ปีการศึกษา 2531

การใช้ 8031 ในการวัดความเร็วของมอเตอร์

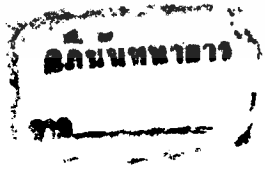
โดย

นาย สุรพงษ์ ศรีกาญจนเพริศ

นาย สุรัตน์ เกวลิน

อาจารย์ที่ปรึกษา

อาจารย์ ธำรงศักดิ์ สุกใส



ปริญญานิพนธ์ ปีการศึกษา 2531

ภาควิชา ระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การใช้ 8031 ในการวัดความเร็วของมอเตอร์

ผู้จัดทำ

นาย สุรพงษ์ ศรีกาญจนเพริศ 28.1280

นาย สุรัตน์ เกวลิน 28.1286

..... อาจารย์ที่ปรึกษา

(อาจารย์ รำรงศักดิ์ สุกใส)

การใช้ 8031 ในการวัดความเร็วของมอเตอร์

นาย สุรพงษ์ ศรีกาญจนเพริศ

นาย สุรัตน์ เกวลิน

อาจารย์ ธำรงศักดิ์ สุกใส อาจารย์ที่ปรึกษา

ปีการศึกษา 2531

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เรียบเรียงขึ้นจากผลงานที่ได้ประยุกต์การใช้งานของ 8031 ไมโครโปรเซสเซอร์ (Microprocessor) มาใช้ในการวัดความเร็วของมอเตอร์ โดยใช้การคำนวณความเร็วจากความถี่ที่สร้างจากเอนโคดเดอร์ (Encoder) ด้วยวิธีการคำนวณแบบใหม่ ซึ่งสามารถวัดความเร็วของมอเตอร์ในระยะเวลาสั้น มีความเที่ยงตรงและความละเอียดสูง

8031 MICROPROCESSOR FOR SPEED MEASUREMENT

Surapong Srikanjanapert

Surat Gevalin

Thamrongsak Suksai Adviser

1989

Abstract

This thesis is an application of 8031 Microprocessor to measure speed regulator for motor drive. The proposed speed measurement system has been developed with a pulse generator ,counter and 8031 Microprocessor which allows a measured variable to be obtained within a short detecting time period with high accuracy and high resolution.

สารบัญ

บทที่ 1

บทนำ

ในระบบควบคุมความเร็วของมอเตอร์ที่มีความเที่ยงตรง และความละเอียดสูง จำเป็นต้องมีระบบการวัดความเร็วของมอเตอร์ และความละเอียดสูงเช่นกัน นอกจากนั้นยังต้องใช้เวลาสั้นในการวัดความเร็ว ในปัจจุบันระบบวัดความเร็วของมอเตอร์ที่ใช้การควบคุมด้วยไมโครคอมพิวเตอร์ ส่วนมากจะใช้วิธีการสร้างพัลส์จากเอนโคดเดอร์ โดยที่ความถี่ของพัลส์ เป็นสัดส่วนกับความเร็วของมอเตอร์ ค่าความถี่นี้จะถูกคำนวณ เพื่อหาค่าความเร็วของมอเตอร์ต่อไป

วิธีการคำนวณความเร็วของมอเตอร์ดั้งเดิมมีอยู่ 2 วิธีคือ

1. วิธีการคำนวณแบบแรก (M-method)

รูปที่ 1.1

ในการคำนวนวิธีนี้จะคำนวณความเร็วของมอเตอร์จากจำนวนของพัลส์ในช่วงเวลาที่กำหนดให้(T_c)ดังรูป1.1 ซึ่งจำนวนของพัลส์ที่นับได้(m_1) จะนำมาคำนวณความเร็วได้ ดังสมการ

$$N = \frac{60.m1}{P.Tc} \quad \text{rpm.}$$

โดย P คือจำนวนของพัลส์ที่ได้จากการหมุนของมอเตอร์ 1 รอบ

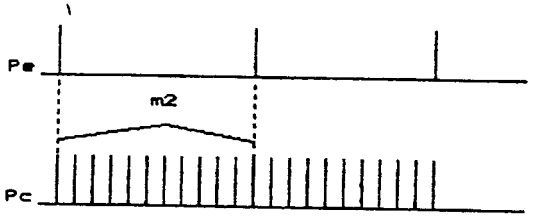
T_c คือช่วงเวลาที่ใช้ในการนับจำนวนของพัลส์

m_1 เป็นจำนวนของพัลส์ที่นับได้ในช่วงเวลา Tc

N คือค่าความเร็วของมอเตอร์ หน่วยเป็น รอบต่อนาที (rpm.)

ในการคำนวณวิธีนี้สามารถใช้ได้ดีในกรณีที่ความเร็วสูง ๆ แต่ถ้ากรณีที่ความเร็วต่ำ ๆ ค่าความละเอียด และความเที่ยงตรงจะมีค่าต่ำ

2. วิธีการคำนวณแบบที่สอง (T-method)



รูปที่ 1.2

ในการคำนวณวิธีนี้ จะคำนวณความเร็วจากระยะเวลาระหว่างพัลส์สองลูกต่อกัน ซึ่งค่าความกว้างของพัลส์จะเป็นส่วนกลับกับความเร็วของมอเตอร์ แต่แทนที่จะนับความกว้างของพัลส์เป็นหน่วยเวลา ก็จะนับจำนวนของพัลส์จากพัลส์ที่ถูกสร้างขึ้นเพื่อเปรียบเทียบ(Pc)ใน 1 ช่วงพัลส์จากเอ็นโคดเดอร์(Pe) ดังรูปที่ 1.2 ซึ่งจำนวนของพัลส์ที่นับได้ m_2 จะนำไปคำนวณเป็นค่าความเร็วของมอเตอร์ จากสมการ

$$N = \frac{60.fc}{P.m2} \quad rpm.$$

โดย P คือจำนวนของพัลส์ที่ได้จากการหมุนของมอเตอร์ 1 รอบ

f_c คือค่าความถี่ของพัลส์ที่สร้างขึ้นเพื่อเปรียบเทียบ(Pc)

m_2 คือจำนวนพัลส์ของสัญญาณที่ใช้เปรียบเทียบ(Pc)ที่วัดภายในหนึ่งช่วงกว้างของสัญญาณจากเอ็นโคดเดอร์(Encoder)

N คือค่าความเร็วของมอเตอร์ หน่วยเป็น รอบต่อนาที (rpm.)

วิธีนี้สามารถใช้ได้ดีที่ความเร็วของมอเตอร์ต่ำ ๆ แต่ในกรณีที่มอเตอร์มีความเร็วสูงการคำนวณจะมีค่าความเที่ยงตรง และความละเอียดต่ำ

เพื่อแก้ไขปัญหาข้อเสียในการคำนวณความเร็วแบบเดิมทั้งสองแบบข้างต้น จึงใช้วิธีการคำนวณแบบใหม่เรียกว่า วิธี M/T โดยจะมีค่าความเที่ยงตรง ความละเอียดสูง และใช้เวลาในการวัดสั้น ซึ่งจะมีวงจรนับสัญญาณสองชุด ชุดแรกนับจำนวนของพัลส์ที่ออกจากเอนโคดเดอร์ในช่วงเวลาที่ทำการวัด ชุดที่สองนับจำนวนของพัลส์ของสัญญาณความถี่คงที่ซึ่งกำหนดในช่วงเวลานั้น แล้วจึงนำจำนวนพัลส์ที่นับได้ทั้งสองชุดมาคำนวณหาความเร็วของมอเตอร์ต่อไป

บทที่ 2

ทฤษฎีระบบวัดความเร็ว

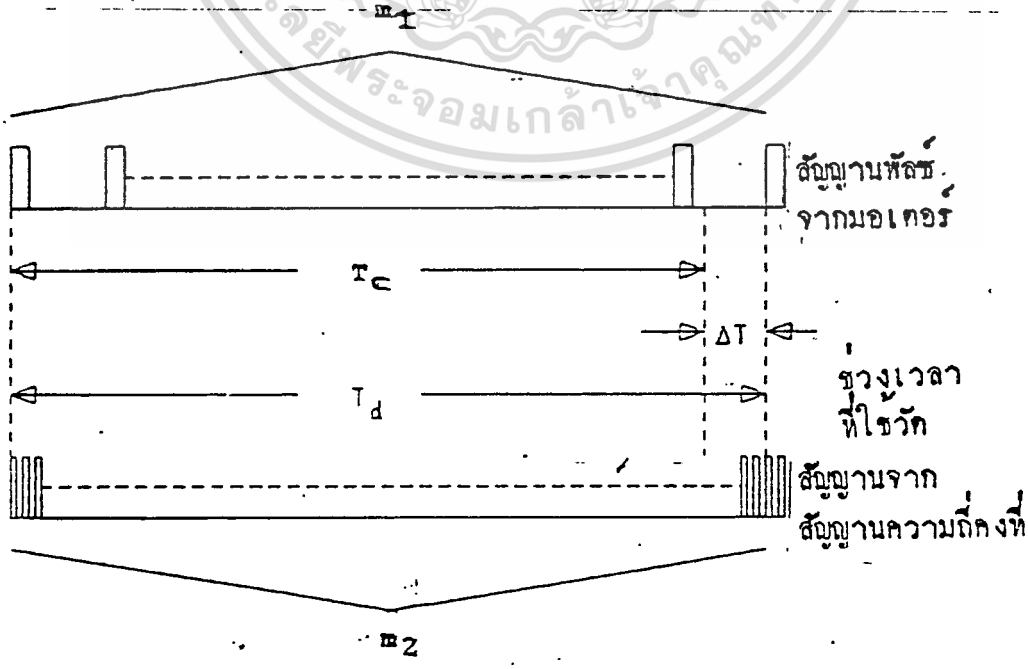
2.1 ทฤษฎี

คุณสมบัติที่สำคัญของเทคนิคการวัดความเร็วของมอเตอร์ โดยใช้ไมโครคอมพิวเตอร์ คือ

1. ความละเอียดสูง (High Resolution) ระบบควบคุมมอเตอร์ที่ต้องการพิสัยการควบคุมกว้าง จึงต้องมีระบบวัดความเร็วที่มีความละเอียดสูง

2. ความเที่ยงตรงสูง (High Accuracy) ระบบวัดความเร็วที่มีความเที่ยงตรงสูงจะมีผลทำให้ ความเที่ยงตรงในการควบคุมที่จุดสภาวะคงตัว (Steady State) ดี

3. เวลาในการวัดสั้น (Short detecting Time) เนื่องจากค่าความเร็วที่วัดได้ไม่ได้เป็นความเร็วชั่วขณะ แต่เป็นความเร็วเฉลี่ยในช่วงเวลาที่วัด นอกจากนี้ช่วงเวลานี้ยังคงเป็นตัวแปรซึ่งกำหนดคุณสมบัติความเร็วในการตอบสนองของระบบควบคุมมอเตอร์ด้วย ดังนั้นถ้าต้องการให้ระบบควบคุมมอเตอร์มีการตอบสนองอย่างรวดเร็ว ช่วงเวลาที่ใช้วัดควรจะมีค่าต่ำ



รูปที่ 2.1 แสดงหลักการเทคนิคการวัดความเร็วแบบวิธี M/T

หลักการของเทคนิคการวัดความเร็วแบบใหม่ แสดงดังรูปที่ 2.1

ช่วงเวลาที่ใช้วัด T_d (วินาที) ถูกกำหนดโดยพัลส์ลูกแรกต่อจากช่วงเวลา T_g (วินาที) ที่กำหนด N_r (รอบต่อนาที) เป็นความเร็วของมอเตอร์ที่วัดได้ และ X (เรเดียน) เป็นระยะทางเชิงมุมที่มอเตอร์เคลื่อนไปในช่วงเวลา T_d ดังนั้นความเร็วของมอเตอร์ที่วัดได้คือ

$$N_r = \frac{60.X}{2.\pi.Td} = \frac{60.X}{2.\pi.(Tc+\Delta T)} \qquad (2.1)$$

m_1 เป็นจำนวนพัลส์ที่ได้จากมอเตอร์ในช่วงเวลา T_d และเมื่อมอเตอร์หยุดหมุนหนึ่งรอบมีจำนวนพัลส์ P พัลส์ นั่นคือ

$$X = \frac{2.\pi.m1}{P} \qquad (2.2)$$

สัญญาณความถี่คงที่มีความถี่ f_c (Hertz) และสัญญาณนี้นับได้ m_2 ในช่วงเวลา T_d

$$T_d = \frac{m2}{fc} \qquad (2.3)$$

แทนค่าสมการ (2.2),(2.3) ลงในสมการที่ (2.1)

$$N_r = \frac{60.fc.m1}{P.m2} \qquad (2.4)$$

2.2 <u>คุณสมบัติ</u>

พิจารณาคำนวณสมการของค่าความละเอียด (Resolution) Q_n (รอบต่อนาที) ค่าความเที่ยงตรง (Accuracy) E (เปอร์เซ็นต์) และช่วงเวลาที่ใช้วัด T_d (วินาที)ของการวัดแบบ วิธี M/T

ค่าความละเอียด Q_n (รอบต่อนาที) สามารถคำนวณได้จากสมการที่ (2.4) เมื่อ m_2 เปลี่ยนไปจาก m_2 เป็น m_2-1

$$Q = \frac{60.fc.m1}{P}\left[\frac{1}{m2-1} - \frac{1}{m2}\right] = \frac{60.fc.m1}{P.m2.(m2-1)} \qquad (2.5)$$

ถ้าค่าความเที่ยงตรงเชิงมุมของพัลส์สองต่อกันมีค่า E_p (เปอร์เซนต์) ดังนั้น เมื่อนับพัลส์จากมอเตอร์ได้ m_1 พัลส์ ค่าความเที่ยงตรง E จะมีค่า

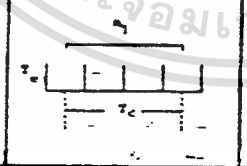
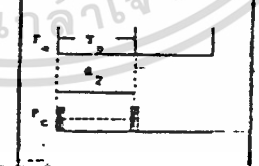
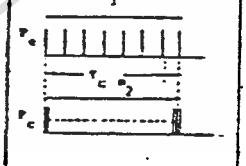
$$E = \frac{Ep}{m1} \qquad (2.6)$$

ช่วงเวลาที่ใช้วัดแสดงได้ดังสมการ

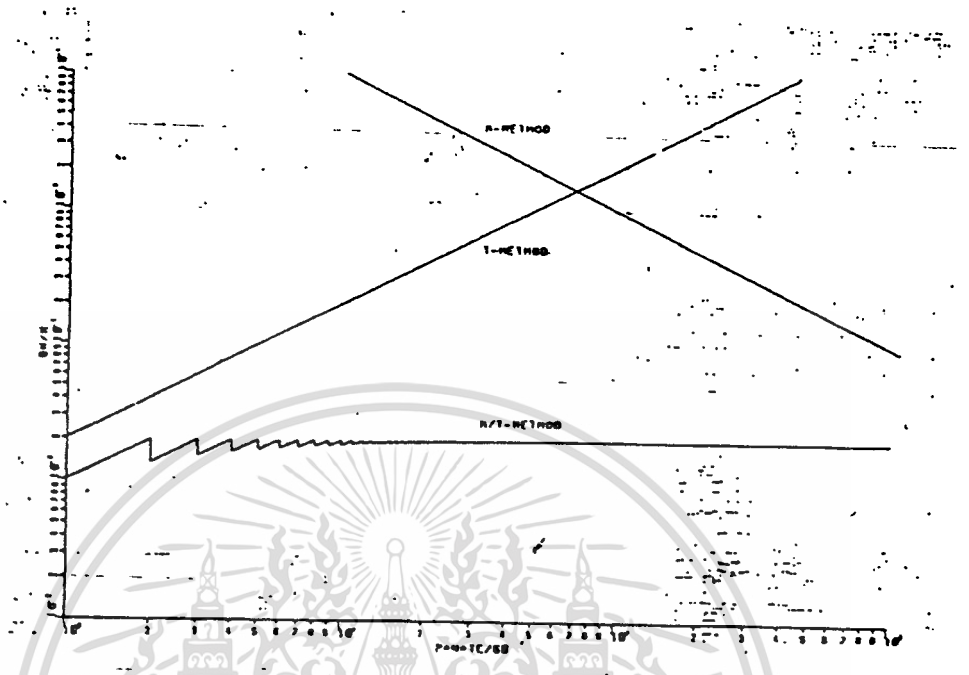
$$T_d = T_m \qquad (T=0)$$

$$T_d = \left\{\left[\frac{P.N.Tc}{60} + 1\right]\sqrt{\frac{60}{P.N}}\right\} \qquad (T\neq 0) \qquad (2.7)$$

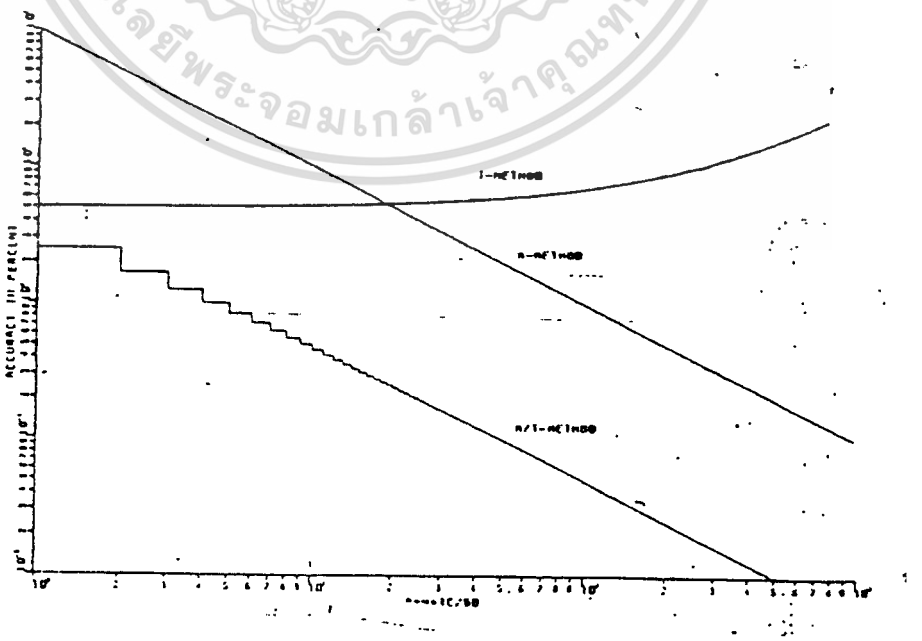
ตารางที่ 2.1 แสดงการเปรียบเทียบหลักการ สมการความเร็วที่วัดได้ สมการค่าความละเอียด สมการค่าความเที่ยงตรง และสมการช่วงเวลาที่ใช้วัดของวิธีการวัดแบบวิธี M, วิธี T, และวิธี M/T รูปที่ 2.2 ,รูปที่ 2.3 และรูปที่ 2.4 แสดงผลการคำนวณเปรียบเทียบระหว่างเทคนิคการวัดทั้งสามแบบ เมื่อ P=300 พัลส์/รอบ f_m = 500kHz T_m =10 mSec

METHOD	M	T	M/T
PRINCIPLE			
MEASURED VALUE $N_f(T=0)$	$60m_1/PT_c$	$60f_c/Pm_2$	$60f_c m_1/Pm_2$
DETECTING TIME T_d/T_c	1	$60/PNT_c$	$60\left[\left[PNT_c/60\right]+1\right]/PNT_c$
RESOLUTION δ_n/N_f	$1/m_1-\left(m_1-1\right)PT_c/60(m_1+1)$	$1/\left[\left(60f_c/PN_f\right)-1\right]$	$1/\left[\left(60f_c m_1/PN_f\right)-1\right]$
ACCURACY (%)	δ_n/N	$\varepsilon_p+\left[100/(m_2-1)\right]$	ε_p/m_1

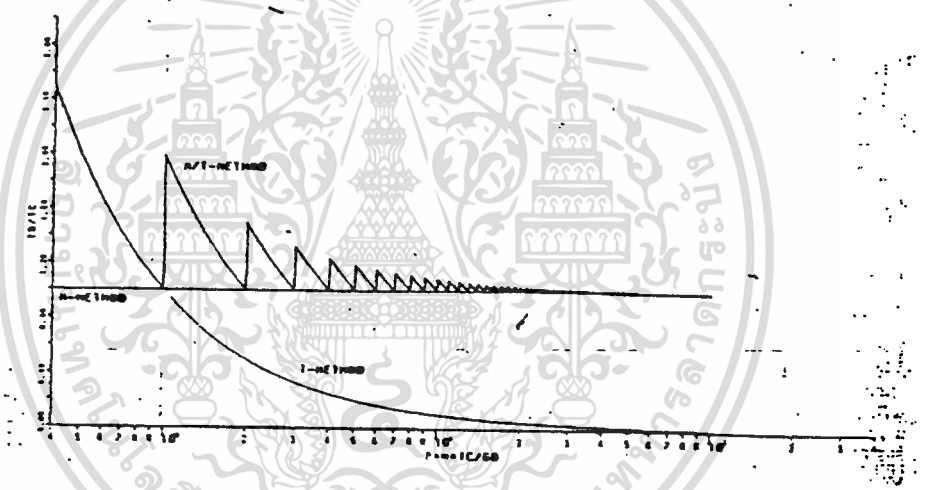
ตารางที่ 2.1



รูปที่ 2.2 แสดงการคำนวณเปรียบเทียบค่าความละเอียดระหว่าง วิธีการวัดแบบ วิธี M วิธี T และวิธี M/T



รูปที่ 2.3 แสดงการคำนวณเปรียบเทียบค่าความเที่ยงตรงระหว่าง วิธีการวัดแบบ วิธี M วิธี T และวิธี M/T



รูปที่ 2.4 แสดงการคำนวณค่าช่วงเวลาที่ใช้เปรียบเทียบระหว่างวิธีการวัดแบบ
วิธี M วิธี T และวิธี M/T

บทที่ 3

8031 ไมโครโปรเซสเซอร์

8031 ไมโครโปรเซสเซอร์ เป็นไมโครโปรเซสเซอร์ตระกูล MCS-51 ซึ่งไม่มีหน่วยความจำชนิดรอม (ROM) อยู่บนชิพเดียวกัน ซึ่งลักษณะหลักของ 8031 ไมโครโปรเซสเซอร์ จะประกอบด้วย

1. ใช้ HMOS เทคโนโลยีสร้าง และทำงานได้ด้วยแหล่งจ่ายไฟขนาด 5 โวลต์แหล่งเดียว

2. ซีพียู (CPU) มีขนาด 8 บิท

3. มีวงจรออสซิลเลเตอร์ (Oscillator) และวงจรนาฬิกาบนชิพ

4. ชุดแบงค์ (BANK) รีจิสเตอร์มีขนาด 4 ชุดด้วยกัน โดยมีชุดละ 8 รีจิสเตอร์ ทำงานเช่นเดียวกับ MCS-48

5. ตัวตั้งเวลาและตัวนับ ขนาด 16 บิท มี 2 ตัว

6. พอร์ทขนานไอโอ (I/O) แบบสองทิศทาง พอร์ทละ 8 บิท จำนวน 16 เส้น และอีก 16 เส้นใช้ในการเข้าถึงทางแอดเดรส (Address) และข้อมูล

7. พอร์ทอนุกรมสามารถที่จะโปรแกรมการรับส่งแบบ FULL DUPLEX ที่ความเร็วสูง

8. หนึ่งวัฏจักรคำสั่งจะใช้เวลา 1 ไมโครวินาที ด้วยการใช้คริสทัล 12 เม็กกะเฮิรตซ์ แต่ในโครงงานชิ้นนี้ได้เลือกใช้คริสทัล 6 เม็กกะเฮิรตซ์ ดังนั้นหนึ่งวัฏจักรคำสั่งจะใช้เวลา 2 ไมโครวินาที

9. แอดเดรสข้อมูลภายนอกได้ 64 กิโลไบต์

10. แอดเดรสโปรแกรมภายนอกได้ 64 กิโลไบต์

11. สามารถกำหนดเลขที่อยู่ข้อมูลขนาดไบต์ หรือบิทได้โดยตรง

12. มีซอฟต์แวร์แฟลก (Flag software) สำหรับผู้ใช้ที่จะกำหนดเองได้ถึง 128 ตำแหน่งบิท

13. โครงสร้างอินเทอร์รัพท์ (Interrupt) ทำได้ 5 แหล่ง พร้อมด้วยการจัดไพรโอริตี้ (Priority) ได้ 2 ระดับ

14. ตัวโปรเซสเซอร์สามารถใช้งานแบบบูลีน (Boolean) ได้ สำหรับการใช้งานควบคุม

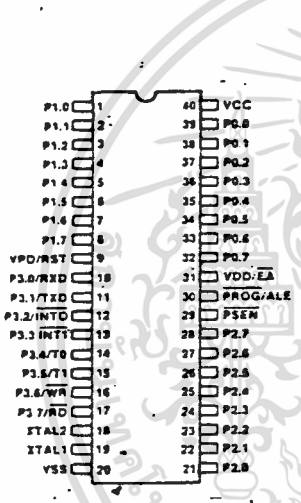
15. มีคำสั่งคูณ และหารทางฮาร์ดแวร์ทำได้ภายใน 4 ไมโครวินาที

