตอร์ต้องการที่จะส่งข้อมูลที่จะมาเก็บในหน่วยความจำของตัวอุปกรณ์

ส่วนของหน่วยความจำบนตัวนี้ ใช้หน่วยความจำ แบบแรม (RAM) เบอร์ 6264 ซึ่งเป็นหน่วยความจำที่มีขนาด 8 กิโลไบท์ มีวงจรดีโค๊ดเพื่อที่จะถอดรหัสเพื่อจะเข้ามาอ่านหรือเขียนหน่วยความจำ โดยนำเอาสัญญาณ IOR (READ)และ IOW (WRITE)ของเครื่องไอบีเอ็ม มาดีโค๊ดร่วมกับสัญญาณ PSEN จากไมโครโปรเซสเซอร์ 8031

3.2.2 <u>การเขียนโปรแกรมเพื่อส่งข้อมูล</u>

การเขียนโปรแกรมนี้ได้ทำการออกแบบตามวงจรที่ประกอบขึ้นแล้ว โดยมีการเขียนโปรแกรมตัวนี้ได้ถูกทำการเขียนให้อยู่ใน ภาษา เทอร์โบปาสคาล และ รูปแบบของการเขียนโปรแกรมดังรูปที่ 3.3

******<u>โปรแกรมการส่งข้อมูลเพื่อไปเก็บในอุปกรณ์พัฒนาโปรแกรม</u>******

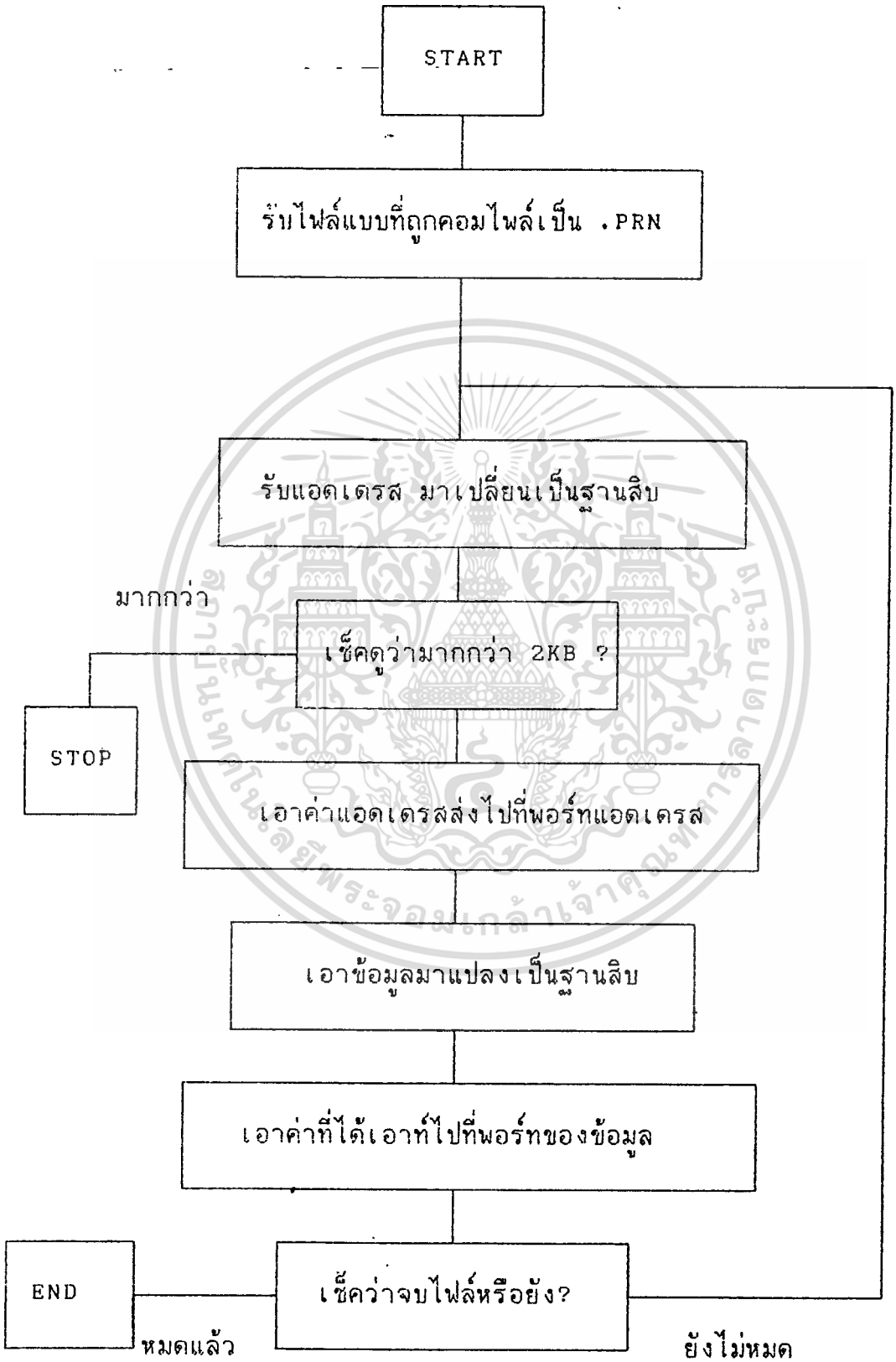
```
PROGRAM SERIES48_ASSEMBLER;
uses  crt;
TYPE source_str = string[80];
     source_strlab = string[10];
     jcode_str = string[4];
     mcode_str = string[2];
VAR   I,J,L,M,N,count : integer;
    perma : source_str;
    tm,ot,s_name : string[15];
    C,D,E : mcode_str;
    T,TOA : TEXT;
```

