ปีการศึกษา 2530

ระบบควบคุมด้วยคอมพิวเตอร์

โดย

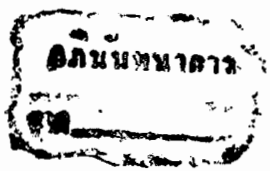
กนกชาติ ลิมป์วรรณธะ

วิศาล นิรินธนชาติ

อาจารย์ที่ปรึกษา

อาจารย์ จงกล งามวิวิทย์

อาจารย์ สุเธียร เกียรติสุนทร



ปริญญานิพนธ์ปีการศึกษา 2530

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมด้วยคอมพิวเตอร์

ผู้จัดทำ

1. กนกชาติ ลิมป์วรรณธะ
2. วิศาล นิรินธนชาติ

อาจารย์ที่ปรึกษา

(อาจารย์ จงกล งามวิวิทย์)

อาจารย์ที่ปรึกษา

(อาจารย์ สุเธียร เกียรติสุนทร)

ระบบควบคุมด้วยคอมพิวเตอร์

กนกชาติ ลิมป์วรรณธะ
วิศาล นิรินธนชาติ
อาจารย์จงกล งามวิวิทย์ อาจารย์ที่ปรึกษา
อาจารย์สุเธียร เกียรติสุนทร อาจารย์ที่ปรึกษา
ปีการศึกษา 2530

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เรียบเรียงขึ้นจากผลงานที่ได้พัฒนา คอมพิวเตอร์เป็นตัวควบคุมระบบ โดยนำสัญญาณจากเครื่องมือวัดของระบบ ซึ่งเป็นความต่างศักดาไฟฟ้า 1 ถึง 5 โวลท์ มาผ่านวงจร A/D คอนเวอทเตอร์ (ANALOG TO DIGITAL CONVERTER) เพื่อเปลี่ยนสัญญาณ ดิจิตอล (DIGITAL) ส่งให้กับคอมพิวเตอร์ จากนั้นคอมพิวเตอร์จะเป็นตัวคำนวน หาค่าที่เหมาะสมที่จะส่งออกมาควบคุมระบบ โดยใช้หลักการประมาณค่าของ อนาลอค PID ชนิด นอนอินเตอร์แอกติงในอุดมคติ (IDEAL NONINTERACTING PID CONTROLLER ALGORITHM) จากนั้นค่านี้จะถูกเปลี่ยนเป็นปริมาณกระแสไฟฟ้า โดยผ่านวงจร D/A คอนเวอทเตอร์ (DIGITAL TO ANALOG CONVERTER) และ V/I คอนเวอทเตอร์ (VOLTAGE TO CURRENT CONVERTER) เพื่อส่งไปควบคุมระบบ

โปรแกรมใช้งานเขียนด้วยภาษาซี (TURBO C) ทำงานจอภาพแบบกราฟิก (GRAPHIC MODE) สามารถทำการควบคุมได้หลายรูปแบบคือ ควบคุมแบบลูปเดียว (SINGLE LOOP CONTROL) ควบบคุมแบบต่อเนื่อง (CASCADE CONTROL) และควบคุมอัตราส่วน (RATIO CONTROL) ซึ่งผู้ควบคุมสามารถจะกำหนดพารามิเตอร์ต่าง ๆ ได้โดยตรงจากโปรแกรม

COMPUTER CONTROL SYSTEM

Kanokchat Limwannata

Wisan Nirinthanachat

Jongkol Ngamwiwit advisor

Suthean Kiatsunthon advisor

1987

Abstract

This project is a computer application on control system. The standard signal from system (1-5 V) is changed to be digital signal by A/D converter for sending to the computer. Computer use this data to calculate the control signal by using the Ideal Noninteracting PID algorithm and send it to the D/A converter and the V/I converter to change to standard signal (4-20 mA) to sent to control system.

Program is written on Turbo C and running on IBM PC/XT color graphic mode (CGA). User can use four type of control, i.e. Single Loop Control, Cascade Control, Ratio Control and manual, and set all system parameter direct to computer

สารบัญ

บทที่ 1
บทนำและทฤษฎี

ในปัจจุบัน มีการใช้งานตัวควบคุมแบบดิจิตอล (DIGITAL CONTROLLER) ในการควบคุมระบบเพิ่มขึ้นอย่างรวดเร็ว เนื่องจากการควบคุมระบบด้วยระบบดิจิตอล (DIGITAL SYSTEM) สามารถทำให้ระบบเข้าถึงจุดที่ดีที่สุดได้ (OPTIMAL PERFORMANCE) คือทำให้ได้ผลผลิตสูงสุดในค่าใช้จ่ายที่ต่ำที่สุด นอกจากนี้การพัฒนาของหน่วยประมวลผล (MICROPROCESSOR) ในปัจจุบันพัฒนาขึ้นและมีราคาถูกลงมาก ดังนั้นจึงมีการนำไปใช้งานในหลายรูปแบบ ตั้งแต่ระบบหุ่นยนต์ไปจนถึงอุปกรณ์เครื่องใช้ต่างๆ ภายในบ้าน

ในงานของวิศวกรรมระบบควบคุม ปรกติจะใช้งานคอมพิวเตอร์ในสองรูปแบบคือ ใช้ในการวิเคราะห์และจำลองระบบควบคุมที่ซับซ้อน และใช้เป็นตัวควบคุม(CONTROLLER) ในระบบ ซึ่งในปริญญานิพนธ์นี้ได้สร้างตัวควบคุม โดยใช้เครื่องคอมพิวเตอร์ TAVON PC-16 ซึ่งเป็นเครื่องซึ่งมีลักษณะเหมือนกับเครื่อง IBM (IBM PC COMPATIBLE) การทดสอบ ได้ทำการทดลองควบคุมระดับน้ำ (LEVEL CONTROL) และควบคุมอัตราการไหลของน้ำ (FLOW CONTROL) ในลักษณะต่างๆ

