บทที่ 3

ทฤษฎีที่เกี่ยวข้อง

3.1 กล่าวนำ

ไดนาโมมิเตอร์เป็นเครื่องมือเพื่อใช้สำหรับวัดแรงบิด ความเร็วรอบ หรือกำลังงานทางกลของเครื่องจักรกลต่างๆที่มีลักษณะของการหมุน-แบ่งได้เป็น 3 ชนิด คือ ไดนาโมมิเตอร์ชนิดขับอาศัยหลักการส่งผ่านพลังงานไปให้อุปกรณ์ที่ต้องการทดสอบ อย่างเช่น เครื่องสูบน้ำ เครื่องอัดอากาศ พัดลม หรือโบลเวอร์ เพื่อหาค่าแรงบิดของอุปกรณ์ในขณะทำงานหรือใช้หาค่าสมรรถนะของอุปกรณ์นั้นๆ ไดนาโมมิเตอร์ชนิดส่งกำลังใช้สำหรับหาค่าแรงบิดของเครื่องจักรที่มีระบบส่งกำลังประเภท เฟือง สายพาน โซ่ ทำให้รู้การสูญเสียพลังงานทางกลของระบบส่งกำลังในเครื่องจักร ไดนาโมมิเตอร์ชนิดซึมซับอาศัยหลักการของแรงเบรคเพื่อต้านกำลังของเครื่องจักรกลเหมาะสำหรับใช้วัดแรงบิดของเครื่องต้นกำลัง อย่างเช่น เครื่องยนต์ เทอร์ไบน์ และมอเตอร์ไฟฟ้า ไดนาโมมิเตอร์ทั้ง 3 แบบ ยังสามารถแบ่งย่อยไปได้อีกหลายชนิด ในที่นี้จะกล่าวเฉพาะไดนาโมมิเตอร์ชนิดซึมซับ แบบกระแสหมุนวนซึ่งเป็นไดนาโมมิเตอร์ชนิดที่ใช้ในการศึกษาครั้งนี้ เนื่องจากมีหลักการทำงานที่ง่าย การสร้างไม่ซับซ้อน การบำรุงรักษาง่าย และสามารถควบคุมแรงบิดได้อย่างเที่ยงตรง แรงเบรคของไดนาโมมิเตอร์แบบกระแสหมุนวน เกิดจากปฏิกิริยาระหว่างสนามแม่เหล็กที่สร้างขึ้นโดยการจ่ายกระแสไฟฟ้าให้กับขดลวด และสนามแม่เหล็กที่เกิดจากกระแสหมุนวน ที่ถูกเหนี่ยวนำขึ้นบนโรเตอร์ ทำให้ความเข้มของสนามแม่เหล็กโดยรวมลดลงและมีค่าน้อยที่สุดบริเวณกึ่งกลางของโรเตอร์ ดังนั้นการควบคุมแรงบิดสามารถทำได้โดยการควบคุมฟลักซ์แม่เหล็กของขดลวด หรือการควบคุมปริมาณกระแสไฟฟ้าที่จ่ายให้ขดลวด ผู้วิจัยจึงได้นำวงจรปรับแรงดันไฟฟ้าแบบ PWM และตัวต้านทานแบบปรับค่าได้มาประยุกต์ใช้เพื่อควบคุมปริมาณกระแสไฟฟ้า ในส่วนการวัดแรงบิดของไดนาโมมิเตอร์ที่มีใช้กันในปัจจุบัน มักจะใช้หลักของการทำสมดุลแรงที่เกิดขึ้นบนสเตเตอร์ โดยการต่อแขนยื่นออกมาจากสเตเตอร์ เพื่อทำหน้าที่เป็นแขนโมเมนต์ และติดตั้งอุปกรณ์วัดแรงอย่างเช่น โหลดเซล หรือ ตาชั่งสปริง ที่ส่วนปลายของแขน แรงบิดที่กระทำบนสเตเตอร์ก็คือ ผลคูณระหว่างระยะตั้งฉากจากศูนย์กลางแกนเพลาไปยังแนวเส้นการกระทำของแรง กับขนาดของแรงที่อ่านได้จากตาชั่งสปริงหรือโหลดเซล แต่วิธีการวัดแรงบิดทั้ง 2 วิธี มีข้อเสีย คือ ความไม่สะดวกในวิธีการวัดของตาชั่งสปริง เพราะต้องคอยปรับตำแหน่งแขนของสเตเตอร์ให้ตั้งฉากกับแนวแรงอยู่เสมอ และราคาของโหลดเซลที่ยังคงมีราคาสูง

ในวิทยานิพนธ์นี้ผู้วิจัยจึงได้คิดค้นวิธีการวัดแรงบิดรูปแบบใหม่ขึ้นมา ให้สามารถวัดแรงบิดได้ในแบบอัตโนมัติ และมีการทำงานในระบบดิจิตอล อีกทั้งอุปกรณ์ที่ใช้มีราคาที่ไม่สูงมากนัก เพราะเครื่องมือที่สร้างขึ้นได้นำไมโครคอนโทรลเลอร์ซึ่งมีใช้งานกันอย่างกว้างขวางมาใช้ในการทำงาน ดังนั้น ทฤษฎีส่วนใหญ่ในบทนี้จะกล่าวถึงระบบไมโครคอนโทรลเลอร์และการเขียนโปรแกรมเพื่อติดต่อกับไมโครคอนโทรลเลอร์ ส่วนในหัวข้อแรก (หัวข้อที่ 3.2) กล่าวถึงหลักการทำงานของเครื่องไดนาโมมิเตอร์แบบกระแสหมุนวน และหัวข้อที่ 3.3 กล่าวถึงทฤษฎีเกี่ยวกับการออกแบบสปริงเพื่อให้สามารถนำไปคำนวณหาขนาดของสปริงและใช้ประเมินค่าคงที่ของสปริงได้

3.2 การทำงานของเครื่องไดนาโมมิเตอร์ชนิดกระแสหมุนวน

ไดนาโมมิเตอร์ชนิดกระแสหมุนวนเป็นไดนาโมมิเตอร์แบบซึมซับชนิดหนึ่ง ทำหน้าที่เพื่อใช้วัดกำลังทางกลของอุปกรณ์ต้นกำลัง อย่างเช่น เครื่องยนต์เผาไหม้ภายใน หรือมอเตอร์ไฟฟ้า ไดนาโมมิเตอร์ชนิดนี้ใช้หลักการพื้นฐานที่ว่า เมื่อมีวัสดุตัวนำเคลื่อนที่ผ่านสนามแม่เหล็ก จะทำให้เกิดแรงเคลื่อนไฟฟ้าบนตัวนำและส่งผลให้เกิดการไหลของกระแสไฟฟ้าขึ้น ถ้าตัวนำนั้นเป็นสายไฟที่ต่อกันเป็นวงจรปิด ก็จะทำให้เกิดกระแสไฟฟ้าไหลอยู่ภายในวงจรนั้น ถ้าตัวนำเป็นแท่งวัสดุชิ้นหนึ่งและไม่ได้เป็นส่วนหนึ่งในวงจรปิด ก็ยังคงมีแรงเคลื่อนไฟฟ้าถูกเหนี่ยวนำขึ้นบนตัวนำเหมือนเดิม โดยกระแสจะไหลเป็นวงจรเล็กๆอยู่ภายในแท่งตัวนำเรียกกระแสนี้ว่า กระแสหมุนวน ซึ่งจะสูญสลายไปในรูปของความร้อน

ไดนาโมมิเตอร์แบบกระแสหมุนวนประกอบไปด้วยแผ่นจานโลหะแผ่นหนึ่งที่หมุนตัดผ่านสนามแม่เหล็ก สนามแม่เหล็กนี้จะสร้างขึ้นจากคอยล์ที่ถูกกระตุ้นด้วยแหล่งกำเนิดไฟฟ้าจากภายนอก โดยคอยล์นี้จะติดอยู่กับตัวเรือนของไดนาโมมิเตอร์ซึ่งถูกรองรับอยู่บนแบริ่ง ทำให้ตัวเรือนของไดนาโมมิเตอร์หมุนได้อย่างอิสระ เมื่อแผ่นจานโลหะถูกทำให้หมุนตัดผ่านสนามแม่เหล็กด้วยเครื่องต้นกำลัง จะทำให้เกิดการไหลของกระแสหมุนวนขึ้นบนแผ่นจานโลหะ ปฏิกิริยาของสนามแม่เหล็กจะดึงให้ตัวเรือนของไดนาโมมิเตอร์หมุนตามแผ่นจานโลหะไปด้วย แต่การหมุนของตัวเรือนไดนาโมมิเตอร์จะถูกรั้งไว้ด้วยอุปกรณ์วัดแรง อย่างเช่น ตาชั่งสปริง หรือโหลดเซล ซึ่งติดไว้ที่ส่วนปลายของแขนโมเมนต์ที่ต่อยื่นออกมาจากตัวเรือนของไดนาโมมิเตอร์ เมื่อรู้แรงจากการวัดของอุปกรณ์วัดแรงและรัศมี r ของแขนโมเมนต์เราสามารถคำนวณหาแรงบิด (T) ได้จากความสัมพันธ์

$$T = Fr$$

ถ้ารู้ความเร็วรอบเชิงมุมของจานโลหะ สามารถคำนวณหากำลังได้จากความสัมพันธ์

$$P = 2\pi Tn$$

หรือ ทำให้อยู่ในเทอมของแรงม้า

$$bhp = \frac{2\pi Frn}{33000}$$

เมื่อ T คือ แรงบิด,ft-lb

F คือ แรงที่วัดได้ที่รัศมี r ,lb

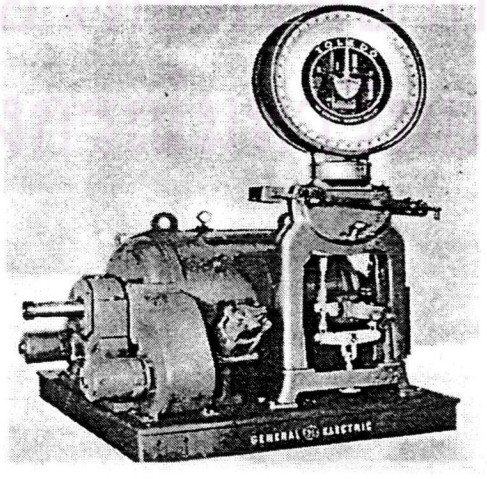
r คือ ความยาวแขนโมเมนต์,ft

P คือ กำลัง, ft-lb/min

bhp คือ แรงม้าเบรก

n คือ ความเร็วในการหมุน , รอบ/นาที

ข้อได้เปรียบของไดนาโมมิเตอร์ชนิดนี้เมื่อเปรียบเทียบกับชนิดอื่นที่กำลังความจุเดียวกัน คือมีขนาดที่ค่อนข้างเล็ก และควบคุมได้ดีที่ความเร็วรอบต่ำ ในรูปที่ 3.1 แสดงภาพ Cradle d-c dynamometer ขนาดความจุ 5000 hp ใช้ประโยชน์เป็นไดนาโมมิเตอร์ได้ทั้ง 2 แบบ คือ แบบซึมซับ และแบบขับ โดยพื้นฐานแล้วอุปกรณ์นี้ก็คือ d-c motor / generator ที่มีระบบควบคุมเพื่อใช้เปลี่ยนโหมดการทำงาน เมื่อใช้เป็นไดนาโมมิเตอร์แบบซึมซับ การทำงานจะเป็น d-c generator พลังงานกลที่เข้ามาจะถูกแปลงไปเป็นพลังงานไฟฟ้าและสูญสลายไปในตัวต้านทาน ซึ่งจะแตกต่างจากไดนาโมมิเตอร์แบบกระแสหมุนวนตรงที่พลังงานความร้อนที่สูญเสียไปมาจากภายนอกเครื่องจักร การวัดแรงบิดสามารถวัดได้ทั้งสองทิศทางขึ้นอยู่กับทิศของการหมุนและโหมดการทำงาน เมื่อถูกกำหนดให้ทำงานเป็นไดนาโมมิเตอร์ชนิดขับ อุปกรณ์จะถูกใช้เหมือน d-c motor ปัญหาที่สำคัญของ d-c motor คือ ไม่มีแหล่งจ่ายไฟที่มากพอให้กับ d-c motor จึงต้องใช้ชุดของ a-c motor มาขับ d-c generator หรือใช้ rectifier เพื่อแปลงไฟกระแสสลับให้เป็นไฟกระแสตรง