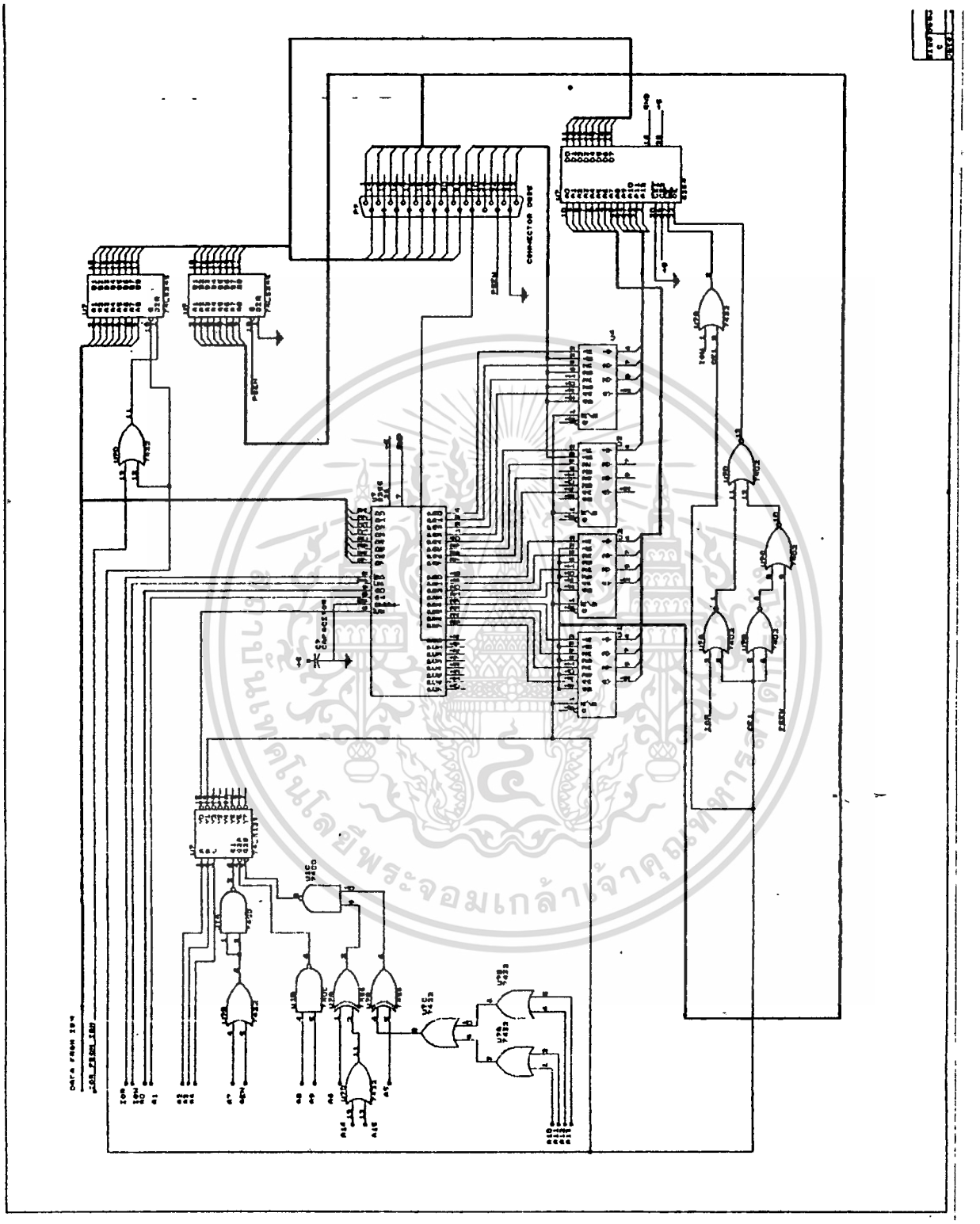
```
    K,X,Y,CONTROL_W    : BYTE;
label LOOP,final;
FUNCTION Revaule(VAR P:mcode_str ):INTEGER;
   var I,J,K  :integer;
       PERMA2 :STRING[1];
   BEGIN
        K:=0;
        FOR I:=1 TO 2 DO
          begin
            perma2 := copy (p,i,1);
            j := integer (perma2);
            j := j div 256 - integer('0');
            if j > 10 then j := j - 7;
            k := k * 16 + j ;
          end;
 Revaule := k;
   END;
BEGIN
     CLRSCR;
     WRITELN('This MCS51-series SEND CODE program  ');
     REPEAT
            J := 13;
            WRITE('Enter filename ');
            READLN(tm);
            IF copy(tm,2,1) = ':' THEN J := 15;
```

```
            I := POS('.',tm);
  UNTIL (LENGTH(tm) < J) AND (I < J - 3);
    ASSIGN(T,tm);
    RESET(T);
loop : readln(t,perma);
    WRITELN(PERMA);
    repeat
        L := 7;
        D := COPY(PERMA,L,1);
        IF D = ' ' THEN  goto loop
        ELSE
      begin
        control_w  := $80 ;
        port[$303] := control_w;
        L := 3;
        M := $301;
        repeat
          D:= COPY(PERMA,L,2);
          K:= REVAULE(D);
          IF L = 3 THEN X:=K
          ELSE
              BEGIN
                Y:= K;
                IF Y > 80 THEN WRITELN ('ERROR - MEMORY LIMIT'
              END;
```

```
    PORT[M] := K;
    M   := M - 1;
    L   := L - 2;
  until L <1;
  L := 7;
  REPEAT
    D := COPY(PERMA,L,2);
    IF L <> 7 THEN
                BEGIN
                  C:=COPY(PERMA,L,2);
                  IF C = '  ' THEN GOTO LOOP
                  ELSE
                      begin
                        Inc(X);
                        IF X = 0 THEN
                  begin
                  Inc (Y)
                  if Y>80 then
                writeln('ERROR-MEMORY LIMIT');
                end;
                        PORT[$301]:= X;
                        PORT[$300]:= Y;
                    end
              END;
    K :=REVAULE(D);
```

```
            PORT[$304] := K;
            L := L + 2;
            UNTIL L >= 13;
            readln (t,perma);
            writeln(perma);
            END;
        UNTIL (PERMA = ' ') OR (EOF(T));
final:close(t);
writeln('**********THIS OPERATE SUCCESSED**********');
writeln('PRESS ENTER TO RETURN');
readln;
END.
```