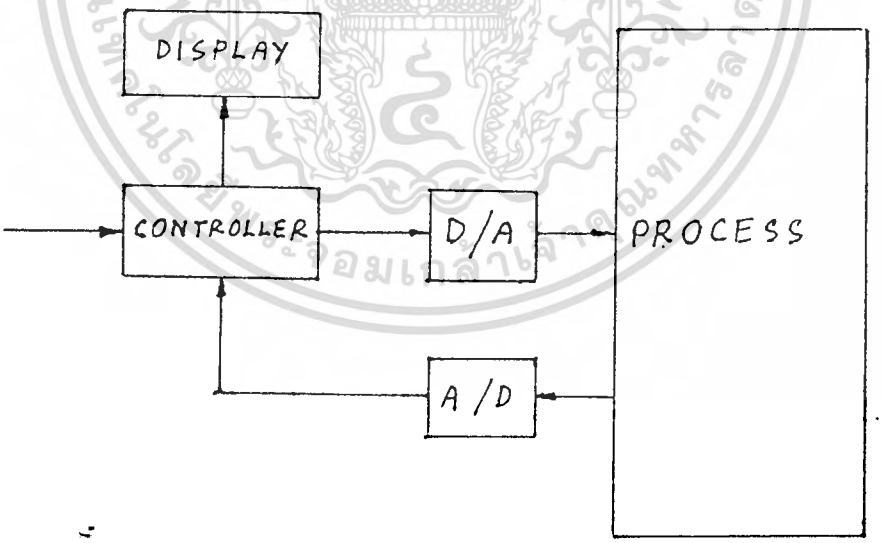
1.1 การควบคุมแบบอนาลอคและดิจิตอล

การควบคุมในลักษณะที่ใช้คอมพิวเตอร์ต่อเข้ากับ ลูปควบคุม(CONTROL LOOP) เพื่อประมวลผลสัญญาณ เรียกว่า การควบคุมแบบดิจิตอลโดยตรง (DIRECT DIGITAL CONTROL) เริ่มมีใช้มาตั้งแต่ทศวรรษที่ 1960 ในอุตสาหกรรมขนาดใหญ่ แต่ในปัจจุบันมีการใช้งานทั่วๆไปกันอย่างกว้างขวาง

การควบคุมแบบดิจิตอลนั้น เป็นการทำงานเกี่ยวกับตัวเลข การคำนวนต่างๆ จึงตรงไปตรงมาทำให้ทำการควบคุมที่ต้องใช้สมการยุ่งยากได้ง่าย การเขียนและแก้ไข

โปรแกรมก็สามารถทำได้อย่างสมบูรณ์ นอกจากนี้ ในแบบดิจิตอลจะมีปัญหาสัญญาณรบกวนน้อยกว่าในแบบอนาลอกอีกด้วย จึงมักจะใช้แก้ปัญหาจุดที่ดีที่สุดในการควบคุมของโรงงานอุตสาหกรรม และเนื่องจากการควบคุมแบบดิจิตอลใช้งานได้หลายอย่าง จึงใช้แก้ปัญหาเกี่ยวกับการควบคุมที่ไม่ลิเนียร์ (NONLINEAR) มากกว่าแบบอนาลอก

การควบคุมแบบดิจิตอล ถ้าระบบเป็นระบบอนาลอก จะต้องทำการแปลงสัญญาณอนาลอกที่ได้ให้เป็นค่าตัวเลขก่อน แล้วจึงนำค่าตัวเลขนี้มาคำนวน หลังจากนั้นจึงจะแปลงค่าตัวเลขที่คำนวนได้กลับเป็นสัญญาณอนาลอกอีกครั้ง ดังรูปที่1.1 ซึ่งจะเกิดการผิดพลาดขึ้นระหว่างการแปลง เนื่องจากการแปลง จะต้องประมาณค่าสัญญานที่ต่อเนื่อง เป็นตัวเลขที่มีค่าใกล้เคียง และนอกจากนี้ยังต้องใช้เวลาในการคำนวนอีกด้วย ดังนั้นสัญญาณควบคุมที่ได้จึงช้ากว่าแบบอนาลอก ซึ่งจะเกิดปัญหาเมื่อนำตัวควบคุมนี้ไปควบคุมระบบซึ่งมีอัตราการเปลี่ยนแปลงรวดเร็ว เมื่อเทียบกับ อัตราการอ่านข้อมูลของตัวควบคุมเอง เนื่องจากข้อมูลที่ได้คลาดเคลื่อนไปมาก ทำให้คำนวนหาขนาดของสัญญานควบคุมผิดพลาดไป



รูปที่ 1.1

ในปริญญานิพนธ์นี้ ได้จัดสร้างตัวควบคุมแบบดิจิตอล PID (DIGITAL PID CONTROLLER) ซึ่งมีลักษณะการทำงานคล้ายกับตัวควบคุมแบบอนาลอค PID (ANALOG PID CONTROLLER) ซึ่งใช้กันในงานควบคุมทั่วๆไป มีสมการในการคำนวนหาค่าขนาดของสัญญาณควบคุม ดังนี้

$$m_{(t)} = K\left[e_{(t)} + [1/T_i]\int_0^t e_{(t)}dt + T_d * de_{(t)}/dt\right]$$

