รายการอ้างอิง

1. Volker Braschel, Neuwied, "Method of Controlling the Brake Pressure in an Antilock Vehicle Brake System", U.S. Patent Number 5033799 Jul, 1991.
2. J.L. Harned, L.E.Johnston, and G. Scharpf, "Measurement of Tire Brake Force Characteristics as related to Wheel Slip (Antilock) Control System Design", SAE Paper 690214, 1986.
3. Masao Watanabe, Noboru Noguchi, "New Algorithm for ABS to Compensate for Road - Disturbance", SAE Transactions v. 99, n. Sect 6, p.271 - 279, 1990.
4. Satohiko Yoneda, Yasuo Naitoh, and Hideo Kigoshi, "Rear Brake Lock-up Control System of Mitsubishi Starion", SAE Paper 830482, 1983.
5. Rhee S.K. ,"Friction Coefficient of Automotive Friction Materials -Its Sensitivity to Load , Speed and Temperature", SAE Paper 740415, 1974.
6. John G. Bollinger, Neil A. Duffie, "Computer Control of Machines and Processes", Addison - Wesley, 1988.
7. William J. Palm "Control System Engineerings", John Wiley & Sons, 1986.
8. Stephen C. Gates, Jordan Becker, "Laboratory Automation Using The IBM PC", Prentice-Hall Inc., 1989.
9. Beckwith T. G., Lewis Buck N. "Mechanical Measurements", Addision-Wesley Inc., 1969.

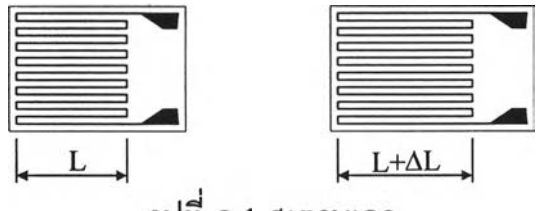
ภาคผนวก ก.

ทรานสดิวเซอร์ความดัน

มีชิ้นส่วนยืดหยุ่นหลายชนิด อาจใช้ในการวัดความดันได้ แต่อุปกรณ์ส่วนใหญ่ที่ใช้กันอยู่จะเป็นท่อบูร์ดอง (bourdon) , ไดอะแฟรม (diaphragm) หรือเบลโลว์ (bellow) เป็นส่วนที่ไวต่อความดัน การยุบหรือยืดตัวของชิ้นส่วนเหล่านี้ อาจใช้ขับเข็มชี้โดยตรง หรือผ่านระบบลิ้งเกจ และเฟือง หรือ การเคลื่อนที่ อาจเปลี่ยนเป็นสัญญาณทางไฟฟ้าโดยวิธีใดวิธีหนึ่ง บางครั้งก็ใช้สเตรนเกจติดโดยตรงกับไดอะแฟรม

สเตรนเกจ (strain gage)

ในการทำโครงการวิทยานิพนธ์นี้ จะใช้สเตรนเกจเป็นตัวตรวจวัดการยืดหดของไดอะแฟรมของอุปกรณ์วัดความดัน (ทรานสดิวเซอร์ความดัน) ที่เราได้สร้างขึ้น โดยตัวสเตรนเกจนั้นจะทำการติดตั้งที่ไดอะแฟรมบริเวณที่จะเกิดการเปลี่ยนแปลงด้านความเครียดมากที่สุด สำหรับความเครียดที่เกิดขึ้นที่สเตรนเกจนี้ โดยพื้นฐานแล้ว เมื่อเกิดความเครียดจะทำให้ความต้านทานในตัวสเตรนเกจมีค่าเปลี่ยนไป



รูปที่ ก.1 สเตรนเกจ

$$\frac{\Delta R/R}{\Delta L/L} = GF \qquad (ก.1)$$

และ $$\varepsilon = \frac{\Delta L}{L} = \frac{\Delta R}{R.GF} \qquad (ก.2)$$

และ $$\Delta R = GF.R.\frac{\Delta L}{L} \quad (ก.3)$$

และ $$\frac{\Delta R}{R} = GF.\varepsilon \quad (ก.4)$$

เมื่อ GF =เกจแฟคเตอร์ (gauge factor)

ε =ค่าความเครียด

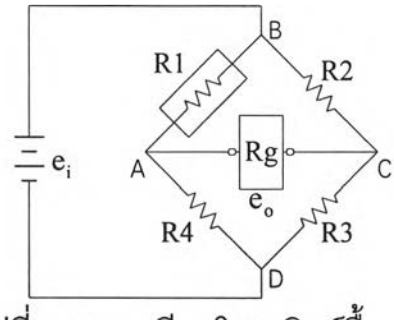
L =ความยาวเดิมของฟอยลที่ตัวสเตรนเกจ

ΔL =ความยาวที่เปลี่ยนแปลง

R =ความต้านทานของตัวสเตรนเกจ

ΔR =ความต้านทานที่เปลี่ยนแปลง

วงจรวีทสโตนบริดจ์ (wheatstone bridge circuit)



รูปที่ ก.2 วงจรวีทสโตนบริดจ์พื้นฐาน

สำหรับค่าความต้านทานที่เปลี่ยนแปลง ΔR นั้นจะมีค่าน้อย ทำให้ไม่สามารถทำการวัดได้โดยตรง วิธีการที่นิยมใช้และทำให้เราสามารถวัดค่าการเปลี่ยนแปลงความต้านทานนี้ได้เราจะต้องทำการติดตั้งสเตรนเกจเป็นส่วนหนึ่งของวงจรไฟฟ้าที่เรียกว่าวงจรวีทสโตนบริดจ์ (wheatstone bridge circuit) ซึ่งจะช่วยในการแปลงค่าการเปลี่ยนแปลงของความต้านทานที่มีค่าน้อยมากนี้ให้สามารถวัดได้ในรูปของค่าความต่างศักย์ไฟฟ้าตามรูปที่ ก.2

ความต่างศักย์ไฟฟ้าออก (เอาท์พุท) e_o จะมีค่าความสัมพันธ์เป็นสัดส่วนตรงกับความเครียดที่เกิดขึ้น และถ้าในสภาวะเริ่มต้นความต่างศักย์ไฟฟ้าเป็นศูนย์ ต่อมาเกิดมีการเปลี่ยนแปลงความต้านทานขึ้นในวงจรก็จะไม่สมดุลย์และมีความต่างศักย์ไฟฟ้าเกิดขึ้น เมื่อเราทำการใส่ความดันที่รู้ค่าต่อระบบ เราจะสามารถทำการปรับเทียบ (calibrated) เพื่อหาคุณสมบัติของระบบ ที่แสดงความสัมพันธ์ระหว่างความดันกับความต่างศักย์ไฟฟ้าออก e_o ได้

โดยวิธีการเดียวกันและถ้าความต้านทานทั้ง 4 ตัว ในวงจรบริดจ์สามารถเปลี่ยนแปลงความต้านทานได้ เนื่องจากใส่ความดันเข้าระบบ เราจะได้ค่าสัญญาณความต่างศักย์ออก e_o ที่สัมพันธ์กับค่าความต้านทานทั้งหมดที่เปลี่ยนแปลงไปดังนี้

$$e_o = \frac{e_i Rg}{4(R+Rg)}\left(\frac{\Delta R1}{R1} - \frac{\Delta R2}{R2} + \frac{\Delta R3}{R3} - \frac{\Delta R4}{R4}\right) \qquad \text{(ก.5)}$$

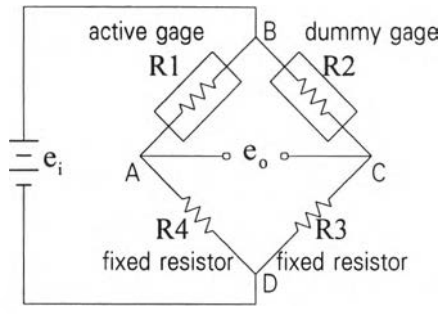
เมื่อ e_o =ค่าความต่างศักย์ไฟฟ้าออกของวงจรวีทสโตนบริดจ์

e_i =ค่าความต่างศักย์ไฟฟ้าเข้าของวงจรวีทสโตนบริดจ์

Rg =ความต้านทานกัลวาโนมิเตอร์ หรือ ความต้านทานทางออก

การชดเชยอุณหภูมิ

อุณหภูมิเป็นสัญญาณรบกวนตัวสำคัญ ในการใช้สเตรนเกจวัดความเครียดเนื่องจากความต้านทานของสเตรนเกจ เปลี่ยนไปตามความเครียดและอุณหภูมิ การเปลี่ยนแปลงความต้านทาน เนื่องจากความเครียดมีค่าน้อย ดังนั้นผลของอุณหภูมิอาจทำให้การวัดไม่ได้ผล ความไวต่ออุณหภูมิอาจมีผลมาจากการขยายตัวที่แตกต่างกันกับอุณหภูมิของวัสดุที่เป็นตัวอย่าง และวัสดุที่ทำเกจ ซึ่งอาจทำให้เกิดความต้านทานเปลี่ยนแปลงเนื่องจากความเครียด ถึงแม้ว่าวัสดุจะไม่รับแรงเลยก็ตาม ผลของอุณหภูมินี้ สามารถจะลบล้างออกด้วยการติดสเตรนเกจไว้ที่บริเวณไม่มีความเครียดไว้ใกล้กับสเตรนเกจที่ใช้งาน เพื่อให้เป็นอุณหภูมิเดียวกัน สเตรนเกจที่ไม่ได้ใช้งาน (dummy gage) นี้ จะต่อกับวงจรวีทสโตนบริดจ์ ดังแสดงในรูป ก.3



รูปที่ ก.3 วิธีชดเชยอุณหภูมิในการวัดความเครียด

ทรานสดิวเซอร์ความดัน (pressure transducer)

ทรานสดิวเซอร์ความดันใช้สเตรนเกจ จะทำการติดสเตรนเกจโดยตรงกึ่งกลางไดอะแฟรม ไดอะแฟรมมีความไม่เป็นเชิงเส้นเมื่อมีระยะยุบตัวมาก เมื่อการยืดตัวมีผลต่อการดัดธรรมดา ทำให้ไดอะแฟรมกระด้างขึ้น ความไม่เป็นเชิงเส้นของความเค้นนี้ใกล้เคียงกับความไม่เป็นเชิงเส้นของระยะยุบตัวที่จุดศูนย์กลาง และสามารถเขียนเป็นสมการดังนี้

$$P = \frac{16E.t^4}{3R^4(1-\upsilon^2)}\left(\frac{Yc}{t} + 0.488\left(\frac{Yc}{t}\right)^3\right) \qquad (ก.6)$$

เมื่อ P =ความดันแตกต่างบนสองข้างของไดอะแฟรม

E =ค่ามอดูลัสการยืดของไดอะแฟรม

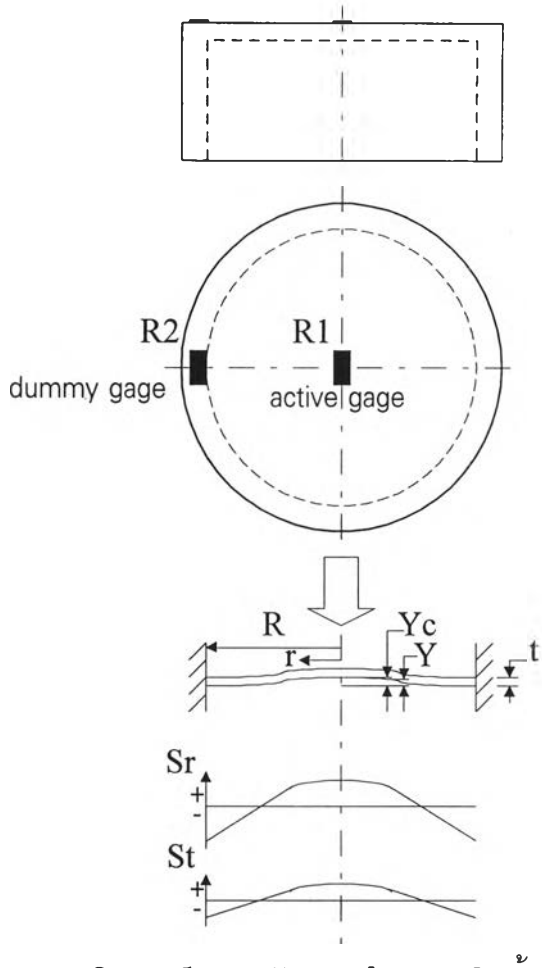
t =ความหนาของไดอะแฟรม

=อัตราส่วนพอยซอน

R =รัศมีของไดอะแฟรมที่จุดยึดที่ขอบ

Yc =ระยะยุบตัวที่จุดศูนย์กลางของไดอะแฟรม

ถ้าออกแบบ Yc/t น้อยพอ ก็อาจมีความเป็นเชิงเส้นได้ แต่ค่า Yc/t น้อย จะทำให้ความเครียดน้อยและความต่างศักย์ไฟฟ้าออก e_o ก็จะน้อยไปด้วย



รูปที่ ก.4 ทรานสดิวเซอร์ความดันและตำแหน่งติดตั้งสเตรนเกจ

ไดอะแฟรมที่ถูกความดันกระทำจะเกิดเปล่งตัวออกหรือยืดตัว จะมีความเค้นทางด้านความดันต่ำในทางรัศมี Sr และความดันแทนเจ้นท์ St เกิดขึ้น ดังสมการ ก.7 และ ก.8

$$Sr = \frac{3P.R^2\upsilon}{8t^2}\left(\left(\frac{1}{\upsilon}+1\right)-\left(\frac{3}{\upsilon}+1\right)\left(\frac{r}{R}\right)^2\right) \qquad (ก.7)$$

$$St = \frac{3P.R^2\upsilon}{8t^2}\left(\left(\frac{1}{\upsilon}+1\right)-\left(\frac{1}{\upsilon}+3\right)\left(\frac{r}{R}\right)^2\right) \qquad (ก.8)$$

เมื่อ r =รัศมีของไดอะแฟรมที่จุดใด ๆ

และสมการระยะยุบตัวของไดอะแฟรม ดังสมการที่ ก.9

$$Y = \frac{3P(1-\upsilon^2)(R^2 - r^2)^2}{16E.t^3} \qquad (ก.9)$$

เมื่อ Y =ระยะยุบตัวที่จุดใด ๆ ของไดอะแฟรม

เราจะใช้สมการ ก.7 และ ก.8 โดยตรงไม่ได้เพราะผิวของไดอะแฟรมอยู่ในความเค้นสองแกน คือในทั้งทางแทนเจ้นท์และทางรัศมี การเกี่ยวเนื่องของความเค้นสองแกนให้สมการดังนี้

$$\varepsilon_r = \frac{Sr - \upsilon.St}{E} \qquad (ก.10)$$

$$\varepsilon_t = \frac{St - \upsilon.Sr}{E} \qquad (ก.11)$$

ในการติดสเตรนเกจ R1 (active gage) ไว้ที่จุดกึ่งกลางของไดอะแฟรม เพื่อวัดความเครียดทางแทนเจ้นท์เพราะความเครียดที่จุดนี้สูงสุด และติดสเตรนเกจ R2 (dummy gage) เพื่อเป็นการชดเชยอุณหภูมิ ไว้ที่ขอบนอกไดอะแแฟรม เพื่อให้เป็นอุณหภูมิเดียวกัน สมการความเครียดทางแทนเจ้นท์ที่จุดกึ่งกลาง ใช้สมการ ก.7 , ก.8 , ก.10 และ ก.11 จะได้

$$\varepsilon_t = \frac{3P.R^2}{8t^2E}(1-\upsilon^2) \qquad (ก.12)$$

จากการติดตั้งสเตรนเกจไว้ที่จุดกึ่งกลางไดอะแฟรมเพียงตัวเดียว เมื่อทำการแทนลงในสมการ ก.5 โดยให้ค่าความต้านทาน Rg มีค่าเข้าใกล้อนันต์ ด้วยการหาค่าลิมิตของ e_o

$$e_o = \lim_{(Rg \to \alpha)} e_o = \frac{E\Delta R}{4R}$$

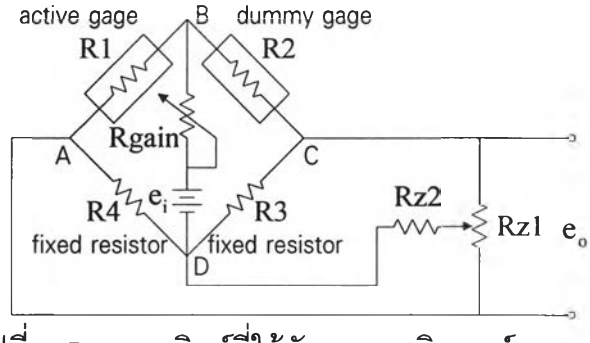
$$e_o = \frac{E.GF.\varepsilon}{4} \qquad \text{(ก.13)}$$

การออกแบบทรานสดิวเซอร์ความดัน

สเตรนเกจที่ใช้ รุ่น B-FAE-5-12 ความยาวเกจ (gage length) 5 มม.ความต้านทาน 120 ±0.2 โอห์ม เกจแฟคเตอร์ 2.1

ไดอะแฟรมที่ใช้เป็นทองเหลือง มีขนาดเส้นผ่าศูนย์กลาง 15 มม. จะมีค่ามอดูลัสความหยุ่น E = 106.0 Gpa , อัตราพอยซอน υ = 0.324 , ความหนาแน่น ρ = 8.55 Mg/m^3 ความดันใช้งานที่ 20 บาร์ แต่ใช้ความปลอดภัยเท่ากับ 2 จึงใช้ความดันที่ 40 บาร์ ในการออกแบบ ค่าความเครียดที่ 1,000 ไมโครสเตรน $\mu\varepsilon$ เมื่อทำการคำนวณใช้สมการ ก.12 จะได้ความหนา t = 0.844 มม. หรือประมาณ 0.9 มม.

ชุดวงจรบริดจ์ที่ใช้กับทรานสดิวเซอร์ความดัน



รูปที่ ก.5 วงจรบริดจ์ที่ใช้กับทรานสดิวเซอร์ความดัน

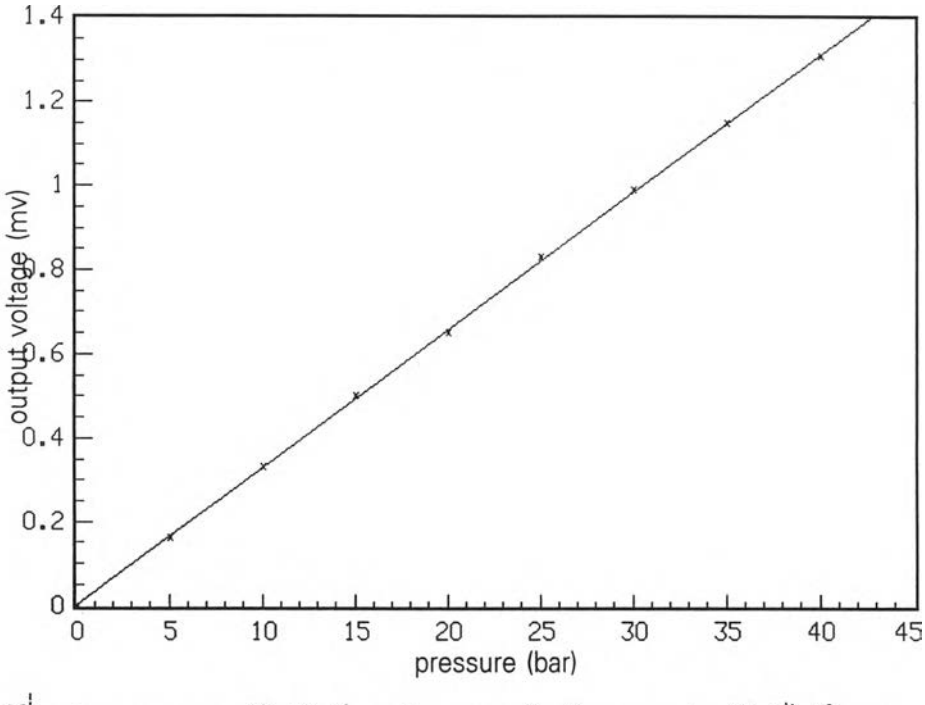
รายการอุปกรณ์

R1 , R2	สเตรนเกจ ความต้านทาน 120 Ω เกจแฟคเตอร์ 2.1
R3 , R4	ความต้านทาน 120 Ω
Rz1	ความต้านทานปรับค่าได้ 50 KΩ

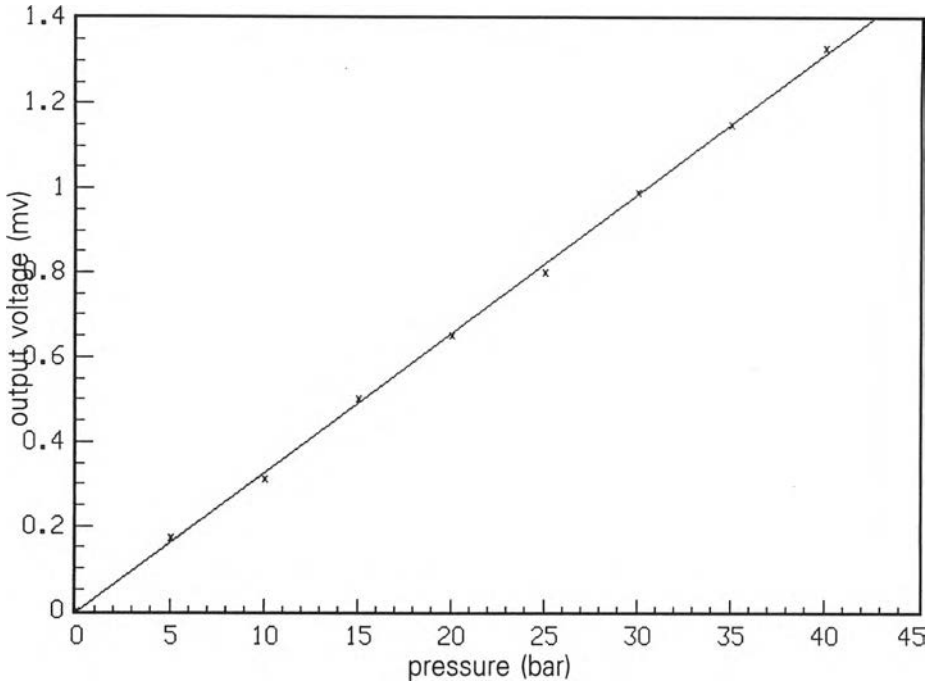
Rz2	ความต้านทาน 12 KΩ
Rgain	ความต้านทานปรับค่าได้
e_i	ความต่างศักย์เข้า 15 โวลต์

การปรับเทียบค่า

จากรูปที่ ก.6 และ ก.7 จะแสดงกราฟของความสัมพันธ์ระหว่างความดันกับความต่างศักย์ไฟฟ้าออก (วิธีการกำลังสองน้อยที่สุด) สำหรับทรานสดิวเซอร์ความดัน1 และทรานสดิวเซอร์ความดัน2



รูปที่ ก.6 แสดงความสัมพันธ์ระหว่างความดันกับความต่างศักย์ไฟฟ้าออกของทรานสดิวเซอร์ความดัน1



รูปที่ ก.7 แสดงความสัมพันธ์ระหว่างความดันกับความต่างศักย์ไฟฟ้าออกของทรานสดิวเซอร์ความดัน2

ภาคผนวก ข.