รูปที่ 3.1 Cradle d-c Dynamometer

โดยทั่วไปแล้ว มอเตอร์ไฟฟ้าและ เจนเนอเรเตอร์ อาจถูกประยุกต์เพื่อใช้เป็นไดนาโมมิเตอร์ก็ได้ และจะมีการทำงานที่ดี หากใช้ชนิดที่เป็น dc มากกว่าที่จะใช้เป็นแบบ ac การนำไปใช้เป็นไดนาโมมิเตอร์แบบซึมซับหรือ แบบขับ อาจจะทำได้โดยแขวนมอเตอร์หรือเจนเนอเรเตอร์ไว้บนแบริ่ง การวัดแรงบิด ความเร็ว และกำลังงานก็อาจจะคำนวณได้เช่นเดียวกัน แต่ทั้งนี้การออกแบบและการประกอบก็ต้องทำให้เกิดแรงเสียดทานให้น้อยที่สุดด้วย การปรับความเร็วและโหลดสามารถทำได้โดยผ่านระบบควบคุมการสร้างสนามแม่เหล็ก

ถ้า d-c generator ถูกนำไปใช้เป็นไดนาโมมิเตอร์แบบซึมซับ

$$\text{hp absorbed} = \frac{1.341 \times kilowatts}{efficiency}$$

ในทำนองเดียวกัน หาก dc motor ถูกใช้เป็นไดนาโมมิเตอร์ชนิดขับ

$$\text{hp input} = 1.341 \times kilowatts \times efficiency = \frac{1.341 \times (e)(i)(efficiency)}{1000}$$

เมื่อ e คือ แรงดันไฟฟ้า และ i คือ กระแสไฟฟ้า ในทั้ง 2 กรณี

$$Torque = \frac{33{,}000hp}{2\pi n}$$

ในการนำไปใช้งานทั่วไป อาจจะต้องการเพียงแค่ผลการวัดอย่างหยาบ ซึ่งประสิทธิภาพของมอเตอร์หรือ เจนเนอเรเตอร์จากโรงงานผู้ผลิตก็เพียงพอต่อการนำไปใช้ได้ แต่สำหรับผลการวัดที่มีความถูกต้องมากขึ้น การทำเป็นไดนาโมมิเตอร์นั้นก็จำเป็นต้องใช้เครื่องจักรที่มีประสิทธิภาพมากขึ้น โดยปกติแล้ว การประยุกต์ใช้เครื่องจักรกลหมุนทางไฟฟ้ามาทำเป็นไดนาโมมิเตอร์นั้นจำเป็นต้องพิจารณาให้เป็นพิเศษ และผลที่ได้จากการประยุกต์ใช้นั้นก็ยังไม่ดีเท่ากับอุปกรณ์ที่ได้ออกแบบมาเพื่อใช้งานเป็นไดนาโมมิเตอร์โดยเฉพาะ

3.3 การออกแบบสปริง

จุดประสงค์หลักของการใช้สปริง จะอยู่ในรูปการของการเก็บพลังงานเอาไว้ในตัวสปริง ความเค้นที่เกิดขึ้นในสปริงขณะใช้รับแรงจะมีค่าสูงมาก วัสดุที่นำมาใช้ทำสปริงจึงต้องมีความแข็งแรงสูง โดยทั่วไปเหล็กที่ใช้ทำสปริง จะเป็นเหล็กที่มีคาร์บอนสูงกว่า 0.5 % แล้วผ่านกรรมวิธีทางความร้อนเพื่อให้มีความยืดหยุ่นสูง เนื่องจากความสามารถในการยืดหยุ่นเป็นคุณสมบัติที่สำคัญของสปริง

3.3.1 คุณสมบัติทางกลของขดลวดสปริง

ค่าความต้านทานแรงของวัสดุที่ใช้ทำขดลวดสปริงจะเปลี่ยนไปตามขนาดของขดลวดสปริง รูปสมการที่ใช้หาค่าความต้านแรงของวัสดุลวดสปริงที่มีขนาดเส้นผ่านศูนย์กลางใดๆ มีดังนี้

$$\sigma_u = \frac{A}{d^x} \qquad (3.1)$$

$$\tau_n = \frac{B}{d^y} \qquad (3.2)$$

โดยที่ คือ σ_u ความต้านแรงดึงต่ำสุด (N/mm^2)

คือ τ_n ความต้านแรงเฉือนทนทาน (N/mm^2)

คือ d ขนาดเส้นผ่านศูนย์กลางของขดลวดสปริง (mm)

ค่าคงที่โดยประมาณของสมการที่ 3.1 และสมการที่ 3.2 แสดงไว้ในตารางที่ 3.1 ค่าความต้านแรงมีหน่วยเป็น N/mm^2

ชนิดของวัสดุ	ขนาดลวด (mm)	x	y	A	B
Hard drawn wire (ASTM A227)	0.50-16.00	0.190	0.340	1780	560
Music wire (ASTM A228)	0.10-6.35	0.154	0.154	2150	565
Oil tempered wire (ASTM A229)	0.50-16.00	0.190	0.340	1855	560
Valve spring steel (ASTM 230)	1.50-6.25	0.100	0.150	1730	515
Cr-V steel (ASTM 231)	0.50-12.50	0.166	0.150	1976	515
Cr-Si steel (ASTM A401)	0.80-12.00	0.107	0.150	1965	515
Stainless steel (ASTM A313)	0.20-12.50	0.140	0.170	1840	360

ตารางที่ 3.1 คุณสมบัติทางกลของขดลวดสปริง

สำหรับค่าโมดูลัสความยืดหยุ่นและโมดูลัสเฉือนของวัสดุในตารางที่ 3.1 ให้ใช้ค่าประมาณดังนี้

E = 200 kN/mm^2 G = 80 kN/mm^2

สำหรับเหล็กกล้าไร้สนิมให้ใช้

E = 180 kN/mm^2 G = 70 kN/mm^2

สำหรับค่าความต้านแรงเฉือนครากให้ใช้ค่าประมาณ

$\tau_y = 0.6\sigma_u$ สำหรับวัสดุทั่วไป

$\tau_y = 0.47\sigma_u$ สำหรับเหล็กกล้าไร้สนิม

3.3.2 ความเค้นในสปริงขดรับแรงกด

พิจารณาสปริงขดซึ่งรับแรงกด F ในแนวแกนของสปริงดังรูปที่ 3.2 และกำหนดให้

D_o คือ เส้นผ่านศูนย์กลางภายนอกของขดสปริง (mm)

D_i คือ เส้นผ่านศูนย์กลางภายในของขดสปริง (mm)

$D = (D_o + D_i)/2$ เป็นเส้นผ่านศูนย์กลางเฉลี่ยของขดสปริง (mm)

d คือเส้นผ่านศูนย์กลางของขดสปริง (mm)

รูปที่ 3.2 สปริงขด

เมื่อตัดส่วนหนึ่งของสปริงมาพิจารณาจะเห็นว่า ลวดสปริงอยู่ภายใต้แรงภายใน F และโมเมนต์ T ค่าความเค้นเฉือนสูงสุดที่เกิดขึ้นในขดลวดสปริง คือ

$$\tau = \frac{Tr}{J} + \frac{F}{A} \qquad (3.3)$$

แต่ $$T = \frac{FD}{2}$$

และ $$\frac{J}{r} = \frac{(\pi d^4/32)}{d/2}$$

$$= \frac{\pi d^3}{16}$$

$$A = \frac{\pi d^2}{4}$$

จึงเขียนสมการที่ 3.3 ใหม่ได้เป็น

$$\tau = \frac{8FD}{\pi d^3} + \frac{4F}{\pi d^2} \qquad (3.4)$$

ถ้าให้ C=D/d ซึ่งเรียกว่าดัชนีสปริง (spring index) สมการที่ 3.4 สามารถจัดรูปใหม่ได้เป็น

$$\tau = K_s \frac{8FD}{\pi d^3} \qquad (3.5)$$

โดยที่ $K_s = 1 + \frac{0.5}{C}$

ค่า K_s นี้เรียกตัวประกอบความเค้นเฉือน (shear stress correction factor) ซึ่งเป็นค่าทำให้ความเค้นเฉือนในสปริงเพิ่มขึ้น อันเนื่องมาจากความเค้นเฉือนตรง F/A

เนื่องจากสูตรโมเมนต์บิด Tr / J เป็นสูตรสำหรับการบิดชิ้นส่วนตรง เช่น เพลา แต่ในสปริงนั้น การที่เส้นลวดบิดวนไปตามส่วนโค้งของสปริงจะทำให้เกิดความเค้นหนาแน่นสูงสุดบริเวณด้านในของสปริงขด จึงได้มีการแก้ไขความเค้นในสมการที่ 3.5 โดยเพิ่มตัวประกอบความโค้ง (curvature correction factor) K_C เข้าไปในสมการ ค่าตัวประกอบความเค้นหนาแน่นนี้สามารถคำนวณได้จากสมการ

$$K = K_C K_S$$

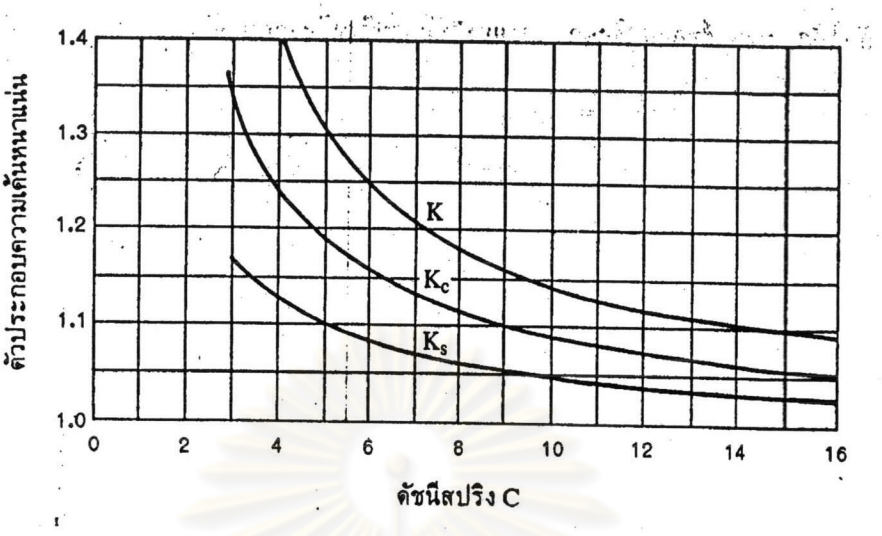
$$= \frac{4C-1}{4C-4} + \frac{0.615}{C} \qquad (3.6)$$

ค่า K นี้เรียกตัวประกอบของวาห์ล (wahl 's factor) สามารถหาค่าได้โดยการคำนวณจากสมการที่ 3.6 หรืออ่านค่าได้จากรูปที่ 3.3 ดังนั้นความเค้นเฉือนในสมการที่ 3.5 จึงกลายเป็น