$m_{(t)}$ = ขนาดของสัญญาณควบคุมที่คำนวนได้

K = อัตราขยาย

$e_{(t)}$ = ขนาดของสัญญาณที่เข้ามายังตัวควบคุม

T_i = ค่าคงที่ (INTEGRAL TIME CONSTANT)

T_d = ค่าคงที่ (DERIVATIVE TIME CONSTANT)

ซึ่งสมการแบบอนาลอคนี้ จะเปลี่ยนเป็นสมการแบบดิจิตอล เพื่อใช้ในการคำนวน ของตัวควบคุมแบบดิจิตอล PID ได้ดังรูปที่ 1.2 จะเห็นได้ว่าอัตราขยายของแบบดิจิตอล PID นี้มีค่าน้อยกว่าแบบอนาลอค PID อยู่ Ki/2 ดังนั้น พารามิเตอร์ที่เหมาะสม ของตัวควบคุมทั้งสองจะไม่เท่ากัน

ในระบบตัวควบคุมแบบดิจิตอลนั้น จะต้องอ่านข้อมูลเป็นช่วงๆ เพื่อเข้ามาคำนวนหาสัญญาณควบคุมที่เหมาะสม จึงต้องใช้เวลาระยะหนึ่งในการปฏิบัติการ ทำให้ข้อมูลที่ได้มาเป็นข้อมูลโดยประมาณเท่านั้น ซึ่งความคลาดเคลื่อนจะขึ้นอยู่กับการเปลี่ยนแปลงของข้อมูล ถ้าข้อมูลมีการเปลี่ยนแปลงมากในระหว่างการอ่านข้อมูล ก็จะเกิดความคลาดเคลื่อนมาก นอกจากนี้ยังมีข้อผิดพลาดอีกชนิดหนึ่ง คือ เมื่ออ่านข้อมูลเข้าไปเมื่อคำนวนหาขนาดของสัญญาณควบคุม สัญญาณควบคุมนี้จะเป็นสัญญาณควบคุมของในเวลาขณะที่เริ่มอ่านข้อมูล ไม่ใช่สัญญาณควบคุมขณะปัจจุบัน ซึ่งมีข้อมูลเปลี่ยนไปแล้ว ดังแสดงในรูปที่ 1.3 ดังนั้นในปริญญานิพนธ์นี้ ได้ทำการแก้ไขโดยการคำนวนหาค่าโดยประมาณของข้อมูลในขณะที่จะส่งสัญญาณควบคุมออกไป แล้วใช้ข้อมูลนี้ในการคำนวนแทน ซึ่งจะทำให้หาสัญญาณควบคุมใกล้เคียงยิ่งขึ้น ดังรูปที่ 1.4 วิธีการนี้จะใช้ได้ดี เมื่อระบบมีการเปลี่ยนแปลงไม่มากนัก มิฉะนั้นจะทำให้ข้อมูลผิดพลาดมากยิ่งขึ้น

$$m(kT) = K\left\{e(kT) + \frac{T}{T_i}\left[\frac{e(0)+e(T)}{2} + \frac{e(T)+e(2T)}{2} + \cdots + \frac{e((k-1)T)+e(kT)}{2}\right] + T_d\frac{e(kT)-e((k-1)T)}{T}\right\}$$

or

$$m(kT) = K\left\{e(kT) + \frac{T}{T_i}\sum_{h=1}^{k}\frac{e((h-1)T)+e(hT)}{2} + \frac{T_d}{T}[e(kT)-e((k-1)T)]\right\} \qquad (3\text{-}65)$$

Define

$$\frac{e((h-1)T)+e(hT)}{2} = f(hT), \qquad f(0)=0$$

Figure 3–37 shows the function $f(hT)$. Then

$$\sum_{h=1}^{k}\frac{e((h-1)T)+e(hT)}{2} = \sum_{h=1}^{k} f(hT)$$

Taking the z transform of this last equation, we obtain

$$\mathcal{Z}\left[\sum_{h=1}^{k}\frac{e((h-1)T)+e(hT)}{2}\right] = \mathcal{Z}\left[\sum_{h=1}^{k} f(hT)\right] = \frac{1}{1-z^{-1}}[F(z)-f(0)]$$

$$\frac{1}{1-z^{-1}}F(z)$$

$$F(z) = \mathcal{Z}[f(hT)] = \frac{1+z^{-1}}{2}E(z)$$

Hence

$$\mathcal{Z}\left[\sum_{h=1}^{k}\frac{e((h-1)T)+e(hT)}{2}\right] = \frac{1+z^{-1}}{2(1-z^{-1})}E(z)$$

Then the z transform of Eq. (3–65) gives

$$M(z) = K\left[1 + \frac{T}{2T_i}\frac{1+z^{-1}}{1-z^{-1}} + \frac{T_d}{T}(1-z^{-1})\right]E(z)$$

This last equation may be rewritten as follows:

$$M(z) = K\left[1 - \frac{T}{2T_i} + \frac{T}{T_i}\frac{1}{1-z^{-1}} + \frac{T_d}{T}(1-z^{-1})\right]E(z)$$