รูปที่ 3.3 แสดงวงจรของอุปกรณ์พัฒนาโปรแกรม

บทที่ 4

วิจารณ์และสรุป

สรุปการสร้างอุปกรณ์ 2 ชิ้นนี้ เพื่อใช้ในการพัฒนาโปรแกรมและช่วยในรับส่งข้อมูลทางอนุกรม ในการสร้างบัฟเฟอร์ของการรับส่งแบบอนุกรมนั้นขณะนี้ได้ทำเป็นบัฟเฟอร์ของโมเด็มก่อน และในต่อไปอาจจะสร้างขึ้นเพื่อที่จะนำไปใช้กับ เครื่องแฟกซิมิลี หรือ อุปกรณ์ตัวอื่นๆ ที่มีการส่งแบบอนุกรมแต่อาจจะต้องมีการดัดแปลงทางโปรแกรมบ้างเพียงเล็กน้อยเท่านั้น ขณะนี้ที่ประเทศเรายังไม่นิยมที่จะใช้โมเด็ม แต่ในต่างประเทศมีการนิยมใช้โดยแพร่หลายและส่วนใหญ่เป็นไปเพื่อการศึกษา โดยทางมหาวิทยาลัยจะทำเป็นสถานีของการส่งโมเด็มเพื่อทำการส่งข้อมูลของวิทยานิพนธ์หรือปริญญานิพนธ์ของนักศึกษาที่ได้ทำการศึกษาอยู่ที่มหาวิทยาลัยนั้น

4.1 ประโยชน์ของบัฟเฟอร์ของการรับส่งแบบอนุกรม

บัฟเฟอร์ของการรับส่งแบบอนุกรมเป็นอุปกรณ์ที่สามารถช่วยในการที่จะรับหรือส่งข้อมูลทางอนุกรมได้เป็นอย่างมาก คือในการรับโมเด็มนั้นจำเป็นที่จะต้องเปิดเครื่องคอมพิวเตอร์อยู่ตลอดเวลาที่จะมี การติดต่อกับโมเด็มโดยตรง อุปกรณ์ตัวนี้สามารถที่จะช่วยให้ ไม่จำเป็นที่จะต้องเปิดเครื่องคอมพิวเตอร์ทิ้งเอาไว้ เพราะว่าในบางเวลาหรือในบางโอกาสอาจจะมีการส่งข้อมูลผ่านทางโมเด็มมาสู่เครื่องคอมพิวเตอร์ก็เป็นไปได้ ซึ่งอาจจะทำให้ต้องเสียเวลามาเปิดคอมพิวเตอร์หรืออาจจะเป็นเวลาที่ไม่มีใครอยู่ก็เป็นได้ ซึ่งทำให้อาจจะต้องเปิดเครื่องคอมพิวเตอร์ทิ้งเอาไว้ทั้งคืนก็เป็นไปได้ แต่ถ้ามีอุปกรณ์ตัวนี้ก็จะทำหน้าที่แทนเครื่องคอมพิวเตอร์ได้ และสามารถที่จะรับข้อมูลเข้ามาเก็บได้โดยอัตโนมัติ เนื่องจากว่าได้มีการโปรแกรมไว้เพื่อที่จะตรวจเช็คว่าขณะนี้มีสัญญาณโทรศัพท์เข้ามากรือไม่อยู่ตลอดเวลา และถ้ามีสัญญาณโทรศัพท์เข้ามาก็จะทำการรับและตรวจ