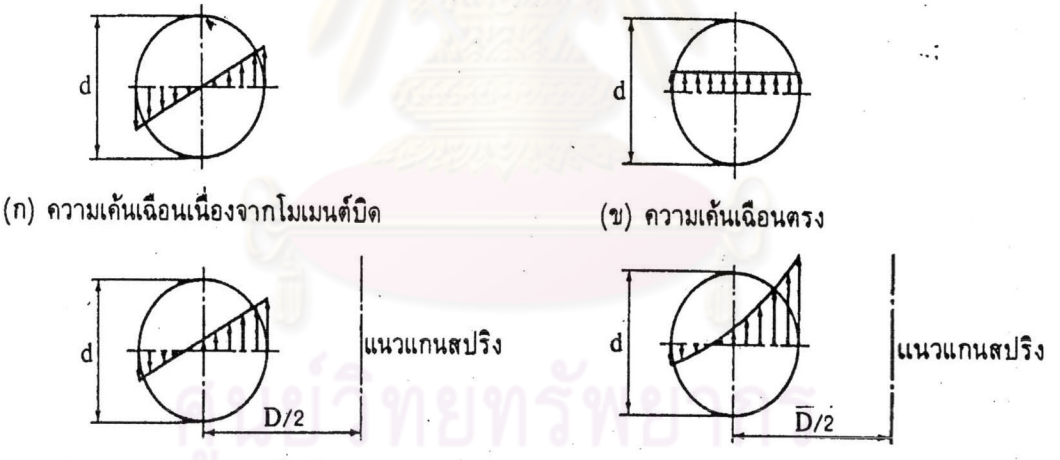
$$\tau = K \frac{8FD}{\pi d^3}$$

$$\tau = K \frac{8FC}{\pi d^2} \qquad (3.7)$$

การกระจายความเค้นเฉือนในเส้นลวดสปริงซึ่งเกิดจากโมเมนต์บิด T และแรงเฉือน F จะดูได้จากรูปที่ 3.4ก และ 3.4ข เมื่อรวมความเค้นเฉือนที่เกิดขึ้นดังรูปที่ 3.4ก และรูปที่ 3.4ข ก็จะได้ความเค้นเฉือนดังรูปที่ 3.4ค แต่เมื่อรวมความหนาแน่นอันเนื่องมาจากความโค้งของลวดสปริงเข้าไปด้วยกันจะได้ดังรูปที่ 3.4ง ซึ่งจะเห็นได้ว่าความเค้นเฉือนสูงสุดเกิดขึ้นที่ด้านในของลวดสปริง



รูปที่ 3.3 ค่าตัวประกอบ K

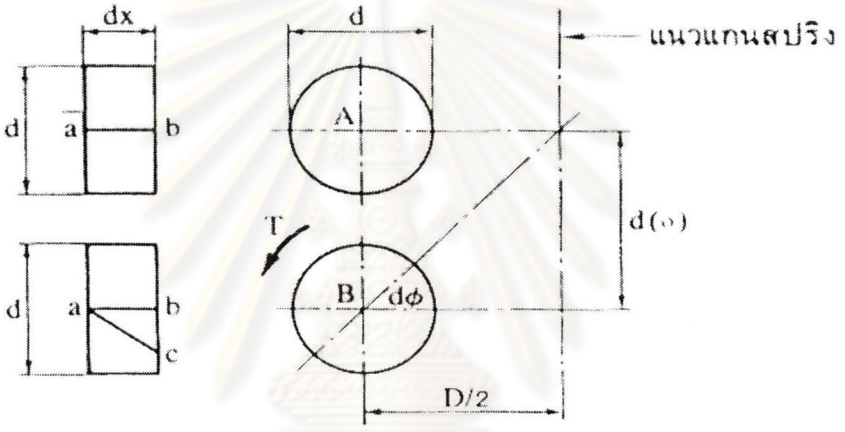


รูปที่ 3.4 การรวมความเค้นในสปริงขด

3.3.3 การยืดหดของสปริงขด

พิจารณาขดสปริงซึ่งตัดออกมาเป็นชิ้นเล็กๆ ยาว dx ก่อนมีแรงมากระทำในแนวดิ่ง ส่วนของลวดสปริงนี้จะอยู่ที่ A หลังจากรับแรงภายนอกแล้วลวดสปริงจะเลื่อนมาอยู่ที่ตำแหน่ง B โดยมีการยุบตัว $d\delta$ และมุมบิดเล็กๆของลวดเท่ากับ $d\phi$ นั่นคือเส้น ab จะบิดมาอยู่ที่ตำแหน่ง ac จากสูตรการบิด

$$d\phi = \frac{Tdx}{GJ}$$



รูปที่ 3.5 การยืดหดของลวดสปริง

ถ้ามุมบิดบนลวดสปริงชิ้นเล็กๆนี้เล็กมาก (เนื่องจากสปริงมีขดถี่) ระยะยุบตัว คือ

$$d\delta = \frac{D(d\phi)}{2} = \frac{TD(dx)}{2GJ}$$

ถ้าสปริงมีความยาวทั้งหมด L ระยะยุบตัวทั้งหมด คือ

$$\delta = \frac{TD}{2GJ}\int_0^L dx = \frac{TDL}{2GJ} \qquad (3.8)$$

ถ้าจำนวนขดทำการ (active coils) เท่ากับ n ขด ความยาว L ของสปริง n ขด โดยประมาณเท่ากับ πDn

และเมื่อแทนค่า T=FD/2 และ $J = (\pi/32)d^4$ ลงในสมการที่ 3.8 จะได้ว่า

ระยะยุบตัว $$\delta = \frac{8FD^3n}{Gd^4} = \frac{8FC^3n}{Gd} \qquad (3.9)$$

3.3.4 ความแข็งตึงของสปริง

ความแข็งตึงของสปริง (spring stiffness) หมายถึงปริมาณของแรงที่ใช้ในการทำให้สปริงยืดหดหนึ่งหน่วยความยาวในแนวแกนของสปริง ถ้าให้ δ เป็นระยะยุบตัวภายใต้แรงกด F ค่าคงที่ของสปริง คือ

$$k = \frac{F}{\delta} \tag{3.10}$$

แทนค่าสมการที่ 3.9 ลงในสมการที่ 3.10 จะได้

$$k = \frac{Gd^4}{8D^3n} = \frac{Gd}{8C^3n} \tag{3.11}$$

3.3.5 สปริงขดแบบดึง

การคำนวณสปริงขดแบบดึงจะใช้สูตรแบบเดียวกับการคำนวณสปริงขดแบบกด แตกต่างกันตรงที่แรงที่ใช้ดึงสปริงขดแบบดึงนั้นจะมีแรงดึงชั้นต้นเพิ่มเข้ามา ซึ่งหมายถึงแรงที่ใช้ดึงสปริงจนกระทั่งถึงจุดที่ขดแยกออกจากกัน(ยังไม่มีการยืด) ดั้งนั้น กฎของฮุคจะใช้ไม่ได้เลยจนกว่าแรงดึงภายนอกจะเอาชนะแรงดึงชั้นต้นแล้ว

ถ้าให้ F คือ แรงภายนอกทั้งหมดที่ใช้ดึงสปริง

F_i คือ แรงดึงชั้นต้น

$$F = F_i + k\delta \tag{3.12}$$

สมการที่ 3.7 และสมการที่ 3.9 ก็ยังสามารถนำมาใช้คำนวณหาค่าความเค้นเฉือนและระยะยืดตัวได้ แต่แรงที่ทำให้เกิดความเค้นในสปริงจริงๆ เท่ากับ $F - F_i$ ถ้าสปริงไม่มีแรงดึงชั้นต้นแล้วสมการที่ผ่านมาก็นำมาใช้ได้โดยไม่มีการเปลี่ยนแปลงใดๆ ความเค้นเนื่องจากแรงดึงชั้นต้นหาได้จากสมการที่ 3.7 โดยให้ค่า K=1 (ไม่มีผลจากความเค้นหนาแน่น) ตัวอย่างของความเค้นขั้นต้น τ_i ของผู้ผลิตสปริงขดแห่งหนึ่งแสดงในตารางที่ 3. 2 ซึ่งมีค่าเปลี่ยนแปลงไปตามดัชนีสปริง

C	3	4	5	6	7	8	9	10	11	12	15
τ_i N/mm^2	165	155	138	124	112	100	90	80	73	67	48

ตารางที่ 3.2 ความเค้นขั้นต้นของสปริงขดแบบดึง

ถึงแม้ว่าสปริงขดแบบดึงจะคล้ายกับแบบกด และสามารถใช้สมการในการคำนวณร่วมกันได้ แต่มีข้อควรระวังในการออกแบบ เพราะจะมีความแตกต่างกันดังนี้คือ

ก). สปริงขดแบบดึงสามารถที่จะดึงให้ยืดออกได้มากจนความเค้นเฉือนที่เกิดขึ้นสูงกว่าความต้านแรงเฉือนครากของวัสดุ แต่สปริงขดแบบกดจะยุบตัวลงได้ถึงความยาวแข็งตัวเท่านั้น

ข). ปลายของสปริงขดแบบดึงจะต้องงอให้เป็นขอสำหรับเกี่ยว ทำให้เกิดความเค้นดัดที่ขอ และค่าความหนาแน่นที่ขอก็อาจมีค่าสูงมากได้

จากข้อแตกต่างระหว่างสปริงขดแบบดึงและแบบกดดังกล่าวมาแล้ว จึงแนะนำให้ใช้ค่าความเค้นเฉือนออกแบบสำหรับสปริงขดแบบดึงเพียง 80 % ของค่าที่ให้ไว้ในตารางที่ 3.3

วัสดุลวดสปริง	ความเค้นเฉือนออกแบบ τ_d		
	งานเบา	งานปานกลาง	งานหนัก
Hard drawn wire (ASTM A227)	$0.344\sigma_u$	$0.275\sigma_u$	$0.244\sigma_u$
Stainless steel (ASTM A313)	$0.320\sigma_u$	$0.260\sigma_u$	$0.210\sigma_u$
วัสดุอื่นๆ ในตารางที่ 3.1	$0.405\sigma_u$	$0.324\sigma_u$	$0.263\sigma_u$

ตารางที่ 3.3 ค่าความเค้นเฉือนออกแบบสำหรับวัสดุสปริง

สัดส่วนโดยทั่วไปของสปริงขดแบบดึงที่นิยมใช้กันจะดูได้จากรูปที่ 3.5

โดยที่ $D_i = D - d$

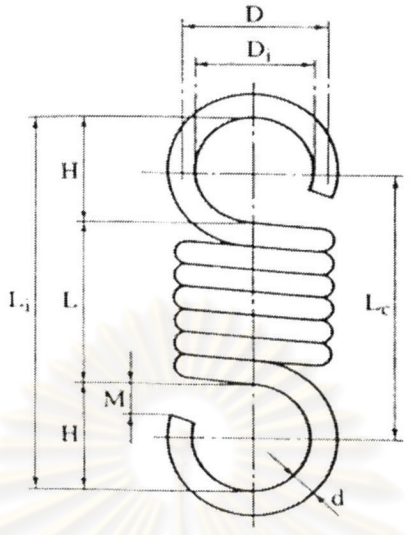
$H = D_i$ สำหรับขอกลม (full hook)