$$= \left[K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1})\right]E(z)$$

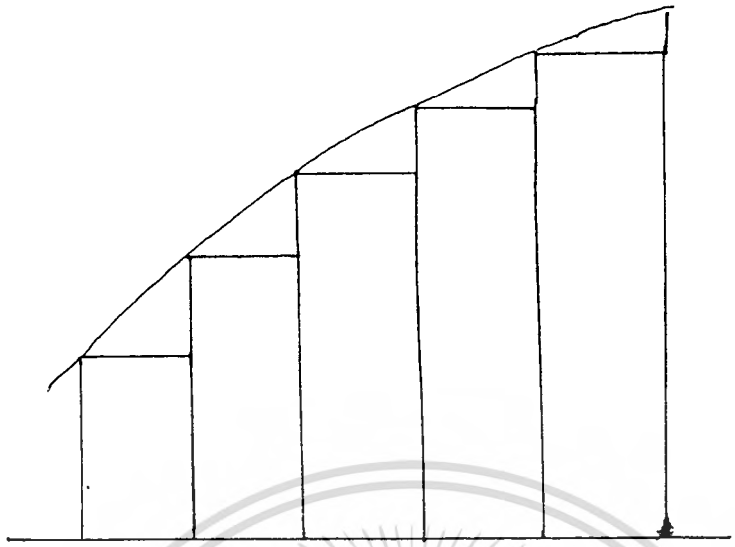
where

$$K_P = K - \frac{KT}{2T_i} = K - \frac{K_I}{2} = \text{proportional gain}$$

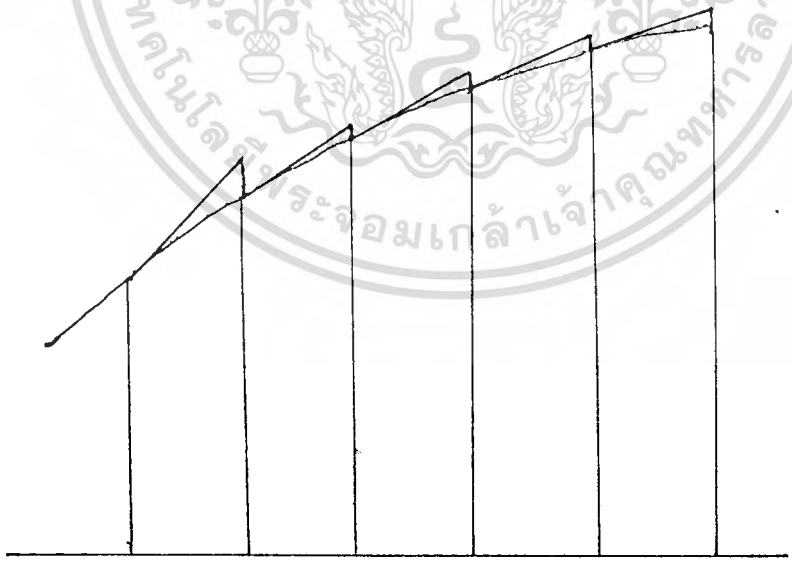
$$K_I = \frac{KT}{T_i} = \text{integral gain}$$

$$K_D = \frac{KT_d}{T} = \text{derivative gain}$$

รูปที่ 1.2



รูปที่ 1.3



รูปที่ 1.4

1.2 ลักษณะของตัวควบคุม

1.2.1 ตัวควบคุมแบบเปิด/ปิด (ON/OFF)

เป็นการควบคุมที่พื้นฐานที่สุด ดังแสดงในรูปที่ 1.5 เมื่อผลต่าง(ERROR) มีค่ามากกว่าที่กำหนด(E) จะทำให้สัญญาณควบคุมเปลี่ยนจาก Ø % เป็น 100 % และเมื่อผลต่างลดลง จะต้องลดลงจนน้อยกว่าอีกค่าหนึ่ง(-E) สัญญาณควบคุมจึงจะเปลี่ยนจาก 100 % เป็น Ø % ทำให้เกิดช่วงเดทแบนท์(DEAD BAND) ขึ้นระหว่างจุดที่ผลต่างเป็นศูนย์ ซึ่งจะเขียนสมการได้ดังนี้