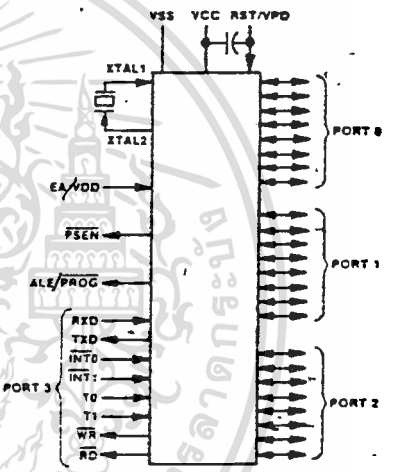
16. ตัวเลขทางคณิตศาสตร์ ใช้ได้ทั้งแบบไบนารี และเดซิมัล

17. การใช้พื้นที่สแต็ก (STACK) สำหรับโปรแกรมย่อยต่าง ๆ ทำได้กว้างขึ้น

18. ชุดคำสั่งของ MCS-51 จะมีมากกว่าชุดคำสั่งของ MCS-48

3.1 การจัดขาลักษณะภายนอกของ 8031





รูปที่ 3.1 (A) ลักษณะขาภายนอกของ 8031 รูปที่ 3.1 (B) สัญญลักษณ์ทางตรรกของ 8031

จากรูปที่ 3.1 เป็นการจัดขาตามลักษณะภายนอกของ 8031 ซึ่งมีรายละเอียดดังนี้ คือ

- ขา Vss ขา 20 เป็นขาสำหรับต่อลงดิน

- ขา Vcc ขา 40 ต้องการแรงดันไฟกระแสตรงขนาด 5 โวลท์ จะเข้าที่ขานี้ และใช้สำหรับการโปรแกรม

- ขา PORT 0 (P0.0 - P0.7/AD0 - AD7) ขา 32-39 เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Biderectional สามารถที่จะรับโหลดทีทีแอลได้ 8 ตัว การเขียนค่า "1" ไปที่พอร์ทนี้ จะเป็นการลอย (FLOAT) ขาของพอร์ทนี้ ทำให้มันทำงานเป็นอินพุท มีสถานะอิมพีแดนซ์สูง ในการให้พอร์ทนี้บริการแบบไอโอ พอร์ท 0 จะทำงานเป็นมัลติเพล็กซ์ ด้วย

สัญญาณแอดเดรสไบต์ต่ำกับบัสข้อมูล สำหรับการใช้งานด้านหน่วยความจำภายนอก ในการใช้งานแบบนี้จะใช้ลักษณะภายในเป็นตัวพูลอัพ พอร์ท 0 ยังใช้งานเป็นตัวส่งข้อมูลออกทางพอร์ทนี้

- ขา PORT (P1.0 - P1.7) ขา 1-8 เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Bidirectional พร้อมด้วยการพูลอัพภายใน ถ้าเป็นพอร์ทเอาท์พุท บัฟเฟอร์สามารถขับโหลดทีทีแอลตระกูลแอลเอสได้ 4 ตัว พอร์ท 1 เมื่อถูกเซียนค่า "1" ด้วยโปรแกรม มันจะมีสถานะสูงด้วยการพูลอัพภายใน การให้สถานะเช่นนี้จะเป็นการ Initial ใช้งานพอร์ทนี้ให้เป็นอินพุท ขณะที่พอร์ท 1 เป็นอินพุทการให้สัญญาณลงต่ำจะเป็นการจ่ายกระแสออกเนื่องจากการพูลอัพภายใน

- ขา PORT 2 (P2.0 - P2.7) ขา 21-28 เป็นพอร์ทไอโอ 8 บิทแบบ Open Drain Bidirectional ด้วยการพูลอัพภายใน พอร์ท 2 ที่ทำหน้าที่เป็นบัฟเฟอร์เอาท์พุทสามารถจ่ายโหลดทีทีแอลตระกูลแอลเอสได้ 4 ตัว พอร์ทจะถูกใช้งานเป็นตัวส่งแอดเดรสไบต์สูงด้วย เมื่อใช้งานร่วมกับหน่วยความจำภายนอกเพื่อให้แอดเดรสได้ถึง 16 บิท ด้วยการใช้งานแบบนี้มันจะมีพูลอัพภายในที่ช่วยให้การส่งค่า "1" ได้ระดับที่แน่นอน

- ขา PORT 3 (P3.0 - P3.7) เป็นพอร์ทไอโอ 8 บิทแบบพูลอัพภายใน นอกจากทำเป็นพอร์ทไอโอที่สามารถรับโหลดทีทีแอลพวกตระกูลแอลเอสได้ 4 ตัวแล้ว ยังใช้งานเป็นพิเศษสำหรับตระกูล MCS-51 คือ

ขาพอร์ท	ขา	การทำงานตามฟังก์ชั่นพิเศษ
P3.0	10	RxD พอร์ทอนุกรมอินพุท
P3.1	11	TxD พอร์ทอนุกรมเอาท์พุท
P3.2	12	INT0 อินเทอร์รัพท์ภายนอกตัวที่ 1
P3.3	13	INT1 อินเทอร์รัพท์ภายนอกตัวที่ 2
P3.4	14	T0 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 0
P3.5	15	T1 สัญญาณกระตุ้นเข้าที่ตัวตั้งเวลาและตัวนับ 1
P3.6	16	WR สัญญาณควบคุมการเขียน
P3.7	17	RD สัญญาณควบคุมการอ่าน

การที่จะให้ทำงานตามฟังก์ชั่นข้างบน จะต้องเริ่มโปรแกรมด้วยการส่งค่า "1" ไปแลทช์ไว้ก่อนที่จะทำงานตามฟังก์ชั่นบน

- ขา RT ขา 9 ต้องคงสถานะสูงเป็นเวลาประมาณอย่างน้อยสองวัฏจักรระหว่างที่ออสซิลเลเตอร์ทำงานขณะที่ต้องการรีเซตทั้งระบบงาน โดยจะต่อรีจิสเตอร์พูลดาวน์ (8.2 กิโลโอห์ม) จากขา RST ปลงดิน และเพื่อให้ตัวชิพรีเซตได้โดยอัตโนมัติขณะเปิดไฟ จะใช้คาปาซิเตอร์ (10 ไมโครฟารัด) ต่อคร่อมระหว่างขา RST กับขา Vcc

- ขา ALE/PROG ขา 30 เป็นขาแอดเดรสแลทช์อีนาเบิ้ลด้วยการส่งพัลส์ออกไปใช้สำหรับแลทช์ค่าแอดเดรสไบท์ต่ำจากพอร์ท 0 ในระหว่างการเข้าถึงข้อมูลจากหน่วยความจำภายนอก ALE จะถูกส่งสัญญาณนาฬิกาออกมา ในอัตราความเร็วคงที่ที่ 1/8 ของความถี่ออสซิลเลเตอร์ตลอดเวลา แม้ว่าจะไม่มีการเข้าถึงข้อมูลจากภายนอก ดังนั้นจึงสามารถที่จะใช้สัญญาณจากขานี้เป็นตัวตั้งเวลาภายนอกหรือเป็นความถี่สัญญาณนาฬิกา แต่อย่างไรก็ตามความถี่สัญญาณนี้จะลดความถี่ลงไปเท่าหนึ่งระหว่างการทำงานแบบการเข้าถึงของหน่วยความจำข้อมูล

- ขา PSEN ขา 29 Program Storage Enable เป็นสโตรบอ่านข้อมูลจากข้อมูลจากโปรแกรมหน่วยความจำภายนอก เมื่อชิพทำงานด้วยโปรแกรมภายนอก ขา PSEN จะไม่มีพัลส์ส่งออก ถ้าชิพทำงานด้วยโปรแกรมหน่วยความจำภายใน

- ขา EA/Vpp ขา 31 มีสถานะสูง ตัวซีพียูในชิพจะทำงานตามโปรแกรมที่อยู่ในหน่วยความจำภายใน (โดยที่โปรแกรมจะต้องไม่ยาวกว่า 4 กิโลไบต์) การทำให้ EA มีสถานะต่ำจะเป็นการควบคุมให้ซีพียูทำงานตามโปรแกรมหน่วยความจำภายนอก ซึ่งขยายโปรแกรมได้ยาวถึง 64 กิโลไบต์ ขา EA จะต้องต่อลงดินเช่นกันแม้ว่าจะไม่มี ROM อยู่ภายในก็ตาม

- ขา XTAL1 ขา 19 ใช้เป็นตัวอินพุทเข้าสู่ตัวออสซิลเลเตอร์ขยายแบบ Invert

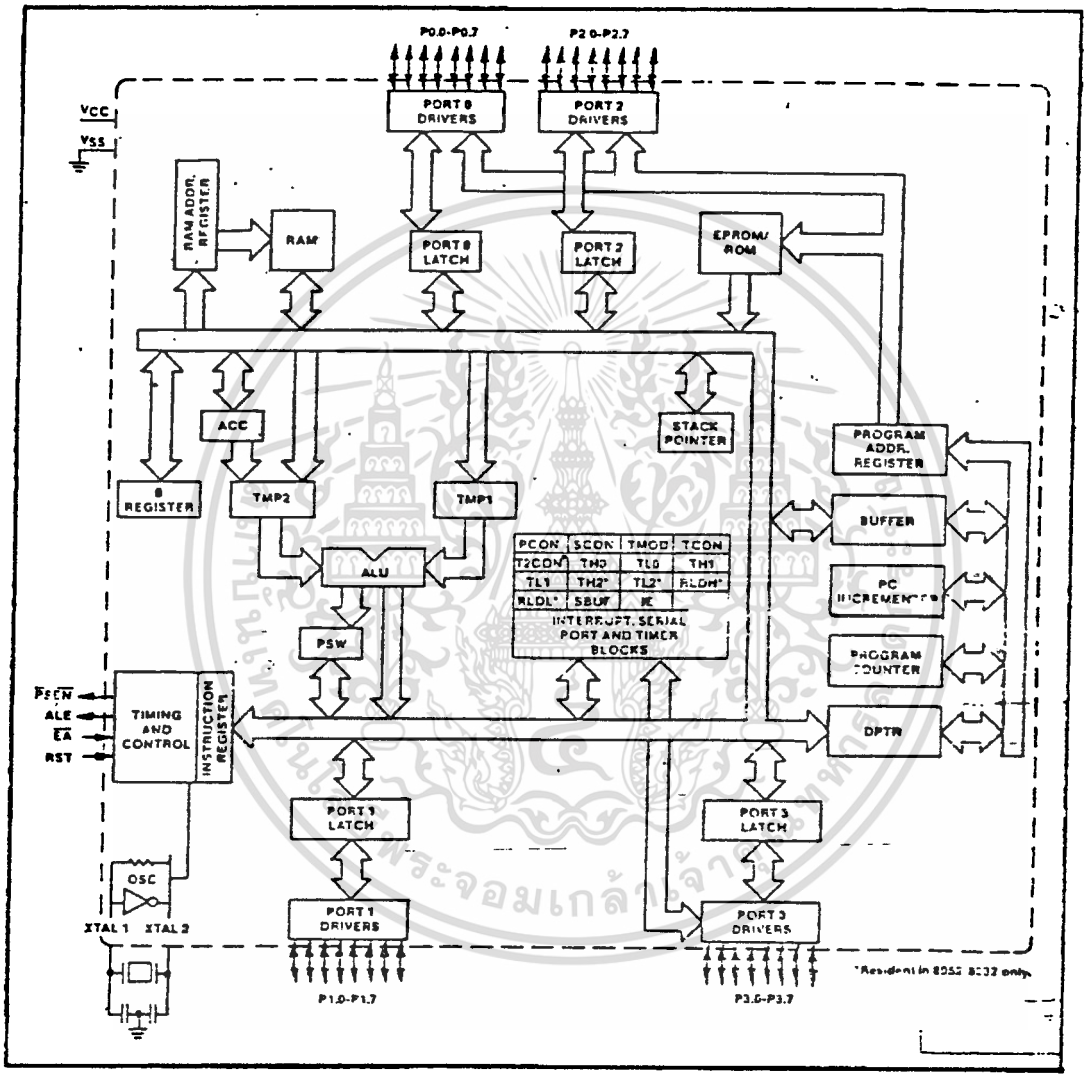
- ขา XTAL2 ขา 18 ใช้เป็นตัวเอาท์พุทจากตัวออสซิลเลเตอร์ขยายแบบ Invert

)

3.2 <u>การจัดการทางสถาปัตยกรรม</u>

รูปที่ 3.2 เป็นบล็อกที่ถูกแบ่งตามลักษณะงานในการจัดการภายในของ MCS-51 โดยชิงเกิ้ลชิพแต่ละตัวของตระกูลนี้จะประกอบด้วยหน่วยศูนย์กลางประมวลผล หน่วยความจำสองชนิดคือ แบบ ROM กับโปรแกรม ROM หรือ EPROM ,พอร์ทเอาท์พุท อินพุท และโหมดรีจิสเตอร์สถานะและข้อมูล ส่วนวงจรตรรกในการ RANDOM ที่จำเป็นสำหรับตัวแปรของฟังก์ชั่นการต่อพ่วง

ส่วนต่าง ๆ ที่กล่าวมานี้จะติดต่อกันด้วยบัสข้อมูลขนาด 8 บิท และจะมีบัฟเฟอร์สำหรับการติดต่อข้อมูลกับภายนอกผ่านพอร์ทไอโอ เมื่อต้องการขยายหน่วยความจำหรือพอร์ทไอโอ



รูปที่ 3.2 สถาปัตยกรรม MCS-51

3.3 หน่วยศูนย์กลางประมวลผลหรือซีพียู

ซีพียูเป็นมันสมอง ของระบบไมโครคอมพิวเตอร์ การอ่านโปรแกรมและทำงานตาม

คำสั่งโปรแกรมจะถูกเก็บไว้ในส่วนนี้ โดยการใช้ส่วนคณิตศาสตร์และตรรกศาสตร์ ทำงานร่วมกับรีจิสเตอร์ A,B,PSW (Program Status Word) และ SP (Stack Pointer) และรีจิสเตอร์ 16 บิทตัวนับโปรแกรม (PC:Program Counter) และตัวชี้ตำแหน่งข้อมูล (DPTR:Data Pointer) ส่วนคณิตศาสตร์และตรรกศาสตร์ (ALU:Arithmetic Logic Unit) เอแอลยูนี้ทำงานในฟังก์ชั่นทางคณิตศาสตร์และตรรกศาสตร์ด้วยตัวแปรต่าง ๆ ขนาด 8 บิท ที่มีลักษณะการทำงานทางคณิตศาสตร์เป็นบวก ลบ คูณ หาร และรวมทั้งทางตรรกศาสตร์เป็น AND OR XOR รวมทั้งการเลื่อนและวนรอบบิท การเคลียร์ค่า และกลับค่า (Complement) ฯลฯ ALU ยังสามารถที่จะตัดสินใจให้กระโดดไปทำคำสั่งโปรแกรมในส่วนอื่น ๆ ตามข้อกำหนดที่ตั้งขึ้น และยังแบ่งรีจิสเตอร์ชั่วคราว ใช้สำหรับให้ข้อมูลเป็นทางผ่านชั่วคราวในการถ่ายเทข้อมูลภายในระบบ คำสั่งอื่นที่ถูกสร้างในการใช้ ALU นี้ ยังมีความสามารถที่จะให้ค่าในรีจิสเตอร์เพิ่มขึ้นครั้งละหนึ่งในลักษณะการบวกด้วยค่าหนึ่ง (Increment) หรือคำนวณเลขที่อยู่ข้อมูลที่จะนำไปเก็บ หรือการลดค่าลงครั้งละหนึ่ง ในลักษณะการลบด้วยค่าหนึ่ง (Decrement) โดยอัตโนมัติ หรือใช้ในการเปรียบเทียบค่าของตัวแปรทั้งสอง

สิ่งที่สำคัญในการทำงานทางสถาปัตยกรรมของ MCS-51 คือ สามารถที่ทำได้ที่ขนาดข้อมูล 8 บิท และ 1 บิท การใช้งานในแต่ละบิทในการเซต เคลียร์ หรือกลับค่า การเคลื่อนย้าย ทดสอบ และใช้ในการคำนวณทางตรรกขนาด 1 บิท ความสามารถเช่นนี้เหมาะสำหรับการใช้งานควบคุมที่มีการคิดและออกแบบทางตรรกด้วย Boolean ของสัญญาณเข้าและออก ซึ่งโดยปกติทำได้ลำบากสำหรับไมโครโปรเซสเซอร์ทั่ว ๆ ไป ด้วยลักษณะงานเช่นนี้จึงได้รื่องานอีกอย่างหนึ่งว่า Boolean Processor

3.3.1 แอกคิวมิวเลเตอร์ (Accumulator : ACC)

MCS-51 ก็เช่นเดียวกับ MCS-48 ใช้ ACC ขนาด 8 บิทเป็นแอกคิวมิวเลเตอร์หลัก โดยคำสั่งส่วนใหญ่จะอ้างถึงตัวรีจิสเตอร์นี้ ถือเป็นค่าตัวตั้งและรับผลลัพธ์จากคำสั่งทางคณิตศาสตร์เช่น บวก ลบ คูณ หาร เข้ามาเก็บไว้ ตัว ACC ยังสามารถใช้เป็นตัวแหล่งกระทำหรือถูกกระทำในการทำงานทางตรรก และใช้ในการถ่ายเทข้อมูลติดต่อกับอุปกรณ์ภายนอก ไอโอและหน่วยความจำภายนอก รวมถึงการตรวจสอบตารางข้อมูล

3.3.2 รีจิสเตอร์ B

เป็นรีจิสเตอร์นิเศษที่ใช้งานในคำสั่งของการคูณและหาร ใช้เป็นตัวคูณ

หรือตัวหาร และเป็นที่เก็บผลลัพธ์ตัวที่สองหลังการคูณและหาร

3.3.3 คำแสดงสถานะโปรแกรม (Program Status Word : PSW)

รีจิสเตอร์ PSW เป็นรีจิสเตอร์ที่แสดงผลที่ได้หลังจากการใช้คำสั่งต่าง ๆ และใช้เป็นตัวเลือกกลุ่มการทำงานของรีจิสเตอร์กลุ่มต่าง ๆ ซึ่งมีรายละเอียดดังรูปที่ 3.3

3.3.4 ตัวชี้สแต็ก (Stack Pointer : SP)

MCS-51 จะรวมเอาสแต็กทางฮาร์ดแวร์ที่ใช้ RAM ภายใน สำหรับการเชื่อมต่อระหว่างโปรแกรมหลัก การผ่านพารามิเตอร์ระหว่างงานในแต่ละส่วนโปรแกรม และการใช้สแต็กเก็บตัวแปรข้อมูลชั่วคราว หรือการเก็บสถานะระหว่างการบริการงานอินเทอร์รัพท์ โดยที่ SP จะมีขนาด 8 บิท จะเพิ่มค่าขึ้นโดยอัตโนมัติก่อนที่ข้อมูลจะนำมาเก็บในหน่วยความจำระหว่างการใช้คำสั่ง PUSH และ CALL และจะลดค่าของ SP ลงหลังจากที่ได้ถ่ายเทข้อมูลไปแล้วในคำสั่ง POP หรือ RETURN โดยทฤษฎีทางสถาปัตยกรรม MCS-51 สามารถใช้สแต็กให้มีเนื้อที่ถึง 128 ไบต์ แต่ในทางปฏิบัติ สำหรับโปรแกรมทั่วไปจะใช้น้อยกว่านี้ SP จะถูกเริ่มตำแหน่งที่ 07H ดังนั้น สแต็กจะเริ่มบรรจุข้อมูลที่ 08H MCS-51 สามารถเปลี่ยนแปลงค่าใน SP ก็จะเปลี่ยนตำแหน่งสแต็กได้ในที่ใด ๆ ของ RAM ภายใน

3.3.5 ตัวชี้ข้อมูล (Data Pointer : DPTR)

DPTR รีจิสเตอร์ขนาด 16 บิทที่ประกอบด้วยไบต์สูง (DPH) และไบต์ต่ำ (DPL) ที่สามารถที่จะเลือกแบ่งรีจิสเตอร์ 8 บิทให้ใช้ได้อย่างอิสระ หรือจะใช้รวมกันทั้ง 16 บิท ในการ Increment หรือ Decrement เพื่อประโยชน์ใช้เป็นฐานเลขที่อยู่รีจิสเตอร์ในการกระโดดโดยอ้อม การใช้คำสั่งเกี่ยวกับตารางข้อมูลและชี้ตำแหน่งเลขที่อยู่ความจำข้อมูลภายนอก

3.3.6 พอร์ท 0 ถึง 3

รีจิสเตอร์ P0,P1,P2 และ P3 ของกลุ่มรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR) จะเป็นตัวแลทช์ค่าของพอร์ท 0,1,2 และ 3 ตามลำดับ

(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	-	P

สัญญลักษณ์	ตำแหน่ง	ข้อกำหนดการทำงาน
CY	PSW7	แฟลกตัวทด จะเซต/เคลียร์ด้วยฮาร์ดแวร์หรือซอฟต์แวร์ระหว่างการใช้คำสั่งทางคณิตศาสตร์ หรือตรรกศาสตร์ที่แน่นอน
AC	PSW6	แฟลกตัวทดของ Auxiliary จะเซต/เคลียร์ด้วยฮาร์ดแวร์ระหว่างการบวกและลบ ที่แสดงผลจากการทดหรือยืมจากบิทที่ 3
F0	PSW5	แฟลก 0 จะเซต/เคลียร์ด้วยซอฟต์แวร์ที่ผู้ใช้กำหนดสถานะแฟลกนี้เอง
RS1	PSW4	รีจิสเตอร์ตัวควบคุมการเลือกแบงค์ด้วยค่า RS1 และ RS0
RS0	PSW3	จะเซต/เคลียร์ด้วยซอฟต์แวร์ เพื่อเลือกกลุ่มรีจิสเตอร์ทำงานแบงค์ โดยปรับค่าใน RS1 และ RS0 ให้อินาเบิ้ลคลุมลักษณะการเลือกแบงค์ต่อไปนี้
OV	PSW2	แฟลก Overflow จะเซต/เคลียร์ด้วยฮาร์ดแวร์ระหว่างการใช้คำสั่งที่แสดงผลถึงการเกิดลักษณะ Overflow
-	PSW1	บิทสำรอง
P	PSW0	แฟลกพาริตี้ จะเซต/เคลียร์ด้วยฮาร์ดแวร์ในแต่ละวัฏจักรคำสั่ง แสดงถึงตัวเลขค่า "1" ในแต่ละบิทของแอกคิวมิวเลเตอร์ เช่น "1" มี 6 ตัว จะเป็นพาริตี้คู่ P บิทจะเท่ากับ 0

RS1	RS0	เลือกแบงค์	ค่าแอดเดรส
0	0	แบงค์ 0	00H - 07H
0	1	แบงค์ 1	08H - 0FH
1	0	แบงค์ 2	10H - 17H
1	1	แบงค์ 3	18H - 1FH

รูปที่ 3.3 รีจิสเตอร์คำแสดงสถานะโปรแกรม PSW

หมายเหตุ อธิบายความหมายฮาร์ดแวร์และซอฟแวร์ในตารางต่าง ๆ ในแต่ละบิทของตัวรีจิสเตอร์ การที่บิทจะเซตหรือเคลียร์นั้น ถ้าเกิดขึ้นจากฮาร์ดแวร์ จะหมายถึงว่าค่าบิทในรีจิสเตอร์จะเกิดเซตตัวเอง เนื่องจากผลของความหมายของการทำงานตามคำสั่งของบิทนั้น เช่น TI จะเซตตัวเองด้วยฮาร์ดแวร์ เมื่อการส่งข้อมูลได้สิ้นสุดถึง STOP บิทแล้ว ช่วยให้เราสามารถตรวจสอบได้ว่าการส่งข้อมูลครั้งละไบต์นั้นสิ้นสุดหรือยัง ถ้ายังจะได้รอต่อไปก่อน หรือมีการคำนวณแล้วผลลัพธ์เกิด Overflow ใน PSW ก็จะเซตตัวเองที่บิท OV ส่วนทางซอฟต์แวร์ หมายถึงว่าเราสามารถที่จะเซตหรือเคลียร์ได้ด้วยการใช้คำสั่งต่าง ๆ ในการเซตหรือเคลียร์ในบิทแต่ละบิทของรีจิสเตอร์ เป็นลักษณะทางซอฟต์แวร์

3.3.7 บัฟเฟอร์ข้อมูลอนุกรม (Serial Data Buffer : SBUF)

บัฟเฟอร์ข้อมูลอนุกรมแบ่งเป็นสองรีจิสเตอร์ ตัวหนึ่งเป็นบัฟเฟอร์ส่ง และอีกตัวเป็นบัฟเฟอร์รับ เมื่อข้อมูลถ่ายเทเข้า SBUF มันจะถ่ายเข้าบัฟเฟอร์ส่งซึ่งเป็นตัวจัดการส่งข้อมูลอนุกรม วิธีการเคลื่อนย้ายเข้า SBUF ขึ้นอยู่กับการเริ่ม (Initial) การส่ง เมื่อข้อมูลย้ายออกจาก SBUF จะเป็นการรับข้อมูลจากบัฟเฟอร์ตัวรับ

3.3.8 รีจิสเตอร์ควบคุม

กลุ่ม SFR ที่เป็น IP, IE, TMOD, TCON, SCON และ PCON จะประกอบด้วยบิทที่ใช้ในการควบคุมและแสดงสถานะของการใช้งานในระบบอินเทอร์รัพท์ ตัวตั้งเวลา/ตัวนับ และพอร์ทอนุกรม