$M = D_i / 3$

$L = d(n+1)$

$L_i = L + 2D_i$

$L_c = L + D_i$

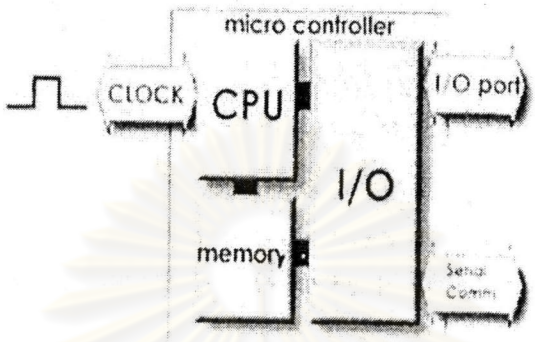


รูปที่ 3.6 สปริงขดแบบดึงซึ่งมีขอกลมในแนวศูนย์กลางขด

3.4 โครงสร้างของระบบไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชั่นการทำงานต่างๆ ไว้ภายในตัวของมันเอง สามารถโปรแกรมการทำงานได้หลายครั้ง และรับข้อมูลในรูปสัญญาณดิจิตอลเข้าไปทำการประมวลผลแล้วส่งผลลัพท์ข้อมูลดิจิตอลออกมาเพื่อนำไปใช้งานตามที่ต้องการได้ เปรียบเสมือนเป็นเครื่องคอมพิวเตอร์ขนาดเล็กที่สร้างขึ้นมาให้อยู่ในไอซีเพียงตัวเดียว นั่นคือ โครงสร้างภายในไมโครคอนโทรลเลอร์นั้นจะประกอบไปด้วย ระบบประมวลผลกลาง (CPU) ,ระบบหน่วยความจำ (memory), ระบบอินพุตเอาท์พุต (I/O) อยู่ในตัวเองทั้งหมด หรืออาจจะเรียกได้ว่าเป็นไมโครโพรเซสเซอร์ชิปเดี่ยว (Single-Chip Microprocessor) เช่นเดียวกับหน่วยประมวลผลกลาง (CPU: Central Processing Unit) ที่ใช้ในคอมพิวเตอร์ แต่ได้รับการพัฒนาแยกออกมาภายหลังเพื่อนำไปใช้ในวงจรทางด้านงานควบคุม คือ แทนที่ในการใช้งานจะต้องต่อวงจรภายนอกต่าง ๆ เพิ่มเติม แต่จะทำการรวมวงจรที่จำเป็น เช่น หน่วยความจำ, ส่วนอินพุต/เอาท์พุต บางส่วนเข้าไปในตัวไอซีเดียวกัน และเพิ่มวงจรบางอย่างเข้าไปด้วยเพื่อให้มีความสามารถเหมาะกับการใช้ในงานควบคุม เช่น วงจรตั้งเวลา วงจรการสื่อสารอนุกรม (Serial Communication) และวงจรที่ทำหน้าที่ทางด้านการแปลงสัญญาณระหว่างแบบอนาล็อกและดิจิตอล (A/D Converter) จึงทำให้มีความนิยมใช้ไมโครคอนโทรลเลอร์ไปควบคุมการทำงานของอุปกรณ์ทางด้านอิเล็กทรอนิกส์และคอมพิวเตอร์ โดยมักจะเป็นการนำไปใช้ฝังในระบบของ

อุปกรณ์อื่น ๆ (Embedded Systems) เพื่อใช้ควบคุมการทำงานบางอย่าง เช่น ใช้ในรถยนต์, เครื่องปรับอากาศ, เครื่องซักผ้าอัตโนมัติ เป็นต้น

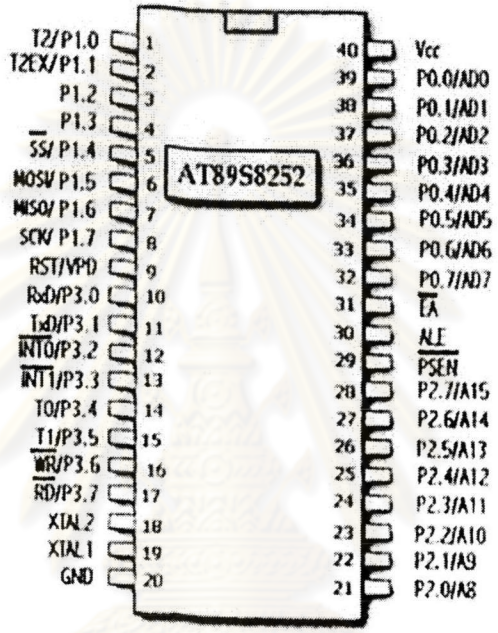


รูปที่ 3.7 แสดงโครงสร้างหลักๆของไมโครคอนโทรลเลอร์

ปัจจุบันเทคโนโลยีทางด้านการสร้างไมโครคอนโทรลเลอร์แบบชิพเดี่ยว (Single Chip) ก้าวหน้าไปมาก คือในไอซีเบอร์ 80C51/80C52 เป็นไมโครคอนโทรลเลอร์ที่สามารถโปรแกรมเก็บเอาไว้ในตัวได้เลย แต่การโปรแกรมนั้นต้องอาศัยเครื่องมือในการเขียนจากโรงงานผลิต ซึ่งไม่เหมาะสำหรับนักพัฒนาทั่วๆไป ทางผู้ผลิตจึงปรับเปลี่ยนรูปแบบการโปรแกรมไอซีมาเป็นแบบ EPROM แต่ก็มีปัญหาในเรื่องการลบข้อมูลในตัวชิพ เพราะต้องอาศัยเครื่องลบแบบแสง UV ต่อมาความต้องการใช้งานไอซีไมโครคอนโทรเลอร์มีมากขึ้น รวมทั้งการผลิตชิ้นงานในโรงงาน และทางด้านการศึกษา จึงได้พัฒนาเป็นไอซีไมโครคอนโทรลเลอร์ ที่เพิ่มคุณสมบัติของหน่วยความจำโปรแกรมภายในเป็นแบบแฟลช (Flash Memory) ซึ่งมีคุณสมบัติสามารถจะลบล้างข้อมูลได้โดยใช้สัญญาณไฟฟ้า และมีการโปรแกรมใหม่ได้ถึงกว่า 1000 ครั้ง ในการลบข้อมูลแต่ละครั้งจะใช้เวลาไม่ถึง 1 นาที โดยจะนำไอซีไปโปรแกรมข้อมูลผ่านเครื่องโปรแกรม การใช้งานก็สะดวกรวดเร็วมากขึ้น โดยเฉพาะในรุ่น AT89Sxxxx สามารถที่จะทำการโปรแกรมข้อมูลลงในหน่วยความจำโปรแกรมได้ โดยที่ไม่ต้องถอดตัวไอซีไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมข้างนอกบอร์ด ทั้งนี้เพราะไมโครคอนโทรลเลอร์รุ่นนี้ถูกออกแบบให้สามารถทำการโปรแกรมชิพด้วยระบบ In-system reprogrammable โดยอาศัยวงจร SPI(Serial Peripheral Interface) ในการเขียนและลบ จึงทำให้การพัฒนาและปรับปรุงข้อมูลในหน่วยความจำโปรแกรมที่อยู่ภายในตัวไอซีทำได้สะดวกและรวดเร็วมากยิ่งขึ้น

3.4.1 การจัดขาของไอซี

ไมโครคอนโทรลเลอร์ในตระกูล MCS-51 เป็นไอซีขนาด 40 ขา ซึ่งรายละเอียดและหน้าที่ของขาต่างๆเป็นดังนี้



รูปที่ 3.8 แสดงการจัดขาของไมโครคอนโทรลเลอร์เบอร์ AT89S8252

ขาที่ 1 ถึง 8 คือ ขาพอร์ต 1 เริ่มจาก P1.0 – P1.7 ทำหน้าที่เป็นพอร์ตอินพุต/เอาท์พุต

ขาที่ 9 คือ ขา RST เป็นขารีเซต

ขาที่ 10-17 คือ ขาพอร์ต 3 เป็นได้ทั้งพอร์ตอินพุต/เอาท์พุต นอกจากนี้แต่ละขายังทำหน้าที่พิเศษอีกหลายอย่างดังนี้

ขาที่ 10 RXD ใช้ในการรับข้อมูลทาง Port อนุกรม

ขาที่ 11 TXD ใช้ในการส่งข้อมูลทาง Port อนุกรม

ขาที่ 12 INT0 รับสัญญาณอินเทอร์รัพท์จากอุปกรณ์ภายนอกตัวไมโครคอนโทรลเลอร์

ขาที่ 13 INT1 รับสัญญาณอินเทอร์รัพท์จากอุปกรณ์ภายนอกตัวไมโครคอนโทรลเลอร์

ขาที่ 14 T0 ใช้งานกับไทเมอร์/เคาน์เตอร์ 0

ขาที่ 15 T1 ใช้งานกับไทเมอร์/เคาน์เตอร์ 1

ขาที่ 16 WR ทำส่งสัญญาณเพื่อบอกแก่หน่วยความจำข้อมูลว่ากำลังส่งข้อมูลจากบัสข้อมูลไปเก็บในหน่วยความจำ ณ ตำแหน่งที่ชี้โดยแอดเดรสบัส

ขาที่ 17 RD ทำส่งสัญญาณเพื่อบอกแก่หน่วยความจำข้อมูลว่าต้องการอ่านข้อมูลจากหน่วยความจำ ณ ตำแหน่งที่ชี้โดยแอดเดรสบัส ผ่านทางบัสข้อมูล

ขาที่ 18-19 คือขา XTAL2 , XTAL1 เป็นขาที่ใช้ต่อกับวงจรกำเนิดสัญญาณสำหรับผลิตความถี่ให้กับ CPU การทำงานของ CPU จะช้าหรือเร็วขึ้นอยู่ความถี่ XTAL ที่นำมาใช้

ขาที่ 20 คือขาที่ใช้สำหรับต่อลงกราวด์

ขาที่ 21-28 คือขาพอร์ต 2 สามารถใช้เป็นได้ทั้งพอร์ตอินพุตและเอาท์พุต นอกจากนี้ยังทำหน้าที่อีกอย่าง คือ เป็นแอดเดรสบัสที่ส่งไบต์สูงออกจากตัวไมโครคอนโทรลเลอร์

ขาที่ 29 คือขา PSEN ถ้าขานี้มีสถานะเป็น 0 หมายความว่า ไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลโปรแกรมจากหน่วยความจำโปรแกรมภายนอก

ขาที่ 30 คือขา ALE เป็นขาที่มีหน้าที่บอกให้รู้ว่ามีการส่งแอดเดรสออกมาจากพอร์ต 0 ของไมโครคอนโทรลเลอร์ ทั้งนี้เพราะ พอร์ต 0 สามารถที่จะส่งได้ทั้งแอดเดรสของหน่วยความจำ และข้อมูล

ถ้า ALE = 1 แสดงว่า ไมโครคอนโทรลเลอร์ส่งแอดเดรสออกมาให้กับอุปกรณ์อื่นๆ

ถ้า ALE = 0 แสดงว่า ไมโครคอนโทรลเลอร์ส่งข้อมูลออกมาให้กับอุปกรณ์อื่นๆ

ขาที่ 31 คือขา EA เป็นขาที่ใช้กำหนดว่า ในการทำงานนั้นให้อ่านข้อมูลโปรแกรมจากภายในตัวไมโครคอนโทรลเลอร์หรืออ่านจากหน่วยความจำโปรแกรมภายนอก

ถ้า EA = 0 แสดงว่า ต้องการให้ไมโครคอนโทรลเลอร์อ่านข้อมูลจากภายในตัวชิพ

ถ้า EA=1 แสดงว่า ต้องการให้ไมโครคอนโทรลเลอร์อ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

ขาที่ 32-39 คือพอร์ต 0 เป็นได้ทั้งพอร์ตอินพุตและเอาท์พุต นอกจากนี้ยังทำหน้าที่เป็นพอร์ตที่ส่งแอดเดรสที่เป็นไบต์ต่ำ และรับส่งข้อมูล