การออกแบบตัวกระตุ้นแบบโซลินอยด์

จากการทดลองสร้างความดันโดยแม่ปั๊มห้ามล้อบนแบบจำลอง ความดันที่สามารถสร้างได้สูงสุดประมาณ 40 บาร์ และความดันที่ใช้ปกติ 10 - 20 บาร์ ความดันที่ใช้ในการออกแบบตัวกระตุ้นแบบโซลินอยด์ 20 บาร์ เลือกขนาดเส้นผ่านศูนย์กลางภายในของลูกสูบเคลื่อนที่ (plunger) ให้มีขนาด 30 มม. เพื่อสวมกับลูกสูบอยู่กับที่ (fixed piston) แรงที่ลูกสูบเคลื่อนที่จะต้องสร้าง 1413.7 N. โดยใช้กระแสสูงสุดในการออกแบบ 10 แอมป์

ให้พิจารณาจากรูป 3.4 จะมีค่าพารามิเตอร์ดังนี้

$G = 0.005$ m.

$x = 0.0$ m.

$\mu_o = 4\pi \times 10^{-7}$ H/m

$a' = 0.0205$ m.

$b' = 0.001$ m.

$l' = 0.01$ m.

$r_1' = 0.015$ m.

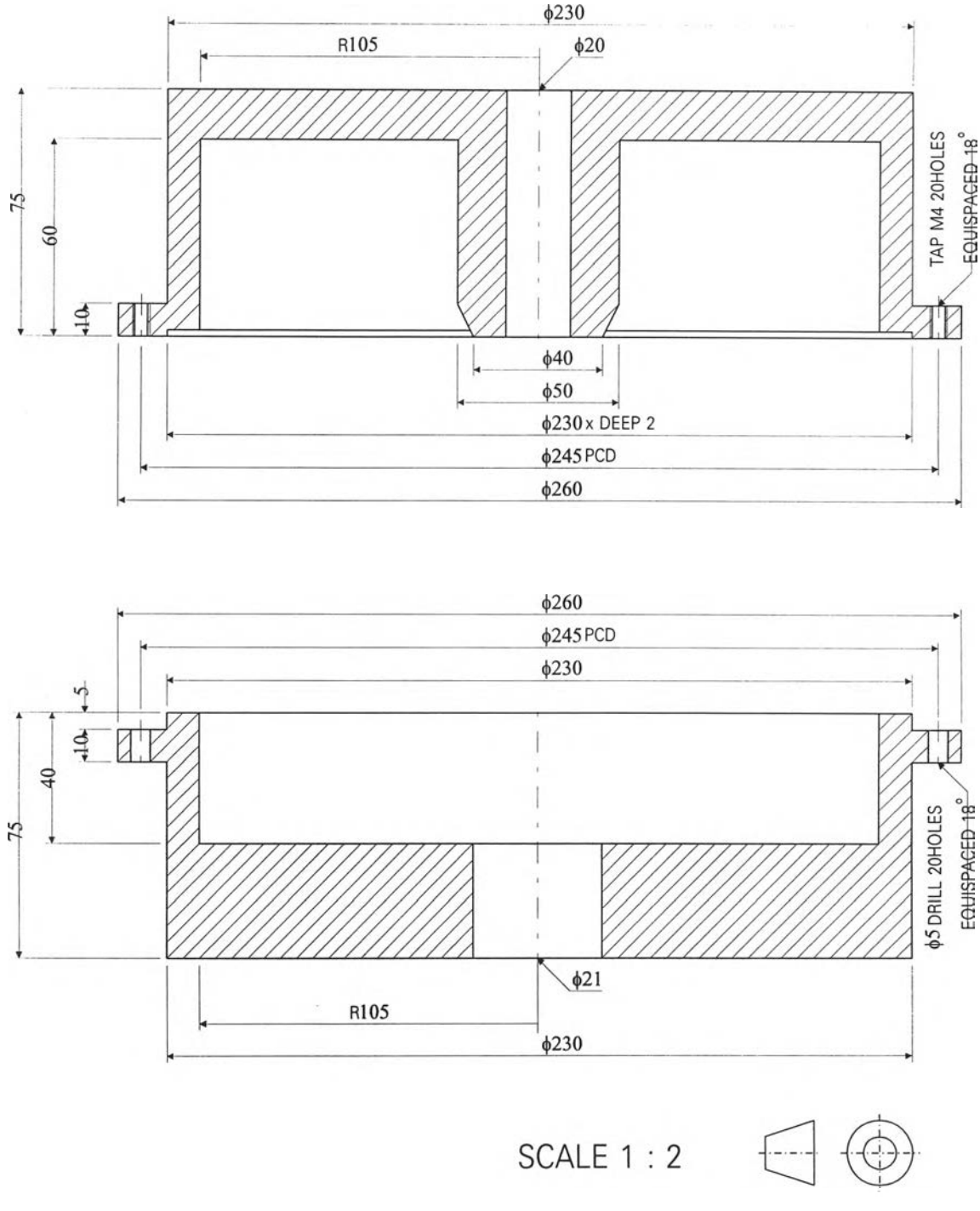
$r_2' = 0.02$ m.

โดยแทนลงในพารามิเตอร์ที่สมการ 3.21 จะได้

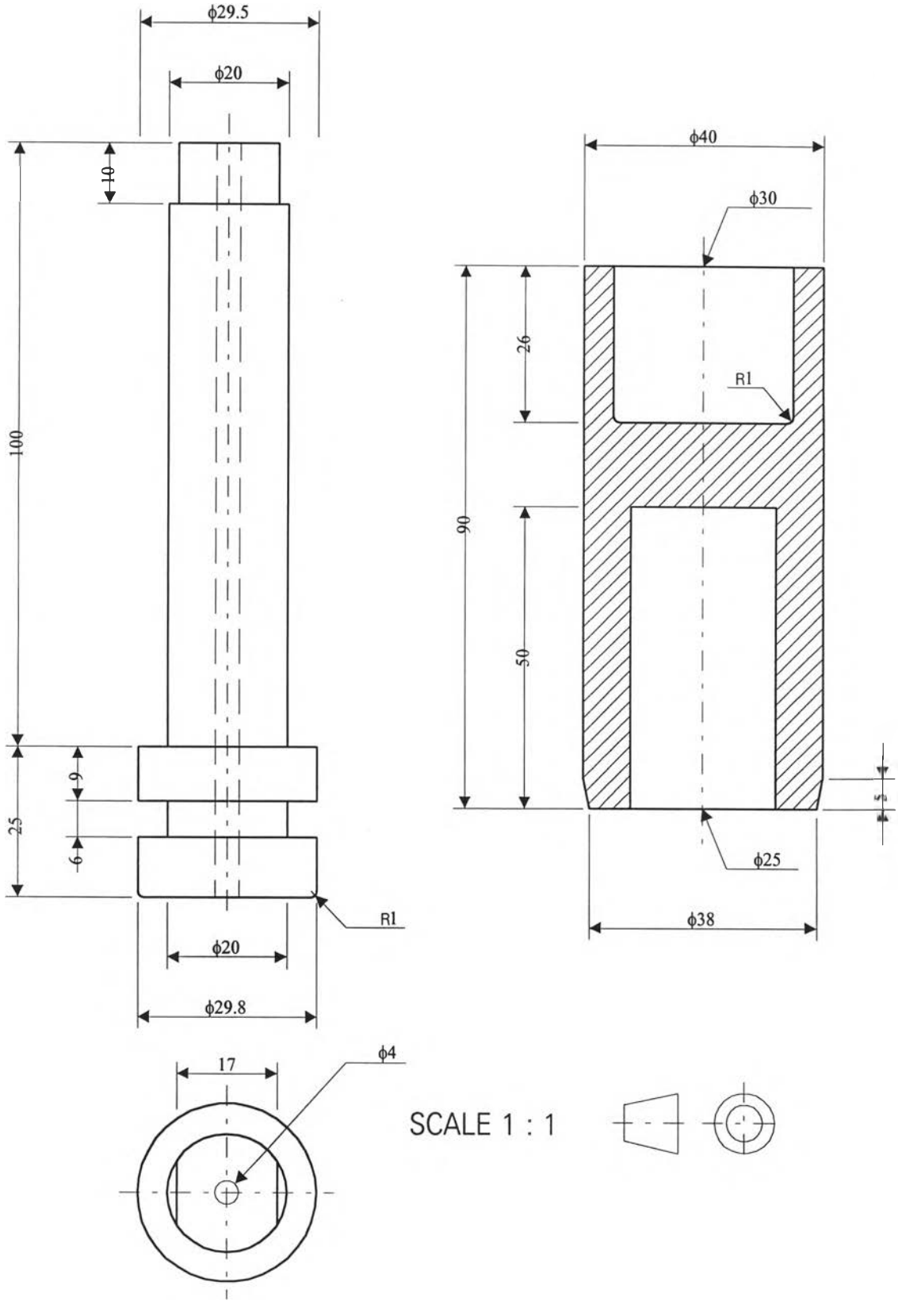
$K_1' = 2.832576463 \times 10^{-13}$

$K_2' = 4.1 \times 10^{-4}$

$K_3' = 1.75 \times 10^{-7}$



รูปที่ ข.1 แบบภาพรายละเอียด กระบอกเสื้อบนและกระบอกเสื้อล่าง ของตัวกระตุ้นแบบโซลินอยด์



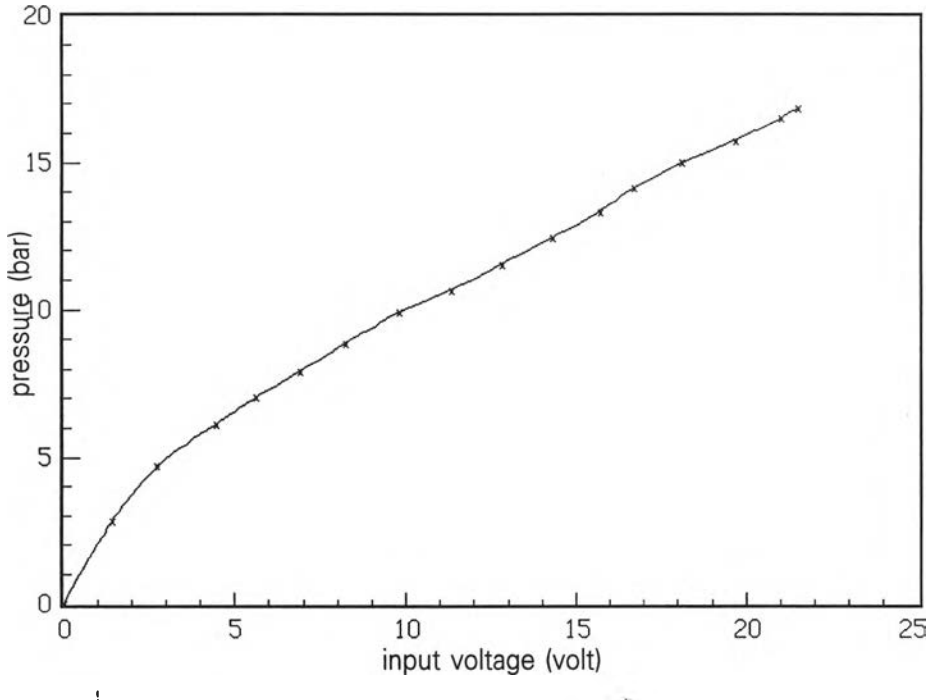
รูปที่ ข.2 แบบภาพรายละเอียด ลูกสูบอยู่กับที่และลูกสูบเคลื่อนที่
ของตัวกระตุ้นแบบโซลินอยด์

จากสมการ 3.22 เป็นสมการแรงของลูกสูบเคลื่อนที่ เพื่อหาจำนวนรอบของขดลวดที่ใช้พันคอยล์ N = 1097.8 รอบ และใช้ค่าแฟคเตอร์ 1.1 ฉะนั้นจะได้จำนวนรอบ 1220 รอบ ใช้ขนาดขดลวด 14 SWG มีเส้นผ่านศูนย์กลางโต 2.032 มม.

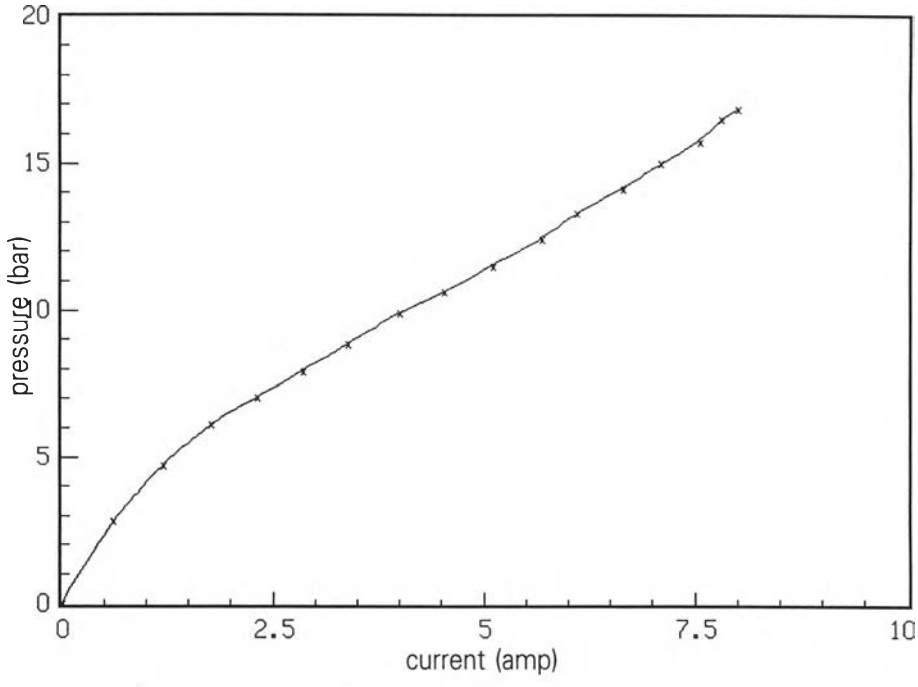
แบบภาพรายละเอียดของตัวกระตุ้นแบบโซลินอยด์ ดังแสดงในรูปที่ ข.1 , ข.2 และเมื่อทำการทดสอบการทำงานที่ภาระเต็มที่ สามารถสร้างความดันที่ 16.8 บาร์ กระแสสูงสุด 8.0 แอมแปร์ ที่ 21.5 โวลต์ ความสิ้นเปลืองกำลังพลังงาน 172 วัตต์

การปรับเทียบค่า

จากรูปที่ ข.3 จะแสดงกราฟของความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้ากับความดัน และ ข.4 จะแสดงกราฟของความสัมพันธ์ระหว่างกระแสไฟฟ้ากับความดัน



รูปที่ ข.3 ความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้ากับความดัน
ของตัวกระตุ้นแบบโซลินอยด์

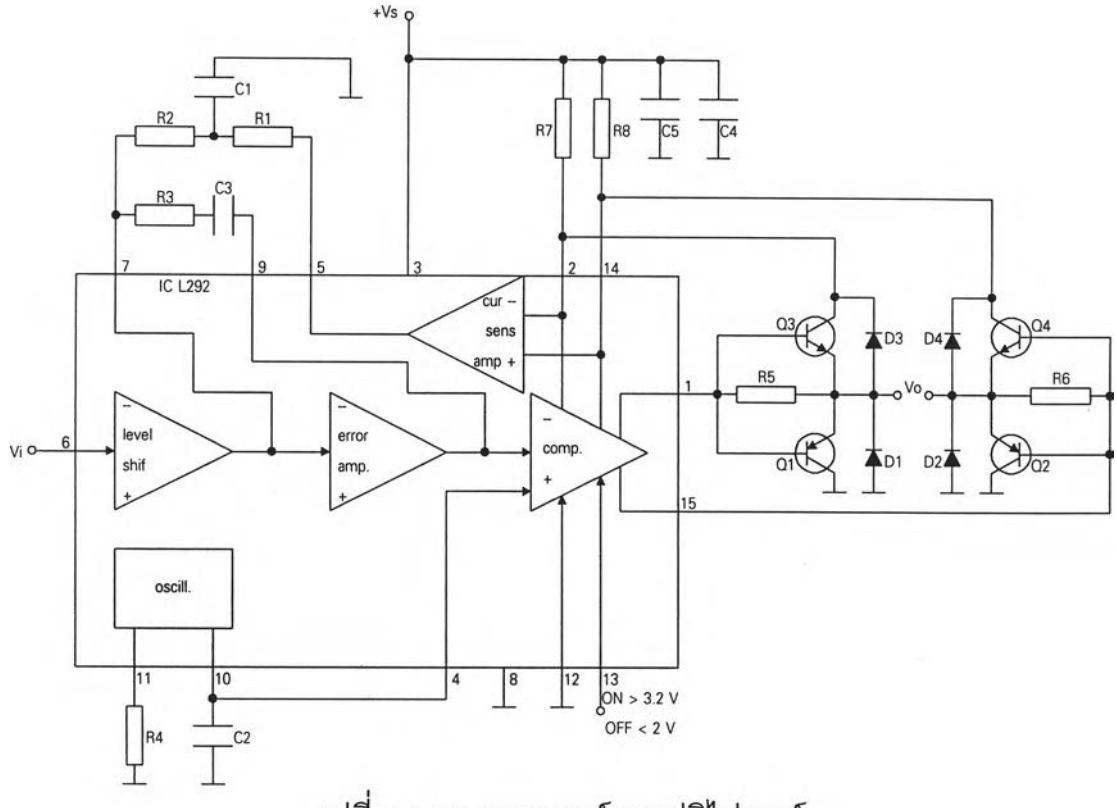


รูปที่ ข.4 ความสัมพันธ์ระหว่างกระแสไฟฟ้ากับความดัน
ของตัวกระตุ้นแบบโซลินอยด์

ภาคผนวก ค.