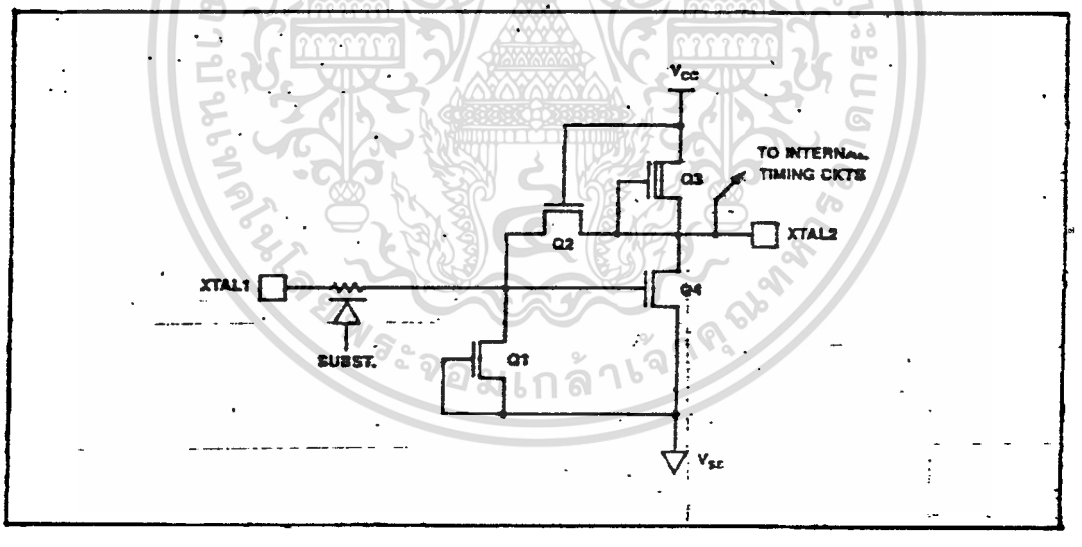
3.4 การจัดการหน่วยความจำ

ตัว MCS-51 จะแยกแอดเดรสสำหรับหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล โปรแกรมหน่วยความจำขยายได้ถึง 64 กิโลไบต์ ความจำข้อมูลมี 128 ไบต์ และอีก 128 ไบต์ใช้สำหรับรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Register : SFR) และหน่วยความจำข้อมูลภายนอกอีก 64 กิโลไบต์

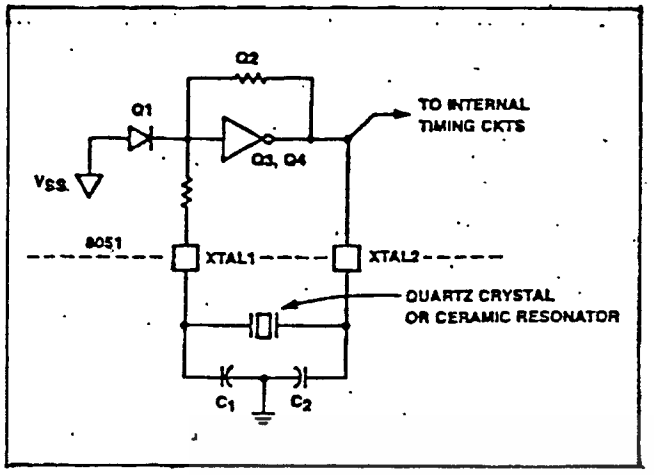
3.5 ออสซิลเลเตอร์และวงจรนาฬิกา

วงจรออสซิลเลเตอร์ที่อยู่ในชิพสำหรับแบบ HMOS จะเป็น Single Linear Inverter ตามรูป 3.4 เพื่อให้ใช้คริสตัลควบคุม เป็นออสซิลเลเตอร์แบบรีแอ็กทีฟบวก ดังรูป

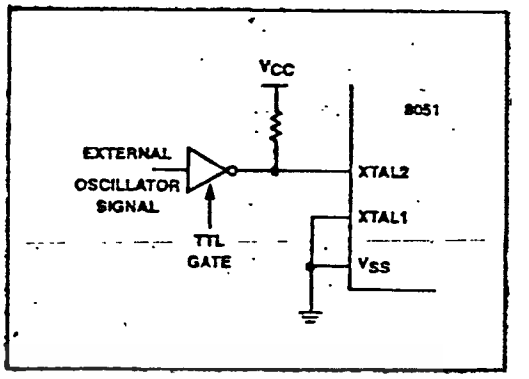
3.5 ในการใช้งานคริสตัลนี้จะทำงานที่โหมด Fundamental เสมือนเป็น Inductance โดยต่อขนานตัวคาปาซิเตอร์ภายนอกกับตัวคริสตัล การกำหนดตัวคริสตัลและค่าคาปาซิเตอร์ C1 และ C2 ในรูป 3.5 ไม่ค่อยวิกฤตนัก อาจจะมีค่าประมาณ 30 PF สำหรับทุกความถี่ของตัวคริสตัลชั้นดี ส่วนการใช้ Ceramic Resonator ค่าคาปาซิเตอร์ที่มาต่อจะมีค่าสูงกว่า โดยมีค่าประมาณ 47 PF การใช้ค่าคาปาซิเตอร์อาจเปลี่ยนแปลงได้ ขึ้นอยู่กับตัว Ceramic Resonator นั้น ๆ การขับตัว HMOS ด้วยสัญญาณนาฬิกาจากภายนอกก็กระทำได้เช่นกัน โดยต่อเข้าที่ขา XTAL2 และต่อลงดินที่ขา XTAL1 ดังรูปที่ 3.6 การใช้พูลอัพต้านทาน ควรจะใช้เพราะระดับสัญญาณที่ XTAL2 ต้องการได้ระดับทีทีแอล



รูปที่ 3.4 วงจรออสซิลเลเตอร์ภายใน MCS-51 แบบ HMOS



รูปที่ 3.5 การใช้วงจรออสซิลเลเตอร์บน HMOS ชิพ



รูปที่ 3.6 การจ่ายสัญญาณนาฬิกาภายนอกในการใช้ HMOS MCS-51

3.6 ช่วงจังหวะเวลาของซีพียู

วัฏจักรแมชชีนประกอบด้วย 6 สถานะ หรือเท่ากับ 12 คาบของออสซิลเลเตอร์ แต่ละสถานะจะแบ่งเป็นเฟส 1 (P1) ครึ่งหนึ่งเป็นช่วงเฟส 1 แอ็กทีฟ และเฟส 2 (P2) เป็นช่วงเฟส 2 แอ็กทีฟ ดังนั้นในแต่ละวัฏจักรแมชชีน จะประกอบด้วย 12 คาบของออสซิลเลเตอร์ เป็นจำนวน S1P1 คือ สถานะที่ 1 เฟสที่ 1 ถึง S6P2 คือ สถานะที่ 6 เฟสที่ 2 โดยปกติการทำงานแบบคณิตศาสตร์และตรรกศาสตร์จะทำในช่วงเฟส 1 และการถ่ายเทข้อมูลภายในระหว่างรีจิสเตอร์จะทำในช่วงเฟส 2

ตามแผนภูมิในรูป 3.7 แสดงถึงช่วงเวลาการเฟทช์ และจะทำงานอ้างถึงลักษณะภายในและเฟส เนื่องจากสัญญาณนาฬิกาภายในผู้ใช้ไม่สามารถที่จะควบคุมการเข้าถึงภายในได้ โดยปกติ ALE จะแอ็กทีฟ 2 ครั้งในแต่ละวัฏจักรแมชชีน และจะเกิดขึ้นระหว่าง S1P2 ถึง S2P1 ครั้งหนึ่ง และระหว่าง S4P2 ถึง S5P1 อีกครั้งหนึ่ง

การทำงานของแต่ละวัฏจักรคำสั่งจะเริ่มที่ S1P2 เมื่ออ็อปโค้ดเก็บเข้าในรีจิสเตอร์คำสั่ง หรืออ่านอ็อปโค้ดเข้ามา ถ้าคำสั่งมีสองไบต์ ไบต์ที่สองจะถูกอ่านในช่วง S4 ภายในวัฏจักรแมชชีนเดียวกัน ก็เป็น 1 ไบต์คำสั่ง และยังคงเฟทช์ที่ S4 แต่ไบต์ที่ถูกอ่าน (ซึ่งควรจะเป็นไบต์ที่สองของคำสั่งเดียวกัน) จะไม่มีผล และตัวนับโปรแกรม (PC) จะยังไม่เพิ่มค่า ไม่ว่ากรณีใด การทำงานจะสมบูรณ์ที่ปลายของ S5P2 ตามรูปที่ 3.7(A) กับ 3.7(B) เป็น

การแสดงแผนภูมิเวลาสำหรับ 1 ไบต์ใน 1 รอบคำสั่ง กับ 2 ไบต์ใน 1 รอบคำสั่ง

คำสั่ง MCS-51 ส่วนใหญ่จะทำงานในช่วงหนึ่งวัฏจักรยกเว้นคำสั่ง MUL (คูณ) DIV (หาร) ที่ใช้มากกว่าสองวัฏจักรในการที่จะทำงานให้สมบูรณ์ได้จะใช้ถึงสี่วัฏจักร ปกติรหัสสองไบต์จะถูกเฟทช์จากหน่วยความจำโปรแกรมช่วงทุกวัฏจักรแมชชีน ยกเว้นคำสั่งพิเศษ คือ MOVX ซึ่งมี 1 ไบต์คำสั่ง แต่จะใช้เวลาสองวัฏจักร ในการเข้าถึงหน่วยความจำข้อมูลภายนอก ระหว่างการทำคำสั่ง MOVX การเฟทช์จะถูกสคิปหรือหายไป ขณะที่หน่วยความจำข้อมูลภายนอกจะถูกแอดเดรสและสโตรบหรือกระตุ้นรับเข้าไปในซีพียู รูป 3.7(C) และ 3.7(D) เป็นการแสดงแผนภูมิเวลาปกติของคำสั่งประเภท 1 ไบต์แต่ใช้ 2 วัฏจักรแมชชีน

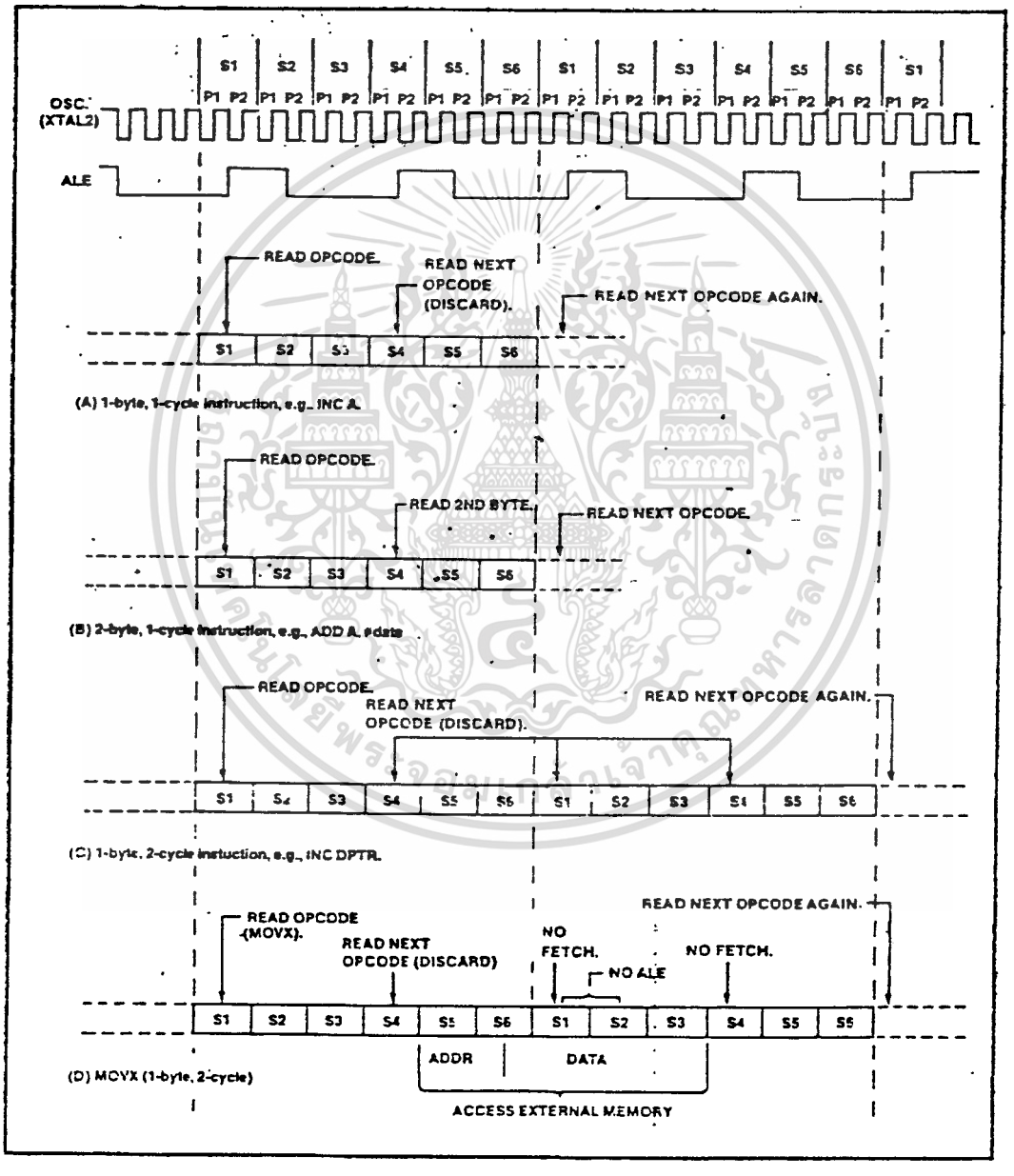
3.7 โครงสร้างพอร์ทและการทำงาน

ทั้งสี่พอร์ทใน MCS-51 เป็นแบบสองทิศทาง แต่ละพอร์ทจะประกอบด้วยแลทช์เป็น P0 ถึง P3 ของ SFR จะมีตัวขับเอาท์พุทและบัฟเฟอร์อินพุท ตัวขับเอาท์พุทของพอร์ท 0 และ 2 และบัฟเฟอร์อินพุทของพอร์ท 0 จะใช้งานในการเข้าถึงหน่วยความจำภายนอก ในการใช้งานที่เอาท์พุทพอร์ท 0 จะเป็นตัวกำหนดไบต์อันดับต่ำของแอดเดรสหน่วยความจำภายนอก ค่าแอดเดรสและค่าข้อมูลจะถูกมัลติเพล็กซ์ด้วยช่วงจังหวะการเฟทช์และการอ่านหรือเขียนข้อมูล และเอาท์พุทพอร์ท 2 จะเป็นตัวกำหนดส่งไบต์สูงของแอดเดรสในการเข้าถึงหน่วยความจำภายนอก

บางขาของตัวขับเอาท์พุทและบัฟเฟอร์อินพุทของพอร์ท 1 และพอร์ท 3 ทั้งหมดสามารถนำไปใช้งานเป็นหลายฟังก์ชั่น (Multifunction) ได้ดังนี้

ขาพอร์ท	การใช้งานตามฟังก์ชั่น
P3.0	RxD (พอร์ทรับข้อมูลอนุกรม)
P3.1	TxD (พอร์ทส่งข้อมูลอนุกรม)
P3.2	INT0 (การใช้อินเทอร์รัพท์ภายนอกตัวที่ 1)
P3.3	INT1 (การใช้อินเทอร์รัพท์ภายนอกตัวที่ 2)
P3.4	T0 (Timer/Counter 0 สัญญาณอินพุทภายนอก)
P3.5	T1 (Timer/Counter 1 สัญญาณอินพุทภายนอก)
P3.6	WR (สโตรบการเขียนหน่วยความจำภายนอก)
P3.7	RD (สโตรบการอ่านหน่วยความจำภายนอก)

ตัวขับเอาท์พุทแลทช์ในการที่จะให้ทำงานตามตารางบน จะต้องเริ่มโปรแกรมด้วย การเซตค่า "1" เก็บในแลทช์ก่อน



รูปที่ 3.7 แสดงถึงช่วงจังหวะการเฟทช์ และการทำงานตามลำดับที่อ้างถึงลักษณะภายในและเฟส

3.8 การเข้าถึงของหน่วยความจำภายนอก

การเข้าถึงของหน่วยความจำภายนอกมี 2 แบบ คือ การเข้าถึงของหน่วยความจำโปรแกรมภายนอก กับของหน่วยความจำข้อมูลภายนอก การเข้าถึงของหน่วยความจำโปรแกรมภายนอกจะใช้ขาสัญญาณ PSEN (Program Store Enable) แอ็กทีฟต่ำ เป็นสโตรบควบคุมการอ่านและการเข้าถึงของหน่วยความจำข้อมูลภายนอก จะใช้ขา RD หรือ WR แอ็กทีฟต่ำเป็นสัญญาณสโตรบควบคุมหน่วยความจำ

การเฟทช์โปรแกรมภายนอกจะใช้ขาแอดเดรส 16 บิทเสมอ ส่วนการเข้าถึงของหน่วยความจำข้อมูลสามารถที่จะใช้กำหนดเลขที่อยู่ได้ทั้ง 16 บิทแอดเดรส เช่น MOVX @DPTR หรือ 8 บิทแอดเดรส เช่น MOVX @R1

เมื่อไรที่ใช้ 16 บิทแอดเดรส ไบต์สูงของค่าแอดเดรสจะส่งออกที่พอร์ท 2 และจะคงสถานะค่านั้นตลอดในช่วงวัฏจักรการอ่านและเขียน ระหว่างช่วงเวลานี้ตัวแลทช์ของพอร์ท 2 ใน SFR จะไม่ต้องประกอบด้วยค่า "1" และค่าข้อมูลใน SFR จะไม่มีการเซต ถ้าช่วงวัฏจักรการใช้หน่วยความจำภายนอกไม่มีการเข้าถึงข้อมูลในวัฏจักรต่อมา ค่าใน SFR ของพอร์ท 2 จะปรากฏค่าเดิมกลับมาใหม่ในวัฏจักรตัวต่อมานี้

ถ้าใช้เป็น 8 บิทแอดเดรส ค่าใน SFR ของพอร์ทจะยังคงค่าเดิมที่ขาพอร์ท 2 ตลอดช่วงวัฏจักรการใช้ความจำภายนอก ซึ่งลักษณะนี้จะเป็นการใช้งานด้านเพจของหน่วยความจำ

ในกรณีใช้แอดเดรสไบต์ต่ำเป็นช่วงเวลามัลติเพล็กซ์กับข้อมูลของพอร์ท 0 ขาสัญญาณแอดเดรส / ข้อมูล จะขับ FET ทั้ง 2 ตัวในพอร์ท 0 เป็นบัฟเฟอร์ส่งข้อมูลออก ดังนั้นในการใช้งานพอร์ท 0 จะไม่มีการรับกระแสเข้า จึงไม่จำเป็นต้องพูลอัพจากภายนอก สัญญาณ ALE (Address Latch Enable) ก็จะใช้เป็นขาควบคุมรับไบต์แอดเดรสเก็บไว้ภายนอก ซึ่งค่าแอดเดรสจะคงที่ที่ช่วงขอบขาลง ALE ดังนั้นในวัฏจักรการเขียนข้อมูลจะถูกเขียนออกไปที่พอร์ท 0 ก่อนที่ WR จะแอ็กทีฟต่ำ ส่วนวัฏจักรการอ่านข้อมูลจะรับเข้ามาที่พอร์ท 0 ก่อนสโตรบการอ่านจะปรากฏเล็กน้อย และระหว่างการเข้าถึงของหน่วยความจำภายนอก ตัวซีพียูจะส่งค่า 0FFH มาเก็บไว้ที่พอร์ท 0 ของ SFR

การใช้หน่วยความจำโปรแกรมภายนอก จะขึ้นอยู่กับสองกรณีคือ

1. เมื่อไรก็ตามที่ EA แอ็กทีฟ หรือ
2. เมื่อไรก็ตามที่ตัวนับโปรแกรม PC ประกอบด้วยตัวเลขที่มีค่า 0FFFH

ในรุ่นที่ไม่มี ROM ในตัว ให้ใช้ค่าแอ็กทีฟต่ำป้อนที่ขา EA เพื่อกำหนดเฟทช์โปรแกรมภายนอกที่มีต่ำกว่า 4 กิโลไบต์ได้

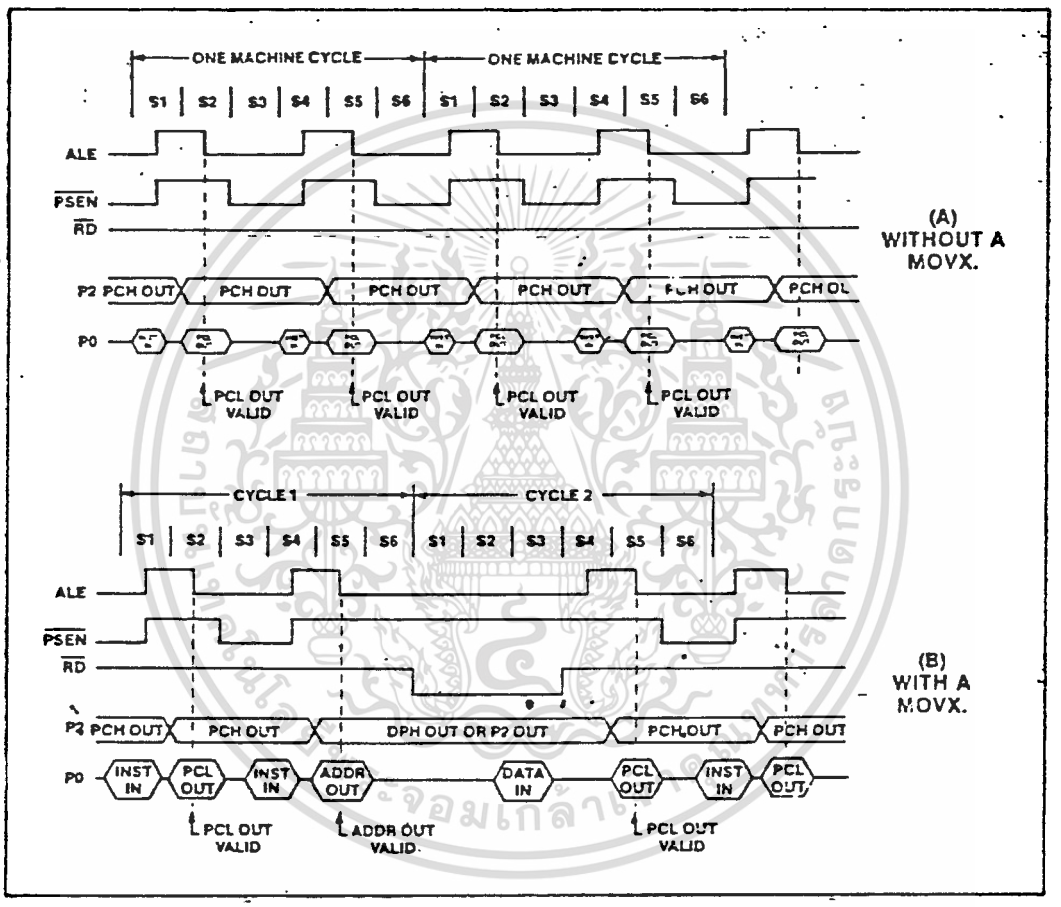
เมื่อซีพียูทำงานด้วยโปรแกรมหน่วยความจำภายนอก ทั้ง 8 บิท ค่าพอร์ท 2 จะส่งค่าแอดเดรสออกมาด้วย ทำให้ไม่สามารถจะใช้งานเป็นไอโอได้ในระหว่างการเฟทช์โปรแกรมภายนอก และระหว่างการเข้าถึงของข้อมูลภายนอก พอร์ท 2 จะส่งทั้ง DPH หรือ SFR ขึ้นอยู่กับการใช้คำสั่งว่าใช้แบบให้คำสั่งส่งเอาท์พุทออกที่ DPH ก็จะใช้คำสั่ง MOVX @DPTR หรือใช้แบบให้ข้อมูลส่งข้อมูลออกที่พอร์ท P2 ของ SFR ก็จะใช้คำสั่ง MOVX @Ri

3.8.1 สัญญาณ PSEN

ใช้เป็นการควบคุมเฟทช์การอ่านโปรแกรมภายนอก PSEN จะไม่แอ็กทีฟถ้ามีการเฟทช์โปรแกรมภายใน เมื่อซีพียูเข้าถึงการใช้โปรแกรมภายนอก PSEN จะแอ็กทีฟ 2 ครั้งในแต่ช่วงวัฏจักรการเฟทช์ ยกเว้นคำสั่ง MOVX ช่วงเวลาของการที่ PSEN เกิดแอ็กทีฟจะไม่เหมือนกับช่วง RD แอ็กทีฟ ช่วงวัฏจักรการอ่านที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอ็กทีฟ และแอ็กทีฟซ้ำลูกที่สองกับสัญญาณการควบคุม RD ที่เกิดพัลส์ต่ำ ประกอบเข้าด้วยกัน ซึ่งจะใช้เวลา 12 คาบสัญญาณนาฬิกา ส่วนช่วงเวลาของ PSEN ที่สมบูรณ์จะรวมเอาช่วงที่ ALE แอ็กทีฟ และแอ็กทีฟซ้ำลูกที่สองกับสัญญาณควบคุม PSEN ประกอบเข้าด้วยกัน ซึ่งจะใช้เวลา 6 คาบสัญญาณนาฬิกา ลักษณะการทำงานตามลำดับของวัฏจักรการอ่านทั้ง 2 แบบ แสดงในรูปที่ 3.8