ว่าเป็นสัญญาณโมเด็มส่งเข้ามาใช่หรือไม่ ถ้าเป็นสัญญาณโมเด็มเข้ามาก็จะตอบรับทันที และก็จะเก็บข้อมูลที่ได้ถูกส่งเข้ามานั้นไว้ในหน่วยความจำที่ได้เตรียมเอาไว้ ซึ่งมีหน่วยความจำถึง 64 กิโลไบท์ ซึ่งคาดว่าเพียงพอต่อข้อมูลที่ส่งเข้ามาภายในเวลา 1 คืน และก็จะมีไฟแสดงว่าขณะนี้ได้มีข้อมูลมาเก็บเอาไว้มนหน่วยความจำ 1 ชุด ถ้ามีข้อมูลเข้ามาอีกในขณะที่ข้อมูลชุดแรกยังอยู่ ไฟแสดงผลก็จะติดอีกดวงหนึ่งซึ่งเป็ยการแสดงว่ามีข้อมูลมาเก็บไว้ที่หน่วยความจำ จำนวน 2 ชุดด้วยกัน ซึ่งสามารถที่จะนำเข้ามาส่งและเก็บเอาไว้ในหน่วยความจำของเครื่องคอมพิวเตอร์ได้ถ้าต้องการเพียงแต่เรียกโปรแกรมตอบรับบนเครื่องคอมพิวเตอร์และทำการส่งข้อมูลที่มีอยู่นั้นด้วยการกดคีย์บอร์ดที่มีอยู่ข้อมูลก็จะถูกส่งเข้ามาเก็บไว้ในหน่วยความจำของคอมพิวเตอร์และสามารถนำเอามาแสดงผลออกบนจอหรือเครื่องพิมพ์ก็ได้

4.2 <u>วิจารณ์บัฟเฟอร์ของการรับส่งแบบอนุกรม</u>

สิ่งที่ควรจะทำการพัฒนาต่อไปคือ น่าจะทำบัฟเฟอร์ของการส่งแบบอนุกรมตัวนี้ให้สามารถใช้ได้กับ แบตเตอรี่และเครื่องชาร์ทด้วย เนื่องจากว่าถ้าเกิดไฟดับขึ้นมาก็ยังที่จะสามารถเก็บข้อมูลที่ได้ถูกส่งมาไว้แล้วให้สามารถคงอยู่ได้ในช่วงเวลาหนึ่ง

4.3 <u>ประโยชน์ของอุปกรณ์พัฒนาโปรแกรม</u>

ประโยชน์ของอุปกรณ์ช่วยพัฒนาโปรแกรมมีดังนี้ คือ ในการที่จะสร้างอุแกรณ์กรือเครื่องมือขึ้นมาที่จำเป็นที่จะต้องใช้ ไมโครโปรเซสเซอร์ 8031 ผู้สร้างจำเป็นที่จะต้องใช้อีพรอมท์ซึ่งเป็นหน่วยความจำที่สามารถจะล้างหรืออัดโปรแกรมเข้าไปได้ แต่ในขั้นตอนของการล้างกรืออัดอีพรอมท์นั้น จำเป็นที่จะต้องใช้เวลาเป็นอย่างมาก ซึ่งถ้ามีอุปกรณ์พัฒนาโปรแกราตัวนี้แล้วก็สามารถที่จะลดขั้นตอนการล้างและอัดอีพรอมท์ออกไปได้มาก คือ ผู้ผลิตหรือผู้ออกแบบอุปกรณ์