เพาเวอร์แอมปลิไฟเออร์

เพาเวอร์แอมปลิไฟเออร์เป็นชุดขับตัวกระตุ้นแบบโซลินอยด์และลิ้นเข็มเซอร์โว จะใช้ด้วยกัน 2 ชุด



รูปที่ ค.1 วงจรเพาเวอร์แอมปลิไฟเออร์

รายการอุปกรณ์

R1,R2	=	470	Ω
R3	=	3.3	MΩ
R4	=	15	KΩ

R5,R6 = 10 Ω

R7,R8 = 0.03 Ω

C1 = 47 ηF

C2 = 1.5 ηF

C3 = 10 ηF

C4 = 0.22 μF

C5 = 470 μF

D1,D2 = 6 A Fast diodes

D3,D4 = 6 A Fast diodes

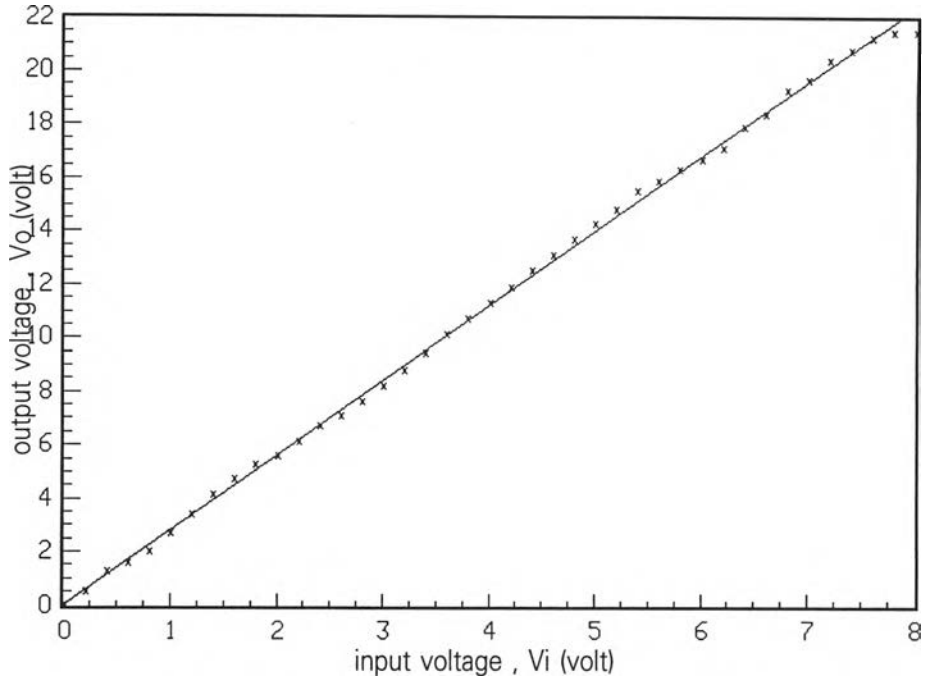
Q1,Q2 = BDW 52 A

Q3,Q4 = BDW 51 A

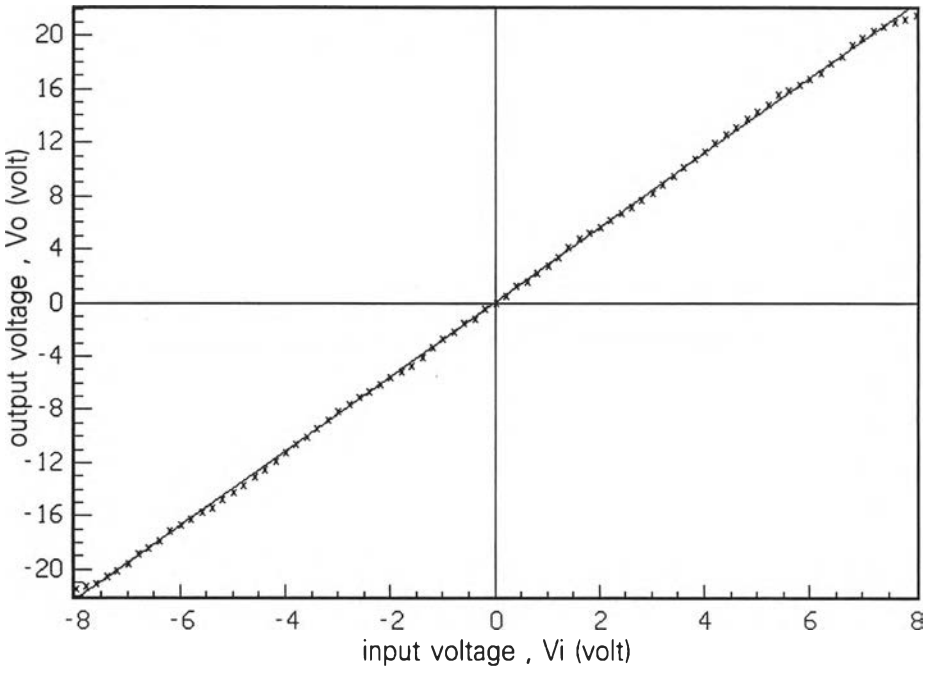
IC L292

การปรับเทียบค่า

จากรูปที่ ค.2 จะแสดงกราฟของความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้าเข้ากับความต่างศักย์ไฟฟ้าออกของชุดเพาเวอร์แอมปลิไฟเออร์สำหรับขับตัวกระตุ้นแบบโซลินอยด์ และ ค.3 จะแสดงกราฟของความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้าเข้ากับความต่างศักย์ไฟฟ้าออกของชุดเพาเวอร์แอมปลิไฟเออร์สำหรับขับลิ้นเข็มเซอร์โว (วิธีการกำลังสองน้อยที่สุด)



รูปที่ ค.2 แสดงความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้าเข้ากับความต่างศักย์ไฟฟ้าออกของชุดเพาเวอร์แอมปลิไฟเออร์สำหรับขับตัวกระตุ้นแบบโซลินอยด์

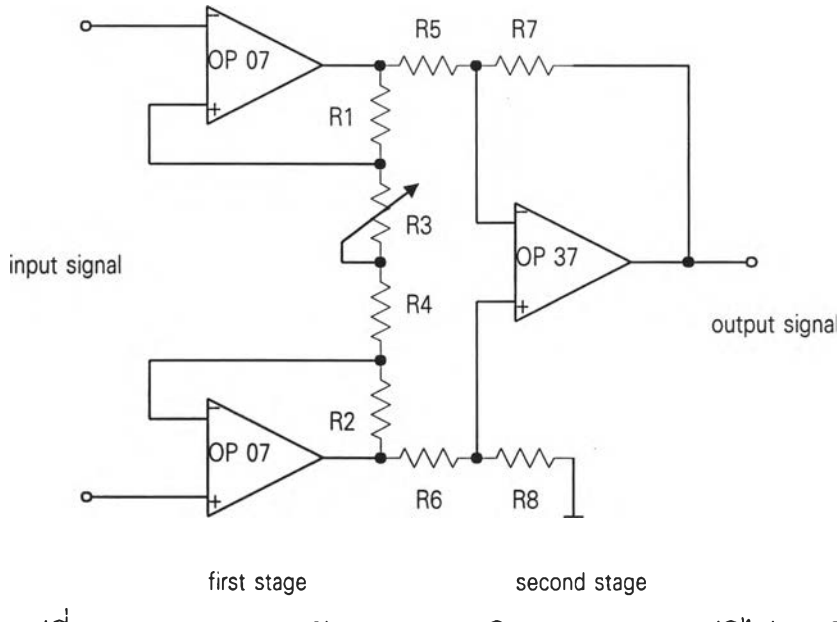


รูปที่ ค.3 แสดงความสัมพันธ์ระหว่างความต่างศักย์ไฟฟ้าเข้ากับความต่างศักย์ไฟฟ้าออกของชุดเพาเวอร์แอมปลิไฟเออร์สำหรับขับลิ้นเข็มเซอร์โว

ภาคผนวก ง.

ชุดขยายสัญญาณแบบอินสตรูเมน แอมปลิไฟเออร์

ชุดขยายสัญญาณแบบอินสตรูเมน แอมปลิไฟเออร์เป็นอุปกรณ์ขยายสัญญาณของทรานสดิวเซอร์ความดัน ทำการขยายสัญญาณประมาณ 5710 เท่า จัดทำ 2 ชุด second stage จะมีอัตราขยายเท่ากับ 200 และ first stage สามารถปรับอัตราขยายได้อยู่ในช่วง 17.67 - 101 ฉะนั้นจะมีอัตราขยายรวม 3533.33 - 20200



รูปที่ ง.1 วงจรชุดขยายสัญญาณแบบอินสตรูเมน แอมปลิไฟเออร์

รายการอุปกรณ์

R1,R2	=	10	KΩ
R3 (trim)	=	1	KΩ
R4	=	200	Ω
R5,R6	=	500	Ω

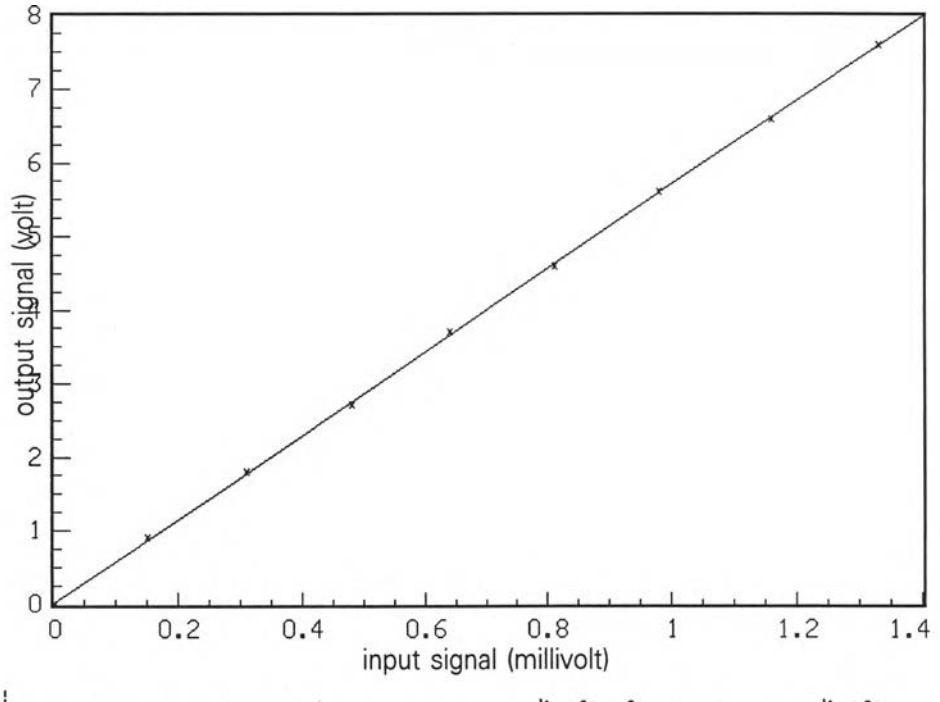
R7,R8 = 100 KΩ

IC OP 07

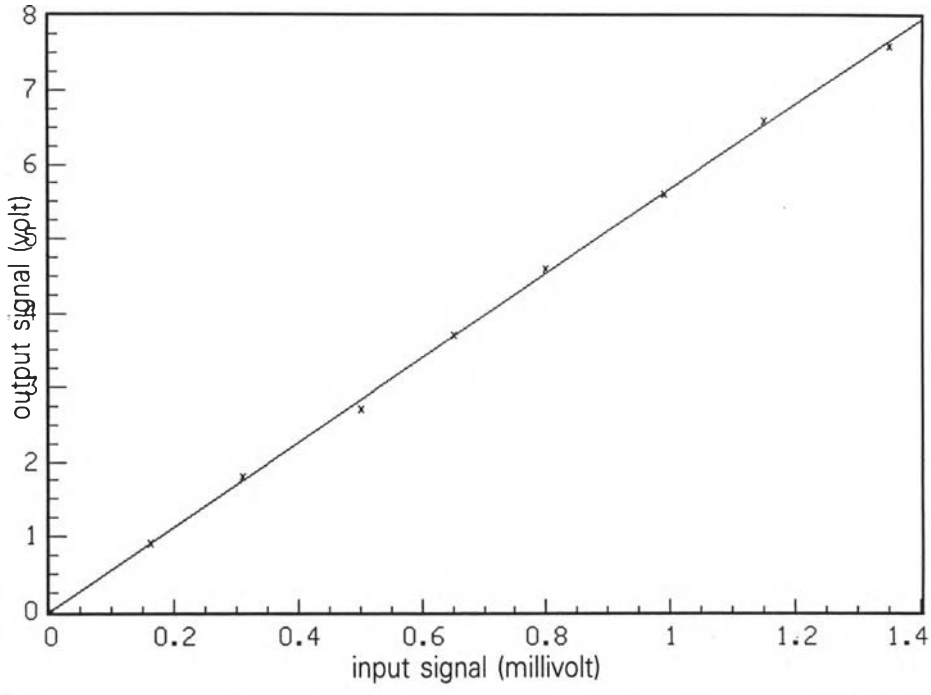
IC OP 37

การปรับเทียบค่า

จากรูปที่ ง.2 จะแสดงกราฟของความสัมพันธ์ระหว่างสัญญาณไฟฟ้าเข้ากับสัญญาณไฟฟ้าออกของชุดขยายสัญญาณแบบอินสตรูเมนแอมปลิไฟเออร์สำหรับทรานสดิวเซอร์ความดัน1 และ ง.3 จะแสดงกราฟของความสัมพันธ์ระหว่างสัญญาณไฟฟ้าเข้ากับสัญญาณไฟฟ้าออกของชุดขยายสัญญาณแบบอินสตรูเมนแอมปลิไฟเออร์สำหรับทรานสดิวเซอร์ความดัน2 (วิธีการกำลังสองน้อยที่สุด)



รูปที่ ง.2 แสดงความสัมพันธ์ระหว่างสัญญาณไฟฟ้าเข้ากับสัญญาณไฟฟ้าออกของชุดขยายสัญญาณแบบอินสตรูเมนแอมปลิไฟเออร์สำหรับทรานสดิวเซอร์ความดัน1

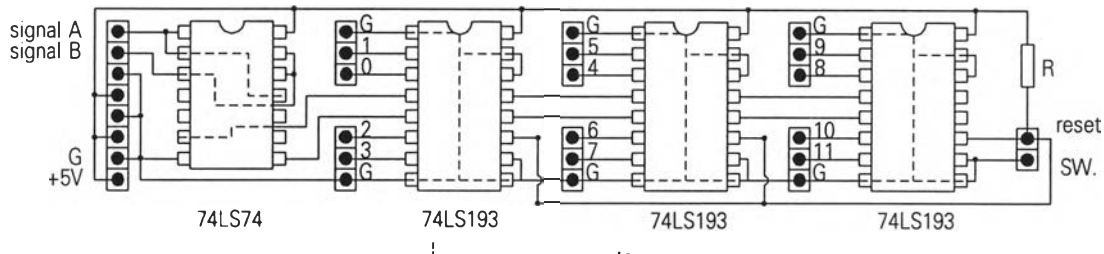


รูปที่ ง.3 แสดงความสัมพันธ์ระหว่างสัญญาณไฟฟ้าเข้ากับสัญญาณไฟฟ้าออก ของชุดขยายสัญญาณแบบอินสตรูเมนแอมปลิไฟเออร์สำหรับทรานสดิวเซอร์ความดัน2

ภาคผนวก จ.

ชุดดีโคดเดอร์

ชุดดีโคดเดอร์เป็นชุดแปลงสัญญาณของชุดตรวจจับความเร็วรอบของล้อรถยนต์ทั้งสอง และลิ้นเข็มเซอร์โว ออกแบบเป็น 12 บิต จัดสร้าง 3 ชุด



รูปที่ จ.1 วงจรชุดดีโคดเดอร์

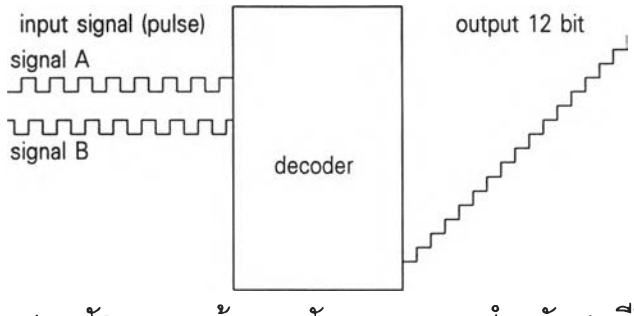
รายการอุปกรณ์

R = 3.6 KΩ

IC 74 LS 74

IC 74 LS 193

สัญญาณเข้าและสัญญาณออก



รูปที่ จ.2 แสดงสัญญาณเข้าและสัญญาณออกสำหรับชุดดีโคดเดอร์

ภาคผนวก ฉ.

การ์ดอินเตอร์เฟส

การ์ดอินเตอร์เฟส จะใช้ด้วยกัน 2 การ์ดคือ 8255 I/O CARD FPC-024 และ PC ADDA-14 CARD FPC-011

8255 I/O CARD FPC-024

The 8255 I/O card is a programmable input/output interface for PC or PC/XT. This card contains 48 I/O line , 3 independent 16 bit counters , each with a count rate of up to 2 MHz. All modes of operation are software programmable.

specifications

- 48 programmable I/O control line
- 3 independent 16 bit count
- 16 LED I/O display

I/O port address :

$1b0 - $1bf

PC ADDA-14 CARD FPC-011

The ADDA-14 card is an analog - digital / digital - analog high performance data conversion card for IBM PC XT/AT and compatible systems. The card operates with two fourteen bit resolution channels for digital to analog in either unipolar or bipolar operation depending on the switches (SW1 & SW2) settings and sixteen

channels with fourteen bit resolution for analog to digital data conversion also operating in either unipolar or bipolar operation.

specifications

A/D :

- 14 - bit resolution
- 16 input channels
- unipolar (0 - 8V) or bipolar (-8V to +8V) input levels
- < 42 μs conversion time
- ±1/2 LSB relative accuracy at 25°C
- 88 ppm/°C temperature coefficient

D/A :

- 14 - bit resolution
- two channels for output
- unipolar (0 - 8V) or bipolar (-8V to +8V) output
- 0.5 μs current setting time

I/O port address :

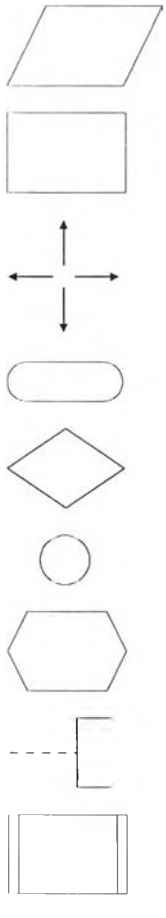
$170 - $17f

ภาคผนวก ซ.

ผังโปรแกรมโครงสร้าง

สัญลักษณ์ของผังงาน

การกำหนดมาตรฐานของผังงานเพื่อสะดวกในการสื่อความหมายให้เข้าใจตรงกันและเป็นระบบสากล เรียกว่ามาตรฐานของ ANSI (The American National Standard Institute) สัญลักษณ์ที่ใช้แทนผังงานแบบต่าง ๆ แสดงไว้ดังนี้



ใช้แสดงคำสั่งที่เกี่ยวกับอินพุตและเอาต์พุต

ใช้แสดงกิจกรรมประมวลผล เช่น คำนวณ

แสดงทิศทาง

แสดงจุดเริ่มต้นหรือจุดสุดท้ายของกิจกรรม

แสดงการตัดสินใจ

แสดงการเชื่อมโยง

แสดงการเตรียมการ

แสดงหมายเหตุ

แสดงโปรแกรมย่อย

ลักษณะของโปรแกรมควบคุมการลื่นไถล

รายละเอียดของโปรแกรมควบคุมการลื่นไถลที่ได้ประดิษฐ์ขึ้นมา จะอธิบายโดยใช้ผังโปรแกรมโครงสร้าง จะมีด้วยกัน 44 หน้า ในแต่ละหน้า ที่ส่วนบนจะเป็นกรอบชื่อ จะมีชื่อโปรแกรมหลักหรือโปรแกรมย่อย , หมายเลขเรียกหน้า และหน้าที่ถูกอ้างอิงถึง เพื่อที่จะสามารถค้นหาได้สะดวก ส่วนด้านซ้ายของหน้าจะเป็นกรอบผังโปรแกรมโครงสร้าง ซึ่งใช้มาตราฐาน ANSI แสดงผังโปรแกรมโครงสร้าง และที่ตำแหน่งสัญลักษณ์แสดงหมายเหตุ จะชี้ถึงหน้าที่อ้างอิงถึง โดยสามารถพลิกไปที่หน้าดังกล่าวโดยดูจากกรอบชื่อ และส่วนด้านขวาของหน้าจะเป็นกรอบรายละเอียดของตัวโปรแกรมหลักหรือโปรแกรมย่อย แสดงด้วยภาษาปาสคาล

โปรแกรมควบคุมการลื่นไถล ประกอบด้วยโปรแกรมหลัก (main program) และ 43 โปรแกรมย่อย (procedures) ลักษณะขั้นตอนที่สำคัญของโปรแกรมควบคุมการลื่นไถลนี้ประกอบด้วย

1. โปรแกรมหลัก [slip_control] อ้างอิงถึง page 1 ในผังโปรแกรมโครงสร้าง

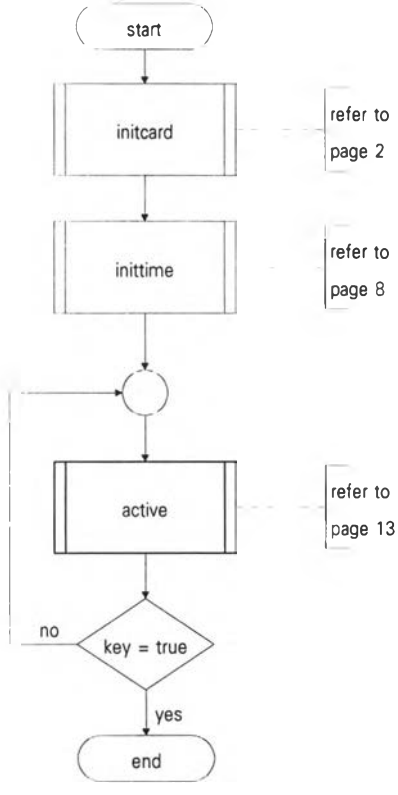
โปรแกรมหลักเริ่มจากการตั้งค่าเริ่มต้นของการ์ดอินเตอร์เฟสด้วยโปรแกรมย่อย [initcard] เพื่อให้การ์ดพร้อมที่จะทำงาน จากนั้นจะเริ่มตั้งค่าเริ่มต้นของเวลาด้วยโปรแกรมย่อย [inittime] เป็นเวลาสุ่ม ช่วงเวลาสุ่มที่ใช้คือ 0.002 วินาที และ 0.02 วินาที เมื่อพร้อมแล้วจะเข้าสู่โปรแกรมย่อย [active] จะกระทำซ้ำจนกว่าจะมีการกดคีย์ เพื่อยุติโปรแกรมควบคุมการลื่นไถล