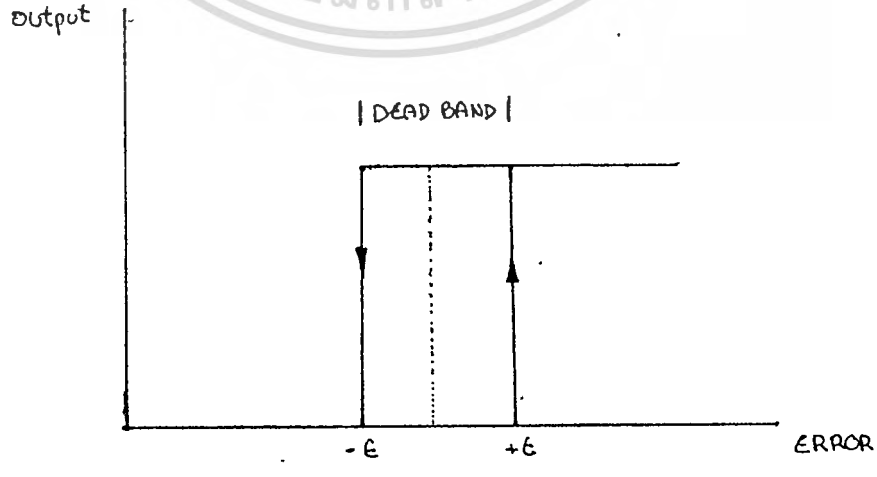
$$C = Ø\ \% \quad e < -E$$

$$C = 100\ \% \quad e > +E$$

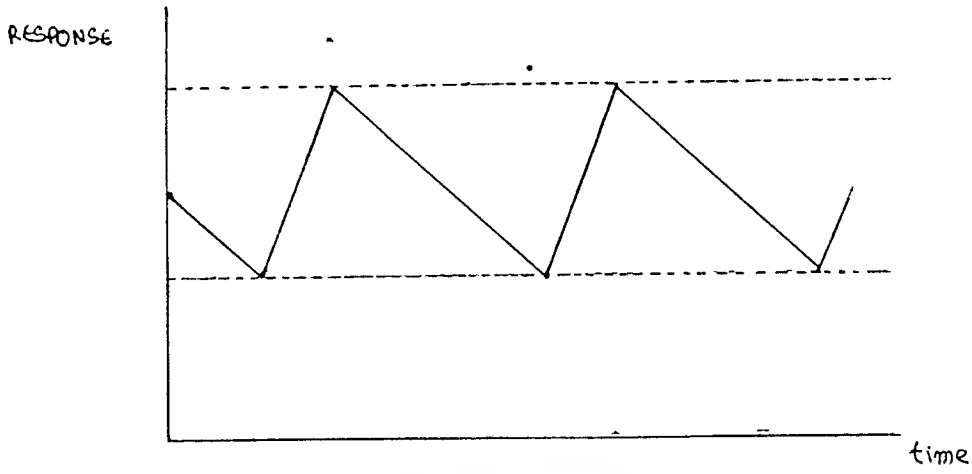
e ผลต่างของค่าที่ตั้งไว้กับค่าที่วัดได้

E ครึ่งหนึ่งของช่วงเดทแบนด์

การควบคุมแบบนี้จะทำให้ระบบเกิด ออสซิเลท(OSCILLATE)ขึ้น ดังรูปที่ 1.6 ซึ่งความถี่ในการออสซิเลตจะเพิ่มขึ้นเมื่อลดค่าช่วงเดทแบนด์ลง ระบบที่ใช้การควบคุมแบบนี้มักจะเป็นระบบที่มีการเปลี่ยนแปลงแบบช้าๆ เช่น ในระบบเครื่องปรับอากาศ เป็นต้น



รูปที่ 1.5



รูปที่ 1.6

1.2.2 ตัวควบคุมแบบ P (PROPORTIONAL)

การควบคุมในลักษณะนี้ ขนาดของสัญญาณควบคุมนี้ จะเป็นสัดส่วนโดยตรงกับผลต่างระหว่างค่าที่วัดได้จากระบบ และค่าที่กำหนด (SET POINT) ดังสมการ

$$C_{(t)} = Ke_{(t)} + B$$

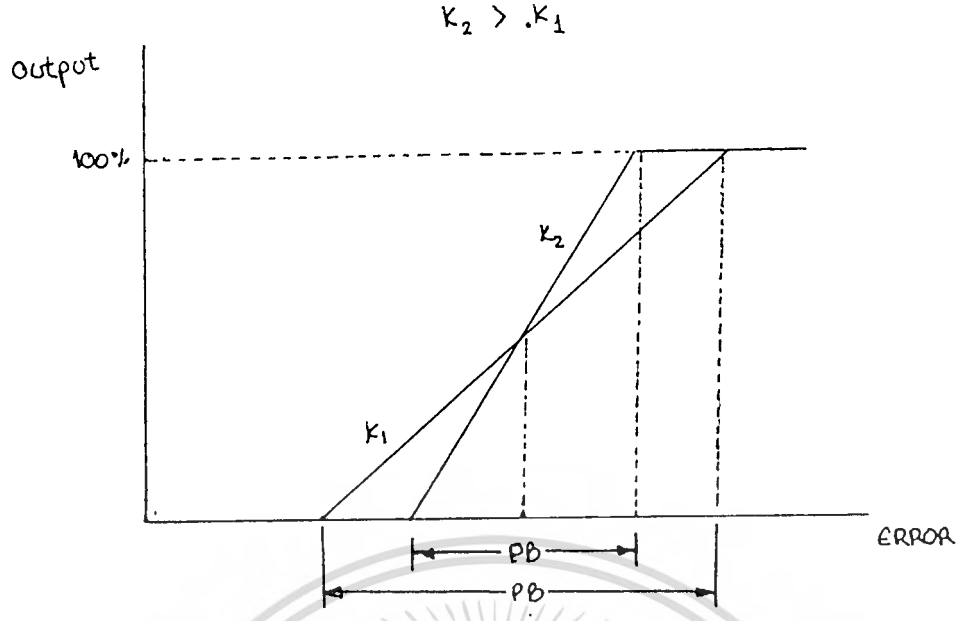
$C_{(t)}$ = ขนาดสัญญาณควบคุมที่เวลา t

K = อัตราขยาย (Porportional gain)

$e_{(t)}$ = ผลต่างระหว่างระบบและค่าที่กำหนด ที่เวลา t

B = สัญญาณควบคุมเมื่อผลต่างเป็นศูนย์

รูปที่1.7 เป็นกราฟแสดงขนาดของสัญญาณควบคุมที่ค่าผลต่างต่างๆ สังเกตได้ว่า สัญญาณควบคุมจะมีขนาดสูงสุดที่ 100% ดังนั้นถ้าสัญญาณควบคุมมีขนาด 100% แล้ว ถึงจะเพิ่มขนาดผลต่างอีก สัญญาณควบคุมก็จะไม่เพิ่มตาม แสดงว่าเกิดการอิ่มตัว (SATURATE) และในทำนองเดียวกัน สัญญาณควบคุมจะมีขนาดต่ำสุดที่ 0% ดังนั้นสัญญาณควบคุมจะแปรตามสัญญาณผลต่างในช่วงๆหนึ่งเท่านั้น เรียกช่วงนี้ว่า PB (PROPORTIONAL BAND) ขนาดของช่วงนี้จะขึ้นอยู่กับอัตราขยาย