สามารถที่จะใช้อุปกรณ์พัฒนาโปรแกรมนี้เข้ามาใช้แทนอีพรอมท์ได้ เนื่องจากว่าสามารถที่จะส่งโปรแกรมสำกรับไมโครโปรเซสเซอร์ 8031 ผ่านอุปกรณ์ตัวนี้โดยตรงได้เลย โดยไม่จำเป็นที่จะต้องทำการอัดหรือล้างอีพรอมท์อีก เพียงแต่ว่าถ้าผู้ผลิตได้พัฒนาโปรแกรมของตนเองไป จนกระทั่งคิดว่าถูกต้องและสามารถที่จะใช้ถาวรก็เพียงแต่ถอดสายอุปกรณ์พัฒนาโปรแกรมตัวนี้ออกแล้วก็นำอีพรอมท์ไปอัดและนำมาใส่แทนที่ ทำให้สามารถที่จะประหยัดเวลาและค่าใช้จ่ายในการล้างและอัดอีพรอมท์เป็นอันมาก ทั้งยังจะสามารถเพิ่มอายุการใช้งานของอีพรอมท์ที่มีอยู่ด้วย เนื่องจากว่าอีพรอมท์แต่ละตัวนั้นมีอายุการใช้งานจำกัด

4.4 <u>วิจารณ์อุปกรณ์พัฒนาโปรแกรม</u>

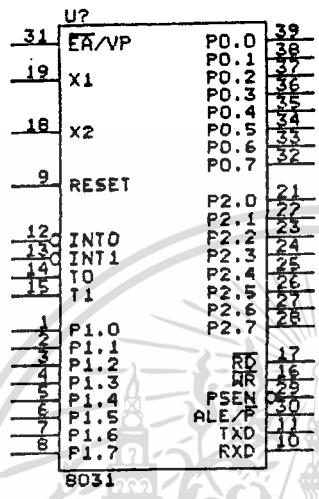
อุปกรณ์ตัวนี้น่าจะสามารถพัฒนาไปเพื่อให้ใช้ได้กับ ไมโครโปรเซสเซอร์ตัวอื่นๆ ได้อีก ก็จะเป็นประโยชน์ต่อการที่จะพัฒนาโปรแกรมของไมโครโปรเซสเซอร์ตัวอื่นด้วย

ภาคผนวก

รูปแบบโครงสร้างภายนอกของไมโครโปรเซสเซอร์ตระกูล MCS-51 เบอร์ 8031 เป็นไปดังรูปต่อไปนี้

รายละเอียดของข้อมูลของขาแต่ละขาของ 8031

- ขาที่ 1 ถึงขาที่ 8 คือขาของพอร์ท 1
- ขาที่ 9 ขา RESET
- ขาที่ 10 ขารับข้อมูลแบบ อนุกรม
- ขาที่ 11 ขาส่งข้อมูลแบบ อนุกรม
- ขาที่ 12 และขาที่ 13 คือ ขาอินเทอรัพท์
- ขาที่ 14 และขาที่ 15 เป็นขาฐานเวลาภายนอก
- ขาที่ 16 คือขา WR
- ขาที่ 17 คือขา RD
- ขาที่ 18 และ 19 คือขาที่ใช้ต่อฐานเวลา
- ขาที่ 20 เป็นขา GROUND
- ขาที่ 21 ถึง 28 เป็นขาของพอร์ท 2
- ขาที่ 29 เป็นขา PSEN
- ขาที่ 30 เป็นขา ALE
- ขาที่ 32 ถึง 39 เป็นขาของพอร์ท 0
- ขาที่ 40 เป็นขาไฟเลี้ยง



รูปโครงสร้างภายนอกของ 8031

กิตติกรรมประกาศ

งานชิ้นนี้ สามารถสำเร็จได้ด้วยดี ข้าพเจ้าขอ ขอบพระคุณ

- ดร. สิทธิชัย โภไคยอุดม อาจารย์ที่ปรึกษา ผู้ให้คำแนะนำ ทางทฤษฎี และการออกแบบ ผู้ให้ความช่วยเหลือ ทางด้านอุปกรณ์ และสถานที่ การทำงาน

- คุณ สุดเจตน์ จันทรังษ์ ผู้ให้คำแนะนำและความช่วยเหลือ ในการทำงาน มา ณ. ที่นี้

หนังสืออ้างอิง

1. INTEL CORPORATION , *Microcontroller Handbook*,1984.

2. Borland International , *TURBO PASCAL 5.0* ,1988.

3. IBM Co,Ltd , *Technical Reference for personal computer XT system* , 1983.