ขาที่ 40 คือขา V_{CC} เป็นขาไฟเลี้ยง 5V

3.5 โครงสร้างพื้นฐานของโปรแกรมภาษาซีที่ใช้กับไมโครคอนโทรลเลอร์ MCS-51

ในการเขียนโปรแกรมเพื่อติดต่อกับไมโครคอนโทรลเลอร์ สิ่งจำเป็นต้องทราบในเบื้องต้นคือโครงสร้างพื้นฐานของโปรแกรม เพื่อให้เข้าใจส่วนประกอบต่างๆภายในโปรแกรม ส่วนประกอบที่สำคัญของโปรแกรมสามารถสรุปได้ดังนี้

1). การประกาศส่วน Preprocessor

ผู้เขียนโปรแกรมต้องประกาศส่วนนี้เป็นส่วนแรกในโปรแกรมภาษาซี โดยขึ้นต้นด้วยเครื่องหมาย (#) แล้วตามด้วยไดเร็กตีฟ include หรือ define เพื่อให้คอมไพเลอร์นำไปใช้ตีความและเลือกไมโครคอนโทรลเลอร์ที่เหมาะสม ทำให้คอมไพเลอร์รู้จักกับรีจิสเตอร์และบิทควบคุมต่างๆภายในไมโครคอนโทรลเลอร์ที่ติดต่อด้วย

ผู้เขียนโปรแกรมจะต้องมีการประกาศใช้ทรัพยากรของไมโครคอนโทรลเลอร์เสียก่อน เพื่อให้คอมไพเลอร์สามารถเข้าถึงทรัพยากรเหล่านั้นได้ โดยไม่ทำให้เกิดข้อผิดผลาดจนเกิดการแจ้งเตือนในขั้นตอนของการแปลซอร์สโปรแกรม(source program)ไปเป็นภาษาเครื่อง โดยทรัพยากรในที่นี้หมายถึงความจุของหน่วยความจำโปรแกรม,หน่วยความจำข้อมูล,รีจิสเตอร์ควบคุม และบิทควบคุม ขึ้นอยู่กับเบอร์ของไมโครคอนโทรลเลอร์ที่ใช้งานด้วยว่ามีความสามารถพิเศษเพิ่มเติมอะไรบ้าง แต่โดยปกติแล้วไมโครคอนโทรลเลอร์ MCS-51 แต่ละเบอร์จะมีรีจิสเตอร์พื้นฐานที่เหมือนกัน

ในการประกาศใช้ทรัพยากรของไมโครคอนโทรเลอร์ ผู้เขียนโปรแกรมจะต้องประกาศด้วยไดเร็กตีฟ #include ที่ส่วนหัวของโปรแกรม แล้วตามด้วยชื่อไฟล์ไลบรารีที่ต้องการเข้าถึง ตัวอย่างไฟล์ไลบรารีของไมโครคอนโทรลเลอร์ MCS-51 และเบอร์ที่ได้รับความนิยมอย่างแพร่หลาย ได้แก่

#include<reg51.h>ใช้ประกาศให้คอมไพเลอร์รู้จักรีจิสเตอร์และบิทควบคุมพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ส่วนใหญ่ใช้งานกับไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89C51/AT891051/ AT892051/AT894051 ฯลฯ ของ Atmel ที่ไม่มีไทเมอร์ 2

#include<reg52.h>ใช้ประกาศให้คอมไพเลอร์รู้จักรีจิสเตอร์และบิทควบคุมพื้นฐานของไมโครคอน โทรลเลอร์ MCS-51 ที่มีไทเมอร์ 2 ส่วนใหญ่ใช้งานกับไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89C52/ AT89SS52/AT89s53

#include<reg8252.h>ใช้ประกาศให้คอมไพเลอร์รู้จักรีจิสเตอร์และบิทควบคุมทั้งหมดที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89S8252 ของ Atmel

#include<C51rd2.h>ใช้ประกาศให้คอมไพเลอร์รู้จักรีจิสเตอร์และบิทควบคุมทั้งหมดที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 เบอร์ P89C51RD2 ของ Phillips

#include<C51ac2.h>ใช้ประกาศให้คอมไพเลอร์รู้จักรีจิสเตอร์และบิทควบคุมทั้งหมดที่มีอยู่ในไมโครคอนโทรลเลอร์ MCS-51 เบอร์ T89C51AC2 ของ Atmel

2). ส่วนประกาศไลบรารีของโปรแกรม

ไลบรารี คือ ไฟล์ที่มีฟังก์ชันบรรจุอยู่ ไฟล์ไลบรารีของภาษาซีจะมีนามสกุลเป็น .h ในการประกาศจะใช้ไดเร็กตีฟ #include ต่อด้วยไฟล์ไลบรารีนามสกุล .h ตัวอย่างเช่น

#include<math.h> ใช้ประกาศเพื่อใช้งานฟังก์ชันคำนวณทางคณิตศาสตร์

#include<string.h> ใช้ประกาศเพื่อใช้งานฟังก์ชันจัดการตัวแปรแบบตัวอักษร

#include<absacc.h>ใช้ประกาศเพื่อใช้งานฟังก์ชันเข้าถึงหน่วยความจำของไมโครคอนโทรลเลอร์

นอกจากนั้นผู้เขียนโปรแกรมสามารถสร้างไลบรารีเพื่อใช้งานได้เอง โดยเขียนฟังก์ชันที่ต้องการแล้วบันทึกให้มีนามสกุลเป็น .h เก็บไว้ที่โฟลเดอร์เดียวกับไฟล์โปรเจ็ก หรือโฟล์เดอร์ที่เก็บไลบรารีของคอมไพเลอร์ก็ได้ เมื่อต้องการเรียกใช้งาน ก็ใช้ไดเร็กตีฟ #include ประกาศไฟล์ไลบรารีที่ต้องการไว้ที่ส่วนหัวของโปรแกรม

3). ส่วนประกาศค่าคงที่ในโปรแกรม

ไดเร็กตีฟ #define ใช้สำหรับประกาศข้อความเพื่อให้แทนค่าคงที่ค่าหนึ่งในโปรแกรม โดยจะประกาศต่อท้ายบรรทัด #include การใช้งานไดเร็กตีฟ #define อาจจะมีหรือไม่มีก็ได้ ขึ้นอยู่กับความจำเป็น ตัวอย่างในการใช้งาน เช่น

#define pi 3.14159 ให้ pi แทนตัวเลข 3.14159

#define on 1 ให้ on แทนตัวเลข 1

4). ส่วนประกอบการใช้งานร่วมทั้งโปรแกรม (global declarations)

เป็นส่วนที่มีการประกาศตัวแปรแบบส่วนรวมหรือแบบโกลบอล(global) กล่าวคือ เป็นการประกาศตัวแปรที่สามารถเรียกใช้งานได้ทุกที่ภายในโปรแกรม หรือภายในฟังก์ชันทุกฟังก์ชัน และเป็นส่วนประกาศฟังก์ชันที่ต้องการให้ถูกเรียกใช้ได้ทุกที่ในโปรแกรม(function prototypes) เพื่อให้ฟังก์ชันอื่นๆมองเห็น และเรียกใช้งานฟังก์ชันนี้ได้โดยไม่เกิดข้อผิดพลาดเนื่องจากลำดับการประกาศฟังก์ชัน

5). ส่วนประกาศฟังก์ชัน

เป็นส่วนที่ใช้ประกาศฟังก์ชันหลักหรือฟังก์ชัน main และ ฟังก์ชันต่างๆที่ทำงานร่วมกัน โดยฟังก์ชัน main จะมีได้เพียงตัวเดียวเท่านั้นในหนึ่งโปรแกรม ภายในฟังก์ชัน main สามารถเรียกใช้ฟังก์ชันอื่นๆที่ฟังก์ชัน main มองเห็น ได้ทุกฟังก์ชัน รวมทั้งฟังก์ชันใดๆที่มีการประกาศในแบบส่วนรวมไว้แล้ว

นอกจากนี้ภายในฟังก์ชันที่เขียนขึ้นมาใช้งาน ในบางครั้งจำเป็นที่จะต้องมีตัวแปรมารับค่า หรือนำมาใช้ภายในฟังก์ชัน นอกจากจะใช้ตัวแปรที่มีการประกาศแบบส่วนรวมแล้ว ผู้เขียนโปรแกรมสามารถกำหนดตัวแปรขึ้นเองภายในฟังก์ชัน และใช้งานตัวแปรที่กำหนดขึ้นมานี้เฉพาะฟังก์ชันหนึ่งๆเท่านั้น โดยเรียกการประกาศตัวแปรดังกล่าวว่า ตัวแปรแบบโลคอล(local)

6). ส่วนข้อความกำกับหรือคอมเมนต์

เป็นส่วนที่ใช้กำกับรายละเอียดเกี่ยวกับโปรแกรมเพื่อขยายความหรือจุดประสงค์การทำงานของโปรแกรม เพื่อให้สามารถทำความเข้าใจโปรแกรมที่เขียนขึ้นได้ง่าย และช่วยให้ใช้เวลาในการทำความเข้าใจรายละเอียดของโปรแกรมลดลง สามารถเขียนได้หลายรูปแบบ ซึ่งใช้เครื่องหมาย // หรือ /*ร่วมกับ*/ ในการกำกับ เช่น

แบบที่1	// ข้อความกำกับ
แบบที่2	/*ข้อความกำกับ*/

3.6 การเขียนโปรแกรมภาษา C เบื้องต้น

การที่ระบบไมโครคอนโทรลเลอร์จะทำงานได้จะต้องมีการป้อนชุดคำสั่งให้กับมัน การนำคำสั่งหลายๆคำสั่งมาต่อเรียงกันจะเรียกว่าการโปรแกรม การเขียนโปรแกรมด้วยภาษาซีนั้นจะต้องมีตัวแปรภาษา (compiler) เพื่อแปลภาษาที่เขียนให้เป็นภาษาเครื่องของชิปตัวนั้นอีกทีหนึ่ง โดยจะอยู่ในรูปของ hex file แล้วนำไปโปรแกรมให้ชิปนั้นทำงาน ในปัจจุบันนี้ได้มีตัวคอมไพเลอร์ภาษาซีออกมาหลายตัว แต่ละตัวจะมีข้อดีข้อเสียแตกต่างกัน ในการศึกษาครั้งนี้เลือกใช้คอมไพเลอร์ภาษาซีของ Raisonance เนื่องจากไม่มีข้อจำกัดเรื่องเวลาในการทดลองใช้ ข้อจำกัดมีเพียงขนาดของโปรแกรมที่ถูกจำกัดไว้ที่ 4 กิโลไบต์ ซึ่งเพียงพอสำหรับใช้งานในการศึกษาครั้งนี้ การเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ มีรายละเอียดดังต่อไปนี้

3.6.1 การกำหนดชนิดค่าคงที่และตัวแปร

ในโปรแกรมภาษา C ที่คอมไพเลอร์(สำหรับ MCS-51) เข้าใจสามารถกำหนดชนิดของตัวแปรและค่าคงที่ได้หลายลักษณะทั้งในแบบบิท,ไบต์,ตัวเลขจำนวนเต็ม(integer) และตัวเลขทศนิยม (floating point) สามารถอธิบายได้ดังนี้

1). บิท

เป็นการกำหนดตัวแปรค่าคงที่ขนาด 1 บิท มีข้อควรระวังในการกำหนดตัวแปรแบบบิทคือไม่สามารถกำหนดเป็นตัวแปรอะเรย์ได้ ตัวอย่างเช่น

```
bit  flag;          /* กำหนดให้ตัวแปร  flag  เป็นตัวแปรขนาด 1 บิท */
flag = 1;           /* เซตค่าตัวแปร  flag  มีค่าเป็น 1 */
```

2). ตัวอักษรแบบมีเครื่องหมาย(signed char) และไม่มีเครื่องหมาย (unsigned char)