2. โปรแกรมย่อย [active] อ้างอิงถึง page 1 , 13 ในผังโปรแกรมโครงสร้าง

เป็นขั้นตอนการควบคุมการทำงานของระบบ ABS เริ่มต้นโดยตั้งค่าพารามิเตอร์ด้วยโปรแกรมย่อย [initparameter] ที่ต้องใช้ในโปรแกรมย่อย [active] นี้ ให้มอเตอร์เซอร์โวหมุนเปิดลิ้นเข็มเซอร์โวที่ 10 รอบ เพื่อให้วงจรไฮดรอลิกต่อถึงกัน จากนั้นจะทำการตรวจสอบด้วยโปรแกรมย่อย [loopcheck] ซ้ำ ๆ จนกว่าอัตราลื่นไถล มีค่าเกินกว่าค่าที่ตั้งไว้ (15%) จะให้มอเตอร์เซอร์โวหมุนปิดลิ้นเข็มเซอร์โวที่ 10 รอบ เพื่อทำการตัดวงจรไฮดรอลิกออกจากกัน ต่อด้วยโปรแกรมย่อย [prewheel] และจะกระทำซ้ำ ๆ ด้วยโปรแกรมย่อย [wheel] จนกว่าจะปล่อยหรือคายคันเหยียบห้ามล้อ หรือมีการกดคีย์

program slip_control

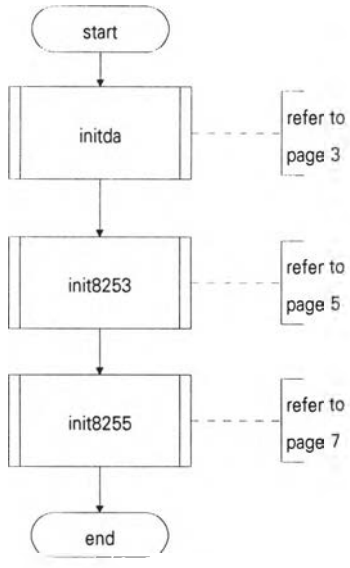
page 1 refer to -




```
{
an automotive slip control braking system program
}
program    slip_control;
uses crt;
type  float = real;
var    t1, t2 : float;
       tim, tdm, tip, tdp, tds, tiw, tdw : float;
       kpm, kim, kdm, km0, km1, km2 : float;
       kpp, kip, kdp, kp0, kp1, kp2 : float;
       kpw, kiw, kdw, kw0, kw1, kw2 : float;
       enm, enm1, enm2, enp, enp1, enp2,
       enw, enw1, enw2 : float;
       rnm, rnp, rnw : float;
       mnm, mnm1, mnp, mnp1, mnw, mnw1 : float;
       pulse, revolve : float;
       intervalpulse : integer;
       pulsestart : word;
       pressure1, pressure2 : float;
       pulsestarta, pulsestartb : word;
       intervalpulsea, intervalpulseb:integer;
       omegaa, omegab, slipratio : float;
       count0, count1, count2 : word;
       counter : word;
       intervaltime0, intervaltime1,
       intervaltime2, intervaltime3 : word;
       time : integer;
       timestart1, timestart2 : byte;
       timestop1, timestop2 : byte;
       upperboundedslip : float;
       key, panel, checkpressure : boolean;
       ch : char;
begin{main}
       initcard;
       inittime;
       repeat
         active;
       until key = true;
end. {main}
```

procedure initcard

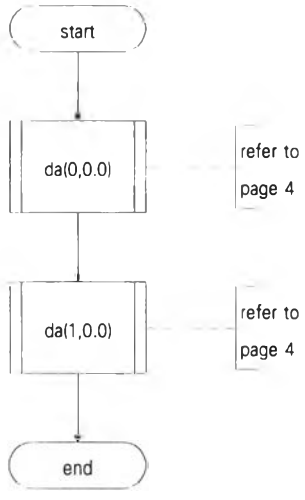
page 2 refer to 1




```
{
initial interface card
}
procedure  initcard;
begin
        initda;
        init8253;
        init8255;
end;
```

procedure initda

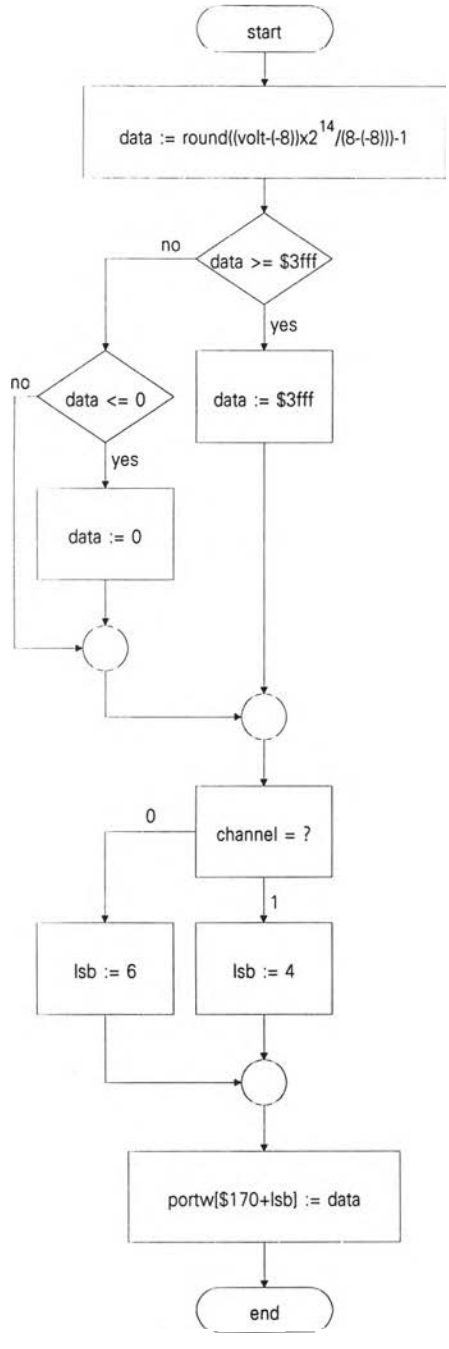
page 3 refer to 2




```
{
initial digital to analog conversion
}
procedure  initda;
begin
      da(0,0.0);
      da(1,0.0);
end;
```

procedure da(channel,volt)

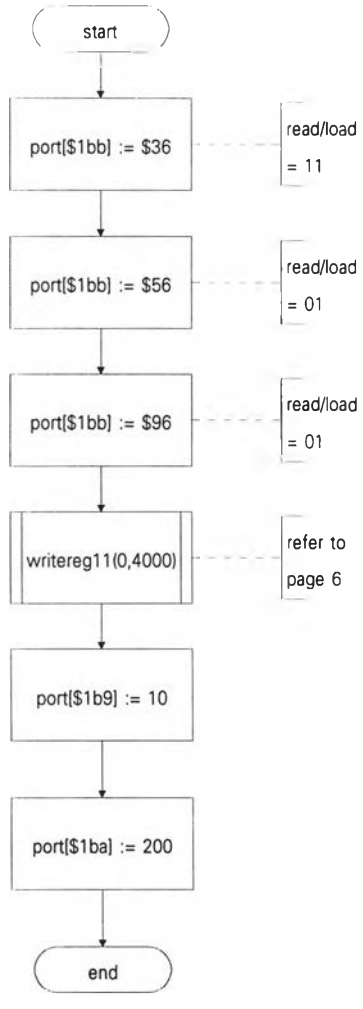
page 4 refer to 3,16,19,25,41




```
{
digital to analog conversion procedure
}
procedure  da(channel:byte;volt:float);
var    data,msb,lsb : word;
begin
       data := round((volt-(-8.0))*16384.0/(8.0-(-8.0)))-1;
       if (data >= $3fff) then data := $3fff else
       if (data <= 0) then data := 0;
       case channel of
         0 : lsb := 6;
         1 : lsb := 4;
       end;
       portw[base+lsb] := data;
end;
```

procedure init8253

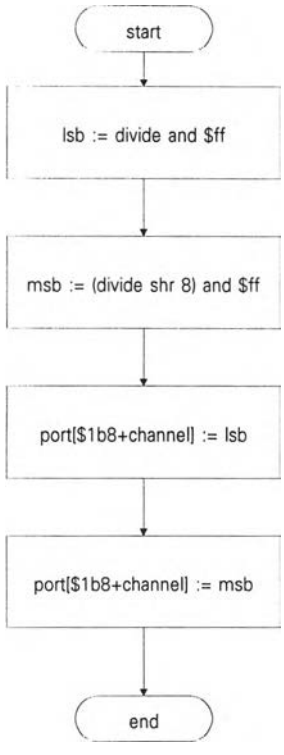
page 5 refer to 2




```
{
initial timer/counter chip for setting up timing loops
and writing on MODE control word register
}
procedure  init8253;
begin
        port[base+11] := $36;
        port[base+11] := $56;
        port[base+11] := $96;
        writereg11(0,4000);
        port[base+9]  := 10;
        port[base+10] := 200;
end;
```

procedure writereg11(channel,divide)

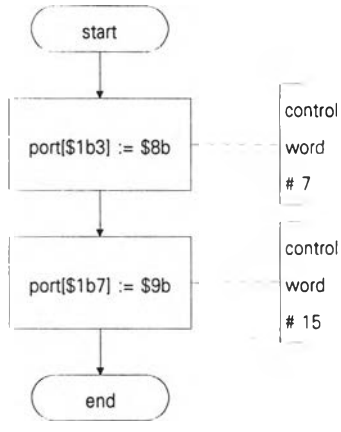
page 6 refer to 5




```
{
writing on MODE control word register
}
procedure  writereg11(channel:byte;divide:word);
var   lsb, msb : byte;
begin
      lsb := divide and $ff;
      msb := (divide shr 8) and $ff;
      port[base+8+channel] := lsb;
      port[base+8+channel] := msb;
end;
```

procedure init8255

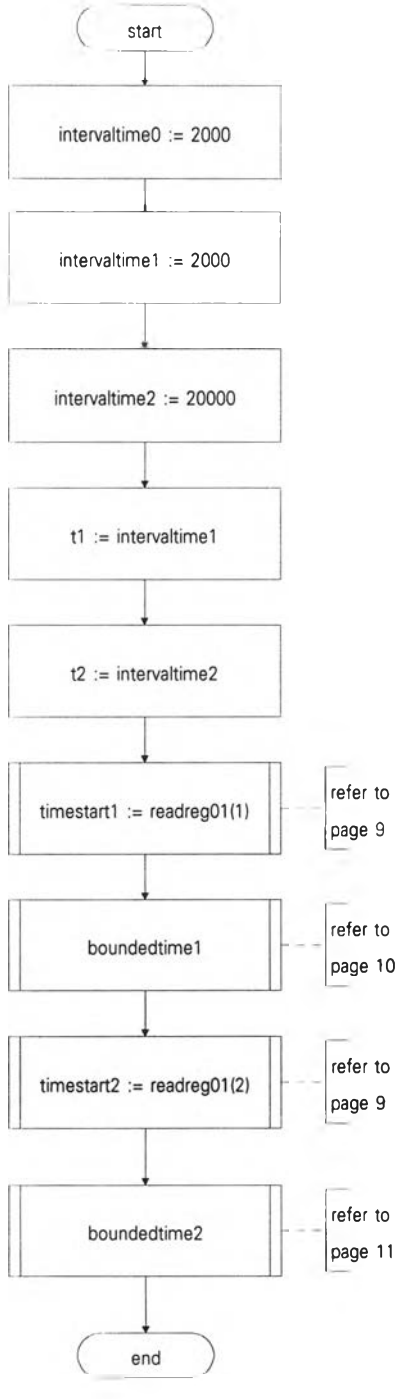
page 7 refer to 2




```
{
initial I/O chip
}
procedure  init8255;
begin
        port[base+3] := $8b;
        port[base+7] := $9b;
end;
```

procedure inittime

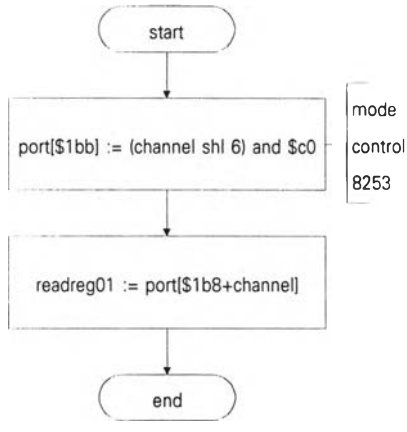
page 8 refer to 1




```
{
initial time for interval time
}
procedure  inittime;
begin
      intervaltime0 := 2000;
      intervaltime1 := 2000;
      intervaltime2 := 20000;
      t1 := intervaltime1;
      t2 := intervaltime2;
      timestart1 := readreg01(1);
      boundedtime1;
      timestart2 := readreg01(2);
      boundedtime2;
end;
```

procedure readreg01(channel)

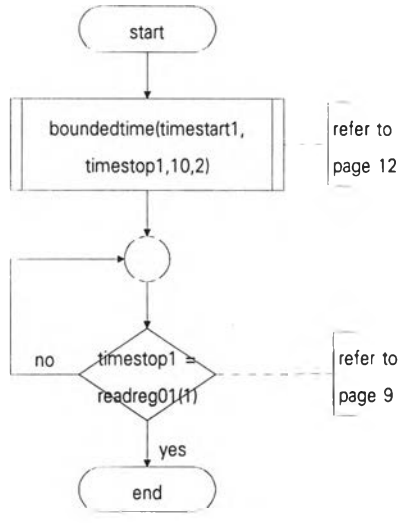
page 9 refer to 8,16,25,32,38




```
{
while counting using WR commands to read MODE
register and loading the MODE register
}
function    readreg01(channel:byte):byte;
begin
      port[$1bb] := (channel shl 6) and $c0;
      readreg01 := port[base+8+channel];
end;
```

procedure boundedtime1

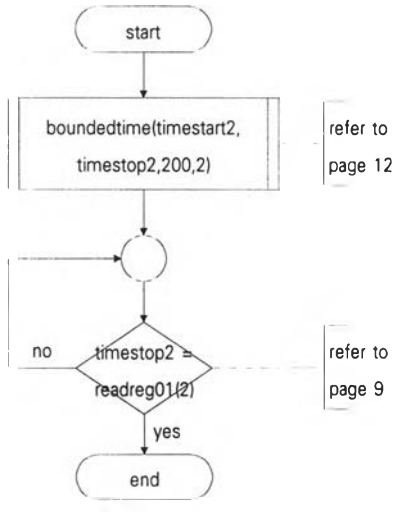
page 10 refer to 8,19,42




```
{
bounded time for counter 1
}
procedure  boundedtime1;
begin
      boundedtime(timestart1,timestop1,10,2);
      repeat
      until timestop1 = readreg01(1);
end;
```

procedure boundedtime2

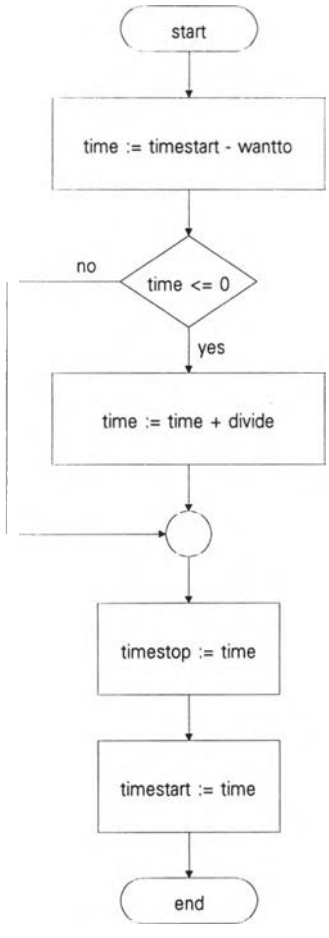
page 11 refer to 8,25,32,43




```
{
bounded time for counter 2
}
procedure  boundedtime2;
begin
      boundedtime(timestart2,timestop2,200,2);
      repeat
      until timestop2 = readreg01(2);
end;
```

procedure boundedtime(timestart,timestop,divide,wantto)

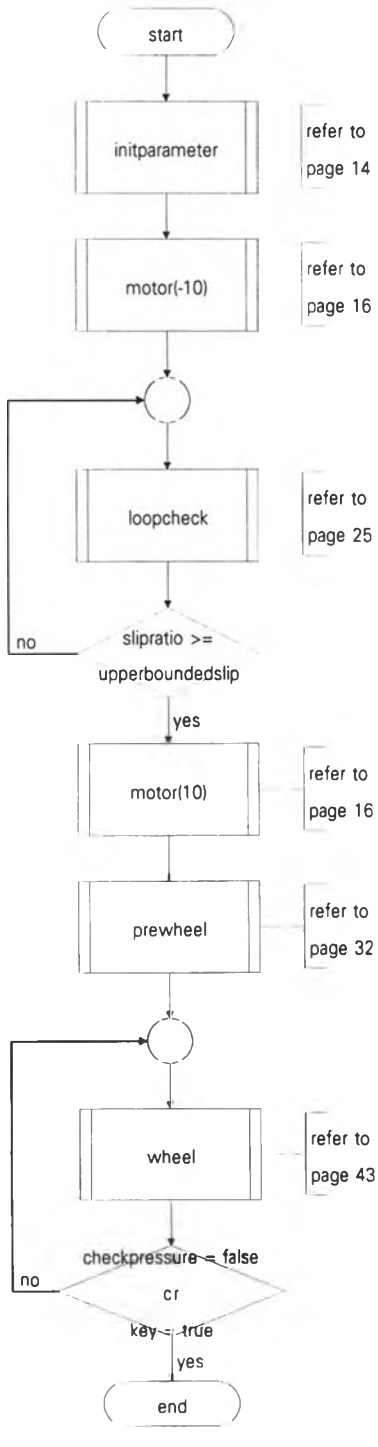
page 12 refer to 10,11




```
{
bounded time loop
}
procedure boundedtime(var timestart,timestop:byte;
                                    divide,wantto:byte);
var   time : integer;
begin
      time := timestart-wantto;
      if time <= 0 then time := time + divide;
      timestop := time;
      timestart := time;
end;
```

procedure active

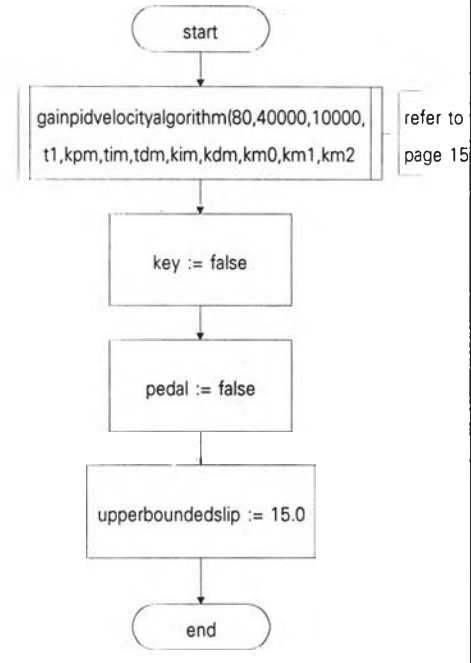
page 13 refer to 1




```
{
an active procedure is slip control when braking
}
procedure active;
begin
      initparameter;
      motor(-10.0);      {open valve}
      repeat
        loopcheck;
      until slipratio >= upperboundedslip;
      motor(10.0);
      prewheel;
      repeat
        wheel;
      until (checkpressure = false) or (key = true);
end;
```

procedure initparameter

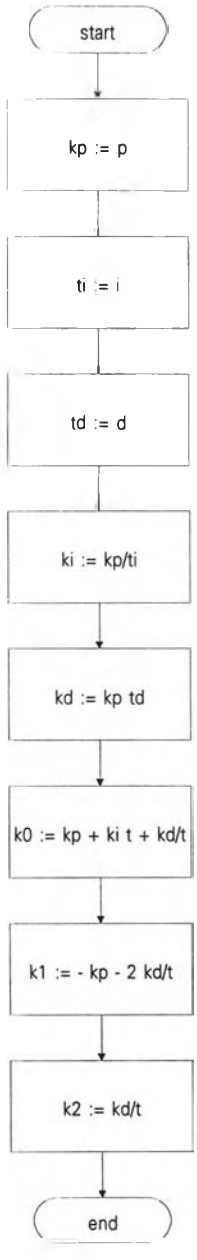
page 14 refer to 13




```
{
initial parameter
}
procedure initparameter;
begin
      gainpidvelocityalgorithm(80,40000,10000,t1,kpm,
                  tim,tdm,kim,kdm,km0,km1,km2);
      key := false;
      panel := false;
      upperboundedslip := 15.0;
end;
```

procedure gainpidvelocityalgorithm(p,i,d,t,kp,ti,td,ki,kd,k0,k1,k2)

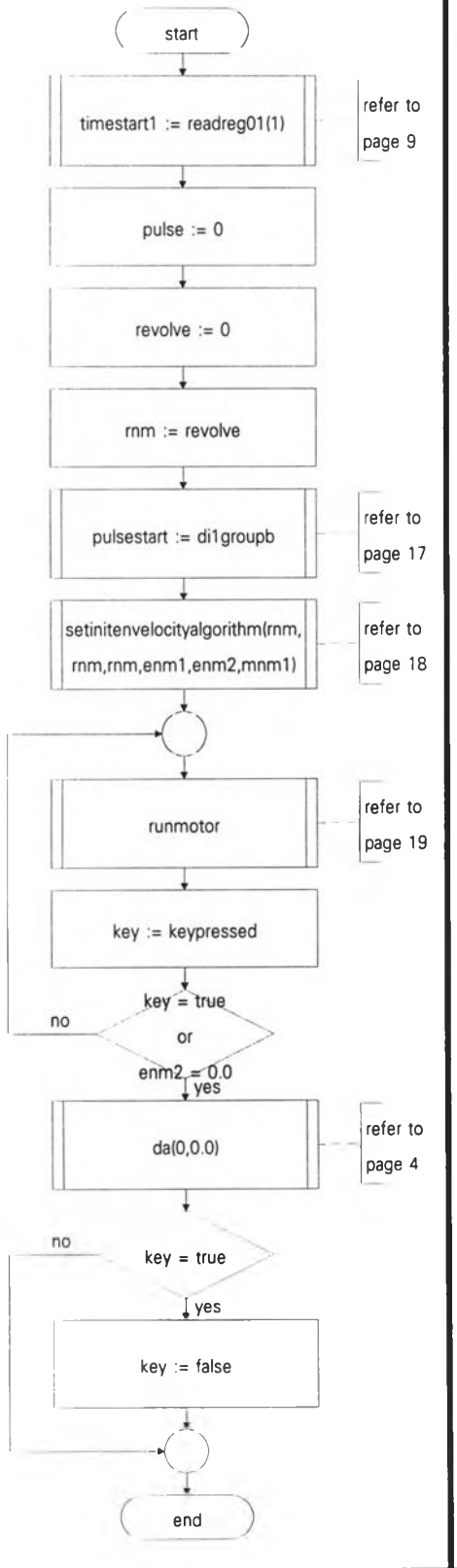
page 15 refer to 14,33,39




```
{
gain of PID control action , written by approximate PID
control and using velocity algorithm.
}
procedure  gainpidvelocityalgorithm(p,i,d:float;
            var t,kp,ti,td,ki,kd,k0,k1,k2:float);
begin
    t := t;
    kp := p;
    ti := i;
    td := d;
    ki := kp/ti;
    kd := kp*td;
    k0 := kp + ki*t + kd/t;
    k1 := -1.0*kp - 2.0*kd/t;
    k2 := kd/t;
end;
```

procedure motor(revolve)