3.8.2 สัญญาณ ALE

ฟังก์ชั่นหลักของ ALE คือ การใช้งานด้านให้จังหวะที่แน่นอนในการแลทช์เอาไบต์ต่ำของแอดเดรสจาก P0 ไปเก็บไว้ที่ภายนอก เพื่อใช้ในการถอดรหัสแอดเดรสโปรแกรมภายนอก โดยจะให้ ALE ทำงานแอ็กทีฟสองครั้งในทุก ๆ วัฏจักรแมชชีน สัญญาณนี้จะเกิดขึ้นตลอดแม้ว่าจะไม่ได้เฟทช์จากภายนอก ช่วงเวลาเดียวเท่านั้นที่ ALE ไม่เกิดพัลส์ คือ ระหว่างการเจ้าถึงหน่วยความจำภายนอก ตามรูปที่ 3.8(B) จะเห็นว่าพัลส์แรกของ ALE ในวัฏจักรที่สองของคำสั่ง MOVX ขาดหายไป หรือมีเพียงพัลส์เดียวในหนึ่งคำสั่ง ลักษณะพัลส์ที่เกิดขึ้นคงที่ในอัตรา 1/6 ของสัญญาณความถี่ออสซิลเลเตอร์ และสามารถนำมาใช้เป็นสัญญาณนาฬิกาภายนอก หรือกำหนดเวลาได้



รูปที่ 3.8 จังหวะการเข้าถึงหน่วยความจำภายนอก

3.9 ตัวจับเวลา/ตัวนับ (Timer/Counter)

MCS-51 มี 16 บิทตัวจับเวลา/ตัวนับ 2 ตัว คือ Timer/Counter0 และ Timer/Counter1 ซึ่งแต่ละตัวจับเวลา/ตัวนับ (Timer/Counter) สามารถที่จะกำหนดให้ทำงานได้เป็นตัวจับเวลาหรือตัวนับ

3.9.1 ตัว Timer/Counter0 และ Timer/Counter1

แต่ละตัวจะถูกกำหนดให้ทำงานเป็นตัวจับเวลาหรือเป็นตัวนับ ได้ด้วยการเซตหรือเคลียร์ที่ตัวควบคุมในรีจิสเตอร์ TMOD ในกลุ่ม SFR

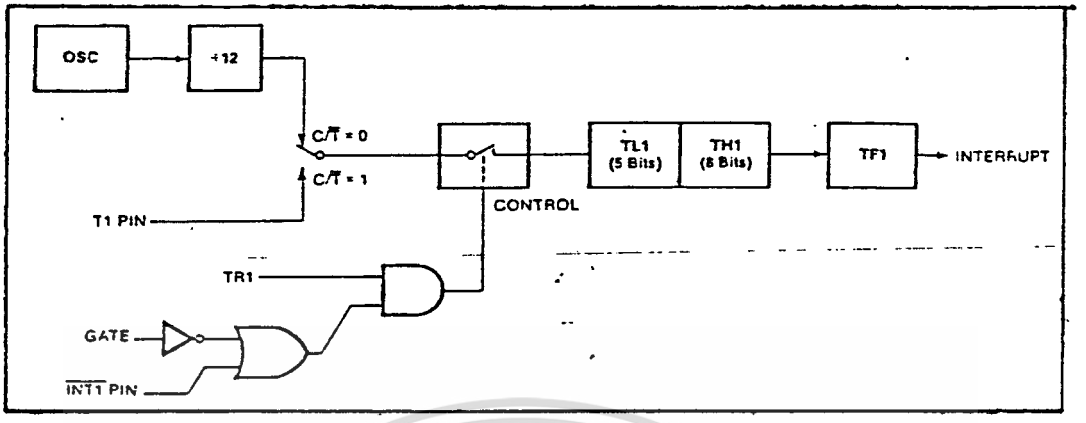
ในฟังก์ชั่นตัวจับเวลา ตัวรีจิสเตอร์จะเพิ่มค่าทุก ๆ ในวัฏจักรแมชชีน ดังนั้น ตัวเลขในรีจิสเตอร์จะเป็นจำนวนของวัฏจักรแมชชีน เนื่องจากแต่ละวัฏจักรแมชชีนประกอบด้วย 12 คาบของออสซิลเลเตอร์ อัตราการนับแต่ละครั้งจะกินเวลาเป็น 1/12 ของความถี่ออสซิลเลเตอร์

ในฟังก์ชั่นตัวนับ รีจิสเตอร์จะเพิ่มค่าทุกครั้งที่มีการเปลี่ยนแปลงสถานะจาก "1" เป็น "0" ที่เข้ามาที่ขา T0 หรือ T1 ในฟังก์ชั่นนี้สัญญาณภายนอกที่เข้ามาจะถูกรับแซมปลิ้ง (Sampling) ระหว่างช่วง S5P2 ของทุกวัฏจักรแมชชีน โดยถ้าแซมปลิ้งสัญญาณเข้าเป็นระดับสูงในวัฏจักรหนึ่ง ดังนั้นถ้าในวัฏจักรตัวต่อมาของสัญญาณเข้าเป็นระดับต่ำ รีจิสเตอร์จะนับเพิ่มหนึ่งค่า โดยที่ค่าใหม่ของตัวนับจะปรากฏที่รีจิสเตอร์ช่วง S3P1 ของวัฏจักร ซึ่งค่าหนึ่งที่รับเข้าไปจะใช้ช่วง 2 วัฏจักรแมชชีน (เท่ากับ 24 คาบ) ในการรับค่าช่วงการเปลี่ยน 1 เป็น 0 ดังนั้น ค่าสูงสุดในการนับจะมีอัตรา 1/24 ของความถี่ออสซิลเลเตอร์ และสัญญาณอินพุทที่นับนั้นจะไม่มีช่วงระยะห่างที่แน่นอนของ Duty Cycle แต่จะถูกนับเมื่อระดับแรงดันที่ถูกแซมปลิ้งในแต่ละครั้งจะต้องมีช่วงคงที่อย่างน้อย 1 วัฏจักรแมชชีนก่อนที่จะเปลี่ยนค่าระดับแรงดันใหม่

ในการเลือกทำงานระหว่างตัวนับเวลากับตัวจับเวลา จะเลือกได้ 4 โหมด คือ โหมด 0, 1 และ 2 เลือกได้ทั้งสองตัวของ Timer/Counter ส่วนโหมด 3 จะทำงานแตกต่างออกไป

โหมด 0

การใช้ตัวจับ/ตัวนับ 0 หรือ 1 ให้อยู่ในโหมด 0 จะทำงานคล้ายกับของ MCS-48 โดยตัวจับเวลาของ MCS-48 มีขนาด 8 บิท มีตัว Prescaler เป็นตัวหาร 12 รูปที่ 3.9 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวนับ1



รูปที่ 3.9 แสดงการทำงานในโหมด 0 ของตัวจับเวลา/ตัวนับ1 จนาด 13 บิท

ในโหมดนี้ รีจิสเตอร์ตัวจับเวลาถูกกำหนดให้มี 13 บิท ด้วยการนับขึ้นเมื่อเป็น "1" หมดทุกบิท จะกลับมาที่ "0" ทุกบิทใหม่ เมื่อกลับเป็น "0" ทุกบิทจะเป็นการเกิด Overflow ไปทดให้แฟลกอินเทอร์รัพท์ TF1 ปรับเป็น "1" การควบคุมให้เริ่มนับตัวอินพุทจะควบคุมด้วยการอินาเบิ้ล TR1 = 1 ; GATE = 0 และขา INT1 = 1 การปรับ GATE = 1 เป็นการตั้งตัวนับให้ถูกควบคุมด้วยสัญญาณจากภายนอกเข้าขา INT1 TR1 จะเป็นบิทควบคุมในรีจิสเตอร์ TMOD ของ SFR

รีจิสเตอร์ตัวนับจะมี 13 บิท ประกอบด้วย TH1 8 บิท และ TL1 อีก 5 บิทอันดับต่ำ ส่วนอีก 3 บิทที่เหลือในอันดับสูงของ TL1 จะไม่ใช้ การปรับแฟลก TR1 ให้ทำงานจะไม่เคลียร์ค่าในรีจิสเตอร์

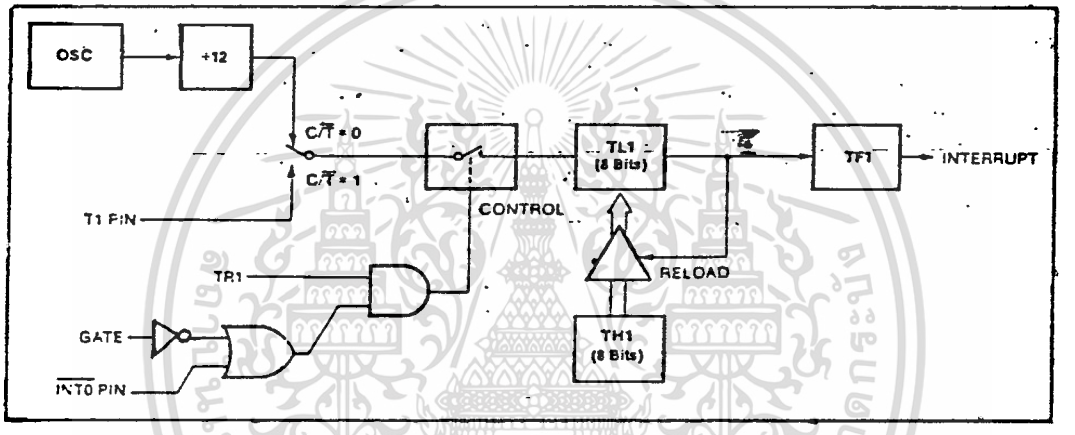
การทำงานในโหมด 0 ในตัวจับเวลา/ตัวนับ จะทำงานเหมือนกับตัวจับเวลา/ตัวนับ1 โดยใช้ TR0 และ INT0 รวมกันควบคุมแทนสัญญาณต่าง ๆ ในรูปที่ 3.9 มีความแตกต่างในการควบคุม คือ บิทของ GATE ทั้งสอง ตัวหนึ่งจะแทนตัวจับเวลา/ตัวนับ1 (TMOD.7) และอีกตัวจะแทนตัวจับเวลา/ตัวนับ0 (TMOD.3)

<u>โหมด 1</u>

โหมด 1 ทำงานเหมือนกับโหมด 0 ต่างกันแต่เฉพาะการใช้รีจิสเตอร์ ตัวจับเวลา/ตัวนับ จะทำงานนับด้วยขนาด 16 บิท

โหมด 2

โหมด 2 มีการทำงานโดยการกำหนดให้ตัวนับ 8 บิท ของ TL1 และจะโหลดใหม่โดยอัตโนมัติทุกครั้งเมื่อมีการ Overflow จาก TL1 ดังรูปที่ 3.10 ไม่เพียงแต่ TF1 จะปรับเป็น "1" แต่ TL1 จะถูกโหลดโดยอัตโนมัติจากค่าที่ตั้งไว้ใน TH1 ซึ่งค่าใน TH1 สามารถจะตั้งค่าได้ด้วยซอฟต์แวร์ และบรรจุเข้าไปใหม่ที่ TL1 ทุกครั้งที่เกิด Overflow THØ และ TFØ จะเป็นตัวร่วมการทำงานในโหมดนี้

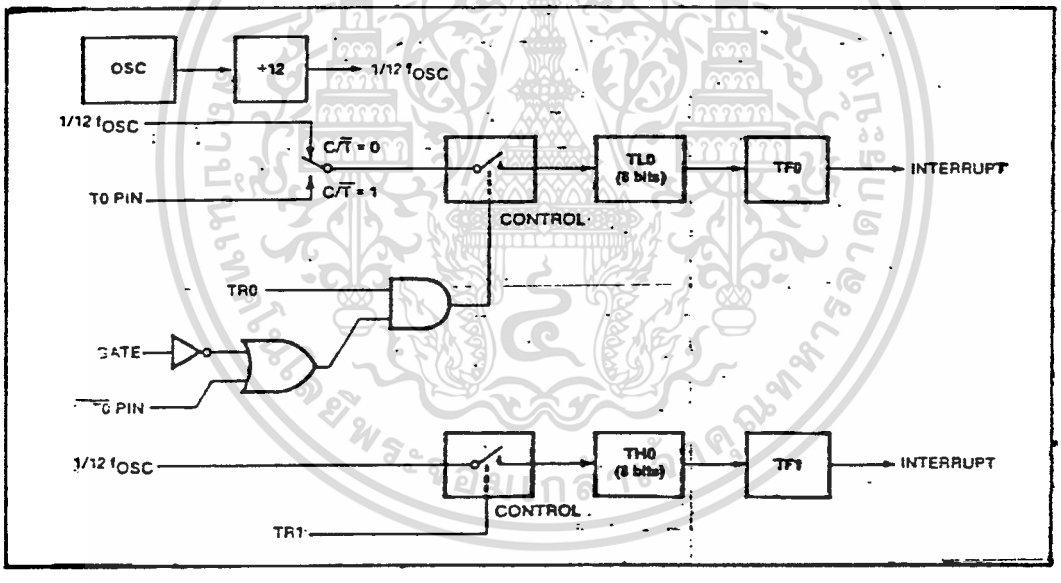


รูปที่ 3.10 ตัวจับเวลา/ตัวนับ1 ทำงานในโหมด 2 แบบโหลดใหม่ 8 บิท

โหมด 3

ใช้ตัวจับเวลา/ตัวนับ1 ในโหมด 3 มีการทำงานเป็นตัวนับ มีผลเช่นเดียวกับการตั้ง TR1 = Ø และใช้ตัวจับเวลา/ตัวนับØ ในโหมด 3 จัดการให้ TLØ และ THØ เป็นตัวนับสองตัวแรกที่แยกออกจากกัน วงจรตรรกควบคุมสำหรับโหมด 3 ที่ใช้ตัวนับเวลา/ตัวจับเวลาØ แสดงในรูปที่ 3.11 TLØ ใช้ตัวจับเวลา/ตัวนับØ เป็นบิทควบคุมของ C/T, GATE, TRØ, INTØ และ TFØ ตัว THØ ถูกล็อคให้ทำงานในฟังก์ชั่นตัวจับเวลา (เป็นตัวนับวัฏจักรแมชชีนได้) และจะใช้งานที่ขา TR1 และ TF1 ของตัวจับเวลา1 เป็นตัวควบคุม ดังนั้นจึงใช้ THØ เป็นตัวจับเวลาถูกควบคุมอินเทอร์รันท์ด้วยตัวจับเวลา1

โดยปกติตัวจับเวลา/ตัวนับ0 ไม่ใช้ในโหมด 3 ถ้าตัวจับเวลา/ตัวนับ1 ยังไม่พร้อมที่จะใช้เป็นตัวกำเนิดอัตราบ๊อด (BAUD) สำหรับพอร์ทอนุกรม โหมด 3 จะถูกกำหนดแบ่งให้ใช้งานเป็นพิเศษสำหรับความต้องการใช้ตัวจับเวลา/ตัวนับสองตัวเป็นอิสระต่อกัน และจะใช้งานสำหรับการทำงานพอร์ทอนุกรม โดยใช้ตัวจับวลา1 เป็นตัวกำเนิดอัตราบ๊อดในโหมด 2 และใช้ตัวจับเวลา0 ในโหมด 3



รูปที่ 3.11 ใช้ตัวจับเวลา/ตัวนับ1 ในโหมด 3 เป็นกลุ่มตัวนับขนาด 8 บิทสองตัว

GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE : ควบคุมเกท เมื่อเซตเป็น "1" จะเป็นอินาเบิ้ล ตัวจับเวลา/ตัวนับเท่านั้น ขณะที่ขา INTx มีสถานะสูง และขาควบคุม TRx ใน TCON จะถูกเซตเป็น "1" เมื่อตัวนับภายในถูกเคลียร์ให้อินาเบิ้ล เมื่อไรก็ตามที่บิทควบคุม TRx ถูกเซตเป็น "1"

C/$\overline{T}$: เลือกการทำงานแบบตัวจับเวลาหรือตัวนับ ถ้าเป็น "0" จะเลือกทำงานเป็นตัวจับเวลา (โดยใช้สัญญาณนาฬิกาภายในเป็นสัญญาณเข้าอ้างอิงถึง) ถ้าเป็น "1" จะเป็นการทำงานแบบตัวนับ และรับสัญญาณเข้าที่ขา Tx

M1	M0	การทำงาน
0	0	ทำงานแบบตัวจับเวลาของ MCS-48 ใช้ TLx เป็นตัวป้อนบิทอีก 5 บิท
0	1	การใช้ตัวจับเวลา/ตัวนับ ขนาด 16 บิท จะใช้ THx และ TLx เป็นตัวนับ ไม่มี Prescaler
1	0	การโหลดขนาด 8 บิท โดยอัตโนมัติที่ตัวนับและตัวจับเวลา โดยใช้ THx เก็บค่าที่ตั้งไว้ และจะถ่ายเข้าที่ TLx ใหม่ทุกครั้งที่เกิด Overflow คือ TLx ถูกนับเป็น "0" หมด
1	1	ตัวจับเวลา0 ทำงาน โดย TL0 และ TH0 เป็นตัวนับแยกกัน

รูปที่ 3.12 TMOD : Timer/Counter Mode Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 TCON.7 ตัวจับเวลา1 แฟลกเป็น "1" เมื่อเกิด Overflow ถูกเซตเป็นหนึ่งด้านฮาร์ดแวร์ทางสัญญาณ เมื่อตัวจับเวลา/ตัวนับเวลา Overflow และจะเคลียร์ตัวเอง เมื่ออินเทอร์รัพท์ไปแล้ว

TR1 TCON.6 ตัวจับเวลา1 เป็นตัวควบคุมบิทให้เริ่มทำงาน จะเซตหรือเคลียร์ด้วยซอฟต์แวร์ที่จะมาทำให้ ตัวจับเวลา/ตัวนับ1 เริ่มหรือหยุดการทำงาน

TF0 TCON.5 ตัวจับเวลา0 แฟลกเป็น "1" เมื่อเกิด Overflow ถูกเซตเป็นหนึ่งด้านฮาร์ดแวร์ทางสัญญาณ เมื่อตัวจับเวลา/ตัวนับเวลา Overflow และจะเคลียร์ตัวเอง เมื่ออินเทอร์รัพท์ไปแล้ว

TR0 TCON.4 ตัวจับเวลา0 เป็นตัวควบคุมบิทให้เริ่มทำงาน จะเซตหรือเคลียร์ด้วยซอฟต์แวร์ที่จะมาทำให้ ตัวจับเวลา/ตัวนับ0 เริ่มหรือหยุดการทำงาน

IE1 TCON.3 อินเทอร์รัพท์1 เป็นแฟลกของสัญญาณ เซตด้วยฮาร์ดแวร์เมื่อสัญญาณของการอินเทอร์รัพท์ปรากฏเข้าที่ขา INT1 และเคลียร์เมื่อการทำงานอินเทอร์รัพท์สิ้นสุด

IT1 TCON.2 อินเทอร์รัพท์1 รูปแบบการควบคุมบิท จะเซตหรือเคลียร์ได้ด้วยซอฟต์แวร์ที่จะเป็นตัวกำหนดให้การกระตุ้นอินเทอร์รัพท์จากภายนอกที่ขอบขาลง หรือระดับแรงต่ำ โดยถ้า IT1 = 1 จะควบคุมอินเทอร์รัพท์ด้วยขอบขาลง และถ้า IT1 = 0 จะควบคุมอินเทอร์รัพท์ด้วยระดับแรงต่ำ

IE0 TCON.1 อินเทอร์รัพท์0 เป็นแฟลกของสัญญาณ เซตด้วยฮาร์ดแวร์เมื่อสัญญาณของการอินเทอร์รัพท์ปรากฏเข้าที่ขา INT0 และเคลียร์เมื่อการทำงานอินเทอร์รัพท์สิ้นสุด

IT0 TCON.0 อินเทอร์รัพท์ต่ำ รูปแบบการควบคุมบิทจะเซตหรือเคลียร์ได้ด้วยซอฟต์แวร์ที่จะเป็นตัวกำหนดให้การกระตุ้นอินเทอร์รัพท์จากภายนอกที่ขอบขาลงหรือระดับแรงต่ำ

รูปที่ 3.13 TCON : Timer/Counter Control Register

3.9.2 Timer/Counter Control และรีจิสเตอร์ Status

การกำหนดโหมดการทำงานและควบคุมฟังก์ชั่นต่าง ๆ ของตัวจับเวลา/ตัวนับ จะควบคุมได้ที่ SFR (Special Function Register) , TMOD และ TCON ด้วยซอฟต์แวร์ โดยที่เมื่อมีคำสั่งเปลี่ยนค่าบิทต่าง ๆ ใน TMOD, TCON ค่าที่ถูกเปลี่ยนก็จะถูกแลทช์เข้าไปที่ SFR และเกิดมีผลตามคำสั่งควบคุมที่ช่วง S1P1 ของวัฏจักรตัวแรกของคำสั่งต่อมา ค่ารีจิสเตอร์ต่าง ๆ ที่ใช้มีแสดงดังรูปที่ 3.8 , 3.13 ตามลำดับ โดยทุกบิทของรีจิสเตอร์เหล่านี้จะถูกเคลียร์ด้วยการรีเซต

3.10 การอินเทอร์รัพท์

โดยทั่วไปความสามารถในการควบคุมการอินเทอร์รัพท์ เป็นการทำงานชนิดหนึ่งของซีพียูที่จะต้องศึกษาถึงความสามารถ และเทคนิคการทำงาน การอินเทอร์รัพท์ของชิงเก็ลชิพมีความสัมพันธ์กันอย่างใกล้ชิด ระหว่างอุปกรณ์ต่อพ่วงกับระบบ การทำงานของอุปกรณ์ต่อพ่วงเหล่านี้มีระบบฮาร์ดแวร์ที่ช่วยให้การส่งสัญญาณ Real Time กับมอนิเตอร์ได้อย่างต่อเนื่อง โดยปราศจากการรบกวนต่อสัญญาณการทำงานของซีพียู ตัวอย่างเช่น ขณะที่มีการรับสัญญาณอนุกรมจากซีอาร์ทีตัวหนึ่ง ก็จะมีการส่งสัญญาณไปยังอุปกรณ์ตัวอื่น และตัวจับเวลา/ตัวนับ ก็จะนับพัลส์การเปลี่ยนแปลงที่เข้ามาอย่างรวดเร็วไปพร้อมกันได้ด้วย ในขณะที่ตัวจับเวลา/ตัวนับอีกตัวก็กำลังวัดความกว้างของพัลส์ที่เข้ามา

ซีพียูตัวนี้ จะรู้ได้อย่างไรว่า เมื่อไรถึงจะมีการรับและส่งสัญญาณอนุกรมซีอาร์ที หรือให้ตัวจับเวลา/ตัวนับ มีการนับจำนวนและวัดความกว้างของพัลส์ว่าจะสิ้นสุดลงเมื่อไร

ตัวโปรแกรม MCS-51 สามารถที่จะเลือกการโปรแกรมได้ 3 วิธีด้วยกัน คือ พิจารณาการโปรแกรมตัวรีจิสเตอร์ TCON และ SCON ที่ประกอบด้วยสถานะบิทที่ถูกเซตทางฮาร์ดแวร์เมื่อตัวจับเวลาตัวหนึ่งเกิด Overflow หรือเมื่อการรับส่งข้อมูลที่พอร์ทอนุกรมสิ้นสุดลง

เทคนิคการโปรแกรมวิธีแรกก็โดยการอ่านสถานะของรีจิสเตอร์ควบคุมเข้าไปยังแอกคิวมิวเลเตอร์ แล้วทดสอบสถานะบิทตามลักษณะการทำงานนั้น ๆ แล้วทำการกระโดดไปยังโปรแกรมย่อยตามผลที่เกิดขึ้นชนิดนั้น ๆ . ลักษณะการทำสอบร่วมกันครั้งละหลายลักษณะงานเช่นนี้เปรียบเสมือนตัวโปรแกรมใช้ระบบไมโครโปรเซสเซอร์หลายตัวควบคุมชิพอุปกรณ์ต่อพ่วงต่าง ๆ ซึ่งผู้โปรแกรมจะต้องทำความเข้าใจอย่างลึกซึ้งถึงระบบ และจังหวะที่จะเกิดในแต่ละงาน และ

การทำสอบแต่ละครั้ง จะใช้คำสั่งไม่น้อยกว่า 3 คำสั่ง

วิธีที่สอง MCS-51 สามารถที่จะทำงานด้วยการกระโดดไปตามสถานะของการควบคุม หรือสถานะบิทของรีจิสเตอร์ควบคุมงาน หรือการรับสัญญาณที่เข้ามาตามขาอินพุทแต่ละบิทด้วยการใช้คำสั่งเพียงคำสั่งเดียว ดังนั้น ลักษณะงานสื่อย่างก็สามารถที่ใช้คำสั่งตรวจสอบได้ภายในสี่คำสั่ง ซึ่งจะใช้เวลาประมาณภายใน 8 ไมโครวินาที