เป็นชนิดตัวแปรที่มีขนาด 1 ไบต์ (8 บิท) ในแบบ signed char ได้กำหนดช่วงจำนวนตัวเลขอยู่ระหว่าง -128 ถึง 127 ส่วนในแบบ unsigned char อยู่ในช่วง 0 ถึง 255 ยกตัวอย่างเช่น Unsigned char var1 /* กำหนดตัวแปร var1 เป็นตัวแปรแบบ unsigned char*/-

3). ตัวเลขจำนวนเต็มแบบมีเครื่องหมาย (signed int) และไม่มีเครื่องหมาย (unsigned int)

เป็นชนิดตัวแปรที่มีขนาด 2 ไบต์ (16 บิท) ในแบบ signed char ได้กำหนดช่วงจำนวนตัวเลขอยู่ระหว่าง -32,768 ถึง 32,767 ส่วนในแบบ unsigned char อยู่ในช่วง 0 ถึง 65,535 ยกตัวอย่างเช่น

```
Unsigned int n1;        /*กำหนดตัวแปร n1เป็นแบบ unsigned int*/
```

4). ตัวเลขจำนวนเต็มยาวแบบมีเครื่องหมาย(signed long) และไม่มีเครื่องหมาย(unsigned long)

เป็นชนิดตัวแปรที่มีขนาด 4 ไบต์ (32 บิท) ในแบบ signed char ได้กำหนดช่วงจำนวนตัวเลขอยู่ระหว่าง -2,147,483,648 ถึง 2,147,483,648 ส่วนในแบบ unsigned char อยู่ในช่วง 0 ถึง 4,294,967,295 ยกตัวอย่างเช่น

```
Unsigned long temp;  /*กำหนดตัวแปร temp เป็นแบบ unsigned long*/
```

5). ตัวเลขจำนวนทศนิยม (float), จำนวนเต็มคู่ (double) และจำนวนเต็มคู่ช่วงยาว(long double)

เป็นชนิดตัวแปรที่มีขนาด 4 ไบต์ (32 บิท) ในแบบตัวเลขจำนวนทศนิยม(float) ,ขนาด 6 ไบต์(48 บิท) ในแบบตัวเลขจำนวนเต็มคู่ (double) และขนาด 7 ไบต์ (56 บิท) ในแบบตัวเลขจำนวนเต็มคู่ช่วงยาว

6). บิทรีจิสเตอร์

เป็นชนิดตัวแปรที่มีขนาด 1 บิท ส่วนใหญ่กำหนดใช้กับรีจิสเตอร์ฟังก์ชันพิเศษ (SFR: Special function register) ในไมโครคอนโทรลเลอร์ MCS-51 หรือข้อมูลที่ระบุหน่วยความจำที่เข้าถึงระดับบิทได้ เช่น ในพื้นที่หน่วยความจำ 0x20 ถึง 0x2F ในไมโครคอนโทรลเลอร์ MCS-51 ยกตัวอย่างเช่น

```
sbit  port0_1 = 0x81; //กำหนดตัวแปร  port0_1  ไปยังพอร์ต 0 บิท 1(P0.1)
                      // ของ MCS-51
sbit  switch  = P1^3;  //กำหนดตัวแปร  switch  ไปยังพอร์ต 1  บิท 3 (P1.3)
                      //ของ MCS-51
```

7). บิทข้อมูล (bdata)

เป็นการกำหนดให้ตัวแปรจองพื้นที่ในหน่วยความจำข้อมูลแรมภายในที่แอดเดรส 0x20 ถึง 0x2F ซึ่งสามารถเข้าถึงในระดับบิทได้ ยกตัวอย่างเช่น

```
Bdata unsigned char led;        // กำหนดตัวแปร led มีขนาด 8 บิท
                                //  ในพื้นที่หน่วยความจำแรมภายในแอดเดรส  0x20-0x2F
                                //ดังนั้นตัวแปร led จึงสามารถเข้าถึงในระดับบิทได้
sbit a=led^0;   /*กำหนดตัวแปร a แทนบิท  0 ของตัวแปร led */
```

8). ไบต์รีจิสเตอร์ (sfr)

เป็นชนิดตัวแปรที่มีขนาด 1 ไบต์ (8 บิท) และกำหนดใช้กับรีจิสเตอร์ฟังก์ชันพิเศษ (SFR) ในไมโครคอนโทรลเลอร์ MCS-51 ซึ่งโดยปกติจะมีกำหนดไว้แล้วในไฟล์ที่แนบส่วนต้นของโปรแกรมอย่าง reg51.h หรือ reg52.h ยกตัวอย่างเช่น

Sfr P1=0x90; /* กำหนดตัวแปร P1 ไปยังพอร์ต 1(แอดเดรส 90) ของ MCS-51

9). ไบต์รีจิสเตอร์คู่ (sfr16)

เป็นชนิดตัวแปรที่มีขนาด 2 ไบต์(16 บิท) และกำหนดใช้กับรีจิสเตอร์ฟังก์ชันพิเศษ (SFR) ในไมโครคอนโทรลเลอร์ MCS-51 ซึ่งโดยปกติจะมีกำหนดไว้แล้วในไฟล์ที่แนบส่วนต้นของโปรแกรมอย่าง reg51.h หรือ reg52.h ยกตัวอย่างเช่น

```
Sfr16  T2= 0xCC;        //กำหนดตัวแปร T2 สำหรับไทเมอร์ 2 ซึ่งประกอบไปด้วย
                        // รีจิสเตอร์ T2L มีแอดเดรสอยู่ที่ 0xCC และ
                        //รีจิสเตอร์ T2H มีแอดเดรสอยู่ที่ 0xCD
```

3.6.2 ตัวดำเนินการในโปรแกรมภาษา C(Operator)

แบ่งออกเป็น 3 กลุ่ม คือ ตัวดำเนินการทางคณิตศาสตร์ (arithmetric operator) , ตัวดำเนินการทางความสัมพันธ์และลอจิก (relation & logic operation) และตัวดำเนินการทางบิท (bitwise operation)

1) ตัวดำเนินการทางคณิตศาสตร์

การทำงานของตัวดำเนินการในกลุ่มนี้ สรุปได้ดังนี้

+	การบวก
-	การลบ
*	การคูณ
/	การหาร
%	การหารแบบเอาเศษ
++	การเพิ่มค่าขึ้นอีกหนึ่งค่า
--	การลดค่าลงอีกหนึ่งค่า
+=	การบวกขึ้นอีกด้วยค่าทางขวามือ
-=	การลดค่าลงอีกด้วยค่าทางขวามือ
*=	การคูณด้วยค่าทางขวามือ
/=	การหารด้วยค่าทางขวามือ
%=	การหารด้วยค่าทางขวามือแบบเอาเศษ

2) ตัวดำเนินการทางความสัมพันธ์และลอจิก

การกระทำของตัวดำเนินการในกลุ่มนี้ จะให้ผลลัพท์จากการตรวจสอบเป็น 1 ถ้าเงื่อนไขเป็นจริง และเป็น 0 เมื่อเงื่อนไขเป็นเท็จ การทำงานของตัวดำเนินการในกลุ่มนี้ สรุปได้ดังนี้

ตัวดำเนินการ	ความหมาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
!	ไม่ใช่(NOT)
&&	และ(AND)
||	หรือ(OR)

3) ตัวดำเนินการทางบิท

การกระทำในกลุ่มนี้เป็นการกระทำที่เข้าถึงข้อมูลในระดับบิท โดยที่ค่าของแต่ละบิทเป็นไปได้ 2 ค่า คือ “1” หรือ “0” เท่านั้น

การทำงานของตัวดำเนินการทางบิทสามารถสรุปได้ดังนี้

ตัวดำเนินการ	ความหมาย
~	กลับค่าบิทของข้อมูล
&	การแอนด์บิท
|	การออร์บิท
^	การเอ็กคลูซีฟ-ออร์บิท
<<	เลื่อนบิทไปทางซ้าย
>>	เลื่อนบิทไปทางขวา
<<=	เลื่อนบิทไปทางซ้ายแล้วให้เท่ากับ
>>=	เลื่อนบิทไปทางขวาแล้วให้เท่ากับ
&=	ทำการแอนด์แล้วให้เท่ากับ
|=	ทำการออร์แล้วให้เท่ากับ
^=	ทำการเอ็กคลูซีฟ-ออร์แล้วให้เท่ากับ

3.6.3 ตัวแปรชนิดอะเรย์

ตัวแปรชนิดอะเรย์ คือ กลุ่มของตัวแปรที่มีชนิดข้อมูลเหมือนกัน และมีที่อยู่ในหน่วยความจำหริอ แอดเดรสเรียงติดต่อกันไป อาจใช้พื้นที่ของหน่วยความจำข้อมูล หรือหน่วยความจำโปรแกรมก็ได้ ขึ้นอยู่กับรูปแบบการประกาศ โดยมีรูปแบบดังนี้

Type name[size];

โดยที่ type คือ ชนิดของข้อมูลตัวแปรอะเรย์

Name คือ ชื่อของตัวแปรอะเรย์

Size คือ ค่าของตัวเลขกำหนดขนาดของอะเรย์ อาจจะไม่กำหนดก็ได้

การเข้าถึงข้อมูลของสมากชิกตัวใดของอะเรย์ ซึ่งอาจเป็นตัวเลขหรือค่าตัวแปรใดๆ หรือการกระทำของนิพจน์ ซึ่งทั้งหมดที่กล่าวมาจะต้องเป็นเลขจำนวนเต็มเท่านั้น ยกตัวอย่างเช่น ถ้าประกาศเป็น char arr[4] ; หมายถึง ตัวแปรอะเรย์ชื่อ arr เป็นตัวแปรที่ประกอบด้วยสมากชิกย่อย 4 ตัว อาจมองว่าเป็นตัวแปร 4 ตัวก็ได้ แต่เวลาที่จะอ้างอิงถึงสมาชิก จะใช้ดัชนีเป็นตัวบ่งบอกว่ากำลังติดต่อใช้งานกับสมาชิกตัวไหน ดังนั้น arr จะสามารถแจกแจงสมาชิกได้ดังนี้

Arr[0] เป็นสมาชิกตัวที่ 1 มีดัชนีที่ชี้เป็น '0'

Arr[1] เป็นสมาชิกตัวที่ 1 มีดัชนีที่ชี้เป็น '1'

Arr[2] เป็นสมาชิกตัวที่ 1 มีดัชนีที่ชี้เป็น '2'

Arr[3] เป็นสมาชิกตัวที่ 1 มีดัชนีที่ชี้เป็น '3'

โดย arr[0],arr[1],arr[2],arr[3] ต่างก็เป็นตัวแปรชนิด char มีขนาด 1 ไบต์ ดังนั้น การประกาศตัวแปร arr จึงใช้เนื้อที่ทั้งสิ้น 4 ไบต์

3.6.4 คำสั่งควบคุมต่าง ๆในภาษา C

คำสั่งในภาษา C สำหรับไมโครคอนโทรลเลอร์ ส่วนใหญ่จะยึดให้ตรงกับมาตรฐาน ANSI C คำสั่งในภาษา C หลักๆประกอบด้วย คำสั่งแบบตรวจสอบเงื่อนไขและคำสั่งวนลูป ดังนี้

1) คำสั่ง if

เป็นคำสั่งที่ใช้ในการตรวจสอบเงื่อนไขว่า เป็นจริงหรือเท็จ มีรูปแบบดังนี้

```
If(condition)
{
  block;
}
```

block หมายถึง ชุดคำสั่งที่อยู่ในช่วงวงเล็บปีกกา '{}' ซึ่งอาจมี 1 คำสั่งหรือหลายคำสั่งก็ได้