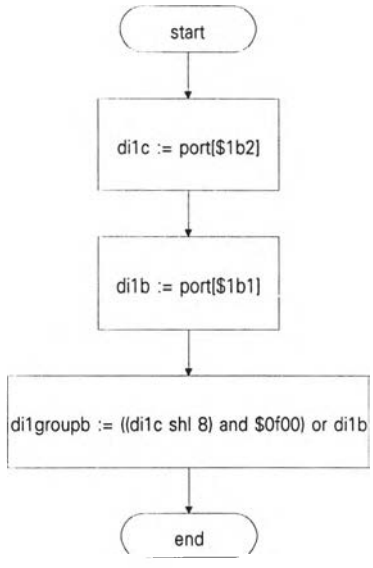
page 16 refer to 13




```
{
motor driving to open and close servo needle valve
}
procedure  motor(value:float);
begin
      timestart1 := readreg01(1);
      pulse := 0.0;
      revolve := 0.0;
      rnm := value;
      pulsestart := di1groupb;
      setinitenvelocityalgorithm(rnm,rnm,rnm,
                        enm1,enm2,mnm1);
      repeat
        runmotor;
        key := keypressed;
      until (key = true) or (enm2 = 0.0);
      da(0,0.0);
      if key = true then key := false;
end; {value < 0.0  - open   valve}
      {value > 0.0  + closed valve}
```

procedure di1groupb

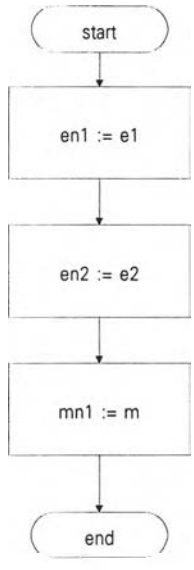
page 17 refer to 16,19




```
{
reading group 1b at port 1c and port 1b
}
function    di1groupb:word;
var   di1b, di1c : byte;
begin
      di1c := port[base+2];
      di1b := port[base+1];
      di1groupb := ((di1c shl 8) and $0f00) or di1b;
end;
```

procedure setinitenvelocityalgorithm(e1,e2,m,en1,en2,mn1)

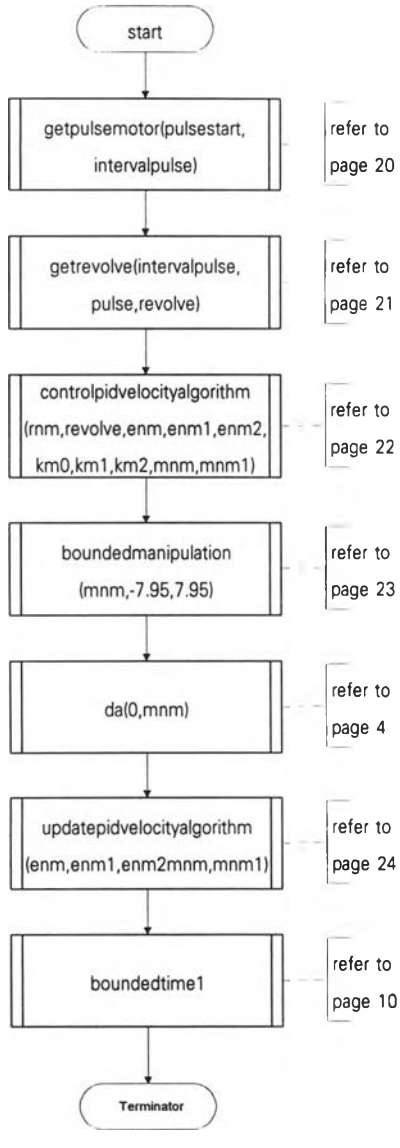
page 18 refer to 16,33,38




```
{
setting up error parameter
}
procedure  setinitenvelocityalgorithm(e1,e2,m:float;
                                      var en1,en2,mn1:float);
begin
      en1 := e1 ;
      en2 := e2;
      mn1 := m;
end;
```

procedure runmotor

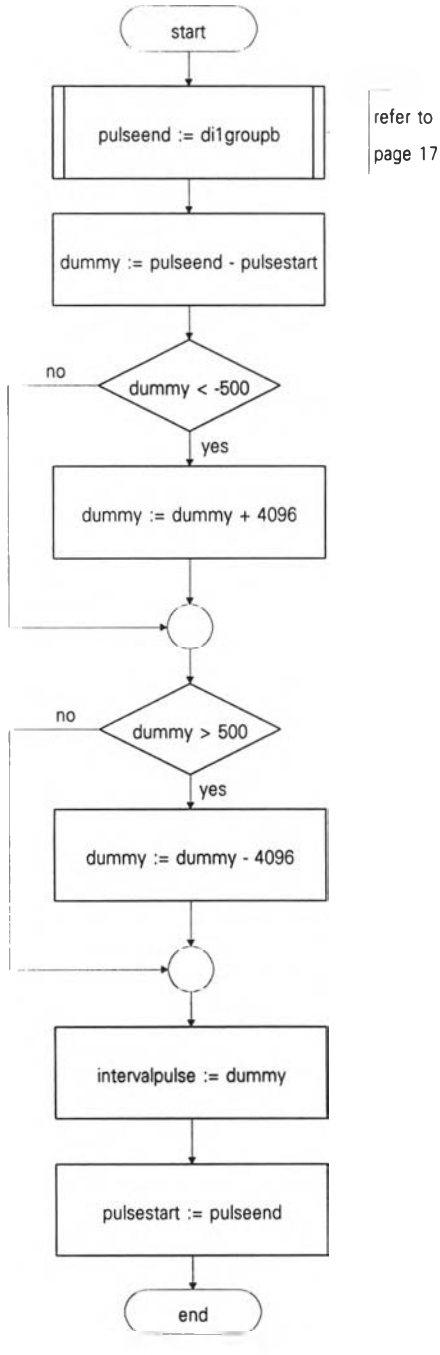
page 19 refer to 16




```
{
servo motor operation controlled by closed loop control
}
procedure  runmotor;
begin
      getpulsemotor(pulsestart,intervalpulse);
      getrevolve(intervalpulse,pulse,revolve);
      controlpidvelocityalgorithm(rnm,revolve,
      enm,enm1,enm2,km0,km1,km2,mnm,mnm1);
      boundedmanipulation(mnm,-7.95,7.95);
      da(0,mnm);
      updatepidvelocityalgorithm(enm,enm1,enm2,
                                 mnm,mnm1);
      boundedtime1;
end;
```

procedure getpulsemotor(pulsestart,intervalpulse)

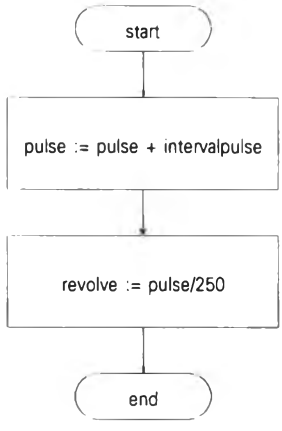
page 20 refer to 19




```
{
reading interval pulse from encoder at the end of the
servo motor
}
procedure  getpulsemotor(var pulsestart:word;
                         var intervalpulse:integer);
var    pulseend : word;
       dummy : integer;
begin
       pulseend := di1groupb;
       dummy := pulseend - pulsestart;
       if dummy < -500 then dummy := dummy + 4096;
       if dummy >  500 then dummy := dummy - 4096;
       intervalpulse := dummy;
       pulsestart := pulseend;
end;
```

procedure getrevolve(intervalpulse,pulse,revolve)

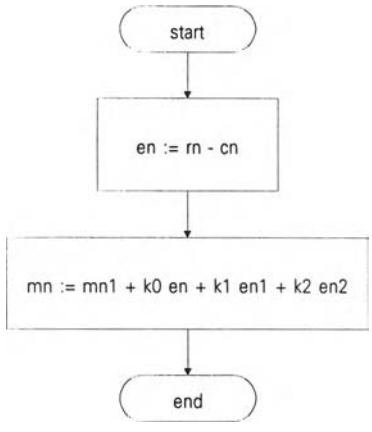
page 21 refer to 19




```
{
getting revolution from encoder at theend of the servo
motor
}
procedure  getrevolve(intervalpulse:integer;
                      var pulse,revolve:float);
begin
      pulse := pulse + intervalpulse;
      revolve := pulse/250;
end;
```

procedure controlpidvelocityalgorithm(rn,cn,en,en1,en2,k0,k1,k2,mn,mn1)

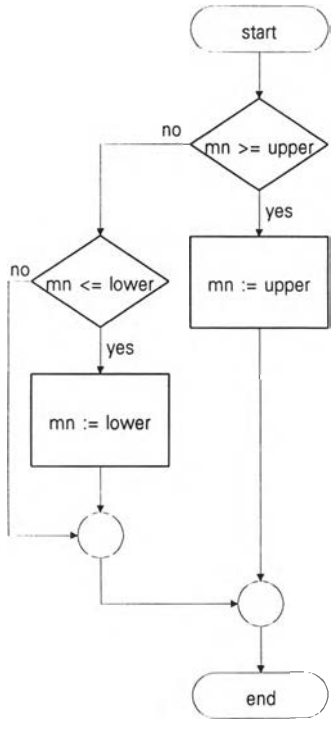
page 22 refer to 19,36,41




```
{
PID control equation
}
procedure  controlpidvelocityalgorithm(rn,cn:float;
                var en:float;en1,en2,k0,k1,k2:float;
                        var mn:float;mn1:float);
begin
      en := rn - cn;
      mn := mn1 + k0*en + k1*en1 + k2*en2;
end;
```

procedure boundedmanipulation(mn,lower,upper)

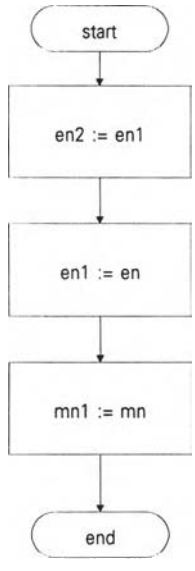
page 23 refer to 19,36,41




```
{
bounded manipulation
}
procedure boundedmanipulation(var mn:float;
                              lower,upper:float);
begin
      if mn >= upper then mn := upper else
      if mn <= lower then mn := lower;
end;
```

procedure updatepidvelocityalgorithm(en,en1,en2,mn,mn1)

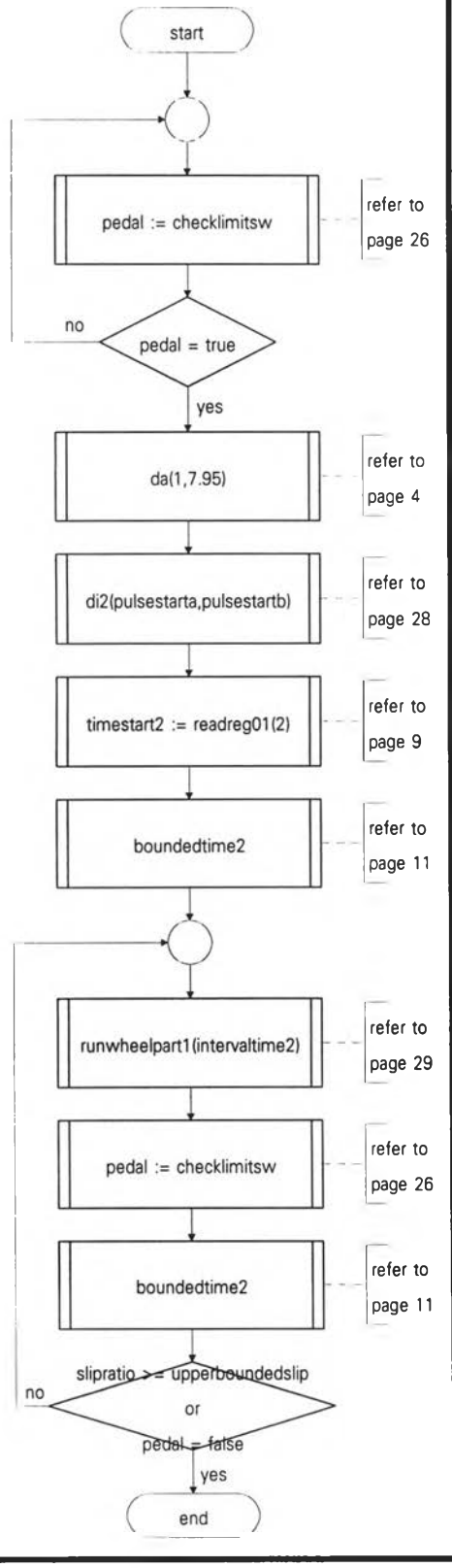
page 24 refer to 19,36,41




```
{
updated error parameter
}
procedure  updatepidvelocityalgorithm(en:float;
    var en1,en2:float;mn:float;var mn1:float);
begin
      en2 := en1;
      en1 := en;
      mn1 := mn;
end;
```

procedure loopcheck

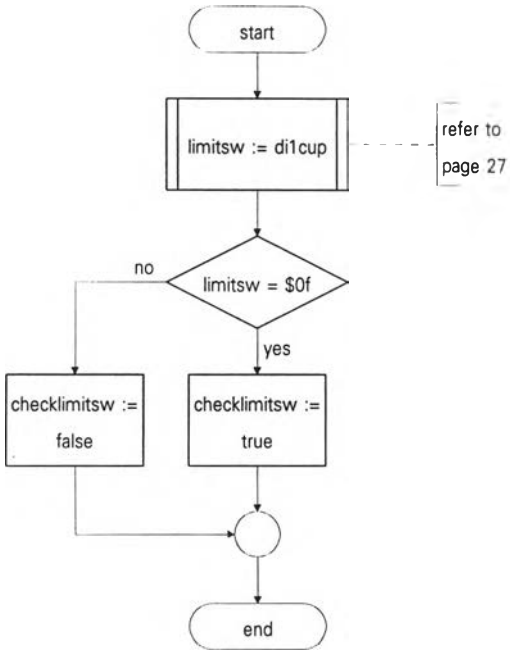
page 25 refer to 13




```
{
loop check for slip ratio when braking
}
procedure  loopcheck;
begin
      repeat
        panel := checklimitsw;
      until panel = true;
      da(1,7.95);
      di2(pulsestarta,pulsestartb);
      timestart2 := readreg01(2);
      boundedtime2;
      repeat
       runwheelpart1(intervaltime2);
        panel := checklimitsw;
         boundedtime2;
      until (slipratio >= upperboundedslip) or
                                  (panel = false);
end;
```

procedure checklimitsw

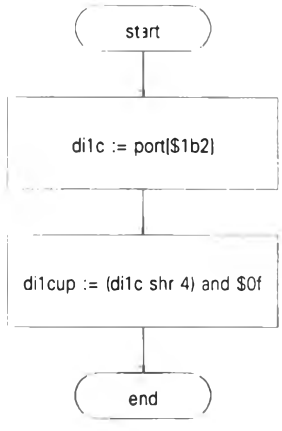
page 26 refer to 25




```
{
checking limit switch when braking
}
function    checklimitsw:boolean;
var    limitsw : byte;
begin
       limitsw := di1cup;
       if limitsw = $0f then checklimitsw := true else
       checklimitsw := false;
end;
```

procedure di1cup

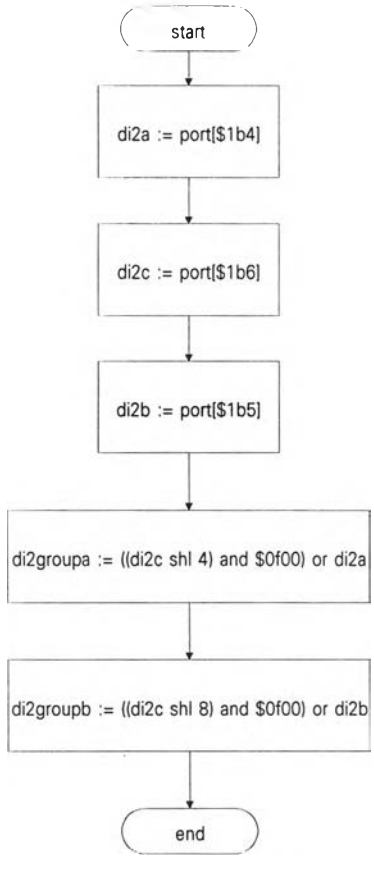
page 27 refer to 26




```
{
reading port 1c upper
}
function    di1cup:byte;
var   di1c : byte;
begin
      di1c := port[base+2];
      di1cup := (di1c shr 4) and $0f;
end;
```

procedure di2(di2groupa,di2groupb)

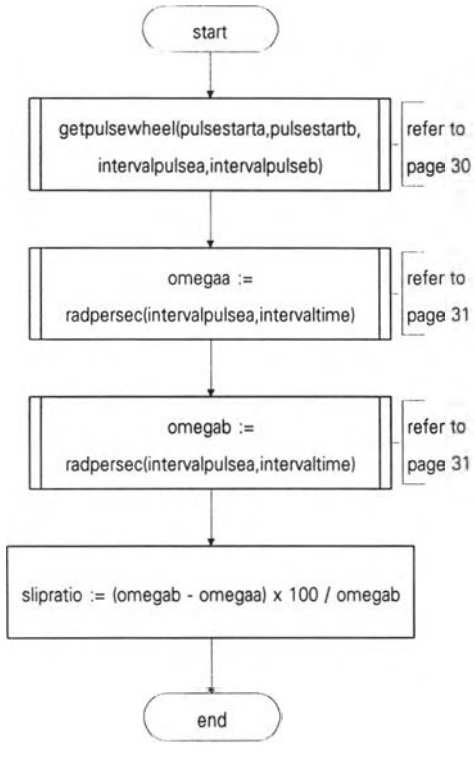
page 28 refer to 25,30




```
{
reading group 2a and group 2b at port 2a , port 2c and
port 2b
}
procedure  di2(var di2groupa,di2groupb:word);
var    di2a, di2b, di2c : byte;
begin
       di2a := port[base+4];
       di2c := port[base+6];
       di2b := port[base+5];
       di2groupa := ((di2c shl 4) and $0f00) or di2a;
       di2groupb := ((di2c shl 8) and $0f00) or di2b;
end;
```

procedure runwheelpart1(intervaltime)

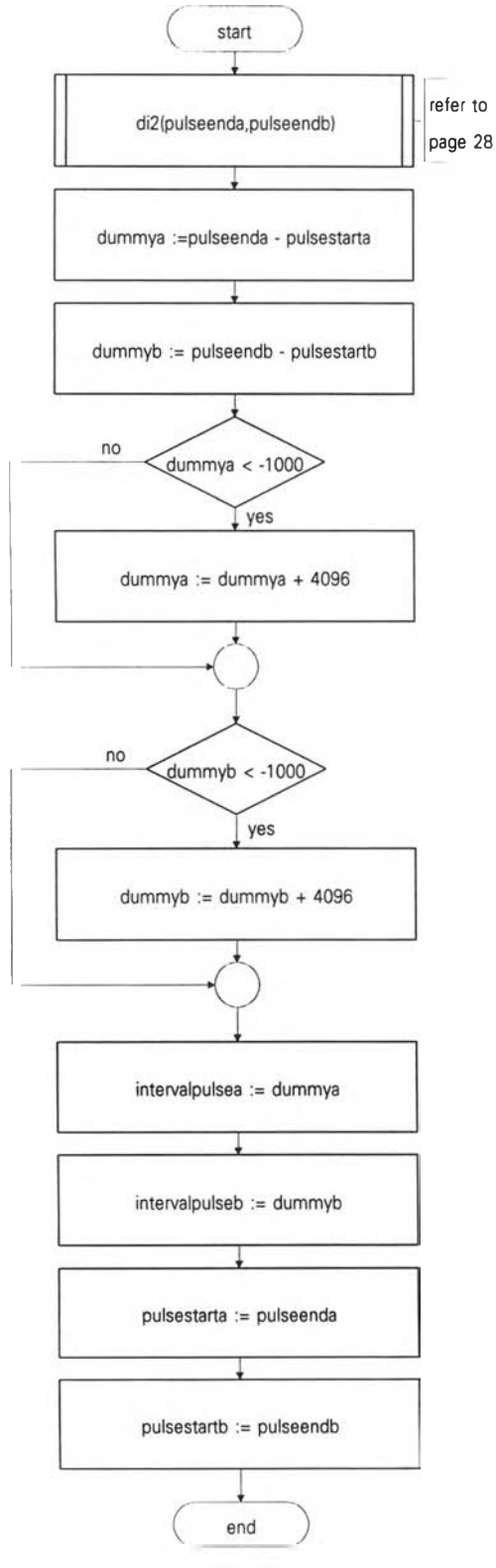
page 29 refer to 25,32,43




```
{
ABS operation in part 1
}
procedure  runwheelpart1(intervaltime:word);
begin
        getpulsewheel(pulsestarta,pulsestartb,
                intervalpulsea,intervalpulseb);
        omegaa := radpersec(intervalpulsea,intervaltime);
        omegab := radpersec(intervalpulseb,intervaltime);
        slipratio := (omegab - omegaa) * 100 / omegab;
end;
```

procedure getpulsewheel(pulsestarta,pulsestartb,intervalpulsea,intervalpulseb)