รูปที่ 1.7

เมื่อผลต่างเป็นศูนย์ สัญญาณควบคุมจะมีขนาดเท่ากับ B ซึ่งอาจจะใช้ได้ในสภาวะหนึ่ง แต่เมื่อมีการเปลี่ยนแปลงเกิดขึ้น(LOAD CHANGE) ค่า B นี้จะต้องทำการแก้ไขเมื่อลดผลต่างลง ในตัวควบคุมแบบอนาลอค ค่านี้อาจจะค่อยๆปรับโดยผู้ควบคุม แต่ในทางดิจิตอลค่านี้เป็นค่าคงที่ในโปรแกรม ซึ่งจะต้องหยุดกระบวนการทั้งหมดก่อนจึงจะปรับค่านี้ได้ ดังนั้นเมื่อค่านี้ผิดไปจะทำให้ผลต่างไม่เท่ากับศูนย์ ผลต่างนี้จะคงอยู่ตลอดไป และมีค่าขึ้นอยู่กับอัตราขยาย เรียกผลต่างนี้ว่าออฟเซต (OFFSET)

1.2.3 ตัวควบคุมแบบ I (INTEGRAL)

การควบคุมในลักษณะนี้ ขนาดของสัญญาณควบคุมจะขึ้นอยู่กับลักษณะเดิมของผลต่างระหว่างค่าที่ได้จากระบบ และค่าที่ตั้งไว้มากกว่าขนาดของผลต่างเอง ลักษณะเดิมนี้จะวัดจากพื้นที่ใต้กราฟของกราฟระหว่างผลต่างและเวลา นั่นคือขนาดของสัญญาณควบคุมเป็นอัตราส่วนกับพื้นที่ใต้กราฟตั้งแต่เริ่มต้นจนถึงเวลาปัจจุบัน ในแง่ของทางคณิตศาสตร์ คือการทำปฏิกิริยานุพันธ์ (INTEGRATE) กราฟผลต่าง ซึ่งจะได้สมการดังนี้

$$C_{(t)} = K\int_{0}^{t} e_{(t)}dt$$

$$C_{(t)} = K\,Ae_{(t)} + C_{(0)}$$

$C_{(t)}$ = ขนาดของสัญญาณควบคุมที่เวลา t

K = ค่าคงที่ (INTEGRAL GAIN)

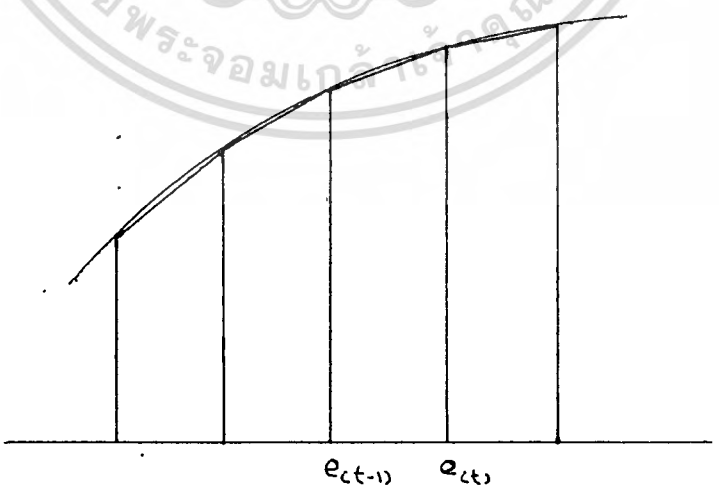
$Ae_{(t)}$ = พื้นที่ใต้กราฟรวมตั้งแต่เวลา t=0 จนถึง t=t

$C_{(0)}$ = ขนาดของสัญญาณควบคุมที่เวลา t=0

เนื่องจาก การควบคุมลักษณะนี้จะขึ้นอยู่กับลักษณะเดิมของสัญญาณผลต่างจึงเป็นไปได้ ที่ขณะที่ผลต่างเป็นศูนย์แล้ว แต่ยังมีการเปลี่ยนแปลงของสัญญาณควบคุม ทั้งนี้เนื่องจากยังมีลักษณะเดิมของสัญญาณผลต่างอยู่ จึงทำให้สามารถแก้ไขขนาดสัญญาณควบคุมเมื่อผลต่างเป็นศูนย์ให้เป็นค่าที่เหมาะสมกับระบบได้ ถึงแม้ระบบจะมีการเปลี่ยนแปลงก็ตาม นั่นคือสามารถกำจัดออฟเซต ให้หมดไปได้

ในการควบคุมแบบดิจิตอล จะรับสัญญาณผลต่างแบบไม่ต่อเนื่อง ดังนั้นการหาพื้นที่ทำได้โดยการประมาณค่า ซึ่งทำได้หลายวิธี ในปริญญานิพนธ์นี้ ใช้วิธีประมาณค่าเป็นพื้นที่สี่เหลี่ยมคางหมู ดังรูปที่ 1.8 ซึ่งจะเขียนเป็นสมการได้ ดังนี้

พื้นที่ ที่เวลา t = $1/2\,[e_{(t-1)} + e_{(t)}]$



รูปที่ 1.8

จะเห็นได้ว่า เมื่อระยะเวลาสุ่มตัวอย่างน้อยลงจะทำให้ประมาณค่าพื้นที่ได้ใกล้เคียงยิ่งขึ้น และถ้ามีการเปลี่ยนแปลงของผลต่างมาก จะทำให้ประมาณค่าผิดมากขึ้นด้วย ดังนั้น ตัวควบคุมจะต้องมีระยะเวลาในการสุ่มตัวอย่าง(SAMPLING TIME) ที่เหมาะสมกับระบบที่ควบคุมด้วย

โดยทั่วไป การควบคุมในลักษณะนี้เพียงอย่างเดียว จะมีปัญหาเมื่อระบบเริ่มต้นทำงาน(START UP) เนื่องจากจะทำให้เกิด โอเวอร์ชูต(OVER SHOOT) สูงมาก ดังนั้นจึงมักจะใช้ร่วมกันในลักษณะอื่น