ถ้าตรวจสอบแล้วพบว่าเงื่อนไขนั้นเป็นจริง ก็จะเข้าไปกระทำคำสั่งภายใน block แต่ถ้าเป็นเท็จก็จะข้ามไปกระทำคำสั่งที่อยู่ถัดไปภายนอก block แต่ถ้าคำสั่งภายใน block มีเพียงคำสั่งเดียว อาจจะไม่ต้องใส่เครื่องหมายปีกกาก็ได้

2) คำสั่ง if else

เป็นคำสั่งที่ตรวจสอบเงื่อนไขว่าเป็นจริงหรือเท็จ แล้วเลือกกระทำอย่างใดอย่างหนึ่ง รูปแบบของคำสั่ง คือ

```
If(condition)
{
    block1;       //ทำในส่วนนี้ถ้าเงื่อนไขเป็น"จริง"
}
else
{
    block2;       //ทำในส่วนนี้ถ้าเงื่อนไขเป็น"จริง"
}
```

จากรูปแบบของการใช้คำสั่ง หากตรวจสอบเงื่อนไขแล้วพบว่าเงื่อนไขแล้วพบว่าเงื่อนไขเป็นจริงก็จะเข้าไปกระทำคำสั่งภายใน block1 แต่ถ้าเงื่อนไขเป็นเท็จ จะเข้าไปกระทำคำสั่งภายใน block2

3) คำสั่ง switch

เป็นคำสั่งที่ใช้ในการตรวจสอบตัวแปรหรือค่าคงที่ ทำงานร่วมกับคำสั่ง case ถ้าตัวแปรหรือค่าคงที่ที่นำมาตรวจสอบไม่ตรงกับ case ใดเลย โปรแกรมจะมองหาคำสั่ง default แทน แต่ถ้าไม่มีการระบุ ก็จะออกจากการทำงานของคำสั่ง switch ทันที มีรูปแบบการใช้งานคำสั่งดังนี้

```
Switch(variable)
{
        case const1 :  block1
                                break;
        case const2 :   block2
                                break;
                default : block3
                                break;
}
```

โดยที่ variable คือ ตัวแปรที่นำมาครวจสอบเงื่อนไข

Const คือ ค่าที่ใช้เปรียบเทียบกับตัวแปรที่นำมาตรวจสอบ ถ้าตรงกับตัวใดจะไปทำงานใน block นั้น

Default คือ ส่วนของโปรแกรมที่กำหนดให้ทำงานในกรณีที่เปรียบเทียบค่าแล้วไม่ตรงกับ case ใดเลย ซึ่งไม่จำเป็นต้องมีเสมอไป

4) คำสั่ง for

เป็นคำสั่งที่กำหนดให้มีการทำงานแบบวนรอบด้วยการตรวจสอบจากเงื่อนไข โดยปกติจะมีการกำหนดรอบของการทำงานเป็นจำนวนที่แน่นอน มีรูปแบบของคำสั่งดังนี้

```
For(initialize;condition;incremental)
{
        block;
}
```

โดยที่ initialize คือ ค่าเริ่มต้นที่กำหนดจากตัวแปรที่นำมาเป็นเงื่อนไขในการทำงานแบบวนรอบ

Condition คือ เงื่อนไขที่ใช้ในการตรวจสอบว่า จะให้กระทำคำสั่งภายใน block หรือไม่ ถ้าเป็นจริง ผลการตรวจสอบมีค่าเป็น "1"จะกระทำคำสั่งใน block แต่ในทางกลับกัน ถ้าเป็นเท็จ ผลการตรวจสอบมีค่าเป็น "0" จะไม่มีการกระทำคำสั่งใน block

Incremental คือ คำสั่งที่กระทำกับตัวแปรซึ่งนำมาเป็นเงื่อนไข

5) คำสั่ง while

เป็นคำสั่งให้ทำงานวนรอบแบบไม่รู้จบ ด้วยการตรวจสอบจากเงื่อนไข มีรูปแบบคำสั่งเป็น

```
While(condition)
{
        block
}
```

โดยที่ condition คือ เงื่อนไขที่จะใช้ตรวจสอบว่าจะให้กระทำคำสั่งภายใน block หรือไม่ ถ้าเงื่อนไขเป็นจริง ผลการตรวจสอบเป็น "1" จะเข้าไปกระทำคำสั่งภายใน block แต่ถ้าเป็นเท็จผลการตรวจสอบเป็น "0" จะไม่มีการกระทำคำสั่งภายใน block เช่น

```
While(1)
{
        block
}
```

เป็นโปรแกรมที่มีการกระทำคำสั่งใน block แบบไม่รู้จบ จะสังเกตเห็นว่า เงื่อนไขจะเป็นจริงตลอดเวลา เพราะที่ตำแหน่งเงื่อนไขมีค่าเป็น "1"

3.7 การใช้งานไทเมอร์/เคาน์เตอร์ของไมโครคอนโทรลเลอร์ mcs-51

ไทเมอร์และเคาน์เตอร์เป็นส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์เนื่องจากในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลา เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณพัลส์เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าการนับ ในไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89S8252 ของ ATMEL ที่นำมาใช้นี้ มีวงจรไทเมอร์/เคาน์เตอร์ขนาด 16 บิท 3 ตัว ประกอบด้วยไทเมอร์O , ไทเมอร์ 1 ,ไทเมอร์ 2 เรียกสั้นๆว่า T0,T1 และ T2 โดยรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ทั้งสามตัวสามารถกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ และตัวนับหรือเคาน์เตอร์ได้อย่างอิสระต่อกัน

3.7.1 การทำงานเป็นไทเมอร์

เมื่อกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นในทุกๆแมชชีนไซเคิล หรือรีจิสเตอร์จะทำการนับค่าของแมชชีนไซเคิลนั่นเอง และเนื่องจากแมชชีนไซเคิลประกอบด้วยคาบเวลาของวงจรกำเนิดสัญญาณนาฬิกา 12 คาบเวลา สำหรับ AT89S8252 ดังนั้นอัตราในการนับจึงเท่ากับ 1/12 ของความถี่สัญญาณนาฬิกา ตัวอย่างเช่น ถ้าหากระบบ MCS-51 ทำงานที่ความถี่สัญญาณนาฬิกา 12 MHz หรือ 1 แมชชีนไซเคิลเท่ากับ 1 ไมโครวินาที และถ้ามีรีจิสเตอร์ตัวจับเวลาเป็นแบบ 16 บิท และมีค่าเริ่มต้นเป็น 0000H เมื่อเวลาผ่านไป 65535 ไมโครวินาที จะเกิดการโอเวอร์โฟล์วขึ้น ดังนั้นถ้าหากใช้เป็นตัวจับเวลา 100 ไมโครวินาที ทำได้โดยโปรแกรมให้ตัวจับเวลามีค่าเริ่มต้นเป็น 65535 – 100 หรือมีค่าเป็น 65435 เมื่อให้ตัวจับเวลาทำงานก็คอยสังเกตที่บิทแฟลก ถ้ามีการโอเวอร์โฟล์วเกิดขึ้นแสดงว่าเวลาได้ผ่านมา 100 ไมโครวินาทีแล้ว

3.7.2 การทำงานเป็นเคาน์เตอร์

เมื่อทำงานเป็นตัวนับหรือเคาน์เตอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิกจาก “1” เป็น “0” เกิดขึ้นที่ขาอินพุตทางฮาร์ดแวร์ของวงจรไทเมอร์/เคาน์เตอร์ ซึ่งก็คือ ขา T0(P3.4),ขา T1(P3.5) และขา T2(P1.0) หรืออาจมองว่า ถ้าจะให้นับอะไร สัญญาณที่จะนับให้ต่อกับขา T0, T1 และ T2 ซึ่งสัญญาณที่เข้ามาเมื่อมีการเปลี่ยนแปลงจาก “1” เป็น “0” จะทำให้ค่าใน รีจิสเตอร์ TLx มีค่าเพิ่มขึ้น 1 ค่า ในการนับสัญญาณที่เข้ามาจะต้องใช้เวลา 2 แมชชีนไซเคิล หรือ 24 ลูกสัญญาณนาฬิกา ซึ่งจะทำให้อัตราการนับสูงสุดของ mcs-51 อยู่ที่ 1/24 ของสัญญาณนาฬิกาที่ป้อนให้กับตัว mcs-51 เอง

3.7.3 รีจิสเตอร์ที่เกี่ยวข้อง

การใช้งานไทเมอร์จะมีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 3 ตัวคือ รีจิสเตอร์ไทเมอร์ รีจิสเตอร์ TMOD และ รีจิสเตอร์ TCON มีรายละเอียด ดังต่อไปนี้

1) รีจิสเตอร์ไทเมอร์

มีด้วยกัน 4 ตัวคือ TL0, TH0, TL1, TH1 แต่ละตัวมีขนาด 8 บิท ในการใช้งานทั่วไปมักใช้ร่วมกันโดยจัดเป็นคู่ TL0 กับ TH0 รวมกันเป็นรีจิสเตอร์ ไทเมอร์0 ขนาด 16 บิท และ TL1 และ TH1 รวมกันเป็นรีจิสเตอร์ ไทเมอร์1 ขนาด 16 บิทโดยใน TL0 และ TL1 เก็บข้อมูล 8 บิทล่าง ส่วน TH0 และ TH1 เก็บข้อมูล 8 บิทบน รีจิสเตอร์ไทเมอร์ทั้ง 2 คู่ เมื่อนำมาใช้ร่วมกันจะสามารถเก็บค่าการนับได้สูงสุด 65,536 หรือ 0xFFFF เมื่อนับถึงค่านี้แล้วจะวนไปเริ่มนับ 0x0000 ใหม่ และเมื่อเกิดการนับรอบใหม่ บิท TF0 หรือ TF1 ในรีจิสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์จะเกิดการเซตเพื่อแจ้งว่านับเกินค่าสูงสุดแล้ว

2) รีจิสเตอร์ TCON

การควบคุมหรือสั่งการให้ ไทเมอร์0 หรือ ไทเมอร์1 ตัวใดทำงานนั้นจะต้องทำผ่านทางรีจิสเตอร์ TCON ซึ่งเป็นรีจิสเตอร์ขนาด 8 บิท สามารถเข้าถึงการทำงานในระดับบิทได้ โดยรายละเอียดในบิทต่างๆของรีจิสเตอร์ TCON เป็นดังนี้

บิท7	บิท6	บิท5	บิท4	บิท3	บิท2	บิท1	บิท0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 : ถ้าบิทนี้เป็น 1 แสดงว่าเกิดการนับเกินหรือเกิดโอเวอร์โฟล์วของไทเมอร์1 บิทนี้จะเป็น 0 เมื่อมีการอินเทอร์รัปต์เกิดขึ้น

TR1 : ใช้ในการเปิดปิดการทำงานของไทเมอร์1 คือถ้าเป็น 1 หมายถึงให้ ไทเมอร์1 ทำงาน แต่ถ้าเป็น 0 หมายถึง ให้หยุดการทำงานของไทเมอร์1

TF0 : ถ้าบิทนี้เป็น 1 แสดงว่าเกิดการนับเกินหรือเกิดโอเวอร์โฟล์วของไทเมอร์0 บิทนี้จะเป็น 0 เมื่อมีการอินเทอร์รัปต์เกิดขึ้น

TR0 : ใช้ในการเปิดปิดการทำงานของไทเมอร์0 คือถ้าเป็น 1 หมายถึงให้ ไทเมอร์0 ทำงาน แต่ถ้าเป็น 0 หมายถึง ให้หยุดการทำงานของไทเมอร์0

IE1 : บิทนี้ใช้ในกระบวนการอินเทอร์รัปต์ บิทนี้จะเป็น 1 เมื่อเกิดสัญญาณอินเทอร์รัปต์ภายนอกตัวที่ 1 (INT1)