แต่วิธีทั้งสองที่กล่าวมาแล้ว จะต้องใช้ตัวซีพียูมาทำการตรวจสอบบิทสถานะต่าง ๆ อยู่ตลอดเวลา ลองเปรียบตัวซีพียูเหมือนกับตัวผู้จัดการของบริษัทซึ่งจะบริหารงานในบริษัทให้ก้าวหน้าได้อย่างดีนั้น จะต้องมีเวลาทำงานให้กับหน้าที่หลักของตัวเองได้อย่างต่อเนื่อง และใช้เวลาเพียงบางส่วนสำหรับพนักงานที่จะเข้ามาขัดจังหวะ เพื่อขอปรึกษาแก้ไขปัญหาเพียงบางเวลาที่จำเป็นเท่านั้น เช่นเดียวกัน แทนที่จะใช้ซีพียูทำงานในลักษณะที่ออกไปตรวจสอบสถานะการทำงานของอุปกรณ์ต่างรอบข้างต่าง ๆ ที่ต้องการจะใช้บริการ ก็จะใช้อุปกรณ์ต่อพ่วงเป็นฝ่ายร้องขอการบริการเข้ามาที่ซีพียูแทน ซึ่งเมื่อซีพียูถูกร้องขอเข้ามา ก็จะปล่อยงานเดิม และเข้าสู่การบริการที่อุปกรณ์ต่อพ่วงชนิดนั้น ๆ ได้ร้องขอเข้ามาชั่วระยะเวลาหนึ่ง แล้วจึงกลับเข้าทำงานหลักต่อไป เมื่อสิ้นสุดงานบริการนั้นแล้ว ทำให้รู้สึกว่าตัวไมโครโปรเซสเซอร์ทำงานพร้อมกันได้หลายงานในเวลาเดียวกัน

การใช้วิธีที่สามจะเป็นวิธีที่ดีที่สุดในการใช้งานลักษณะนี้ด้วยการอินเทอร์รัพท์ทางฮาร์ดแวร์

8031 มีแหล่งการอินเทอร์รัพท์ 5 แหล่ง โดยแต่ละแหล่งสามารถจะโปรแกรมให้ระดับหนึ่งในสองของระดับไพโอริตี้ (Priority) แหล่งการอินเทอร์รัพท์จะมาจากภายนอก 2 แหล่ง ที่เข้ามาที่ขา INT0 และ INT1 และแต่ละแหล่งจากตัวจับเวลา/ตัวนับ สามตัวในการเกิดแฟลก Overflow

อินเทอร์รัพท์แต่ละแหล่งสามารถที่จะอีนาเบิ้ลและดิสเอเบิ้ลด้วยการเซตและเคลียร์ค่าบิทต่าง ๆ ในริจิสเตอร์ IE ซึ่งในรูปที่ 3.14 จะแสดงรายละเอียดของรีจิสเตอร์ IE ในการเซตค่าบิทต่าง ๆ เพื่อควบคุมการอินเทอร์รัพท์แต่ละแบบ

EA	X	ET2	ES	ET1	EX1	ET0	EX0

สัญญลักษณ์	ตำแหน่งบิท	ฟังก์ชั่น
EA	IE.7	จะดิสเอเบิ้ลการอินเทอร์รัพท์ทั้งหมด ถ้า EA = 0 จะไม่มีการอินเทอร์รัพท์ในการตอบรับ ถ้า EA = 1 สามารถที่จะอินเทอร์รัพท์ได้ โดยแต่ละแหล่งอินเทอร์รัพท์จะมีอิสระในการเซตหรือเคลียร์ให้อินาเบิ้ลแต่ละบิทก่อนได้
-	IE.6	สำรอง
ET2	IE.5	จะอินาเบิ้ลหรือดิสเอเบิ้ลอินเทอร์รัพท์ Overflow ของตัวจับเวลา2 ถ้า ET = 0 การอินเทอร์รัพท์ตัวจับเวลา2 จะดิสเอเบิ้ล
ES	IE.4	จะอินาเบิ้ลหรือดิสเอเบิ้ลอินเทอร์รัพท์พอร์ทอนุกรม ถ้า ES = 0 การอินเทอร์รัพท์พอร์ทอนุกรมจะดิสเอเบิ้ล
ET1	IE.3	จะอินาเบิ้ลหรือดิสเอเบิ้ลการอินเทอร์รัพท์ Overflow ของตัวจับเวลา1 ถ้า ET1 = 0 การอินเทอร์รัพท์ตัวจับเวลา 1 จะดิสเอเบิ้ล
EX1	IE.2	จะอินาเบิ้ลหรือดิสเอเบิ้ลอินเทอร์รัพท์จากภายนอก1 ถ้า EX = 1 การอินเทอร์รัพท์จากภายนอก1 จะดิสเอเบิ้ล
ET0	IE.1	จะอินาเบิ้ลหรือดิสเอเบิ้ลการอินเทอร์รัพท์ Overflow ของตัวจับเวลา0 ถ้า ET = 0 การอินเทอร์รัพท์ตัวจับเวลา0 จะดิสเอเบิ้ล
EX0	IE.0	จะอินาเบิ้ลหรือดิสเอเบิ้ลการอินเทอร์รัพท์จากภายนอก0 ถ้า EX0 = 0 การอินเทอร์รัพท์จากภายนอก0 (INT0) จะดิสเอเบิ้ล

รูปที่ 3.14 รีจิสเตอร์การอินเทอร์รัพท์อินาเบิ้ล (IE)

แต่ละแหล่งอินเทอร์รัพท์สามารถที่จะโปรแกรมให้มีระดับไพโอริตี้สูงหรือต่ำได้ด้วยการเซตหรือเคลียร์ค่าบิทต่าง ๆ ใน IP ของ SFR ตามรูปที่ 3.15 เป็นรายละเอียดของ IP โดยที่ตัวแฟลกอินเทอร์รัพท์ทุกตัวสามารถเซตหรือเคลียร์ ได้ด้วยซอฟต์แวร์ซึ่งจะมีผลเช่นเดียวกับผลที่เกิดขึ้นจากฮาร์ดแวร์

-	-	PT2	PS	PT1	PX1	PT0	PX0

สัญญลักษณ์	ตำแหน่งบิท	ฟังก์ชั่น
-	IP.7	สำรอง
-	IP.6	สำรอง
PT2	IP.5	หาระดับความสำคัญการอินเทอร์รัพท์ตัวจับเวลา2 ถ้า PT2 = 1 เป็นการโปรแกรมให้มีระดับการอินเทอร์รัพท์ความสำคัญสูงกว่า
PS	IP.4	การโปรแกรมให้มีระดับการอินเทอร์รัพท์พอร์ทอนุกรม ถ้า PS = 1 เป็นการโปรแกรมให้มีระดับการอินเทอร์รัพท์ความสำคัญสูงกว่า
PT1	IP.3	หาระดับความสำคัญอินเทอร์รัพท์ตัวจับเวลา1
PX1	IP.2	หาระดับความสำคัญอินเทอร์รัพท์ภายนอก1 (INT1)
PT0	IP.1	หาระดับความสำคัญอินเทอร์รัพท์ตัวจับเวลา0
PX0.	IP.0	หาระดับความสำคัญอินเทอร์รัพท์ภายนอก0 (INT0)

รูปที่ 3.15 รีจิสเตอร์ลำดับความสำคัญการอินเทอร์รัพท์ (IP)

3.10.1 โครงสร้างลำดับความสำคัญการอินเทอร์รัพท์

การอินเทอร์รัพท์ความสำคัญต่ำ สามารถที่ถูกอินเทอร์รัพท์ด้วยตัวเองหรือด้วยการอินเทอร์รัพท์จากความสำคัญสูง แต่ไม่สามารถที่จะถูกอินเทอร์รัพท์จากความสำคัญต่ำตัวอื่นได้ การอินเทอร์รัพท์ความสำคัญสูงไม่สามารถที่จะถูกอินเทอร์รัพท์ได้ด้วยการทำงานตามกฎ

เหล่านี้ ระบบการอินเทอร์รัพท์จะประกอบด้วยตัวที่ไม่สามารถกำหนดแอดเดรสสองตัวคือ "Priority Level Active" กับ "Flip - Flop" ตัวหนึ่งเป็นตัวแสดงถึงการอินเทอร์รัพท์ความสำคัญสูงกำลังได้รับการบริการ และการอินเทอร์รัพท์ตัวอื่นจะถูกกันหมด อีกตัวเป็นการแสดงถึงการอินเทอร์รัพท์ความสำคัญต่ำกำลังได้รับบริการและกันตัวอื่นหมด แต่การอินเทอร์รัพท์ความสำคัญสูงยังคงทำงานต่อ

ในเหตุการณ์ที่มีการร้องขอของระดับความสำคัญเดียวกันถูกรับเข้ามาพร้อมกันการหาลำดับการใช้ร่วมกันก่อนหลังภายในเมื่อการร้องขอได้รับบริการ ดังนั้น ระดับความสำคัญภายในแต่ละอัน จะมีการหาระดับโครงสร้างความสำคัญการอินเทอร์รัพท์ที่มีลำดับการใช้ร่วมกันก่อนหลังดังนี้

แหล่งที่มาการอินเทอร์รัพท์	ลำดับความสำคัญภายใน
การอินเทอร์รัพท์0 จากภายนอก	(สูงสุด) 1
การเกิด Overflow ของตัวจับเวลา/ตัวนับ0	2
การอินเทอร์รัพท์1 จากภายนอก	3
การเกิด Overflow ของตัวจับเวลา/ตัวนับ1	4
พอร์ทอนุกรม	5

แหล่งกำเนิดการอินเทอร์รัพท์ทั้งหมดจะถูกตรวจสอบตามลำดับระหว่างช่วงวัฏจักรแต่ละลูก เช่น ถ้าเกิดที่ S6 ของวัฏจักรใดใด โดยทุกตัวการร้องขอการอินเทอร์รัพท์ถูกตรวจพบและมีการจัดลำดับความสำคัญ การตอบสนองการร้องขอของตัวอินเทอร์รัพท์สูงสุดจะถูกให้ทำงานที่สถานะ 1 ของวัฏจักรตัวต่อมา การตอบสนองการอินเทอร์รัพท์ที่กล่าวมานี้จะไม่ถูกกันออกจากเหตุการณ์ที่เกิดขึ้นใด ๆ ต่อไปนี้

1. การอินเทอร์รัพท์ของระดับความสำคัญที่เท่ากันหรือสูงกว่ากำลังทำงานจะสิ้นสุดแล้ว

2. วัฏจักรแมชชีนที่เกิดระหว่างนี้ จะไม่เป็นวัฏจักรสุดท้ายในการทำงานตามคำ

สั่งที่กำลังทำอยู่ หรือการอินเทอร์รัพท์ที่ร้องขอมาจะยังได้รับการตอบสนองจนกว่าการทำคำสั่งขณะนั้นจะสิ้นสุดสมบูรณ์

3. คำสั่งในขณะนั้นเป็น RETI หรือการเข้าถึงรีจิสเตอร์ IE หรือ IP ของ SFR หรือการร้องขอการอินเทอร์รัพท์จะไม่ได้รับการตอบรับหลังคำสั่ง RETI หรือหลังการอ่านและเขียนเข้ารีจิสเตอร์ IE หรือ IP จะได้รับการตอบรับจนกว่าจะต้องทำคำสั่งอย่างน้อยหนึ่งคำสั่งไปแล้ว

ถ้ากรณีใด ๆ จากข้างบนนี้เกิดขึ้น ผลของการใช้อินเทอร์รัพท์ร่วมกันจะถูกละเลย ถ้าไม่เกิดกรณีใดจากข้างบนนี้ปรากฏ ผลของการใช้อินเทอร์รัพท์ร่วมกันจะทำงานช่วงวัฏจักรแมชชีนลูกต่อมา

3.10.2 การอินเทอร์รัพท์จากภายนอก

แหล่งกำเนิดภายนอกสามารถที่จะถูกโปรแกรมเลือกระดับการแอ็กทีฟหรือช่วงการเปลี่ยนแปลงด้วยการเซตหรือเคลียร์บิทที่ IT1 หรือ IT0 ในรีจิสเตอร์ TCON ถ้า ITx = 0 การอินเทอร์รัพท์ภายนอก x จะถูกกระตุ้นด้วยการกระตุ้นระดับต่ำที่ขา INTx แต่ถ้า ITx = 1 การอินเทอร์รัพท์ภายนอก x เป็นการกระตุ้นใช้ขอบสัญญาณ ในโหมดนี้ ถ้าตัวอย่างสัญญาณของขา INTx แสดงถึงระดับสูงในวัฏจักรลูกหนึ่งและต่ำในวัฏจักรอีกลูกหนึ่ง แฟลกการร้องขออินเทอร์รัพท์ IEx ในรีจิสเตอร์ TCON จะถูกเซต ดังนั้นแฟลกบิทของ IEx จะเป็นการแสดงถึงการร้องขออินเทอร์รัพท์

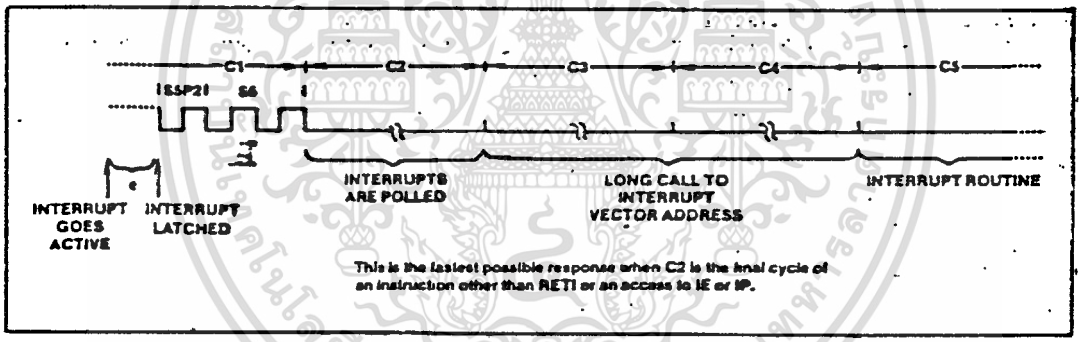
เพราะสัญญาณที่ขาการอินเทอร์รัพท์จะถูกสุ่มตัวอย่างหนึ่งครั้งในแต่ละวัฏจักรแมชชีน สัญญาณที่เข้าจะต้องรักษาระดับสูงหรือต่ำอย่างน้อยภายในช่วง 12 คาบของความถี่ออสซิลเลเตอร์ เพื่อให้มั่นใจในการสุ่มตัวอย่างที่รับเข้าไปได้ค่าแน่นอน ถ้าการอินเทอร์รัพท์ภายนอกถูกแอ็กทีฟเปลี่ยนแปลง แหล่งสัญญาณภายนอกจะต้องรักษาค่าการร้องขอสถานะสูง เป็นเวลาอย่างน้อยหนึ่งลูก และรักษาค่าสถานะต่ำอีกอย่างน้อยเป็นเวลาหนึ่งวัฏจักร เพื่อให้แน่ใจว่าการเปลี่ยนแปลงค่าจะสามารถทำให้แฟลกการร้องขออินเทอร์รัพท์ของ IEx จะถูกเซต ค่าใน IEx จะถูกเคลียร์โดยอัตโนมัติด้วยซีพียู เมื่อโปรแกรมการบริการอินเทอร์รัพท์ถูกเรียกมาใช้

ถ้าการอินเทอร์รัพท์ภายนอกอยู่ในระดับการแอ็กทีฟ แหล่งภายนอกจะต้องเก็บการแอ็กทีฟการร้องขอไว้ จนกว่าสัญญาณการร้องขออินเทอร์รัพท์จะถูกสร้างขึ้นมาเรียบร้อยแล้ว แล้วมันจะต้องกลับมารับแอ็กทีฟการร้องขอใหม่ ก่อนที่การทำงานบริการอินเทอร์รัพท์

เดิมจะสิ้นสุดลง หรือการอินเทอร์รัพท์อีกลูกหนึ่งจะถูกสร้างขึ้นมาใหม่

3.10.3 ช่วงเวลาการตอบสนอง

ระดับของ INT0 และ INT1 จะถูกแลทช์เก็บไว้ในริจิสเตอร์ภายในช่วง S5P2 ของทุก ๆ วัฏจักรแมชชีน ค่าที่เก็บจะยังไม่นำมาใช้ด้วยวงจร จนกว่าจะถึงวัฏจักรแมชชีนลูกใหม่ ถ้าการร้องขอครั้งหนึ่งแอ็กทีฟและข้อแม้ต่าง ๆ ถูกต้อง สำหรับการทำให้มีการตอบรับ ทางฮาร์ดแวร์ก็จะเรียกโปรแกรมย่อยเพื่อตอบรับการบริการการร้องขอของอินเทอร์รัพท์ ในอีกคำสั่งต่อมาจะถูกทำงาน การเรียกโปรแกรมตัวเองจะใช้เวลาสองลูกคลื่น ดังนั้น จะต้องใช้อย่างน้อยสามวัฏจักรแมชชีนช่วงระหว่างการเริ่มการร้องขออินเทอร์รัพท์ภายนอกแอ็กทีฟ จนกระทั่งถึงการเริ่มทำงานคำสั่งแรกของโปรแกรมย่อยการบริการอินเทอร์รัพท์ ดังรูปที่ 3.15 จะแสดงช่วงเวลาการตอบสนองการอินเทอร์รัพท์



รูปที่ 3.15 แสดงช่วงเวลาการตอบสนองการอินเทอร์รัพท์

ช่วงเวลาตอบสนองที่ยาวนานกว่าอาจเกิดขึ้นได้ถ้าการร้องขอถูกบล็อกด้วยข้อแม้ต่าง ๆ ของการจัดลำดับความสำคัญของการอินเทอร์รัพท์สามกรณีที่กล่าวมาแล้ว ถ้าการอินเทอร์รัพท์ที่มีความสำคัญเท่ากัน หรือสูงกว่าได้ทำงานสมบูรณ์ไปแล้ว ช่วงเวลาที่รอทั้งหมดจะขึ้นอยู่กับการใช้เวลาของการอินเทอร์รัพท์นั้น ๆ ถ้ากำลังทำคำสั่งอยู่ที่ยังไม่ถึงวงรอบสุดท้าย ช่วงเวลาที่รอทั้งหมดจะไม่สามารถมีค่าเกินกว่า 3 วัฏจักร เพราะคำสั่งที่ใช้เวลายาวนานที่สุด เช่น MUL และ DIV จะใช้ 4 วัฏจักร

ถ้าอยู่ในช่วงที่กำลังทำคำสั่ง RETI หรือกำลังเข้าถึงริจิสเตอร์ IE หรือ IP เวลา

ที่รอทั้งหมดก็จะไม่สามารถมีค่าเกินกว่า 5 วัฏจักร โดยคำสั่งสูงสุดจะคิดที่หนึ่งวัฏจักรในช่วงที่กำลังทำงานคำสั่งที่เกิดการอินเทอร์รัพท์อยู่ และบวกกับอีกสี่วัฏจักรเพื่อให้สิ้นสุดคำสั่งต่อมา ถ้าคำสั่งต่อมา คือ คำสั่ง MUL หรือ DIV

ดังนั้น ในระบบการอินเทอร์รัพท์ครั้งหนึ่ง ช่วงเวลาที่ตอบสนองจะอยู่ระหว่าง 3 วัฏจักร ถึง 6 วัฏจักรเสมอ

3.11 รีเซต (RESET)

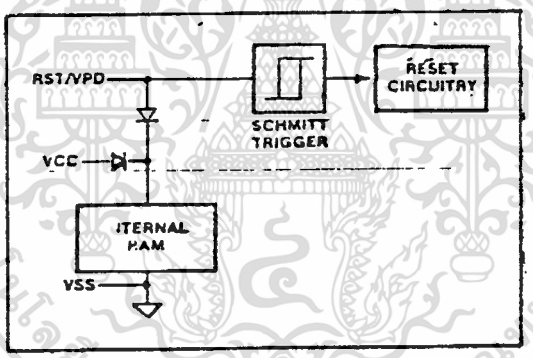
วงจรรีเซตสำหรับ 8031 จะต่อที่ขารีเซตคือขา RST/VPD ดังแสดงในรูปที่ 3.16 วงจร Schmitt trigger ถูกใช้เป็นตัว Input สำหรับขจัดสัญญาณรบกวน Noise และที่ output ของ Schmitt trigger จะถูกสุ่มเก็บเข้าไปด้วยวงจรรีเซต เกิดที่ S5P2 ของทุก ๆ วัฏจักรแมชชีน

8031 จะทำงานได้ด้วยการรีเซตสถานะสูงที่ขา RST/VPD เป็นเวลาอย่างน้อย 2 วัฏจักรแมชชีน ขณะที่ออสซิลเลเตอร์กำลังทำงาน ตัวชิพจะตอบสนองด้วยการทำงานแบบรีเซตภายใน มันจะกำหนดให้ขา ALE และ PSEN เป็นขาอินพุท ซึ่งปกติมันจะเป็น Quasi-Bidirectional การรีเซตภายในจะเริ่มขึ้นระหว่างวัฏจักรที่ 2 ในขณะที่ขา RST มีสถานะสูงและจะซ้ำทุกวัฏจักรแมชชีนจนกว่าขารีเซตมีสถานะต่ำ มันจะทำให้รีจิสเตอร์ภายในมีค่าต่อไปนี้

รีจิสเตอร์	มีค่าข้อมูลเป็น	รีจิสเตอร์	มีค่าข้อมูลเป็น
PC	000H	T2CON	00H
ACC	00H	TH0	00H
B	00H	TL0	00H
PSW	00H	TH1	00H
SP	07H	TL1	00H
DPTR	0000H	TH2	00H
P0-P3	0FFH	TL2	00H

IP	(XX000000)B	RLDH	00H
IE	(0X000000)B	RLDL	00H
TMOD	00H	SCON	00H
TCON	00H	SBUF	Intermediate
PCON	(0XXX0000)B		

* แรมภายในจะไม่มีผลจากการรีเซต เมื่อมีแรงดันไฟจ่ายที่ Vcc ค่าต่าง ๆ ในแรมจะเป็นค่าที่ไม่แน่นอน ถ้าส่วนนั้นไม่ได้กลับมาจากการใช้งานของโหมดลดพลังงาน (Reduced Power Mode)



รูปที่ 3.16 วงจรการจัดการรีเซตของ 8031

บทที่ 4

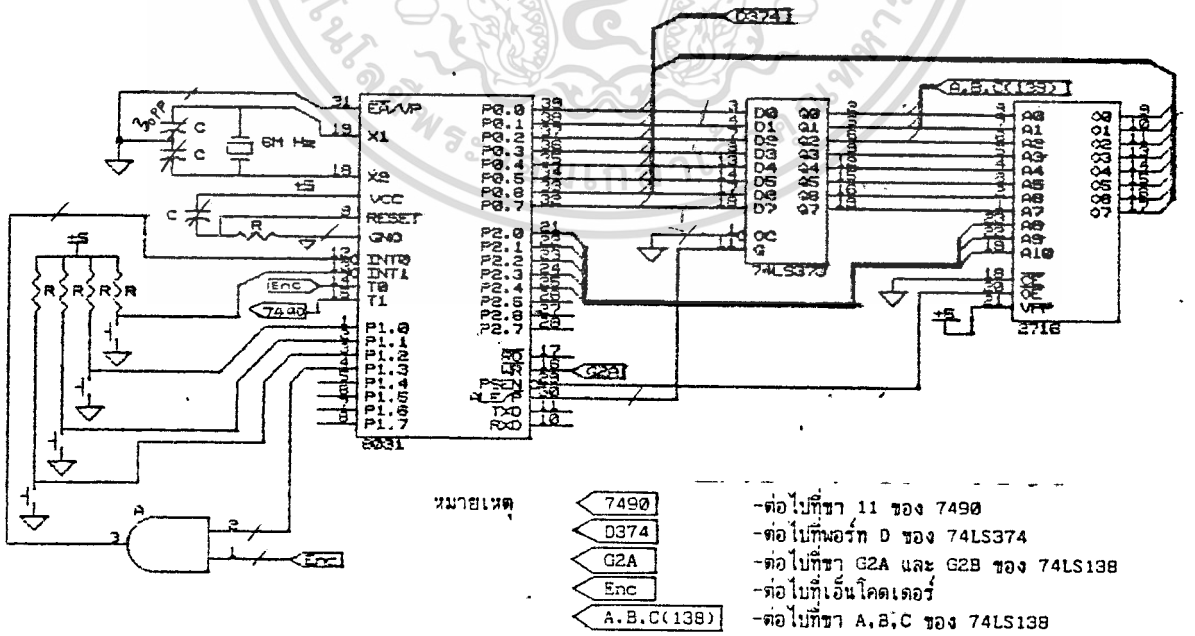
การออกแบบและสร้างวงจร

จากหลักการการวัดความเร็วของมอเตอร์ในการวัดแบบเอ็ม-ที ดังรูป 2.1 จะต้องทำการนับสัญญาณ 2 สัญญาณ คือสัญญาณจากเอ็นโคดเดอร์ และสัญญาณความถี่ที่นำมาเปรียบเทียบ โดยเริ่มนับที่พัลส์พัลส์แรกของสัญญาณทั้งสองพร้อมกัน และหยุดนับที่พัลส์สุดท้ายพร้อมกัน จากนั้นนำค่าจำนวนพัลส์ของสัญญาณทั้งสองมาคำนวณเป็นค่าความเร็วมอเตอร์ต่อไป

ดังนั้นในการออกแบบและสร้างวงจร สามารถแบ่งเป็นส่วน ๆ ได้ดังนี้

ส่วนประมวลผล (MAIN BOARD)

ส่วนประมวลผล ประกอบด้วย 8031 พร้อมทั้งส่วนเก็บความจำของโปรแกรมภายนอก เป็นหน่วยความจำชนิด EPROM ขนาด 2 Kbyte เบอร์ 2716 โดยพอร์ท 0 ของ 8031 เป็นทั้งพอร์ทส่งแอดเดรสของข้อมูล รวมทั้งเป็นพอร์ทที่รับและส่งข้อมูลด้วย จึงต้องใช้ 74LS373 เป็นตัวแลทช์แอดเดรสของข้อมูล ในขณะที่ 8031 กำลังส่งและรับข้อมูล ดังรูป 4.1