1.2.4 ตัวควบคุมแบบ D (DERIVATIVE)

ในลักษณะนี้ ขนาดของสัญญาณควบคุมจะขึ้นอยู่กับอัตราการเปลี่ยนแปลงของสัญญาณผลต่างในขณะนั้น จะไม่ขึ้นอยู่กับขนาดของสัญญาณผลต่าง ดังนั้น จึงเป็นไปได้ที่สัญญาณผลต่างเป็นศูนย์ แต่สัญญาณควบคุมมีขนาดสูงมาก ในทางคณิตศาสตร์ คือการหาอนุพันธ์ (DIFFERENTIATE) ของสัญญาณผลต่าง ดังสมการ

$$C_{(t)} = K [de_{(t)}/dt]$$

$C_{(t)}$ สัญญาณควบคุมที่เวลา t

K ค่าคงที่ (DERIVATIVE GAIN)

$e_{(t)}$ สัญญาณผลต่างที่เวลา t

จะเห็นได้ว่าการควบคุมในลักษณะนี้ ถ้าหากผลต่างไม่มีการเปลี่ยนแปลง ก็จะไม่มีสัญญาณควบคุม ดังนั้น การควบคุมในลักษณะนี้จะต้องใช้ร่วมกับการควบคุมในลักษณะอื่นเสมอ ในตัวควบคุมแบบดิจิตอลจะพบปัญหาในการคำนวนเช่นเดียวกับในการควบคุมแบบ I เนื่องจากสัญญาณเป็นแบบไม่ต่อเนื่อง การคำนวนหาค่าจะเป็นเพียงการประมาณเท่านั้น ซึ่งจะประมาณได้ใกล้เคียงยิ่งขึ้น ถ้าหากระยะเวลาในการสุ่มตัวอย่างลดลง เราสามารถจะประมาณ ได้ดังสมการ

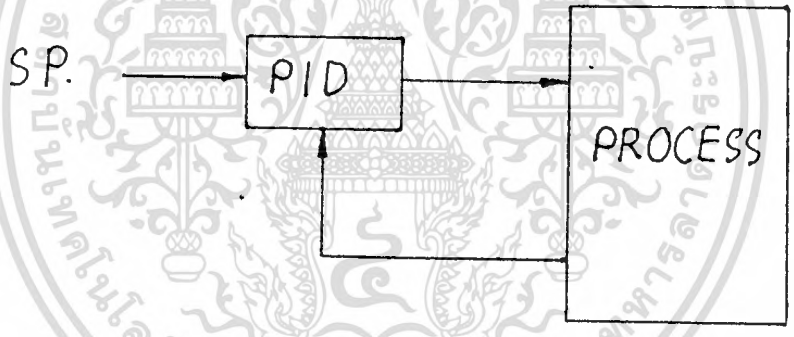
$$C_{(t)} = [K/T][\, e_{(t)} - e_{(t-1)} \,]$$

$e_{(t)}$ คือ ผลต่างที่เวลา t

T คือ ระยะเวลาในการสุ่มตัวอย่าง

1.3 การควบคุมแบบลูปเดียว (SINGLE LOOP CONTROL)

การควบคุมแบบนี้ จะมีตัวควบคุมแบบ PID หนึ่งตัวเป็นตัวควบคุมของระบบ ดังรูปที่ 1.9 การควบคุมแบบ PID นี้จะรวมเอาข้อดีของแบบ P, I และ D เข้าด้วยกัน ทำให้ระบบเข้าสู่สมดุลย์ได้อย่างรวดเร็ว และไม่มีออฟเซต ในปริญญานิพนธ์นี้ ได้สร้างตัวควบคุมชนิดนี้สองชุด



รูปที่ 1.9

ตัวควบคุมแบบอนาลอค PID ที่ใช้ในงานอุตสาหกรรมทั่วๆไปนั้น จะมีฟังก์ชั่นการคำนวนต่างกันออกไปเนื่องจากเหตุผลในจัดสร้าง โดยทั่วไปจะประกอบด้วยสามชนิดใหญ่ๆ คือ

1) $K\ [e_{(t)} + T\int_0^t e_{(t)}dt + D\ de_{(t)}/dt]$

2) $K\ e_{(t)} + T\int_0^t e_{(t)}dt + D\ de_{(t)}/dt$

3) $K\ [e_{(t)} + T\int_0^t e_{(t)}dt][1 + D\ de_{(t)}/dt]$

ทั้งสามแบบนี้จะให้การตอบสนองเมื่อมีการรบกวนระบบ (DISTRUBANCE) คล้ายกัน แต่เมื่อมีการเปลี่ยนจุดในการทำงาน(SET POINT) จะให้การตอบสนองต่างกัน โดยแบบที่ 3 จะไวต่อการเปลี่ยนแปลงของสัญญาณผลต่างมากกว่าแบบที่ 1 จึงเหมาะที่จะเป็นตัวควบคุมของลูปย่อยมากกว่า ส่วนในแบบที่ 1 นั้นจะเหมาะสมที่จะเป็นตัวควบคุมของลูปหลัก เพราะมีการตอบสนองการเปลี่ยนแปลงที่สม่ำเสมอ(smooth) กว่าแบบที่ 3