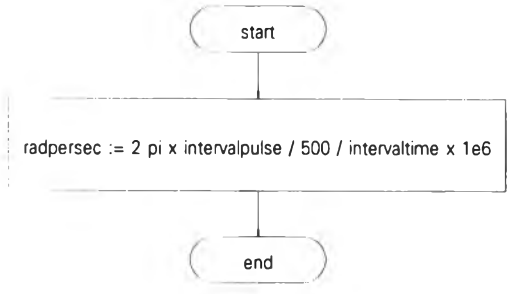
page 30 refer to 29




```
{
reading interval pulse from encoder at two wheels
}
procedure  getpulsewheel(var pulsestarta,
                                pulsestartb:word;
                               var intervalpulsea,
                           intervalpulseb:integer);
var    pulseenda,pulseendb:word;
       dummya,dummyb:integer;
begin
       di2(pulseenda,pulseendb);
       dummya := pulseenda - pulsestarta;
       dummyb := pulseendb - pulsestartb;
       if  dummya < -1000 then
             dummya := dummya + 4096;
       if  dummyb < -1000 then
             dummyb := dummyb + 4096;
       intervalpulsea := dummya;
       intervalpulseb := dummyb;
       pulsestarta := pulseenda;
       pulsestartb := pulseendb;
end;
```

procedure radpersec(intervalpulse,intervaltime)

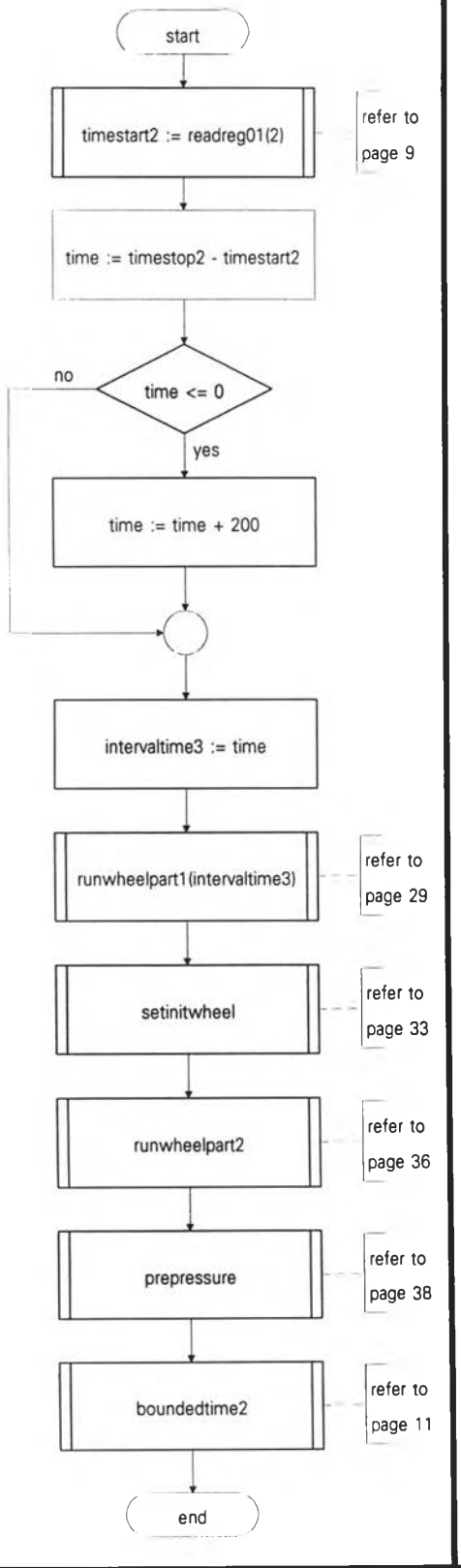
page 31 refer to 29




```
{
changing interval pulse into angular velocity
}
function    radpersec(intervalpulse:integer;
                      intervaltime:word):float;
begin
      radpersec := 2.0*pi*intervalpulse/500/
                   intervaltime*1.0e6;  {rad/sec}
end;
```

procedure prewheel

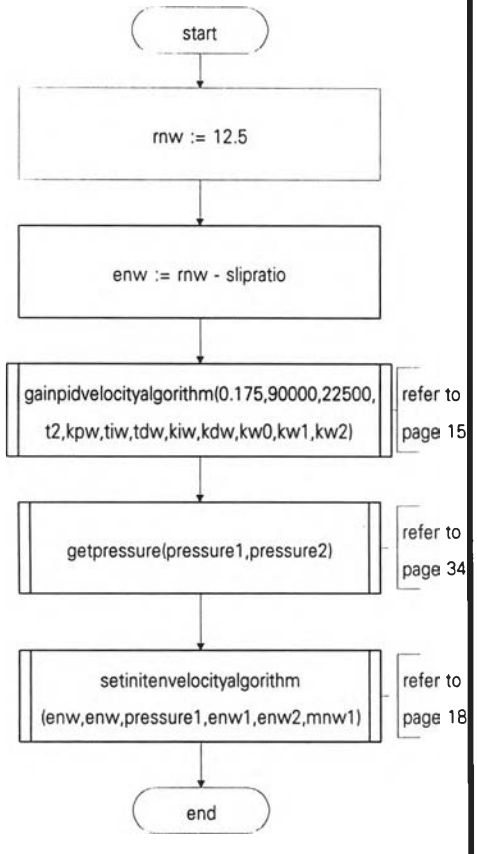
page 32 refer to 13




```
{
ABS operation after the servo needle valve is closed.
}
procedure  prewheel;
begin
      timestart2 := readreg01(2);
      time := timestop2 - timestart2;
      if time <= 0 then time := time + 200;
      intervaltime3 := time;
      runwheelpart1(intervaltime3);
      setinitwheel;
      runwheelpart2;
      prepressure;
      boundedtime2;
end;
```

procedure setinitwheel

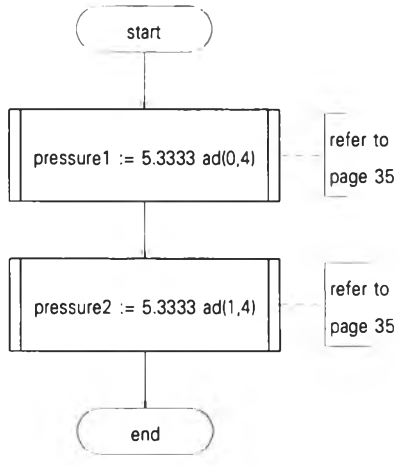
page 33 refer to 32




```
{
setting up initial wheel parameter
}
procedure  setinitwheel;
begin
      rnw := 12.5;
      enw := rnw - slipratio;
      gainpidvelocityalgorithm(0.175,90000,22500,
        t2,kpw,tiw,tdw,kiw,kdw,kw0,kw1,kw2);
      getpressure(pressure1,pressure2);
      setinitenvelocityalgorithm(enw,enw,pressure1,
                                 enw1,enw2,mnw1);
end;
```

procedure getpressure(pressure1,pressure2)

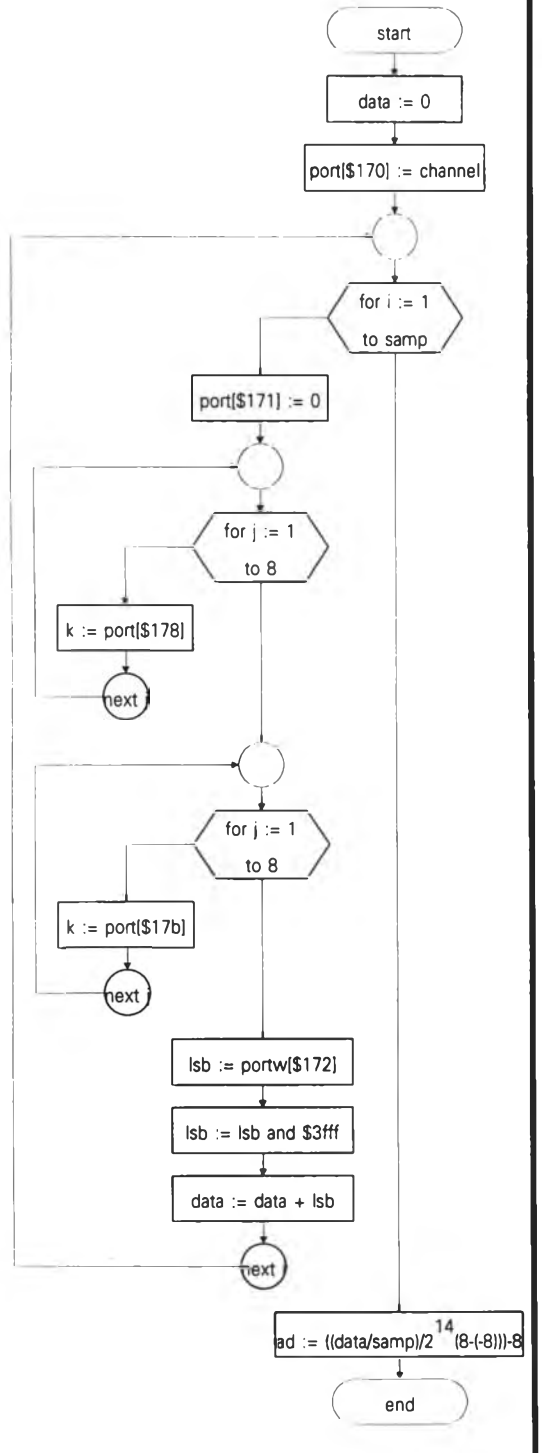
page 34 refer to 33,41




```
{
reading pressure from pressure transducer 1 and
pressure transducer 2
}
procedure  getpressure(var pressure1,pressure2:float);
begin
      pressure1 := 5.3333 * ad(0,4);
      pressure2 := 5.3333 * ad(1,4);
end;
```

procedure ad(channel,samp)

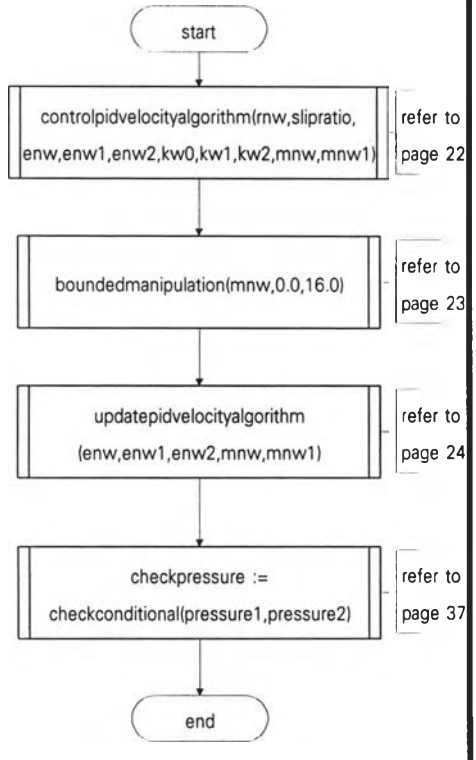
page 35 refer to 34




```
{
analog to digital conversion procedure
}
function    ad(channel,samp:byte):float;
var   data , lsb : word;
      i , j , k : byte;
begin
      data := 0;
      port[base+0] := channel;
      for i := 1 to samp do
      begin
        port[base+1] := 0;
        for j := 1 to 8 do begin k := port[base+ 8]; end;
        for j := 1 to 8 do begin k := port[base+12]; end;
        lsb := portw[base+2];
         lsb := lsb and $3fff;
        data := data + lsb;
      end;
      ad := ((data/samp)/16384.0*(8.0-(-8.0)))-8.0;
end;
```

procedure runwheelpart2

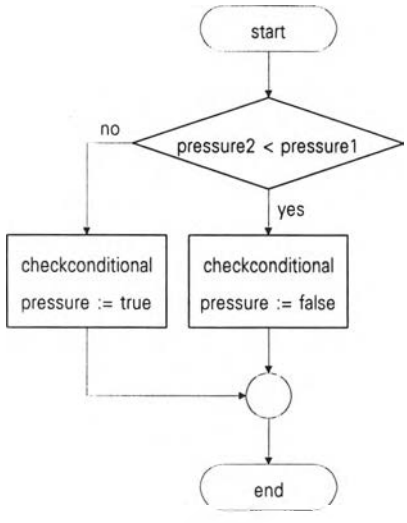
page 36 refer to 32,43




```
{
ABS operation in part 2
}
procedure  runwheelpart2;
begin
      controlpidvelocityalgorithm(rnw,slipratio,
            enw,enw1,enw2,kw0,kw1,kw2,mnw,mnw1);
      boundedmanipulation(mnw,0.0,16.0);
      updatepidvelocityalgorithm(enw,enw1,enw2,
                                    mnw,mnw1);
      checkpressure :=
          checkconditionalpressure(pressure1,pressure2);
end;
```

procedure checkconditionalpressure(pressure1,pressure2)

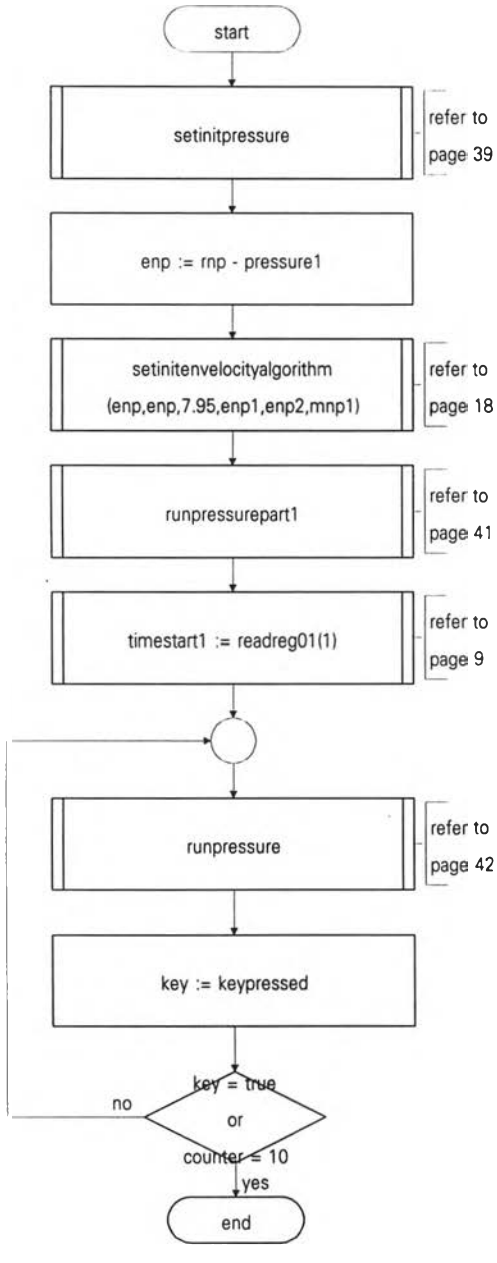
page 37 refer to 36




```
{
checking pressure condition
}
function    checkconditionalpressure(pressure1,
                        pressure2:float):boolean;
begin
      if (pressure2) < pressure1 then
            checkconditionalpressure := false else
            checkconditionalpressure := true;
end;
```

procedure prepressure

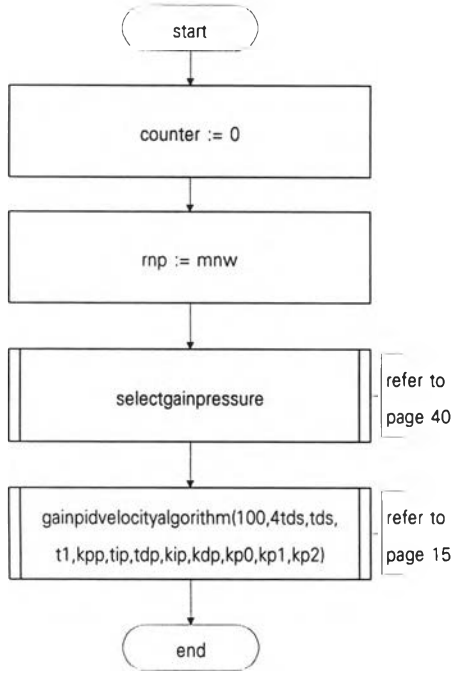
page 38 refer to 32




```
{
solenoid actuator operation after the servo needle valve
is closed.
}
procedure  prepressure;
begin
      setinitpressure;
      enp := rnp - pressure1;
      setinitenvelocityalgorithm(enp,enp,7.95,enp1,enp2,
                                                  mnp1);
      runpressurepart1;
      timestart1 := readreg01(1);
      repeat
        runpressure;
        key := keypressed;
      until (key = true) or (counter = 10);
end;
```

procedure setinitpressure

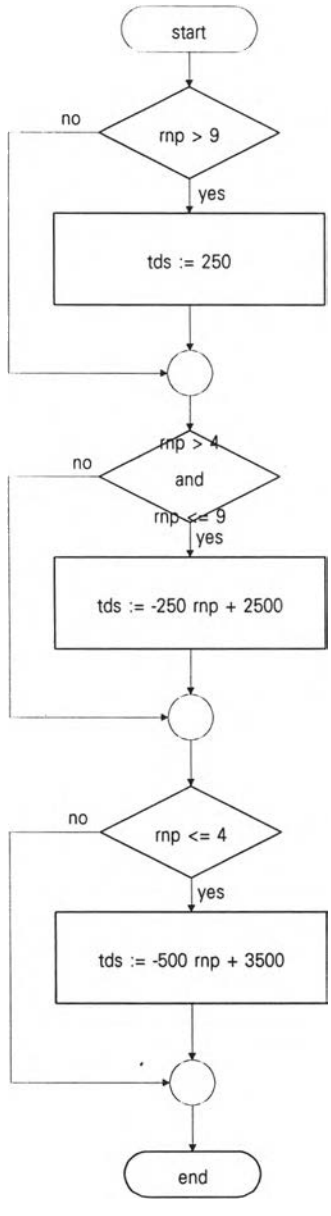
page 39 refer to 38,44




```
{
setting up initial pressure parameter
}
procedure  setinitpressure;
begin
      counter := 0;
      rnp := mnw;
      selectgainpressure;
      gainpidvelocityalgorithm(100,4*tds,tds,
        t1,kpp,tip,tdp,kip,kdp,kp0,kp1,kp2);
end;
```

procedure selectgainpressure

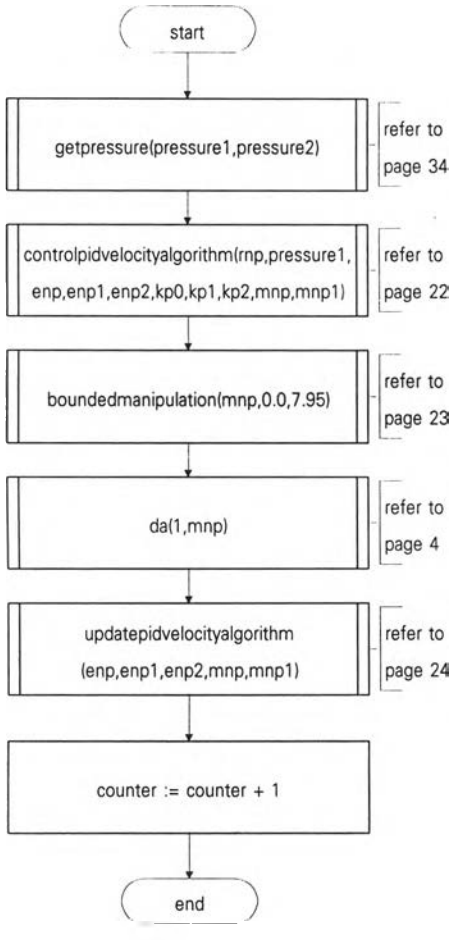
page 40 refer to 39




```
{
gain schedule for solenoid actuator
}
procedure  selectgainpressure;
begin
      if (rnp > 9) then tds := 250;
      if (rnp > 4) and (rnp <= 9) then
            tds := -250 * rnp + 2500;
      if (rnp <= 4) then
            tds := -500 * rnp + 3500;
end;
```

procedure runpressurepart1

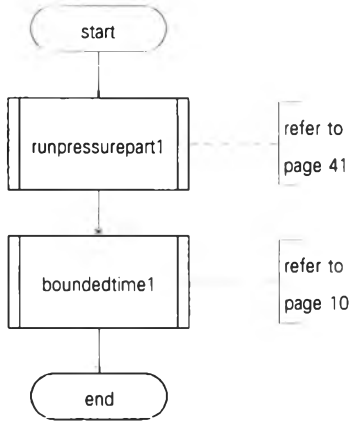
page 41 refer to 38,42




```
{
solenoid actuator operation in part 1
}
procedure  runpressurepart1;
begin
      getpressure(pressure1,pressure2);
      controlpidvelocityalgorithm(rnp,pressure1,
            enp,enp1,enp2,kp0,kp1,kp2,mnp,mnp1);
      boundedmanipulation(mnp,0.0,7.95);
      da(1,mnp);
      updatepidvelocityalgorithm(enp,enp1,enp2,
                                 mnp,mnp1);
      counter := counter + 1;
end;
```