รูปที่ 4.1 แสดงส่วนประกอบของส่วนประมวลผล

สำหรับพอร์ทอื่น ๆ ของ 8031 ที่กำหนดในการใช้งานมีดังนี้

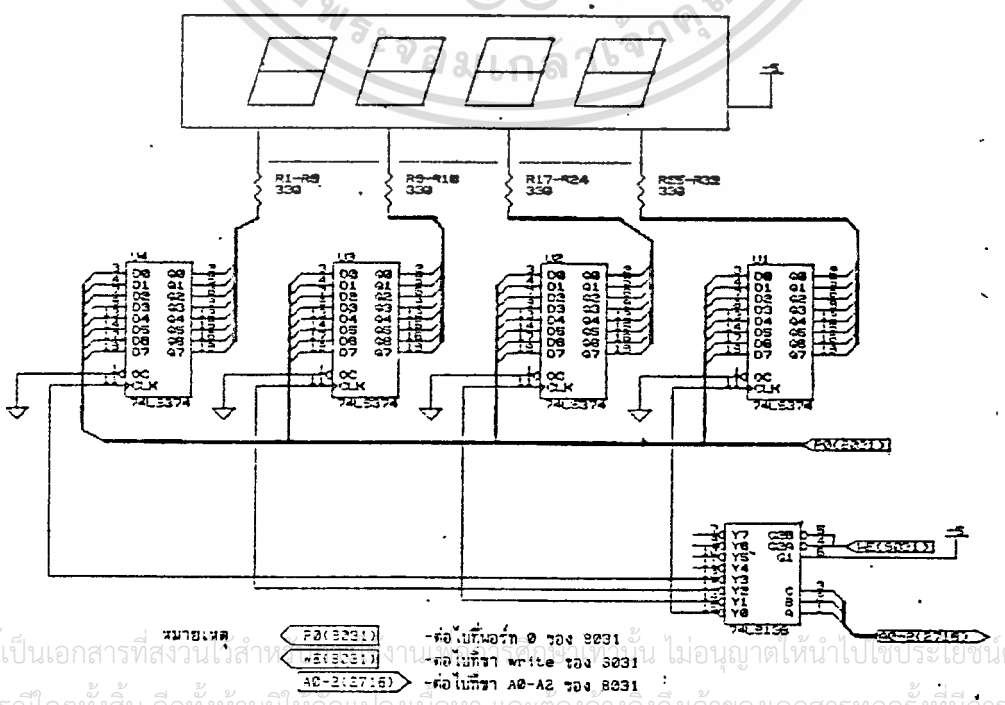
- พอร์ท 1.0, 1.1, 1.2 เป็นส่วนที่ใช้ป้อนค่า P (จำนวนสัญญาณที่ได้จากการหมุนของมอเตอร์ครบหนึ่งรอบ) ซึ่งค่า P นี้ ผู้ใช้จะต้องป้อนให้กับเครื่องวัดความเร็วก่อนที่จะทำการวัด

- พอร์ท 1.3 เป็นส่วนที่ 8031 จะใช้เป็นส่วนที่ส่งสัญญาณเริ่มต้นและหยุดนับเพื่อไปรอทำการ AND กับสัญญาณจากเอ็นโคดเดอร์ และสัญญาณที่ได้จากการ AND แล้วจะถูกป้อนเข้าสู่ขาอินเทอร์รัพท์ 0 ของ 8031 เพื่อให้ 8031 จะได้สั่งให้ตัวนับภายในทำการเริ่มต้นและหยุดนับพร้อมกับสัญญาณจากเอ็นโคดเดอร์ได้พอดี

- ส่วนขาของอินเทอร์รัพท์ 1 เป็นส่วนที่ให้ผู้ใช้ทำการกดสวิทช์ที่ต่อเข้าไปเพื่อให้เริ่มทำการนับค่าความเร็วของมอเตอร์ ซึ่งในขณะที่ทำการวัดความเร็วของมอเตอร์อยู่สามารถที่จะเปลี่ยนค่า P ใหม่ได้ โดยการกดปุ่มอินเทอร์รัพท์ 1 เสียก่อนแล้วจึงป้อนค่า P หลังจากนั้นถ้าทำการกดอินเทอร์รัพท์ 1 ก็จะเป็นการเริ่มต้นนับใหม่อีกครั้งหนึ่ง

<u>ส่วนแสดงผล</u> (DISPLAY BOARD)

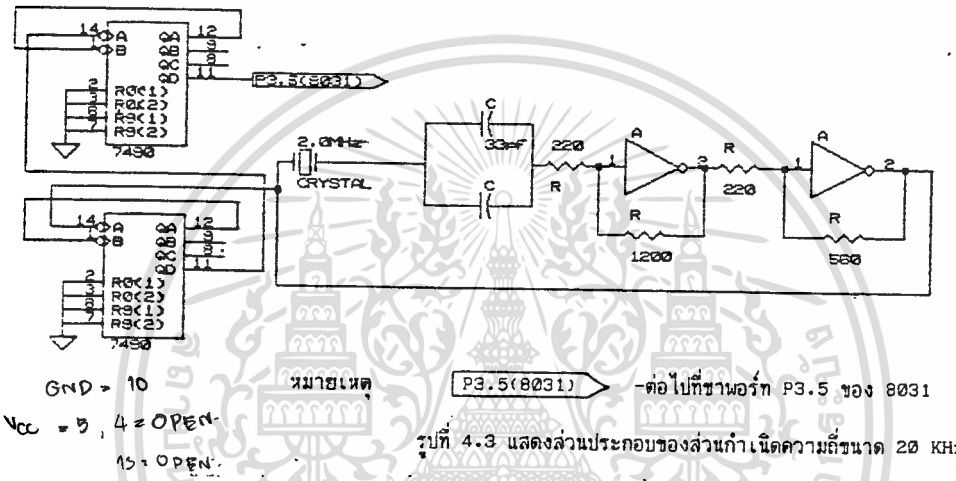
ส่วนแสดงผล ประกอบด้วย 7-segment ชนิด common anode จำนวน 4 ตัว และใช้ 74LS374 จำนวน 4 ตัว ทำหน้าที่แลทช์ข้อมูล เพื่อทำให้การแสดงผลของ 7-segment สามารถแสดงผลพร้อมกันทั้ง 4 ตำแหน่ง ดังรูป 4.2



รูปที่ 4.2 แสดงส่วนประกอบของส่วนแสดงผล

ส่วนกำเนิดความถี่

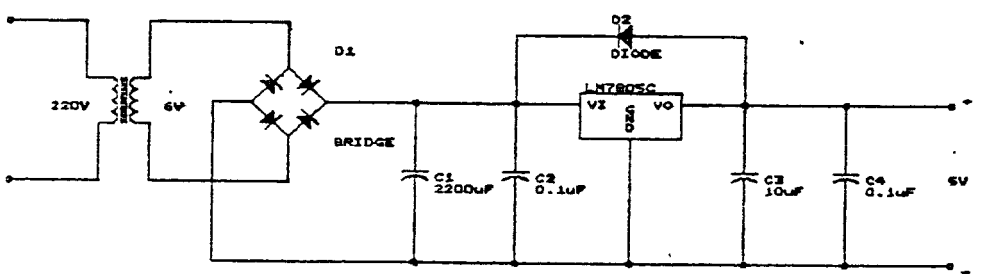
ส่วนกำเนิดความถี่ เป็นส่วนที่กำเนิดความถี่ขนาด 20 KHz เพื่อนำมาเป็นสัญญาณเปรียบเทียบ ประกอบด้วย 74LS00 พร้อมทั้งคริสตัลขนาด 2 MHz เป็นวงจรกำเนิดความถี่ขนาด 2 MHz ซึ่งต้องทำการหารความถี่ให้เหลือเพียง 20 KHz จึงต้องมีวงจรหารความถี่เป็นวงจรหารสิบ 2 ชุด โดยเราใช้ 74LS90 จำนวน 2 ตัว ดังรูป 4.3



รูปที่ 4.3 แสดงส่วนประกอบของส่วนกำเนิดความถี่ขนาด 20 KHz

ส่วนจ่ายพลังงาน

ส่วนจ่ายพลังงาน เป็นส่วนที่แปลงไฟฟ้ากระแสสลับขนาด 220 โวลท์ เป็นไฟฟ้ากระแสตรงขนาด 5 โวลท์ เพื่อจ่ายให้กับส่วนอื่น ๆ ของวงจร ดังรูป 4.4



รูปที่ 4.4 แสดงส่วนประกอบของส่วนจ่ายพลังงาน

<u>ส่วนตรวจจับสัญญาณ</u> (SENSOR)

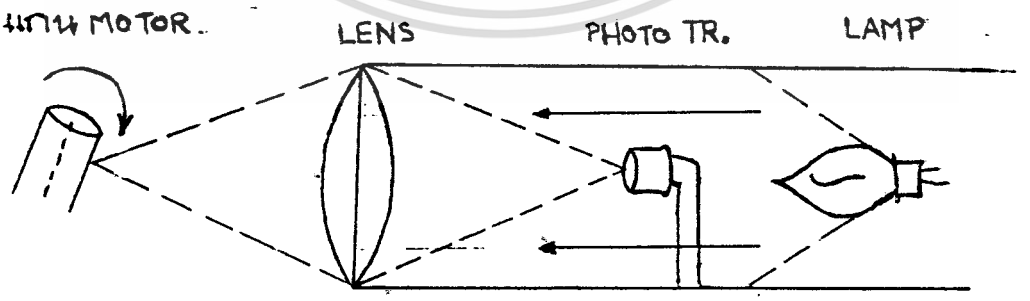
ส่วนตรวจจับสัญญาณ เป็นส่วนที่ใช้ตรวจจับสัญญาณจากแกนของมอเตอร์โดยตรง เพื่อสามารถใช้วัดความเร็วของมอเตอร์ที่ไม่มีเอ็นโคดเดอร์ในตัว ประกอบด้วยส่วนสำคัญ 2 ส่วน คือ

1. โฟโต้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น (NPN PHOTO-TRANSISTOR)
2. วงจรเปรียบเทียบแรงดัน (VOLTAGE COMPARATOR CIRCUIT)

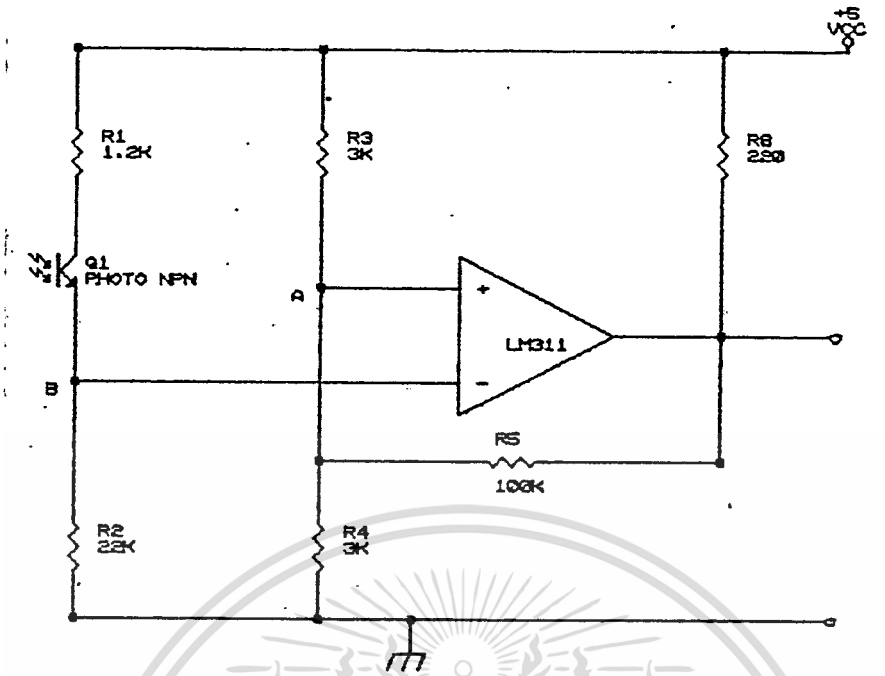
หลักการทำงาน

โฟโต้ทราสซิสเตอร์ ทำการแปลงสัญญาณ โดยการเปลี่ยนความเข้มของแสงเป็นสัญญาณแรงดันทางไฟฟ้า แล้วนำไปเปรียบเทียบกับแรงดันอ้างอิงของวงจรเปรียบเทียบแรงดัน ซึ่งเมื่อโฟโต้ทราสซิสเตอร์ยังไม่ได้รับแสง สัญญาณแรงดันจะต่ำกว่าแรงดันอ้างอิง จะให้สัญญาณเอาท์พุทเป็น 1 และเมื่อโฟโต้ทรานซิสเตอร์ได้รับแสง สัญญาณแรงดันจะสูงกว่าแรงดันอ้างอิง จะให้สัญญาณเอาท์พุทเป็น Ø โดยค่าสัญญาณเอาท์พุทที่ได้นี้จะนำไปป้อนสู่ส่วนประมวลผล เพื่อทำการนับและคำนวณเป็นค่าความเร็วของมอเตอร์อีกทีหนึ่ง

สำหรับในรูปที่ 4.5 แสดงการวางตำแหน่งของส่วนตรวจจับสัญญาณ



รูปที่ 4.5 แสดงการวางตำแหน่งของส่วนตรวจจับสัญญาณ



รูปที่ 4.6 วงจรส่วนตรวจจับสัญญาณ

จากรูปที่ 4.6

พิจารณาเมื่อโฟโต้ทรานซิสเตอร์ยังไม่ได้รับแสง สัญญาณแรงดันที่ B จะต่ำกว่าแรงดันอ้างอิงที่ A ให้ค่าสัญญาณเอาท์พุทประมาณ 5 โวลท์ ($R_5 >> R_6$

ดังนั้นเสมือนว่า R_3 ขนานกับ R_5

ซึ่งจะได้ค่า $R_{รวม} = \frac{3K \times 100K}{103K} = 2.91$ กิโลโอห์ม

แรงดันอ้างอิงที่ A $= \frac{3K \times 5V}{2.91K + 3K} = 2.54$ โวลท์

กระแสที่ไหลผ่าน A $= \frac{5V}{3K + 2.91K} = 0.85$ มิลลิแอมป์

พิจารณาเมื่อโฟโต้ทรานซิสเตอร์ได้รับแล้ว สัญญาณแรงดันที่ B จะสูงกว่าแรงดันอ้างอิงที่ A ให้ค่าสัญญาณเอาท์พุทประมาณ 0 โวลท์

แรงดันที่ B $= \frac{22K \times 5V}{22K + 1.2K} = 4.47$ โวลท์

ซึ่งมีค่ามากกว่าแรงดันอ้างอิงที่ A คือ 2.54 โวลท์

ทำให้แรงดันเอาท์พุทตกลงเป็น 0 โวลท์

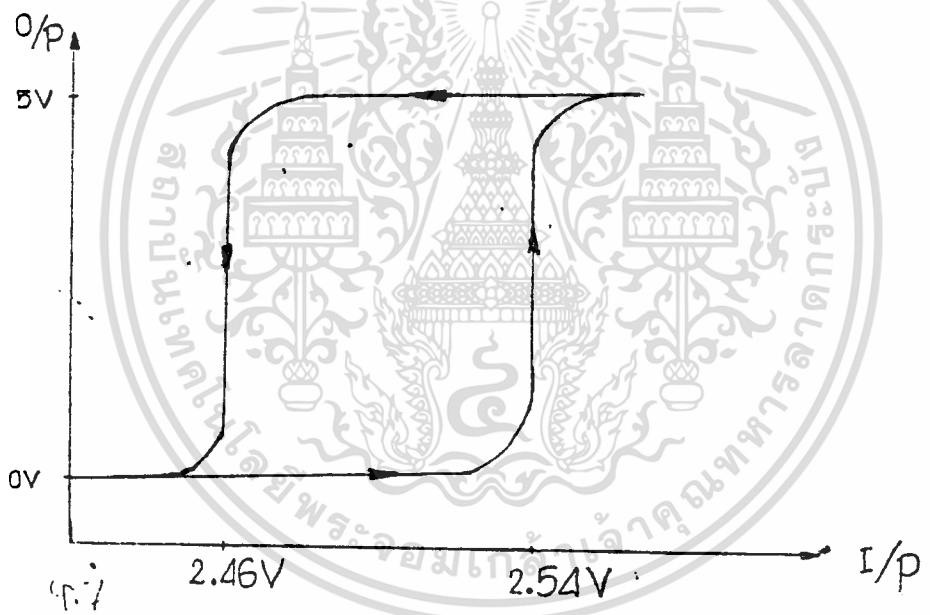
เมื่อแรงดันเอาท์พุทเป็น 0 โวลท์ ทำให้เสมือนกับว่า R_4 ขนานกับ R_5 ซึ่งจะได้ค่า $R_{รวม}$ ที่ต่ำลง

$R_{รวม}$ = $\frac{3K \times 100K}{103K}$ = 2.91 โวลท์

ดังนั้นจะทำให้ค่าแรงดันอ้างอิงที่ A ตกลงเหลือ

V = $\frac{2.91K \times 5V}{3K + 2.91K}$ = 2.46 โวลท์

จะเห็นได้ว่าแรงดันอ้างอิงมีค่าเปลี่ยนแปลงระหว่าง 2.46-2.54 โวลท์ ซึ่งเป็นช่วงของฮิสเตอรีซีส (HYSTERISIS) ดังรูป 4.7 เพื่อที่จะลดค่าความไวของเอ็นโคดเดอร์ไม่ให้เกิดความผิดพลาดเมื่อแรงดันที่ B มีค่าประมาณ 2.5 โวลท์



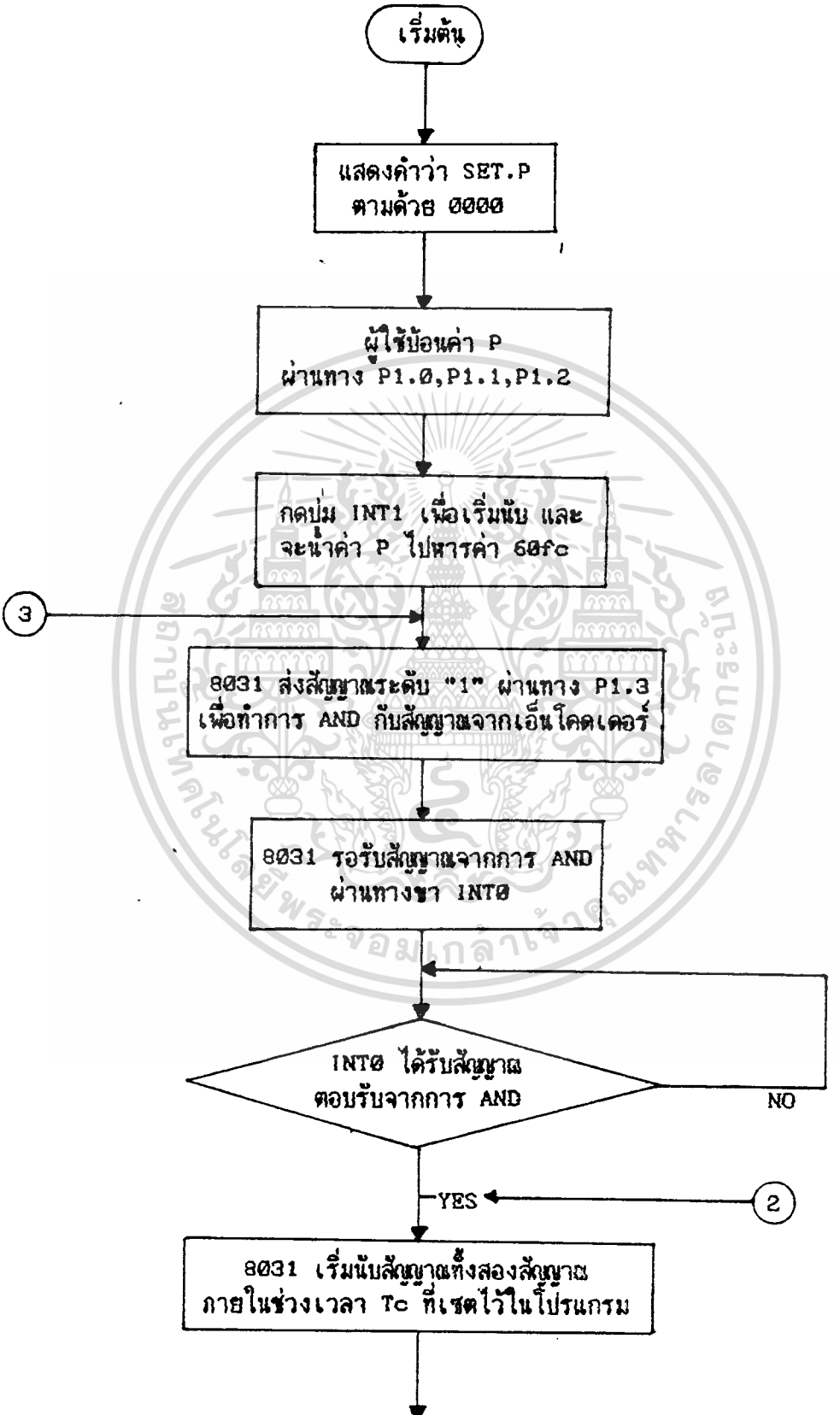
รูปที่ 4.7 แสดงฮิสเตอรีซิสของวงจรส่วนตรวจจับสัญญาณ