IT1 : บิทนี้ใช้ในกระบวนการอินเทอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเทอร์รัปต์จากภายนอกตัวที่1(INT1)

ถ้าเป็น 0 เลือกให้ช่วงขอบขาลงเป็นสัญญาณอินเทอร์รัปต์

ถ้าเป็น 1 เลือกให้ช่วงที่เป็นลอจิกต่ำเป็นสัญญาณอินเทอร์รัปต์

IE0 : บิทนี้ใช้ในกระบวนการอินเทอร์รัปต์ บิทนี้จะเป็น 1 เมื่อเกิดสัญญาณอินเทอร์รัปต์ภายนอกตัวที่ 0 (INT0)

IT0 : บิทนี้ใช้ในกระบวนการอินเทอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเทอร์รัปต์จากภายนอกตัวที่0(INT0)

ถ้าเป็น 0 เลือกให้ช่วงขอบขาลงเป็นสัญญาณอินเทอร์รัปต์

ถ้าเป็น 1 เลือกให้ช่วงที่เป็นลอจิกต่ำเป็นสัญญาณอินเทอร์รัปต์

3) รีจิสเตอร์ TMOD

เป็นรีจิสเตอร์ขนาด 8 บิท ใช้สำหรับควบคุมการทำงานของไทเมอร์ แบ่งการทำงานเป็น 2 ส่วน คือ 4 บิทล่างเป็นการควบคุมไทเมอร์ 0 และ 4 บิทบนจะเป็นการควบคุมไทเมอร์ 1 รีจิสเตอร์นี้ใช้เป็นตัวเลือกการทำงานว่าจะให้ตัวไทเมอร์ทำงานในโหมดใด รายละเอียดในแต่ละบิทแสดงได้ดังนี้

บิท7	บิท6	บิท5	บิท4	บิท3	บิท2	บิท1	บิท0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
ไทเมอร์ 1				ไทเมอร์ 0			

GATE : ใช้เลือกลักษณะการควบคุมการทำงานของไทเมอร์ ถ้าเป็นบิทนี้เป็น 0 ไทเมอร์จะทำงานเมื่อบิท TRx ในรีจิสเตอร์ TCON เป็น 1

C/T (Timer or Counter selector) : ถ้าเป็น 1 เลือกให้ทำงานเป็นเคาน์เตอร์ โดยรับสัญญาณอินพุตทางขา T0 หรือ T1 ถ้าเป็น 0 เลือกทำงานเป็นไทเมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิกาภายใน CPU

M1,M2 (Mode selector bit) : ใช้เลือกโหมดการทำงานของไทเมอร์

ถ้า M1=0,M0=0 แสดงว่าเลือกให้ทำงานในโหมด 0

ถ้า M1=0,M0=1 แสดงว่าเลือกให้ทำงานในโหมด 1

ถ้า M1=1,M0=0 แสดงว่าเลือกให้ทำงานในโหมด 2

ถ้า M1=1,M0=1 แสดงว่าเลือกให้ทำงานในโหมด 3

3.7.4 โหมดการทำงานของไทเมอร์ 0 และ 1

ไทเมอร์มีโหมดในการทำงานอยู่ 4 แบบ คือ

โหมด 0 : ไทเมอร์ 13 บิท

เป็นการนับแบบ 13 บิท โดยใช้ TLx(TL0 หรือ TL1) จำนวน 5 บิทล่างกับ THx(TH0 หรือ TH1) ทั้ง 8 บิทในการนับ ในการนับเมื่อ TLx นับครบ 32 หรือมีค่าเป็น 00011111_2 ก็จะส่งสัญญาณไปยัง THx เพื่อทำการเพิ่มค่าขึ้นอีก 1 แล้ว TLx จะเปลี่ยนค่าเป็น 00000000_2 แล้วก็นับต่อไปเรื่อยๆ จนกว่า TLx เป็น 00011111_2 และ THx เป็น 11111111_2 ก็จะเกิดโอเวอร์โฟล์วโดยแจ้งผ่านบิท TFx(TF0 หรือ TF1) หลังจากนั้นไมโครคอนโทรลเลอร์จะสร้างสัญญาณอินเทอร์รัปต์สำหรับไทเมอร์ขึ้นมา แล้วเปลี่ยนค่าของ TLx กับ THx ให้เป็น 00000000_2 เพื่อเริ่มต้นการนับต่อไป

โหมด 1 : ไทเมอร์ 16 บิท

เป็นการนับแบบ 16 บิท โดยใช้ TLx เก็บข้อมูลไบต์ล่าง และ THx เก็บข้อมูลไบต์สูง ในการทำงานในโหมดนี้คล้ายกับในโหมด 0 แต่จะใช้งานรีจิสเตอร์ TLx และ THx ครบ 8 บิท ดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 16 บิท เวลาทำงานก็จะเริ่มนับค่าจาก 0000_{16} หรือจากค่าที่ตั้งไว้ไปจนกว่าจะถึง $FFFF_{16}$ ก็จะเกิดการโอเวอร์โฟล์วโดยแจ้งผ่านบิท TFx หลังจากนั้นไมโครคอนโทรลเลอร์จะสร้างสัญญาณอินเทอร์รัปต์สำหรับไทเมอร์ขึ้นมา แล้วเปลี่ยนค่าของ TLx กับ THx ให้เป็น 0000_{16} เพื่อเริ่มต้นการนับต่อไป

โหมด 2 : ไทเมอร์ 8 บิทแบบตั้งค่าอัตโนมัติ

เป็นการนับแบบ 8 บิท โดยใช้ TLx เป็นตัวนับ และ THx เป็นตัวเก็บค่าเริ่มต้น เมื่อเริ่มต้นทำงาน ค่าของ THx จะถูกส่งไปยัง TLx เมื่อ TLx นับจนมีค่าเป็น FF_{16} จะเกิดการโอเวอร์โฟล์วโดยแจ้งผ่านบิท TFx จากนั้นไมโครคอนโทรลเลอร์จะสร้างสัญญาณอินเทอร์รัปต์สำหรับไทเมอร์ขึ้นมา แล้วจะมีการส่งค่าเริ่มต้นจาก THx ไปยัง TLx เพื่อเริ่มต้นการนับใหม่ เรียกกระบวนการนี้ว่า รีโหลด (reload)

โหมด 3 : ไทเมอร์ 8 บิท

ในโหมดนี้การทำงานของไทเมอร์ 1 กับไทเมอร์ 0 จะไม่เหมือนกัน ในการทำงานของไทเมอร์ 0 นั้น เป็นการนับแบบ 8 บิท โดยแยกการนับ TH0 และ TL0 ออกจากกัน เมื่อเกิดการโอเวอร์โฟล์วของ TL0 จะแจ้งผ่านบิท TF0 ในขณะที่ TH0 จะแจ้งผ่านทางบิท TF1 ส่วนการทำงานของไทเมอร์ 1 เมื่อเข้าสู่โหมดนี้จะเป็นการสั่งให้ไทเมอร์ 1 หยุดนับ และไม่สามารถสร้างสัญญาณอินเทอร์รัปต์ได้ เพราะไทเมอร์ 0 ได้ใช้ TF1 ไปแล้ว

3.8 การอินเทอร์รัพท์

อุปกรณ์ภายนอกสามารถติดต่อกับไมโครคอนโทรลเลอร์ได้สองวิธี คือ Polling Method ตัว CPU จะคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลาว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ถ้ามีอุปกรณ์ภายนอกหลายตัว ระบบก็ต้องคอยตรวจสอบอุปกรณ์ภายนอกหลายตัว ทำให้เสียเวลาในการทำงานหลักไป การทำงานอีกแบบหนึ่งเรียกว่า การอินเทอร์รัพท์ (Interrupt) คือ การที่ CPU ไม่ต้องคอยตรวจสอบอุปกรณ์อินพุตเอาต์พุต ถ้าอุปกรณ์ตัวใดต้องการติดต่อกับ CPU อุปกรณ์ตัวนั้นจะส่งสัญญาณมาบอก CPU เอง การใช้อินเทอร์รัปต์นี้จะช่วยลดการทำงานของ CPU หรือไมโครคอนโทรลเลอร์ลงได้มาก

ในไมโครคอนโทรลเลอร์ จะมีการอินเทอร์รัพท์อยู่สองประเภท คือ สัญญาณอินเทอร์รัพท์จากภายนอกที่เกิดจากสัญญาณที่เข้ามาทางขา INT0 และ INT1 และสัญญาณภายในที่เกิดจากไทเมอร์0 ไทเมอร์1 และพอร์ตอนุกรม เมื่อไมโครคอนโทรลเลอร์ถูกอินเทอร์รัพท์ มันจะหยุดการทำงานโปรแกรมหลัก และจะต้องไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์ เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัพท์เสร็จแล้ว จะมาทำโปรแกรมหลักที่ทำงานค้างอยู่ต่อไป การใช้อินเทอร์รัพท์ของไมโครคอนโทรลเลอร์ จะมีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 3 ตัวคือ

1).รีจิสเตอร์ IE (Interrupt Enable) ใช้ในการกำหนดว่าจะยอมให้อินเทอร์รัพท์จากแหล่งใดได้บ้าง สามารถเข้าถึงได้ในระดับบิทโดยรายละเอียดแต่ละบิทเป็นดังนี้

บิท7	บิท6	บิท5	บิท4	บิท3	บิท2	บิท1	บิท0
EA	EC	ET2	ES	ET1	EX1	ET0	EX0

EA : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์ทุกตัวทำงาน
ถ้าเป็น 0 หมายความว่าให้ยกเลิกอินเทอร์รัพท์ทุกตัว

ET2 : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์ไทเมอร์2 ทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์ไทเมอร์2

ES : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์สื่อสารอนุกรมทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์สื่อสารอนุกรมทำงาน

ET1 : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์ไทเมอร์1 ทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์ไทเมอร์1

EX1 : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์ภายนอก หรือ INT1ทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์ภายนอก หรือ INT1
ET0 : ถ้าบิทนี้เป็น 0 หมายความว่าให้อินเทอร์รัพท์ไทเมอร์0 ทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์ไทเมอร์0
EX0 : ถ้าบิทนี้เป็น 1 หมายความว่าให้อินเทอร์รัพท์ภายนอก หรือ INT0ทำงาน
ถ้าเป็น 0 หมายความว่ายกเลิกอินเทอร์รัพท์ภายนอก หรือ INT0

2).รีจิสเตอร์ IP (Interrupt Priority) ใช้กำหนดลำดับของการอินเทอร์รัพท์ กรณีที่เกิดการอินเทอร์รัพท์จากหลายแหล่งพร้อมๆกัน

3).รีจิสเตอร์ TCON (Timer Control) รีจิสเตอร์ตัวนี้นอกจากจะใช้ควบคุมไทเมอร์แล้ว ยังใช้ในการอินเทอร์รัพท์อีกด้วย

ในการเขียนโปรแกรมสำหรับการอินเทอร์รัพท์ด้วยภาษาซี จะใช้คำว่า interrupt และหมายเลขอินเทอร์รัพท์ต่อท้ายฟังก์ชัน โดยหมายเลขอินเทอร์รัพท์ของอินเทอร์รัพท์แต่ละตัวจะเป็นดังนี้

แหล่งกำเนิดสัญญาณอินเทอร์รัพท์	หมายเลขอินเทอร์รัพท์
อินเทอร์รัพท์จากภายนอก (IE0)	0
อินเทอร์รัพท์ไทเมอร์0 (TF0)	1
อินเทอร์รัพท์จากภายนอก (IE1)	2
อินเทอร์รัพท์ไทเมอร์1 (TF1)	3
อินเทอร์รัพท์พอร์ตอนุกรม	4