procedure runpressure

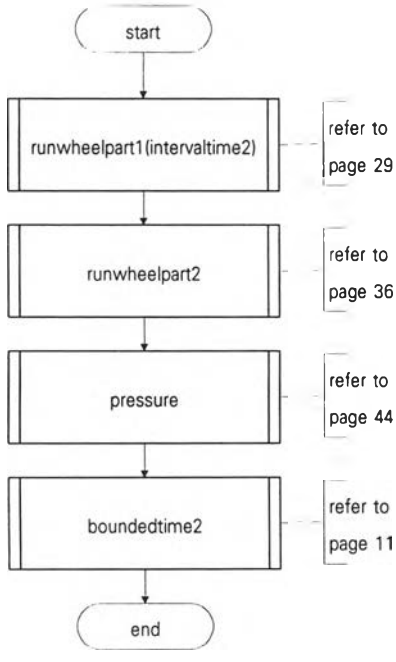
page 42 refer to 38,44




```
{
solenoid actuator operation controlled by closed loop
control
}
procedure runpressure;
begin
      runpressurepart1;
      boundedtime1;
end;
```

procedure wheel

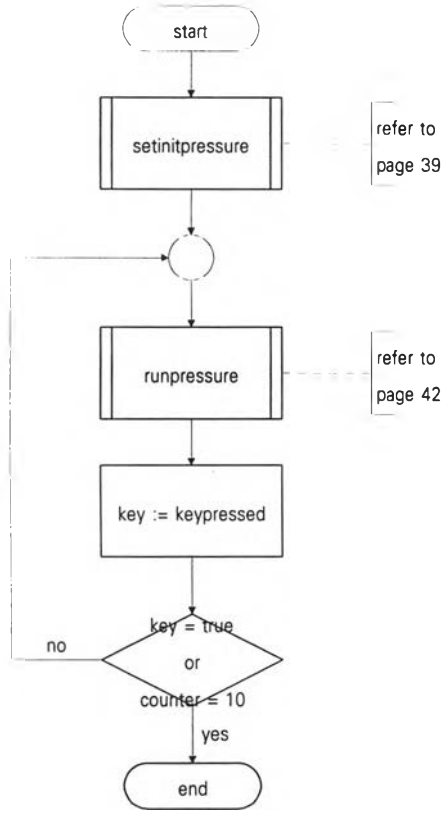
page 43 refer to 13




```
{
ABS operation controlled by closed loop control
}
procedure  wheel;
begin
        runwheelpart1(intervaltime2);
        runwheelpart2;
        pressure;
        boundedtime2;
end;
```

procedure pressure

page 44 refer to 43



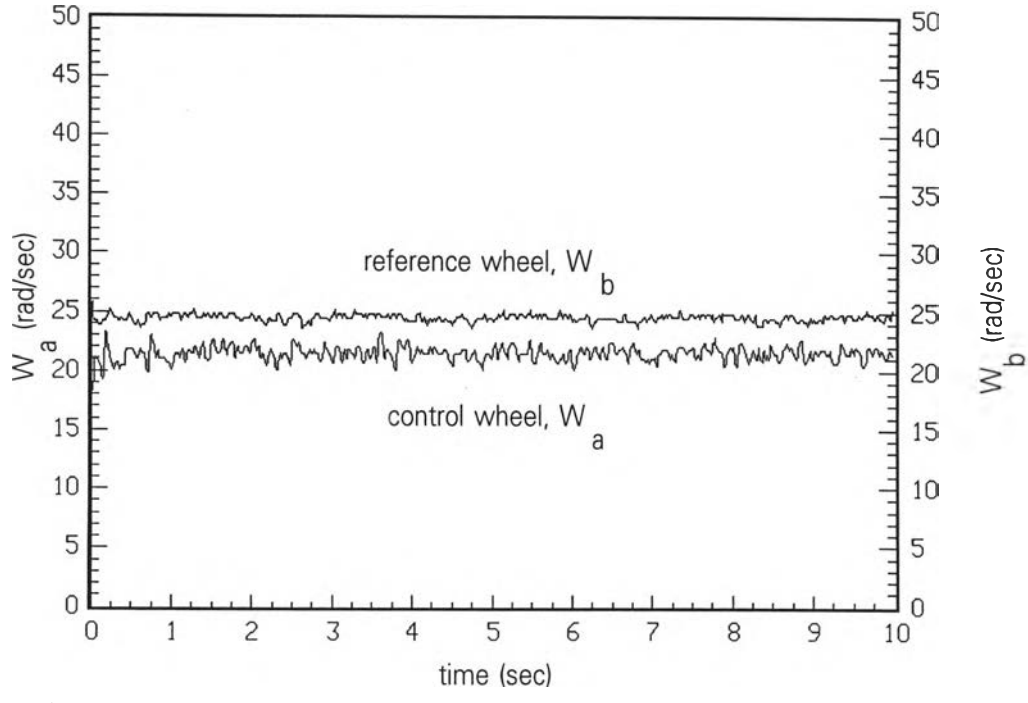

```
{
solenoid actuator operation for 10 times
}
procedure  pressure;
begin
      setinitpressure;
      repeat
         runpressure;
        key := keypressed;
      until (key = true) or (counter = 10);
end;
```

ภาคผนวก ซ.

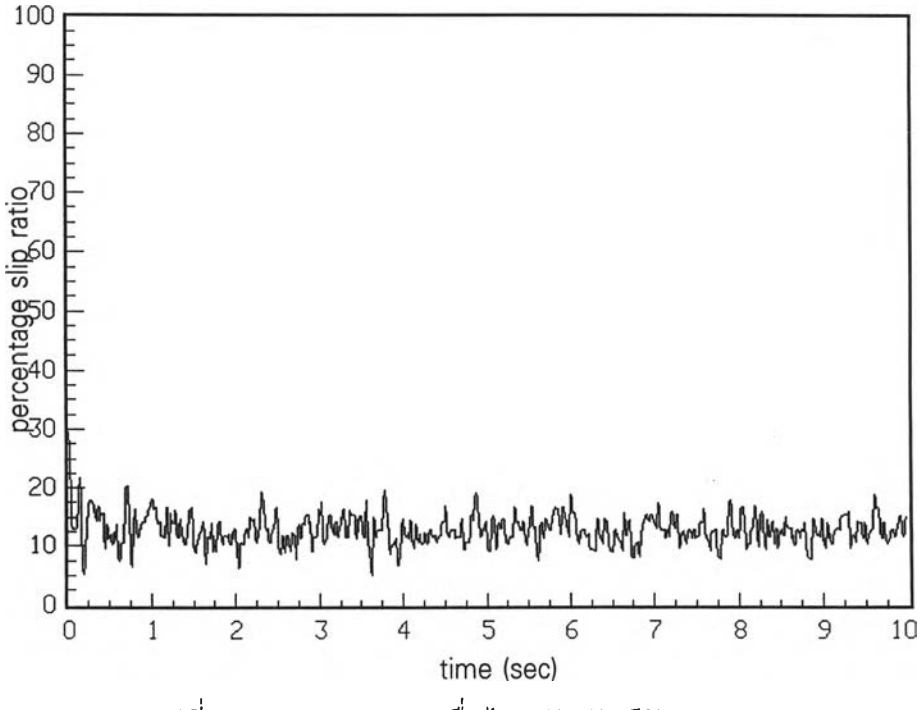
การตอบสนองของล้อทั้งสอง และอัตราการลื่นไถล

การทดสอบโดยการปรับการรับภาระน้ำหนักที่ล้อกดบนลูกกลิ้งเหล็กตั้งแต่ช่วง 300 - 600 กิโลกรัม ทำการปรับให้รับภาระน้ำหนักที่ 600 , 550 , 500 , 450 , 400 , 350 และ 300 กิโลกรัม ตามลำดับ ผลการทดสอบ รูปที่ ซ.1 , ซ.3 , ซ.5 , ซ.7 , ซ.9 และ ซ.11 จะแสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงต่อเวลา ซึ่งให้ผลการควบคุมอัตราลื่นไถลในช่วง 5 - 20% ดังแสดงในรูปที่ ซ.2 , ซ.4 , ซ.6 , ซ.8 , ซ.10 และ ซ.12

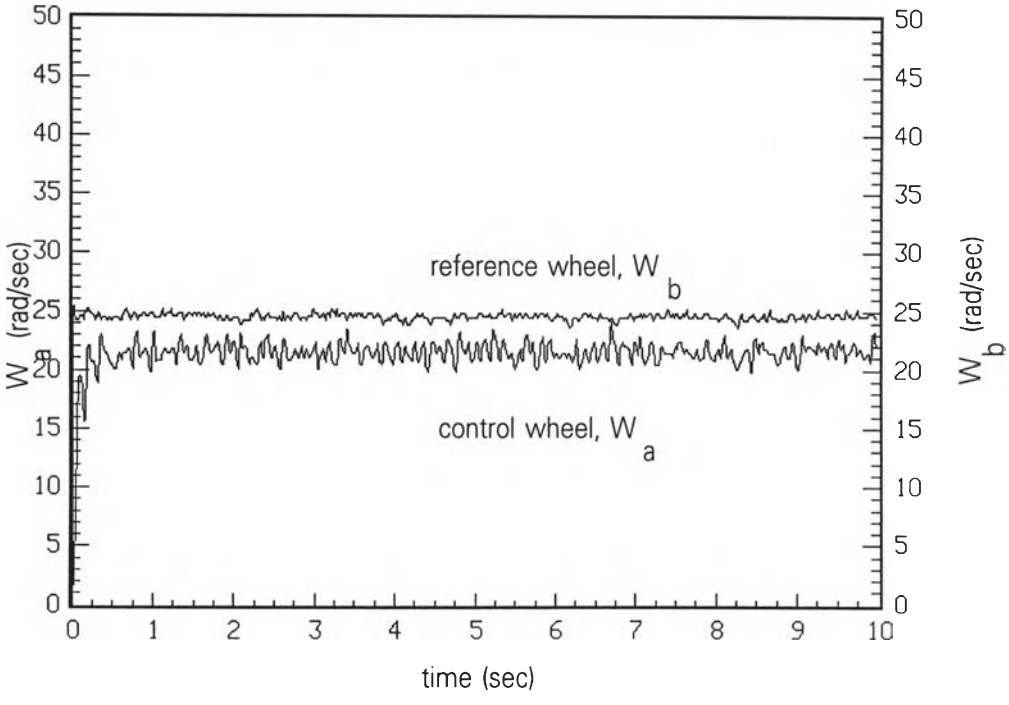
ในรูป ซ.1 , ซ.3 , ซ.5 , ซ.7 , ซ.9 และ ซ.11 เป็นการรับค่าจากเอนโคดเดอร์ ที่นำมาเป็นอุปกรณ์ตรวจจับความเร็วรอบของล้อ เป็นสัญญาณพัลซ์ และผ่านดีโคดเดอร์แปลงเป็นค่าไบนารี่ 12 บิต ค่าที่ได้จะไม่ราบเรียบ โดยดูได้จากการตอบสนองของล้ออ้างอิงที่แสดงในรูปดังกล่าว อีกทั้งการตอบสนองล้อที่ถูกควบคุมในช่วงการตอบสนองชั่วครู่จะหมุนช้ากว่าล้ออ้างอิงมากในช่วงต้น และจะหมุนเร็วขึ้น และมีการแกว่งตัว เมื่อเวลาผ่านไป การตอบสนองคงตัวความเร็วเชิงมุมของล้อที่ถูกควบคุมจะมีการแกว่งตัวมากเมื่อทดสอบรับภาระน้ำหนักน้อย และจะแกว่งตัวน้อยเมื่อทดสอบรับภาระน้ำหนักมาก และการคำนวณหาค่าอัตราการลื่นไถลซึ่งแสดงในรูป ซ.2 , ซ.4 , ซ.6 , ซ.8 , ซ.10 และ ซ.12 การตอบสนองอัตราลื่นไถลเมื่อทดสอบภาระน้ำหนักมากจะมีการแกว่งตัวน้อย ซึ่งจะภายใต้ขอบเขต 5 - 20% และการตอบสนองอัตราลื่นไถลเมื่อทดสอบรับภาระน้ำหนักน้อยจะมีการแกว่งตัวมาก ซึ่งก็ยังอยู่ในขอบเขต 5 - 20% แต่ก็จะมีบางเวลาที่การตอบสนองหลุดออกนอกขอบเขต



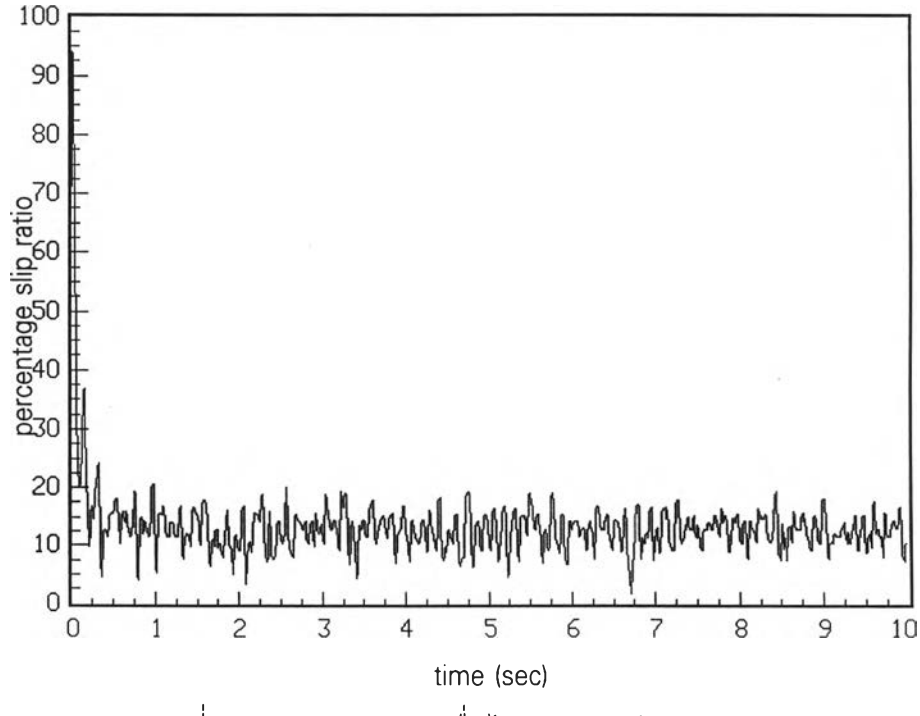
รูปที่ ซ.1 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 600 กิโลกรัม



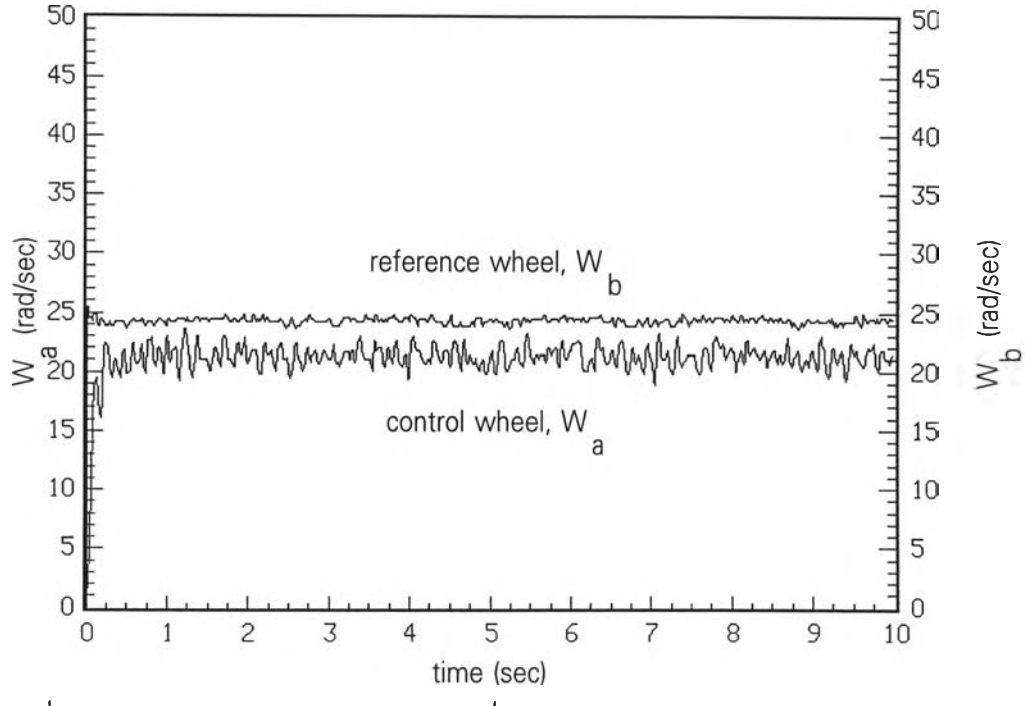
รูปที่ ซ.2 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 600 กิโลกรัม



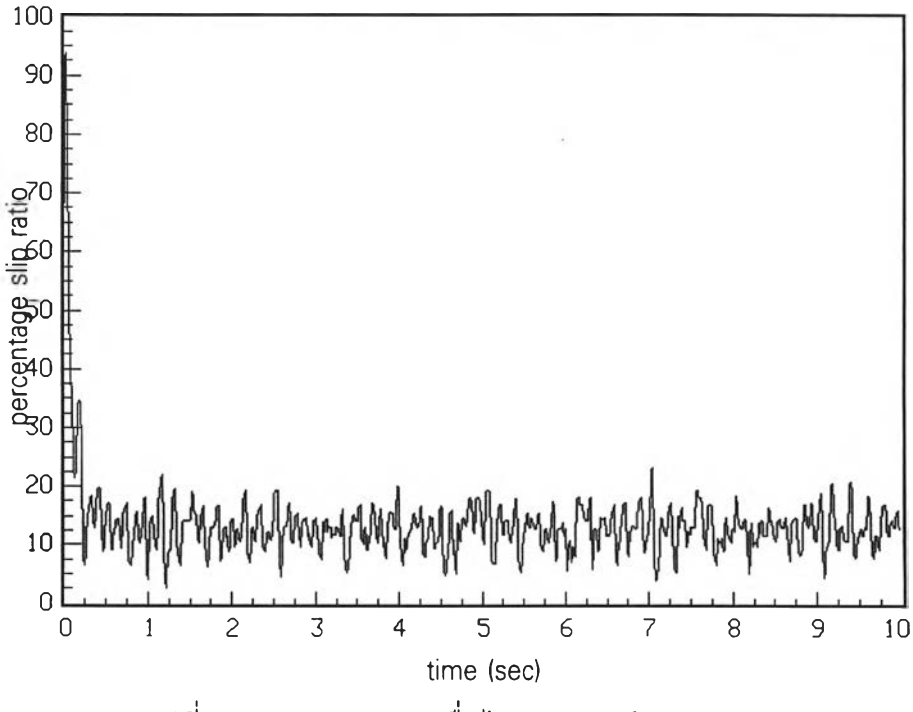
รูปที่ ซ.3 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 550 กิโลกรัม



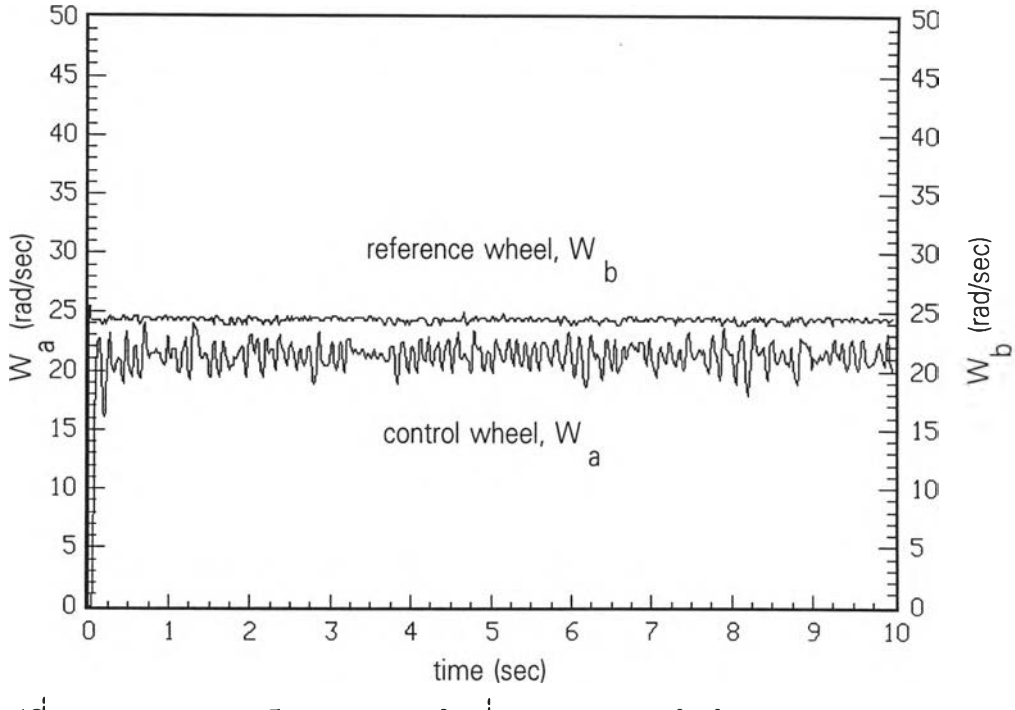
รูปที่ ซ.4 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 550 กิโลกรัม