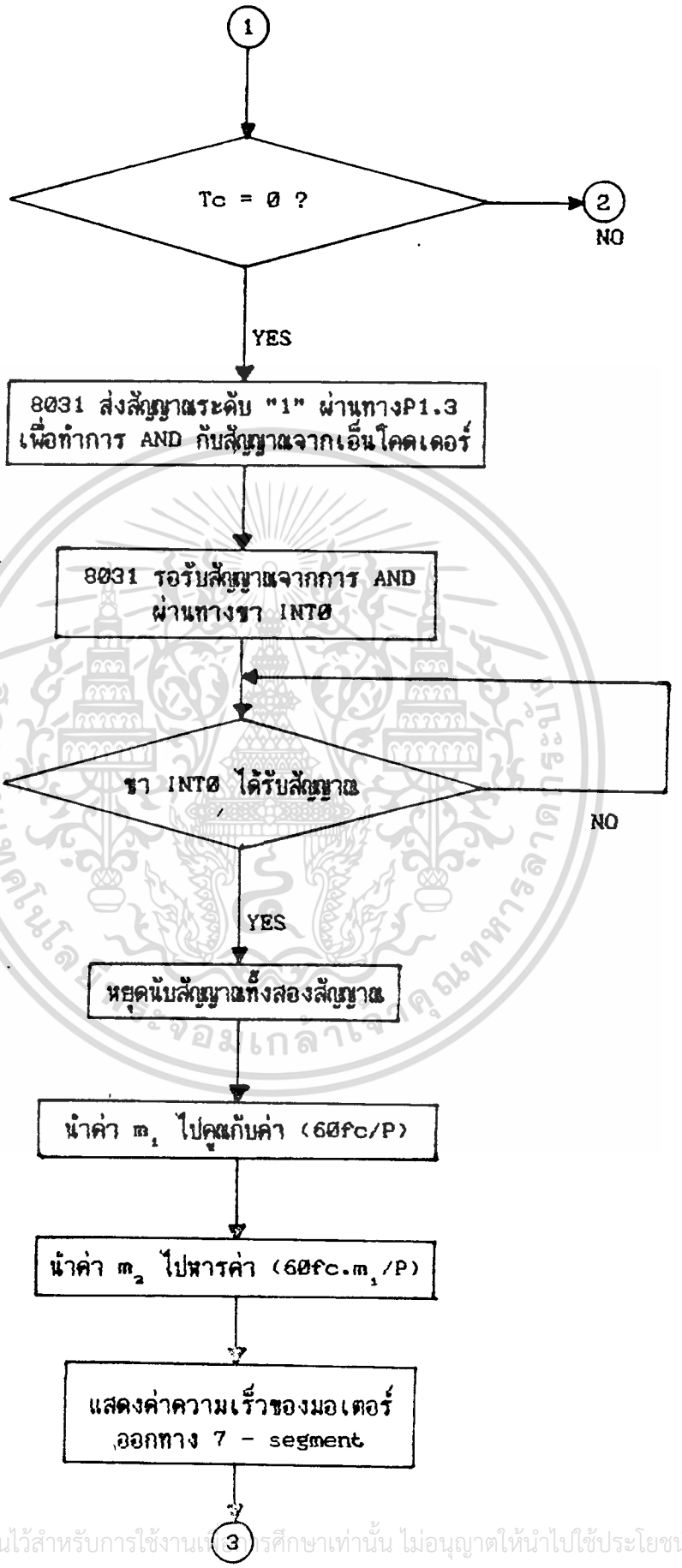
บทที่ 5

การออกแบบโปรแกรมควบคุมการทำงาน

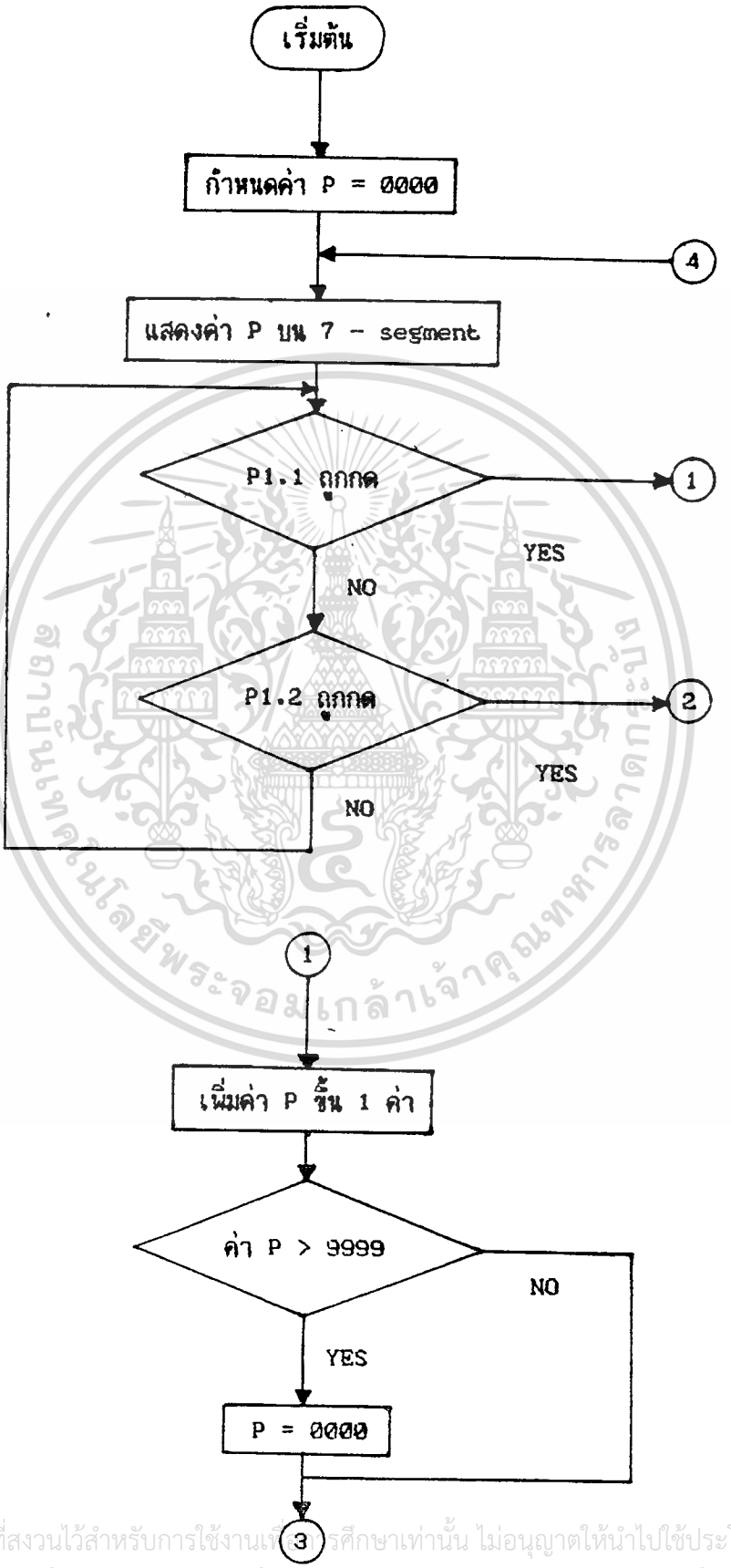
ในการใช้ 8031 วัดค่าความเร็วของมอเตอร์ จะใช้ตัวนับภายในทำหน้าที่นับจำนวนสัญญาณทั้ง 2 สัญญาณ คือสัญญาณจากเอ็นโคดเดอร์ และสัญญาณจากส่วนกำเนิดความถี่ขนาด 20 KHz ซึ่งได้กำหนดให้ตัวนับภายในทั้ง 2 ตัวทำหน้าที่เป็นตัวนับในโหมด 2 ซึ่งสามารถนับได้ 16 บิท ทำให้สามารถนับได้สูงสุดถึง 65,535 พัลส์ ส่วนอินเทอร์รัพท์ภายนอก 1 ให้มีระดับไพออริตี้สูงกว่าอินเทอร์รัพท์ภายนอก 0 และทั้งสองอินเทอร์รัพท์ให้ทำงานที่ขอบขาลงของสัญญาณ ทำให้ต้องเซทค่าภายในรีจิสเตอร์ TCON = #05, IP = #04, TMOD = #66 ซึ่งขั้นตอนการทำงานทั้งหมดได้แสดงเป็น FLOWCHART พร้อมทั้งมีส่วนที่เป็นโปรแกรมทั้งหมดได้จัดแสดงไว้ต่อจากส่วนของ FLOWCHART

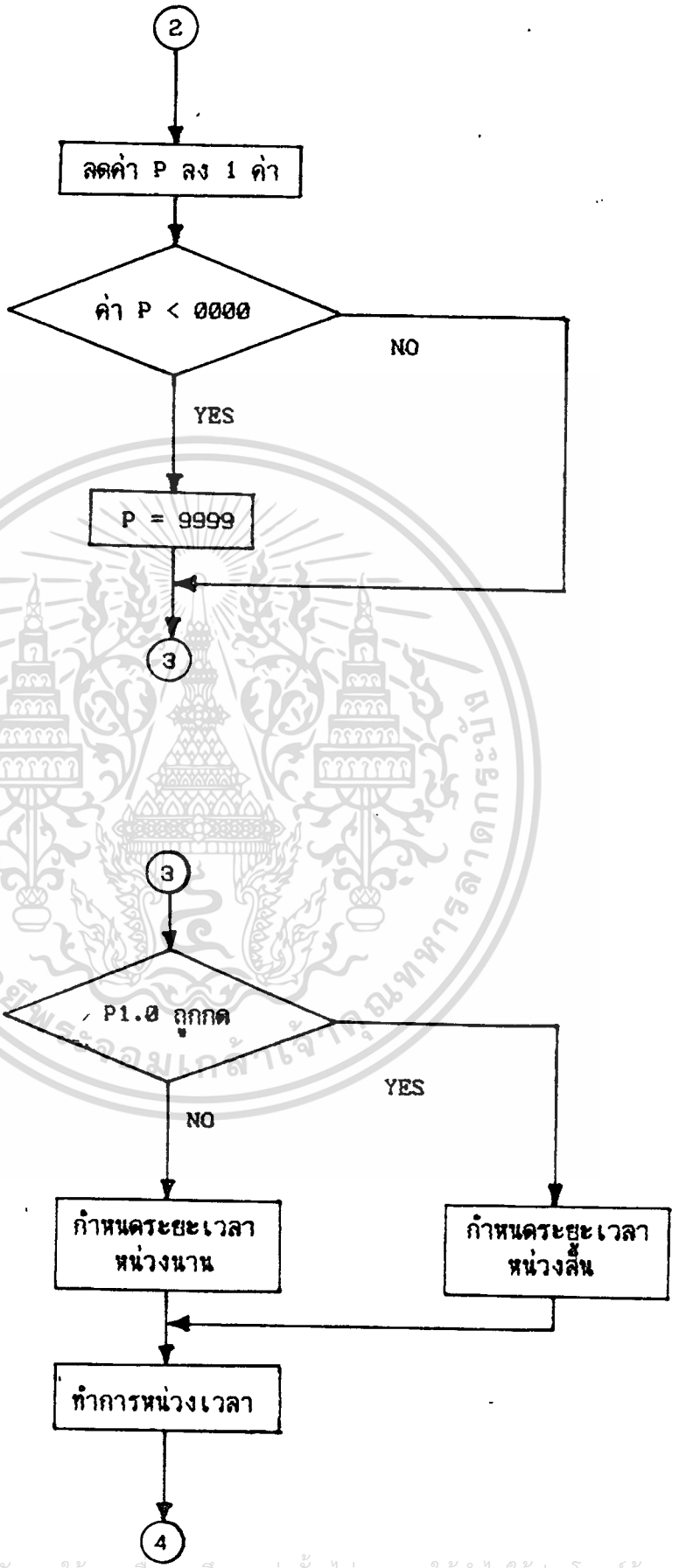
FLOWCHART แสดงขั้นตอนการทำงานของ 8031 ในการวัดค่าความเร็วมอเตอร์





FLOWCHART แสดงการทำงานของโปรแกรมเซตค่า P

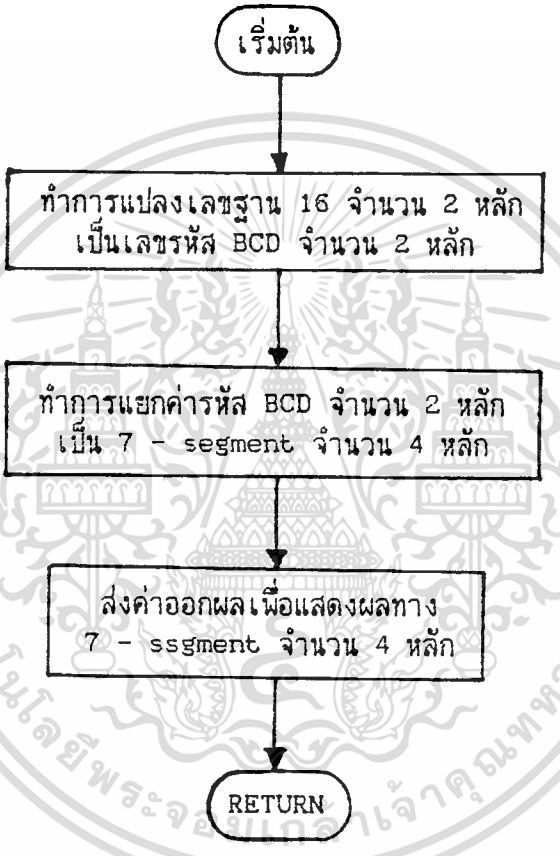




FLOWCHART แสดงการแปลงค่าจากเลขฐาน 16 เป็นเลขฐาน 10 เพื่อแสดงออกทาง 7-segment

<u>กำหนดให้</u> หน่วยความจำภายในที่ @7E, @7F เป็นค่า P ในเลขฐาน 16 โดยเป็น Ph และ P1 ตามลำดับ

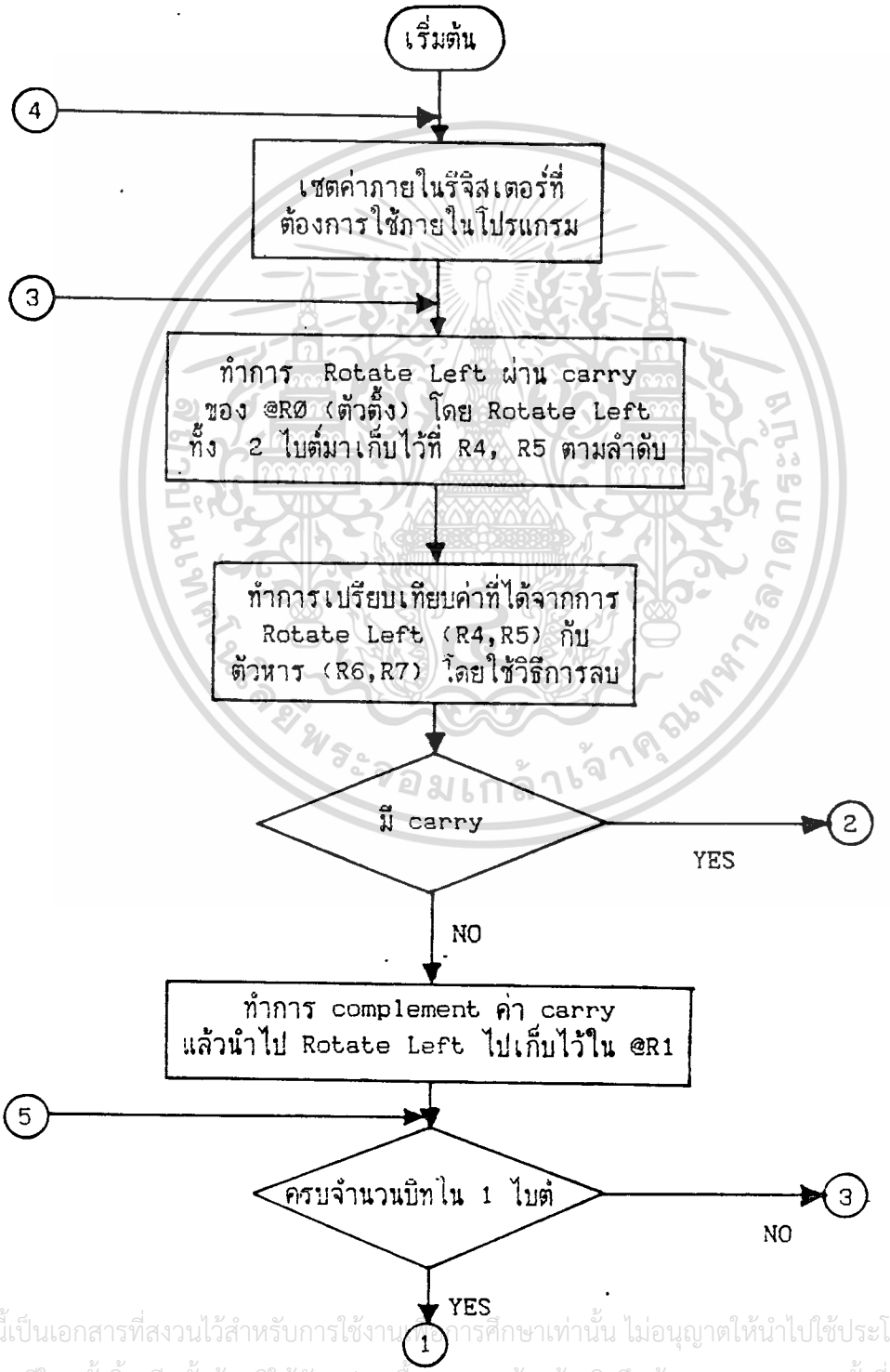
ให้ @7C, @7D เป็นค่าที่จะนำไปแสดงผล (ฐาน 16)
@7A, @7B เป็นค่าที่จะนำไปแสดงผล (รหัส BCD)
@76, @77, @78, @79 เป็น display buffer

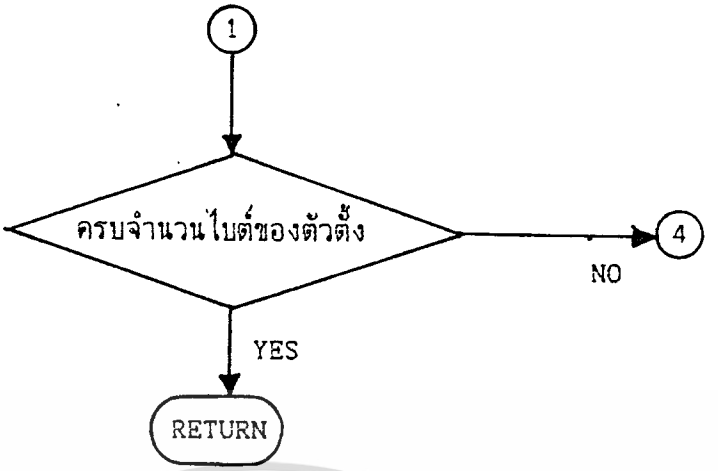


FLOWCHART โปรแกรมการหารเลขฐาน 16 ด้วยเลขฐาน 16

<u>กำหนดให้</u> ตำแหน่งของตัวตั้งเก็บไว้ที่ @RØ , ตัวหารเก็บไว้ที่ R6, R7 ได้ผลลัพธ์เก็บไว้ที่ @R1

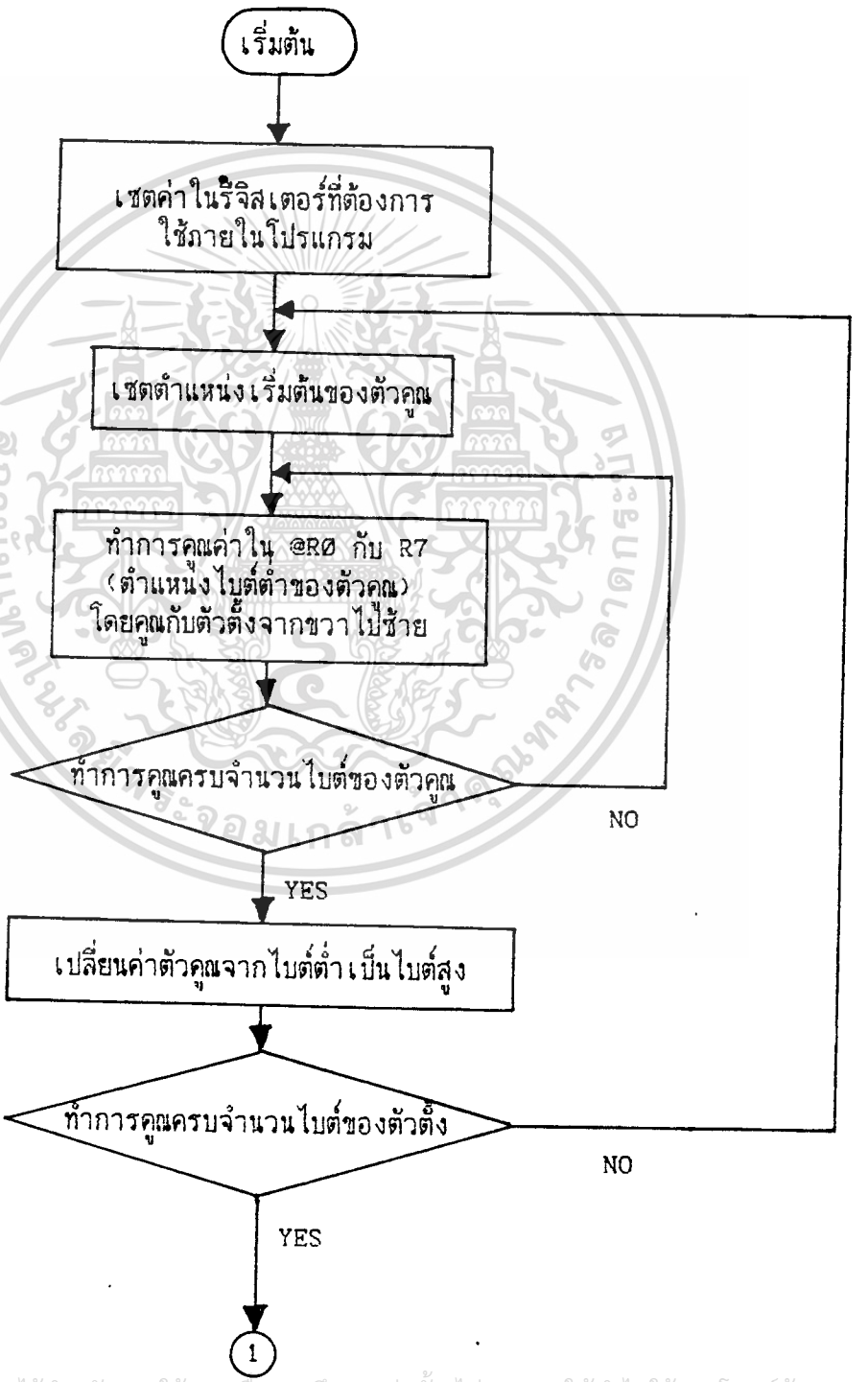
โดย R2 เป็นรีจิสเตอร์ที่คอยเช็คว่าการหารทำครบ 8 บิทใน 1 ไบต์
R3 เป็นรีจิสเตอร์ที่คอยเช็คว่าการหารทำครบจำนวนไบต์ของตัวตั้ง
R4, R5 เป็นรีจิสเตอร์ที่เก็บผลลัพธ์จากการลบ

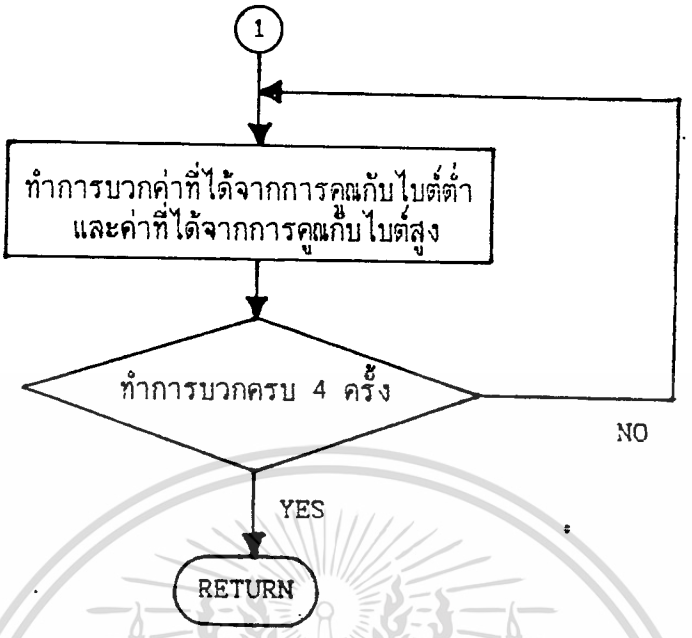




FLOWCHART โปรแกรมการคูณของเลขฐาน 16 โดยตัวตั้งจำนวน 3 ไบต์ และตัวคูณจำนวน 2 ไบต์

<u>กำหนดให้</u> ตัวตั้งเก็บไว้ที่ @RØ , ตัวคูณเก็บไว้ที่ R6,R7 ส่วนผลลัพธ์เก็บไว้ที่ @R1
โดย R3 เป็นรีจิสเตอร์ที่คอยเช็คว่าทำการคูณครบจำนวนไบต์ของตัวตั้ง
R2 เป็นรีจิสเตอร์ที่คอยเช็คว่าทำการคูณครบจำนวนไบต์ของตัวคูณ
R4 เป็นรีจิสเตอร์ที่คอยเช็คว่าทำการบวกครบ 4 ครั้ง





START PROGRAM (โปรแกรมเริ่มต้นการทำงาน)

```
ADDRESS  OBJ.CODE  SOURCE

0000      020044    ORG..     LJMP START      ;ข้ามไปทำที่โปรแกรม START

0044      753000              MOV  @30,#00    ;เซตค่าภายในตำแหน่งหน่วย
0047      753105              MOV  @31,#05     ความจำภายในที่ใช้ในการ
004A      753200              MOV  @32,#00     INT1, INT0 ตามลำดับ
004D      753306              MOV  @33,#06
0050      C293                CLR  P1.3       ;เพื่อป้องกันการ INT0
0052      438805              ORL  TCON,#05   ;เซตค่าภายใน SFR ของ
0055      75B804              MOV  IP,#04      8031
0058      758966              MOV  TMOD,#66
005B      D2AF                SETB EA
005D      757625              MOV  @76,#25    ;ให้ 8031 แสดงคำว่า
0060      75770D              MOV  @77,#0D     SET.P
0063      75780E              MOV  @78,#0E
0066      757919              MOV  @79,#19
0069      121043              LCALL  SHOW
006C      D2A8                SETB EX0        ;เซตให้ INT0 enable
006E      7400      DELAY:    MOV  A,#00       เพื่อทำการล้าง INT0 ที่
0070      7D00                MOV  R5,#00      เกิดขึ้นก่อนการนับ
0072      04        TIME2     INC  A          ;หน่วงเวลา
0073      0D        TIME1     INC  R5
0074      BDFFFC              CJNE R5,#FF,ONE
0077      B4FFF8              CJNE A,#FF,TWO
007A      7C9A                MOV  R4,#9A     ;คอยเช็คว่ามี INT0 ?
007C      C2A8                CLR  EX0        ;disable INT0
007E      D2AA                SETB EX1        ;enable INT1
0080      0201FC              LJMP SETP       ;ไปที่โปรแกรม SETP
```

SETP PROGRAM (โปรแกรมที่ใช้ในการเซตค่า P ; จำนวนพัลส์ต่อการหมุนครบ 1 รอบ)

```
ADDRESS  OBJ.CODE  SOURCE

01FC     D2D1      SETP:   SETB D1          ;เซตบิทที่ใช้ในการ INT1,
01FF     D2D5               SETB D5          INT0
0201     787E               MOV  R0,#7E
0203     7600               MOV  @R0,#00
0205     08                 INC  R0
0206     7600               MOV  @R0,#00     ;แสดงค่า 0000 ออก
0208     E6        LOOP4 ;  MOV  A,@R0       7 - segment และ
0209     797D               MOV  R1,#7D      นำค่าที่ป้อนเข้าไปแสดง
020B     F7                 MOV  @R1,A       ออกทาง 7 - segment
020C     18                 DEC  R0
020D     19                 DEC  R1
020E     E6                 MOV  A,@R0
020F     F7                 MOV  @R1,A
0210     121000             LCALL  DISPY
0213     439007    LOOP0    ORL  P1,#07
0216     C3                 CLR  C
0217     309126             JNB  P1.1,LOOP1
021A     2092F6             JB   P1.2,LOOP0
021D     787F               MOV  R0,#7F
021F     E6                 MOV  A,@R0
0221     9401               SUBB A,#01       ;ลดค่า P
0223     F6                 MOV  @R1,A
0224     18                 DEC  R0
0225     5001               JNC  ONE
0227     16                 DEC  @R0
0228     B6FF05    ONE      CJNE @R0,#FF,LOOP3 ;เช็คว่ามีค่าน้อยกว่า
022B     7627               MOV  @R0,#27       0000 ? ถ้าน้อยกว่า
022D     08                 INC  R0            ให้แสดงค่าจาก
022E     760F               MOV  @R0,#0F       0000 ——→ 9999
0230     787F      LOOP3    MOV  R0,#7F      ;เช็คว่า P1.0 ถูกกด ?
0232     309004             JNB  P1.0,TWO      ถ้าถูกกดให้หน่วงเวลา
0235     7B07               MOV  R3,#07        น้อย ถ้าไม่ถูกกดให้หน่วง
0237     8002               SJMP THREE         เวลามาก
0239     7B7F      TWO      MOV  R3,#7F
023B     120300    THREE    LCALL  DELY
023E     020207             LJMP   LOOP4
0241     787F      LOOP1    MOV  R0,#7F
0243     E6                 MOV  A,@R0
0244     2401               ADD  A,#01       ;เพิ่มค่า P
0246     F6                 MOV  @R0,A
0245     18                 DEC  R0
```

```
ADDRESS  OBJ.CODE  SOURCE

0246      5001              JNC   ZERO
0248      06                INC   @R0
0241      B627E2    ZERO    CJNE  @R0,#27,LOOP3   ;เช็ดว่ามีค่ามากกว่า
0244      08                INC   R0               9999 ? ถ้ามากกว่า
0245      B610DE            CJNE  #R0,#10,LOOP3    ให้แสดงค่าจาก
0248      7600              MOV   @R0,#00          9999 ——→0000
024A      18                DEC   R0
024B      7600              MOV   @R0,#00
024D      80D7              SJMP  LOOP3
```