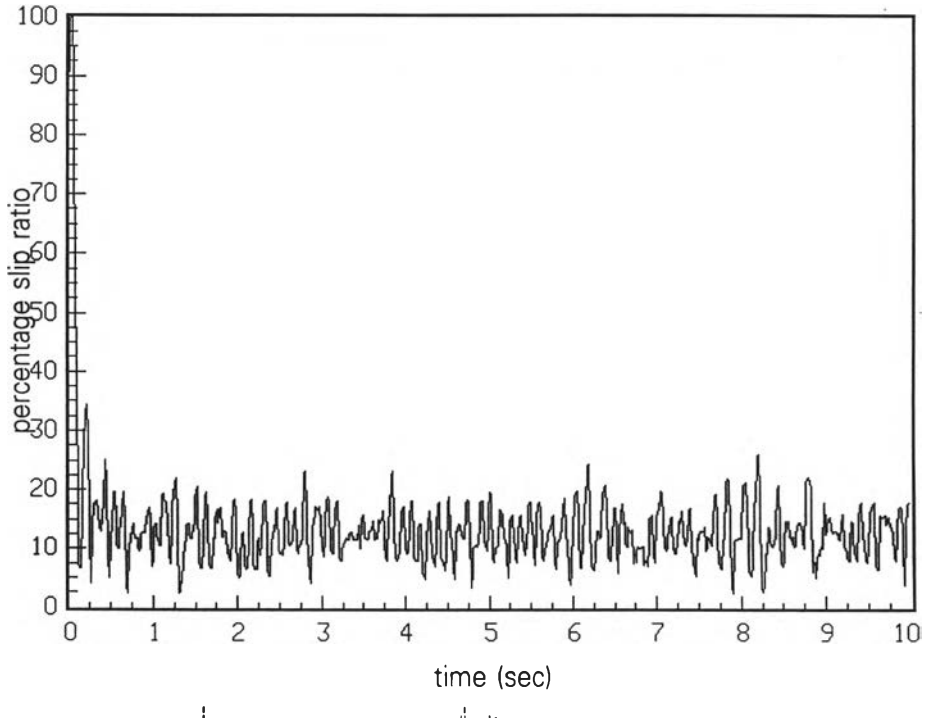
รูปที่ ซ.5 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 500 กิโลกรัม



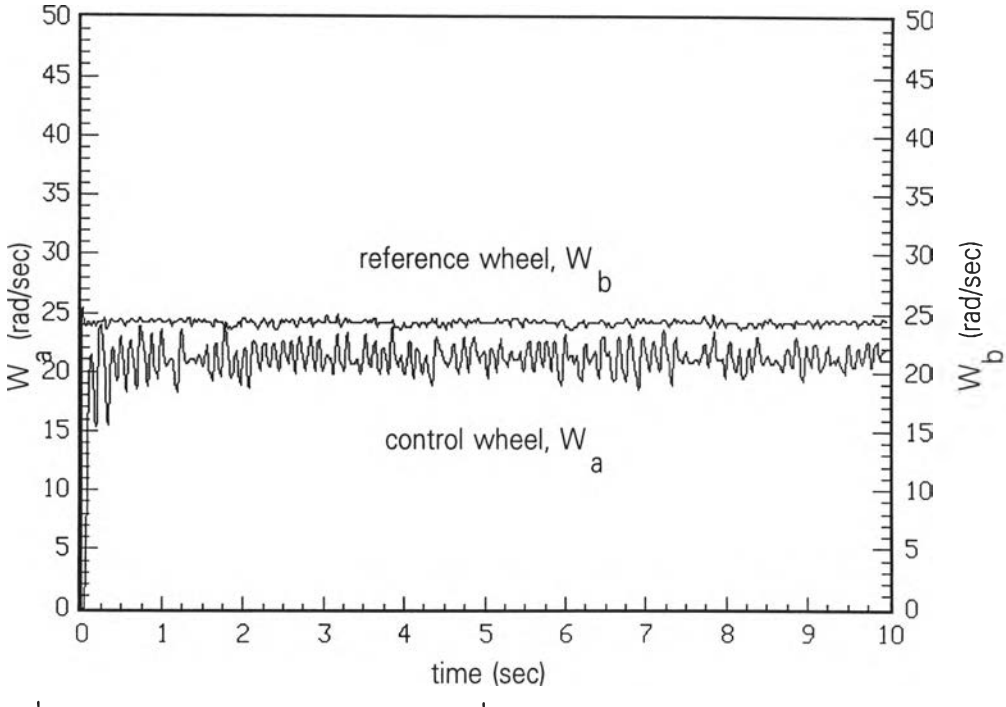
รูปที่ ซ.6 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 500 กิโลกรัม



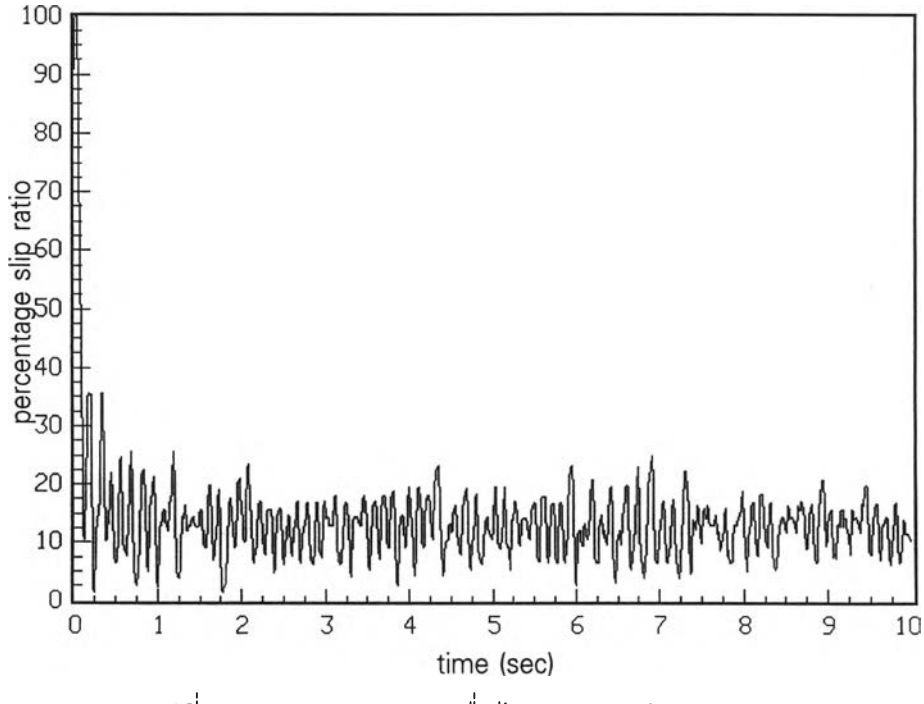
รูปที่ ซ.7 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 450 กิโลกรัม



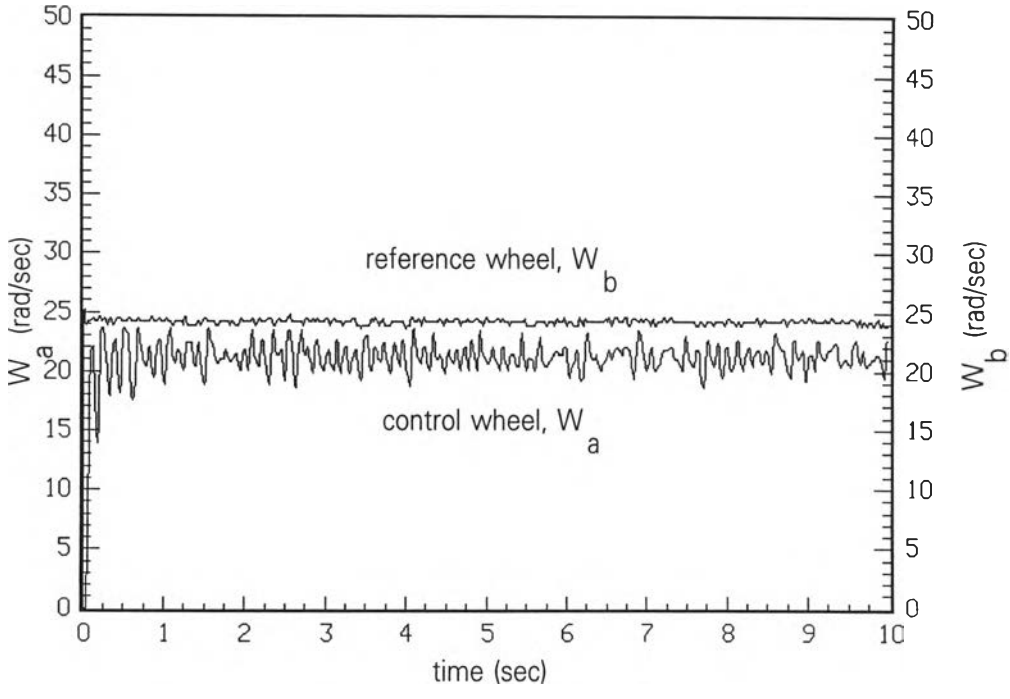
รูปที่ ซ.8 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 450 กิโลกรัม



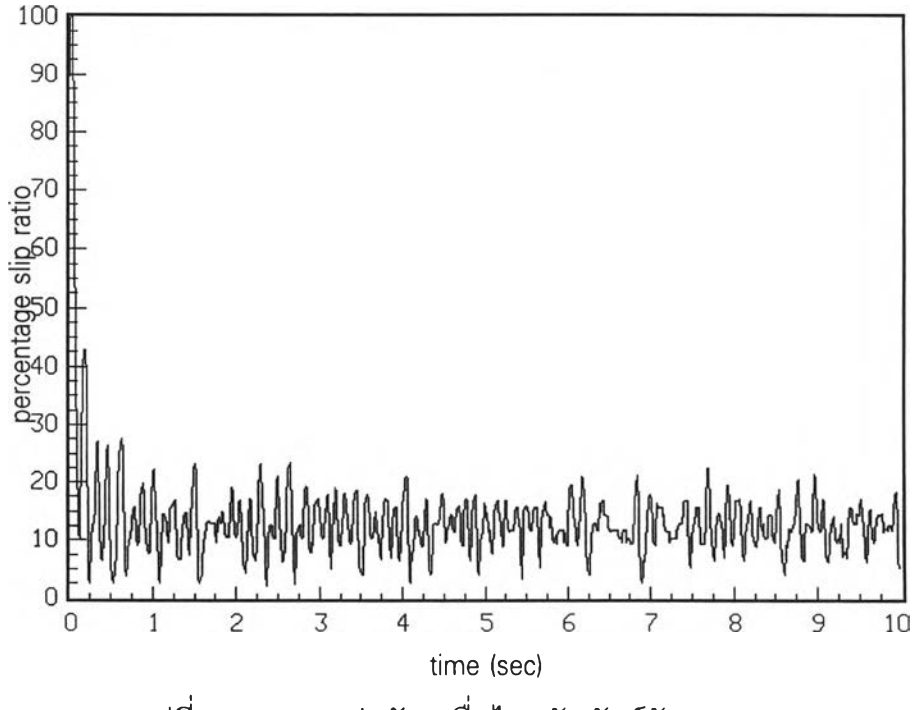
รูปที่ ซ.9 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 400 กิโลกรัม



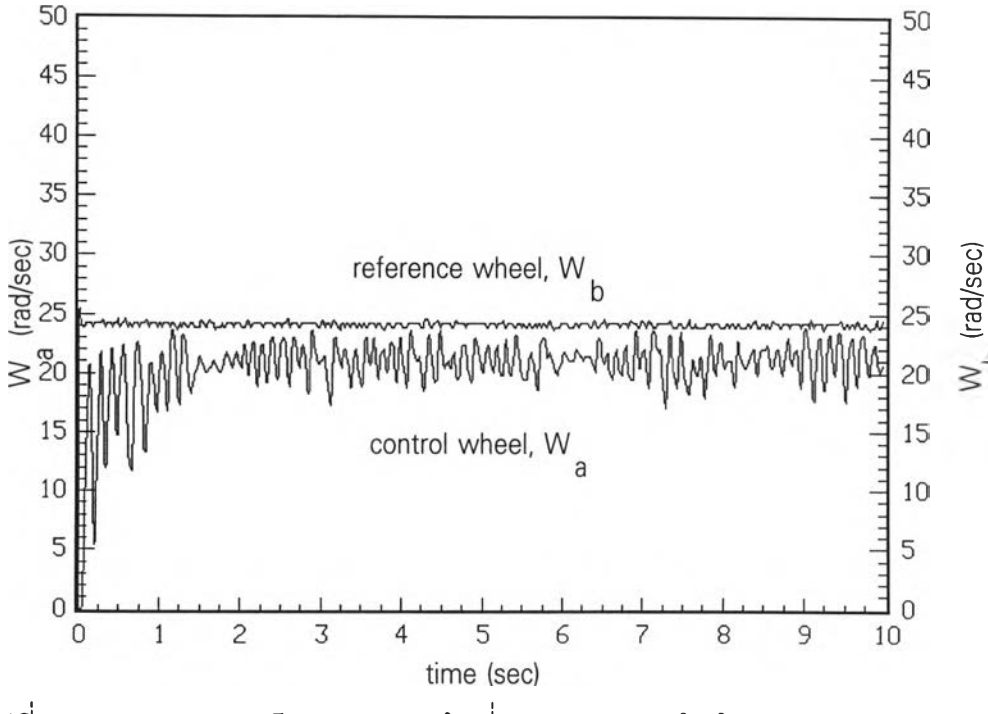
รูปที่ ซ.10 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 400 กิโลกรัม



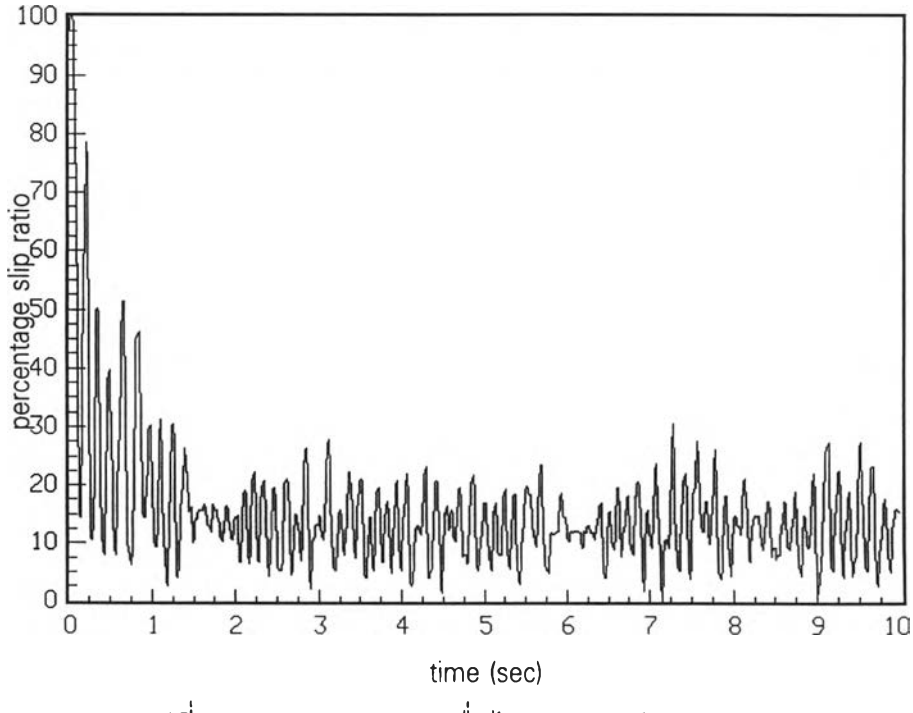
รูปที่ ซ.11 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 350 กิโลกรัม



รูปที่ ซ.12 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 350 กิโลกรัม



รูปที่ ซ.13 แสดงความเร็วเชิงมุมของล้อที่ถูกควบคุมกับล้ออ้างอิงสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 300 กิโลกรัม



รูปที่ ซ.14 แสดงค่าอัตราลื่นไถลสัมพันธ์กับเวลา
ทดสอบรับภาระน้ำหนักที่ 300 กิโลกรัม

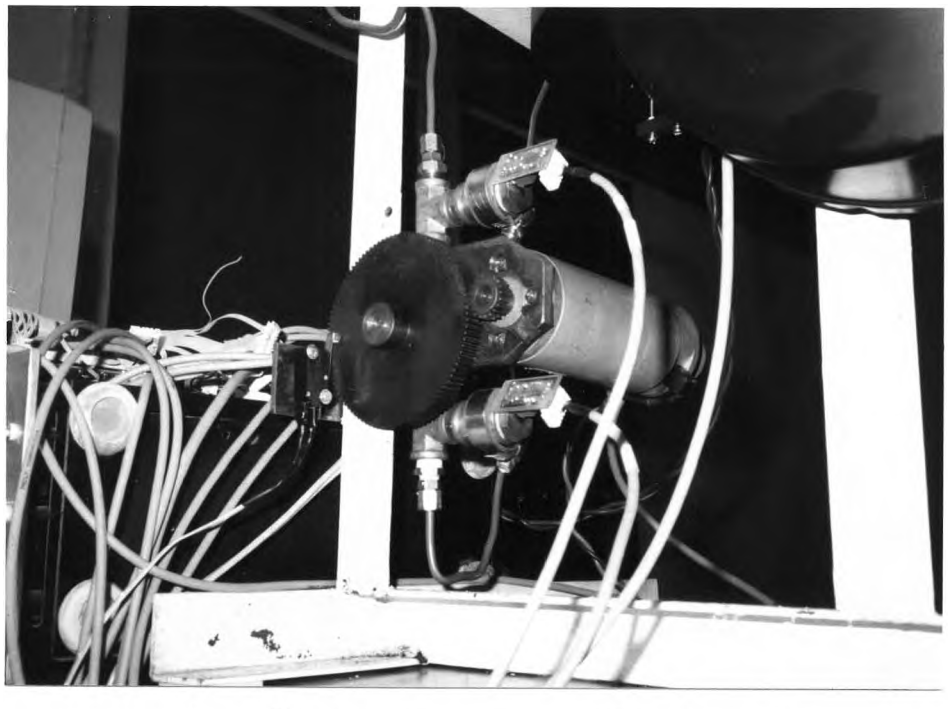
ภาคผนวก ณ.

รูปภาพส่วนประกอบของอุปกรณ์ต่าง ๆ ในงานวิจัย

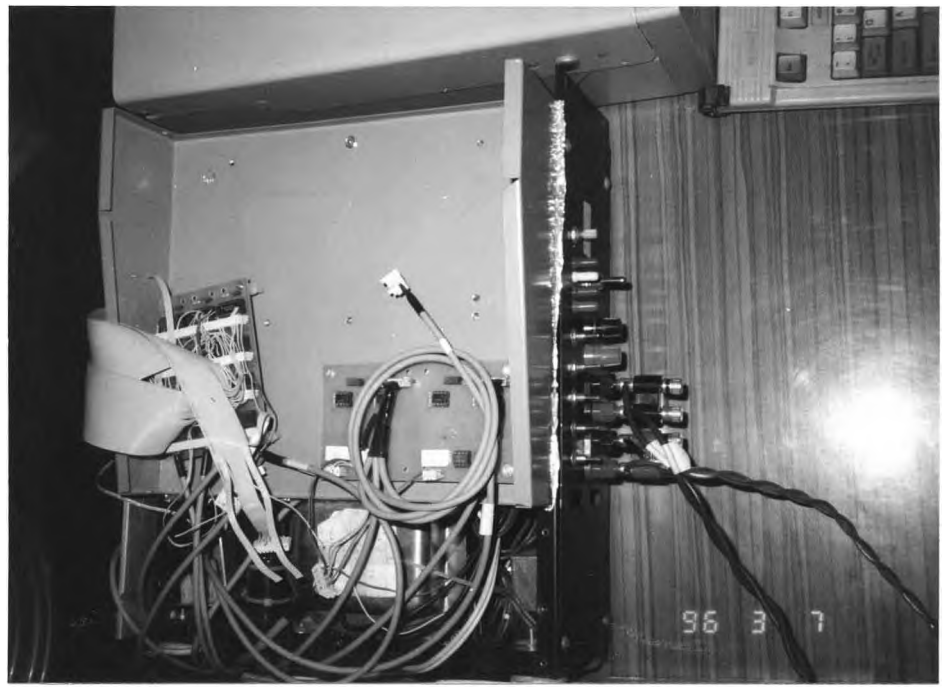
จากในรูปที่ ณ.1 ถึง ณ.5 จะแสดงภาพส่วนประกอบของอุปกรณ์ต่าง ๆ ดังนี้

รูปที่ ณ.1 ชุดแบบจำลอง ABS

รูปที่ ฌ.2 ชุดคันเหยียบห้ามล้อ และตัวกระตุ้นแบบโซลินอยด์



รูปที่ ฌ.3 ลิ้นเข็มเซอร์โว และทรานสดิวเซอร์ความดัน



รูปที่ ฌ.4 ชุดดีโคดเดอร์ , ชุดขยายสัญญาณแบบอินสตรูเมน แอมปลิไฟเออร์ และเพาเวอร์แอมปลิไฟเออร์

รูปที่ ฌ.5 การ์ดอินเตอร์เฟส และคอมพิวเตอร์

ประวัติผู้เขียน

นายสมบุญ ทุนทวีสิน เกิดเมื่อวันที่ 8 กรกฎาคม พ.ศ. 2510 ที่เขตยานนาวา กรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรี วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จากสถาบันเทคโนโลยีพระจอมเกล้าธนบุรี เมื่อปีการศึกษา 2534 และได้เข้าศึกษาในระดับปริญญาโท ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2534 และได้ทำงานที่

- สถาบันราชมงคล วิทยาเขต ฯ เทคนิคกรุงเทพ

  ตำแหน่งอาจารย์ประจำแผนกช่างยนต์ คณะช่างกล

  ตั้งแต่ปี พ.ศ. 2534 ถึง ปีพ.ศ. 2535

- มหาวิทยาลัยสยาม

  ตำแหน่งอาจารย์ประจำภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์

  ตั้งแต่ปี พ.ศ.2535 ถึงปี พ.ศ. 2538.