DELY PROGRAM (โปรแกรมหน่วงเวลา)

```
0300      7AFF      DELY :  MOV   R2,#FF
0302      00        DELY1   NOP
0303      00                NOP             ;ทำการหน่วงเวลา
0304      00                NOP
0305      00                NOP
0306      DAFA              DJNZ  R2,DELY1
0308      DBF6              DJNZ  R3,DELY
030A      22                RET
```

DA & DISP PROGRAM (โปรแกรมที่ทำการแปลงเลขจากฐาน 16 เป็นเลขฐาน 10 แล้วจึงส่งไปแสดงผล)

```
0FF0 CODE: 81,F3,49,61,33,25,05,81,01,21   ;รหัสที่ใช้ในการแสดงเลข 1-10
1000      7400      DISPY:  MOV   A,#00
1002      C3                CLR   C
1003      797B              MOV   R1,#7B
1005      7700              MOV   @R1,#00
1007      19                DEC   R1          ;ทำการแปลงเลขฐาน 2
1008      7700              MOV   @R1,#00       เป็นเลขรหัส BCD
100A      7A10              MOV   R2,#10
100C      787D      FOUR    MOV   R0,#7D
100E      7B02              MOV   R3,#02
1010      E6        FIVE    MOV   A,@R0
1011      33                RLC   A
1012      F6                MOV   @R0,A
1013      18                DEC   R0
1014      DBFA              DJNZ  R3,FIVE
1016      787B              MOV   R0,#7B
1018      7B02              MOV   R3,#02
101A      E6        SIX     MOV   A,@R0
101B      36                ADDC  A,@R0
101C      D4                DA    A
```

```
ADDRESS  OBJ.CODE  SOURCE

101D     F6                MOV  @R0,A
101E     18                DEC  R0
101F     DBF9              DJNZ R3,SIX
1021     DAE9              DJNZ R2,FOUR
1023     9000FF0           MOV  DPTR,#CODE
1026     7876              MOV  R0,#76
1028     797A              MOV  R1,#7A
102A     7A02              MOV  R2,#02
102C     E7        SAVEN   MOV  A,@R1
102D     7B02              MOV  R3,#02
102F     C4        EIGHT   SWAP A
1030     75F00F            MOV  B,#0F
1033     52F0              ANL  B,A
1035     C0E0              PUSH A          ;ทำการแยกค่า BCD จำนวน
1037     E5F0              MOV  A,B         2 หลัก เป็นค่าที่แสดงออก
1039     93                MOVC A,@A + DPTR 7-segment จำนวน 4 หลัก
103A     F6                MOV  @R0,A
103B     08                INC  R0
103C     D0E0              POP  A
103E     DBEF              DJNZ R3,EIGHT
1040     09                INC  R1
1041     DAE9              DJNZ R2,SAVEN
1043     7A04      SHOW:   MOV  R4,#04      R2 ?
1045     7876              MOV  R0,#76
1047     7903              MOV  R1,#03     ;ส่งค่าออกแสดงผลทาง
1049     E6        NINE    MOV  A,@R0       7-segment
104A     F3                MOVX @R1,A
104B     19                DEC  R1
104C     08                INC  R0
104D     DAFA              DJNZ R2,NINE
104F     22                RET
```

DIVIDE PROGRAM (โปรแกรมการหารเลขฐาน 16 โดยตัวหารมีจำนวน 2 ไบต์)

```
1500     7860      DIVIDE: MOV  R0,#60     ;เซตค่าภายในหน่วยความ
1502     7968              MOV  R1,#68      จำภายในที่ใช้ในโปรแกรม
1504     7803              MOV  R3,#03      ←จำนวน
1506     7D00              MOV  R5,#00
1508     7C00              MOV  R4,#00
150A     7A08      BYTE    MOV  R2,#08     ;ทำการ Rotate Left
150C     AE5E              MOV  R6,@5E      ผ่าน carry ของ @60,
150E     AF5F              MOV  R7,@5F      @61 (ตัวตั้ง) โดย
1510     C3        BIT     CLR  C           Rotate Left ทั้ง 2
```

```
ADDRESS  OBJ.CODE  SOURCE

1511      E6                MOV   A,@R0       ไบต์มาเก็บไว้ที่ R4,R5
1512      33                RLC   A           ตามลำดับ
1513      F6                MOV   @R0,A
1514      ED                MOV   A,R5
1515      33                RLC   A
1516      FD                MOV   R5,A
1517      EC                MOV   A,R4
1518      33                RLC   A
1519      FC                MOV   R4,A
151A      ED                MOV   A,R5        ;ทำการเปรียบเทียบค่าที่
151B      C3                CLR   C           ถูก Rotate แล้วกับตัว
151C      9F                SUBB  A,R7        หารโดยใช้วิธีการลบ
151D      FD                MOV   R5,A
151E      EC                MOV   A,R4
151F      9E                SUBB  A,R6
1520      402E              JC    BORROW      ;เช็ค carry ถ้ามีไป BORROW
1522      FC                MOV   R4,A        ;ทำการ Complement ค่า
1523      E7                MOV   A,@R1       carry แล้วนำไปเก็บไว้ใน
1524      B3                CPL   C           @R1
1525      33                RLC   A
1526      F7                MOV   @R1,A
1527      DAE7    CHECK     DJNZ  R2,BIT      ;เช็คว่าครบจำนวนบิทใน
1529      08                INC   R0          1 ไบต์ ?
152A      09                INC   R1
152B      DBDD              DJNZ  R3,BYTE     ;เช็คว่าครบจำนวนไบต์
152C      22                RET               ของตัวตั้ง ?
-------------------------------------------------------------------
SUBROUTINE  BORROW (โปรแกรมย่อยที่ใช้เมื่อการหารมีตัวทด(เกิดขึ้น)

1550      FC      BORROW    MOV   R4,A        ;ทำการ Complement ค่า
1551      E7                MOV   A,@R1       carry แล้วนำไป Rotate
1552      B3                CPL   C           เก็บไว้ใน @R1
1553      33                RLC   A
1554      F7                MOV   @R1,A       ;ทำการบวกของค่าระหว่าง
1555      ED                MOV   A,R5        ตัวหารกับผลลัพธ์ที่ได้จาก
1556      3F                ADDC  A,R7        การหาร เพื่อให้ค่าคงเดิม
1557      FD                MOV   R5,A
1558      EC                MOV   A,R4
1559      3E                ADDC  A,R6
155A      FC                MOV   R4,A
155B      80CA              SJMP  CHECK       ;กลับไปตรวจสอบว่าครบ
                                              จำนวนบิทและไบต์ ?
```

```
MULTIPLY  PROGRAM (โปรแกรมการคูณ โดยตัวตั้งมีจำนวน 3 ไบต์ ตัวคูณจำนวน 2 ไบต์

ADDRESS   OBJ.CODE  SOURCE

1600      753600    MULTIPLY: MOV  @36,#00     ;เซตค่าภายในหน่วยความ
1603      753A00              MOV  @3A,#00      จำภายในที่ใช้ในโปรแกรม
1606      753F00              MOV  @3F,#00
1609      7B02                MOV  R3,#02     ; จำนวนตัวคูณ
160B      AE46                MOV  R6,@46
160D      AF47                MOV  R7,@47
160F      793A                MOV  R1,#3A
1611      7843      LOOP2     MOV  R0,#43     ;เซตตำแหน่งเริ่มต้นของ
1613      7A03                MOV  R2,#03       การคูณ
1615      C3                  CLR  C
1616      18        NEXT      DEC  R0         ;ทำการคูณค่าใน @R0 กับ
1617      E6                  MOV  A,@R0        R7 (ตำแหน่งไบต์ต่ำของ
1618      8FF0                MOV  B,R7         ตัวคูณ) โดยคูณกับตัวตั้งที
161A      A4                  MUL  A,B          ละไบต์จากขวาไปซ้ายจน
161B      37                  ADDC A,@R1        ครบทั้งหมด
161C      F7                  MOV  @R1,A
161D      19                  DEC  R1
161E      A7F0                MOV  @R1,B
1620      DAF0                DJNZ R2,NEXT
1622      AF46                MOV  R7,@46     ;ทำการเปลี่ยนค่าตัวคูณ
1624      793F                MOV  R1,#3F       จากไบต์ต่ำเป็นไบต์สูง
1626      DBEB                DJNZ R3,LOOP2
1628      C3                  CLR  C          ;ทำการบวกค่าที่ได้จากการ
1629      7C04                MOV  R4,#04       คูณของไบต์ต่ำกับค่าที่ได้จาก
162B      783A                MOV  R0,#3A       การคูณของไบต์สูง
162D      7940                MOV  R1,#40
162F      18        PLUS      DEC  R0
1630      19                  DEC  R1
1631      E6                  MOV  A,@R0
1632      37                  ADDC A,@R1
1633      F6                  MOV  @R0,A
1634      DCF9                DJNZ R4,PLUS
1636      22                  RET

-------------------------------------------

INTERRUPT 1  PROGRAM (โปรแกรมที่ใช้ในการเซตค่า P และทำการเริ่มนับ)

0013      020400    INT1      LJMP JINT1
0016      32        RET1:     RETI

0400      D010      JINT1:    POP  @10
0402      D011                POP  @11
```

```
ADDRESS  OBJ.CODE  SOURCE

0404     C030              PUSH @30
0406     C031              PUSH @31
0408     020016            LJMP RET1

0500     20D11D            JB   D1,COUNT1
0503     020FFC            LJMP SETP

0520     752012   COUNT1   MOV  @20,#12     ;ทำการเซตค่าที่ต้องการ
0523     752154            MOV  @21,#54      ทำการหารแล้วข้ามไปยัง
0526     752215            MOV  @22,#15      โปรแกรมการหาร
0529     7820              MOV  R0,#20
052B     792B              MOV  R1,#2B
052D     7803              MOV  R2,#03
052F     857E5E            MOV  @5E,@7E
0532     857F5F            MOV  @5F,@7F
0535     121500            LCALL  DIVIDE
0538     852B1B            MOV  @1B,@2B      ;ทำการเก็บค่าที่ได้จากการ
053B     852C1C            MOV  @1C,@2C      หารสำรองไว้
053E     852D1D            MOV  @1D,@2D
0541     C2D1     AGAIN:   CLR  D1           ;เซตค่าภายในหน่วยความ
0543     7F00              MOV  R7,#00       จำภายในที่ใช้ในการนับ
0545     7E00              MOV  R6,#00
0547     758D00            MOV  TH1,#00
054A     758B00            MOV  TL1,#00
054D     758C00            MOV  TH0,#00
0550     758A00            MOV  TL0,#00
0553     753203            MOV  @32,#03
0556     753306            MOV  @33,#06
0559     D2A8              SETB EX0          ;enable INT0
055B     D293              SETB P1.3         ;เซตให้ P1.3 มีค่าเป็น "1"
055D     00       WAIT1    NOP               ;รอผลของการ AND ของ
055E     80FD              SJMP WAIT1        สัญญาณ เพื่อไปยัง INT0
```

```
INTERRUPT0 PROGRAM (START AND STOP COUNT)

0003     C293     INT0     CLR  P1.3
0005     020450            LJMP  JINT0
0008     32       RET2:    RETI

0450     D012     JINT0:   POP  @12
0452     D013              POP  @13
0454     C032              PUSH @32
0456     C033              PUSH @33
```

```
ADDRESS  OBJ.CODE  SOURCE

0458     020000    LJMP RET2

0600     BC9A4D    CJNE R4,#9A,CLRINT0
0603     20D51A    JB   D5,STARTC
0606     0206A0    LJMP STOPC

0620     43A80A  STARTC  ORL  IE,#0A      ;สั่งให้ตัวนับเริ่มทำการนับ
0623     438850          ORL  TCON,#50
0626     7400            MOV  A,#00       ;ทำการหน่วงเวลาที่ต้อง
0628     7D00            MOV  R5,#00       การใช้ในการนับจำนวน
062A     7C00            MOV  R4,#00       สัญญาณ
062C     04      TC3     INC  A
062D     0D      TC2     INC  R5
062E     0C      TC1     INC  R4
062F     BCFFEC          CJNE R4,#FF,TC1
0632     BDFFF8          CJNE R5,#FF,TC2
0635     B404F4          CJNE A,#04,TC3
0638     C2D5            CLR  D5
063A     D293            SETB P1.3
063C     00      WAIT2   NOP              ;รอผลของการ AND
063D     80FD            SJMP WAIT2        เพื่อไปยังส่วนคำนวณ

0650     020056  CLRINT0 LJMP DELAY
-------------------------------------------------------------
CALCULATE  PROGRAM (โปรแกรมการคำนวณค่าจำนวนพัลส์เป็นค่าความเร็วมอเตอร์

06A0     53A8F5  STOP_C: ANL  IE,#F5      ;สั่งให้ตัวนับทำการหยุดนับ
06A3     53A8AF          ANL  TCON,#AF
06A6     851840          MOV  @40,@1B     ;เซตค่าที่ต้องการคูณแล้ว
06A9     851C41          MOV  @41,@1C      ส่งไปยังโปรแกรมการคูณ
06AC     851D42          MOV  @42,@1D
06AF     8E46            MOV  @46,R6
06B1     858A47          MOV  @47,TL0
06B4     8F5E            MOV  @5E,R7
06B6     858B5F          MOV  @5F,TL1
06B9     121600          LCALL  MULTIPLY
06BC     7836            MOV  R0,#36      ;เซตค่าที่ต้องการหารแล้ว
06BE     792B            MOV  R1,#2B       ส่งไปยังโปรแกรมการหาร
06C0     7B05            MOV  R3,#05
06C2     121506          LCALL  DIVIDE
06C5     7576FF          MOV  @76,#FF     ;ทำการเคลียร์
06C8     7577FF          MOV  @77,#FF      7-segment
06CB     7578FF          MOV  @78,#FF
```

```
ADDRESS   OBJ.CODE  SOURCE

06CE      7579FF            MOV  @79,#FF
06D1      121043            LCALL  SHOW
06D4      852E7C            MOV  @7C,@2E      ;ส่งค่าที่คำนวณเสร็จเป็น
06D7      852F7D            MOV  @7D,@2F       ค่าความเร็วมอเตอร์ออก
06DA      121000            LCALL  DISPY       ไปแสดงผล
06DD      D2D5              SETB D5           ;ข้ามไปยังส่วนเริ่มต้นนับ
06DF      020541            LJMP AGAIN         เพื่อทำการวัดความเร็ว
                                               ความเร็วรอบใหม่
```

บทที่ 6

การทดลองและผลการทดลอง

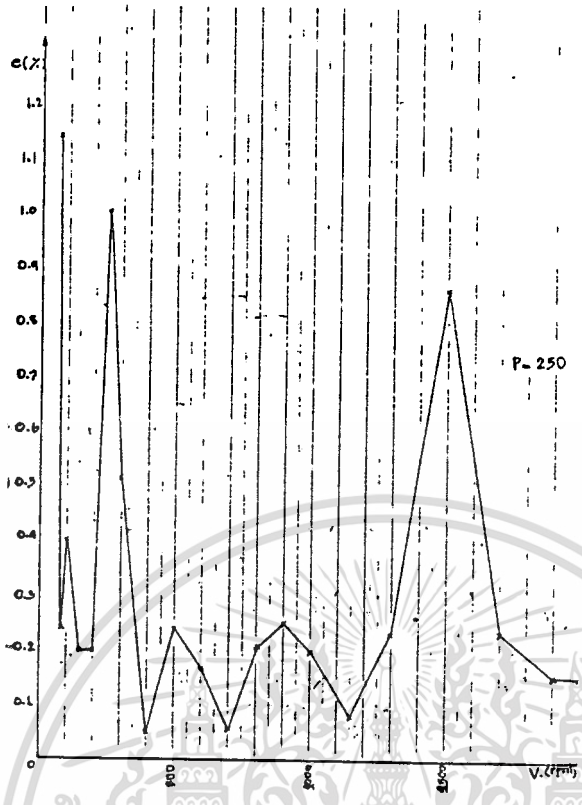
หลังจากที่ได้ทำการออกแบบและสร้างวงจรพร้อมทั้งส่วนของโปรแกรม ได้มีการทดสอบเครื่องวัดความเร็วของมอเตอร์ที่ทำการสร้างขึ้น เปรียบเทียบกับเครื่องวัดความเร็วมอเตอร์ที่เป็นมาตรฐาน โดยในขั้นแรกได้ทำการวัดค่าความเร็วของมอเตอร์โดยรับสัญญาณเอ็นโคดเดอร์ที่มีอยู่ภายในตัวมอเตอร์ที่ทำการทดสอบ โดยมีค่า P = 250 แล้วทำการบันทึกผลการทดลอง หลังจากนั้นจึงนำส่วนตรวจจับสัญญาณที่ได้สร้างขึ้นมาทำการตรวจจับสัญญาณแทนเอ็นโคดเดอร์ที่มีอยู่ภายในตัวของมอเตอร์ ซึ่งในกรณีนี้จะได้ค่า P = 1 แล้วทำการบันทึกผลการทดลอง หลังจากนั้นจึงได้คำนวณค่าความผิดพลาด พร้อมทั้งบันทึกไว้ดังตารางที่ 6.1 และ 6.2 ตามลำดับ ซึ่งในส่วนของค่าความผิดพลาดได้นำมาพล็อตเป็นเส้นกราฟดังแสดงไว้ในรูปที่ 6.1 และ 6.2 ตามลำดับ

ค่าความเร็วมอเตอร์ (rpm) ทาโคมิเตอร์	8031	แรงดันที่จ่ายให้มอเตอร์ (V)	ค่าความผิดพลาด (%)
65.75	65	1.22	1.14
81.2	81	1.39	0.246
99.4	99	1.58	0.402
151.3	151	2.13	0.198
200.4	200	2.64	0.199
250.9	248	3.16	1.077
302.6	301	3.70	0.529
397.2	397	4.66	0.050
501.2	500	5.71	0.239
600	601	6.71	0.167
699.6	700	7.72	0.057
803.3	805	8.77	0.212
897.7	900	9.73	0.256
999	1001	10.75	0.200
1147	1148	12.20	0.087
1297	1300	13.70	0.231
1498	1502	15.71	0.867
1704	1708	17.75	0.235
1897	1900	19.65	0.158
2006	2009	20.60	0.150
2072	2076	21.20	0.193

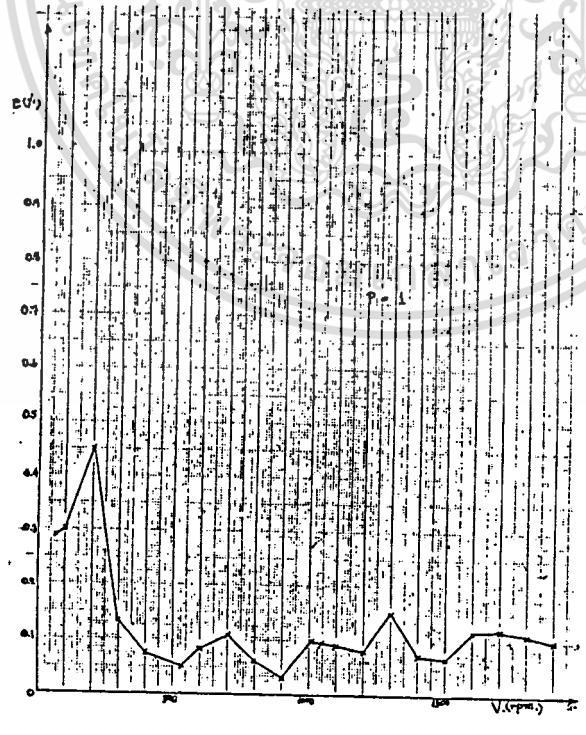
ตารางที่ 6.1 ผลการทดลองเมื่อค่า P = 1

ค่าความเร็วของมอเตอร์ (rpm) ทาโคมิเตอร์	8031	แรงดันที่จ่ายให้มอเตอร์ (V)	ค่าความผิดพลาด (%)
68.2	68	1.2	0.293
99.3	99	1.5	0.302
199.1	192	2.3	0.452
298.4	298	2.8	0.134
399.7	400	4.3	0.075
534.7	535	6.0	0.056
601.5	601	6.4	0.083
703.2	704	7.9	0.114
798.5	799	8.4	0.063
903.7	904	9.9	0.032
1010	1011	11.0	0.099
1107	1108	12.0	0.090
1202	1203	13.0	0.083
1300	1302	14.0	0.154
1402	1403	14.9	0.071
1499	1500	15.6	0.067
1598	1600	16.9	0.125
1697	1699	17.7	0.118
1800	1802	18.6	0.111
1903	1905	19.9	0.105
2000	2002	20.5	0.100
2141	2143	22.0	0.095

ตารางที่ 6.2 ผลการทดลองเมื่อค่า P = 250



รูปที่ 6.1 กราฟแสดงค่าความผิดพลาดเมื่อ P = 250



รูปที่ 6.2 กราฟแสดงค่าความผิดพลาดเมื่อ P = 1

บทที่ 7

บทวิจารณ์และสรุป

ในการวัดความเร็วของมอเตอร์ สามารถเลือกใช้อุปกรณ์ในการวัดได้หลายแบบขึ้นอยู่กับความต้องการของผู้ออกแบบ แต่ในโครงงานนี้ได้เลือกใช้ 8031 ไมโครโปรเซสเซอร์เป็นส่วนประมวลผล เนื่องจากว่าทำให้การออกแบบวงจรสามารถทำได้ง่ายขึ้น ทั้งยังสามารถนำค่าความเร็วมอเตอร์ที่วัดได้ไปใช้ในการควบคุมมอเตอร์อีกทีหนึ่ง โดยเครื่องวัดความเร็วมอเตอร์ที่ได้จัดทำขึ้นนี้ใช้ระยะเวลาในการวัดสั้น, สามารถวัดค่าความเร็วมอเตอร์ในช่วงที่กว้าง, ทั้งยังให้ค่าความผิดพลาดน้อย จากผลการทดลองได้ค่าความผิดพลาดประมาณ 0-1 % โดยในกรณีที่ค่าความผิดพลาดในการวัดมีค่าถึง 1% นั้นเกิดขึ้นจากการที่เครื่องวัดที่ได้จัดทำขึ้น ไม่สามารถแสดงผลได้ถึงตำแหน่งทศนิยม

กิตติกรรมประกาศ

ในการจัดทำวิทยานิพนธ์ครั้งนี้ที่สามารถสำเร็จลุล่วงไปด้วยดี ต้องขอขอบคุณ อาจารย์ธำรงศักดิ์ สุกใส ผู้เป็นอาจารย์ที่ปรึกษาที่ได้ให้ความช่วยเหลือ และกำลังใจในการทำวิทยานิพนธ์นี้มาตลอด ขอบคุณอาจารย์เกียรติวรรณ ทรงสัตย์ ที่ได้ให้ความสะดวกในการจัดหาอุปกรณ์ ขอบคุณอาจารย์วันชัย ริ้วรุจา ที่ได้ช่วยให้คำปรึกษา ขอบคุณพี่วันเพ็ญ ม่วงเขียว ที่ให้ความสะดวกในการติดต่อ ขอบคุณอาจารย์วรศักดิ์ จิตรภักดิ ที่ได้ให้แนวทางในการทำวิทยานิพนธ์ ขอบคุณรุ่นพี่ ๆ คือ พี่ชัยศรี, พี่อ่าว, พี่โรจน์ และเพื่อน ๆ อันได้แก่ สมมาศ, สุพจน์, นนท์, โอ ขอขอบคุณทุก ๆ ท่านที่ได้ให้ความช่วยเหลือ แต่ไม่ได้กล่าวนามไว้ และสุดท้ายขอขอบพระคุณบิดา, มารดา ผู้ที่ให้กำเนิด และให้กำลังใจในการทำวิทยานิพนธ์

ผู้จัดทำ

เอกสารอ้างอิง

1. "8031 MICROCONTROLLER DATA BOOK"

2. A. Kusko. "Application of microprocessor to AC and DC electricmotor drive system." in Conf. Rec. IEEE Ind. Appl. Soc. Ann. Mect, Oct. 1977, pp. 1079-1081.

3. M. Demerle and J. Fromont. "Speed measure and speed control with a multi-microprocessor system on DC motors," in IEEE-IECI'80 Proc., Mar. 1980 pp. 40-44.

4. E.P. McCarthy. "A digital instantaneous frequency meter." IEEE Trans. Instrum. Meas., vol. IM-28. pp. 224-226. Sept. 1979

5. T. Ohmae et al., "A Microprocessor-controlled fast-response speed regulator with dual mode current loop for DCM drives," IEEE Trans. Ind. Appl. vol. IA-16.pp. 388-394, May/June 1980.