ดังจะเห็นได้ว่า ทั้งสามแบบนี้จะมีข้อดีข้อเสียต่างกันจึงเหมาะสมกับงานต่างกัน และค่าพารามิเตอร์ที่เหมาะสมกับตัวควบคุมชนิดหนึ่ง อาจจะไม่เหมาะสมกับอีกชนิดหนึ่งก็ได้ ซึ่งในปริญญานิพนธ์นี้ ได้เขียนโปรแกรมสำหรับสร้างตัวควบคุมแบบดิจิตอล PID โดยมีสมการดังแบบที่ 1 ทั้งสองชุด ดังนี้

$$C_{(t)} = K [e_{(t)} + K_i * A_{(t)} + [K_d/T][e_{(t)} - e_{(t-1)}]]$$

$$A_{(t)} = A_{(t-1)} + [T/2][e_{(t)} + e_{(t-1)}]$$

C ขนาดสัญญาณควบคุม

K อัตราขยาย

K_i ค่าคงที่ (INTEGRAL GAIN)

K_d ค่าคงที่ (DERIVATIVE GAIN)

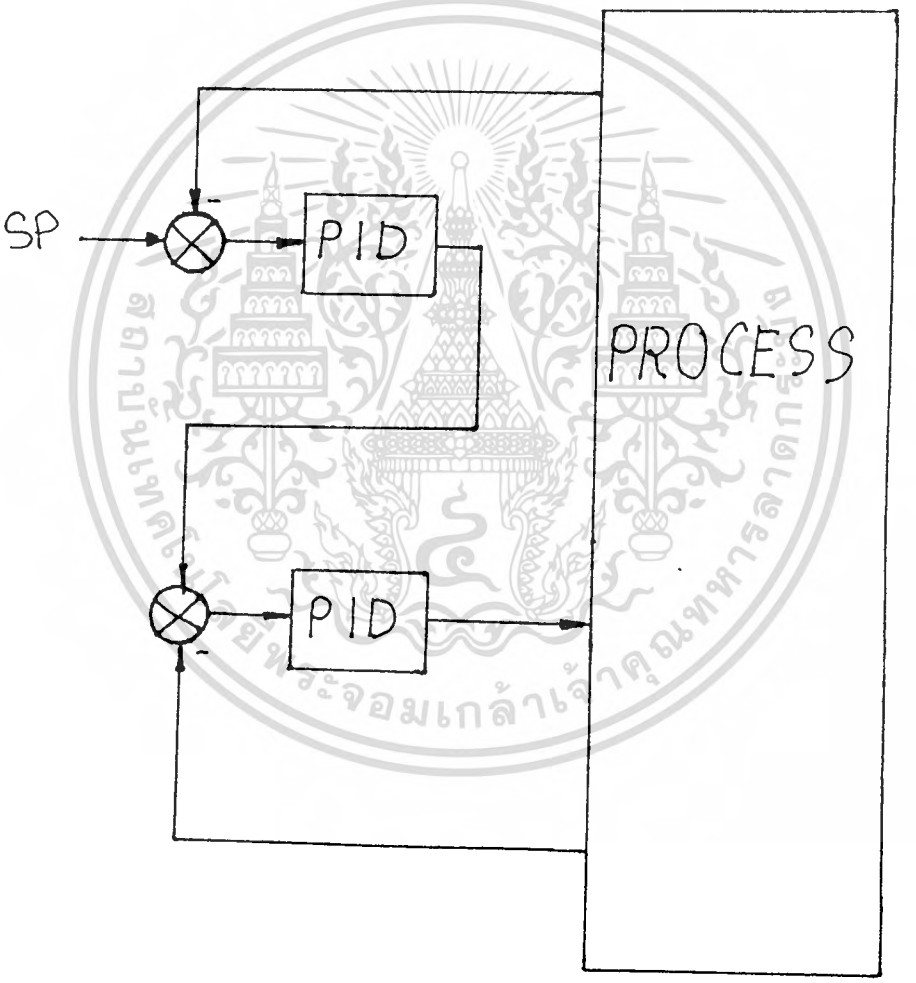
T คาบเวลาในการสุ่มตัวอย่าง

A พื้นที่ใต้กราฟผลต่างรวมจนถึงเวลา t

เนื่องจาก ในระบบดิจิตอลขนาดของสัญญาณควบคุมเกิดจากการคำนวณค่าทางคณิตศาสตร์โดยตรงซึ่งต่างกับในระบบอนาลอก จึงต้องมีการจำกัดค่าสูงสุดและต่ำสุดของพื้นที่(A)ไว้ที่ค่าค่าหนึ่งซึ่งจะทำให้ค่าโอเวอร์ชูต(OVERSHOOT) ลดลง และระบบสามารถเข้าสู่สมดุลย์ได้เร็วขึ้น แต่ยังสามารถแก้ออฟเซตได้เช่นเดิม ในปริญญานิพนธ์นี้กำหนดไว้ที่ ช่วง ø ถึง 1øø เมื่อค่าพื้นที่เกินค่าที่กำหนดนี้ก็จะมีค่าเท่าเดิมไม่เปลี่ยนแปลง

1.4 การควบคุมแบบต่อเนื่อง (CASCADE CONTROL)

การควบคุมแบบต่อเนื่องจะมีลักษณะเป็นตัวควบคุม 2 ตัวต่อกันอยู่โดยที่แต่ละตัวมีการรับข้อมูลของตัวเอง แต่ตัวควบคุมตัวแรกจะเป็นอิสระ ส่วนตัวที่สองนี้จะขึ้นอยู่กับตัวแรก สัญญาณควบคุมของตัวควบคุมตัวแรกจะส่งมายังตัวควบคุมตัวที่สอง ดังนั้นจึงมีเพียงสัญญาณของตัวควบคุมตัวที่สองเท่านั้นที่ส่งออกไปควบคุมระบบ จึงเรียกลักษณะของการควบคุมเช่นนี้ว่า การควบคุมแบบต่อเนื่อง (CASCADE CONTROL) ดังรูปที่ 1.10



รูปที่ 1.10

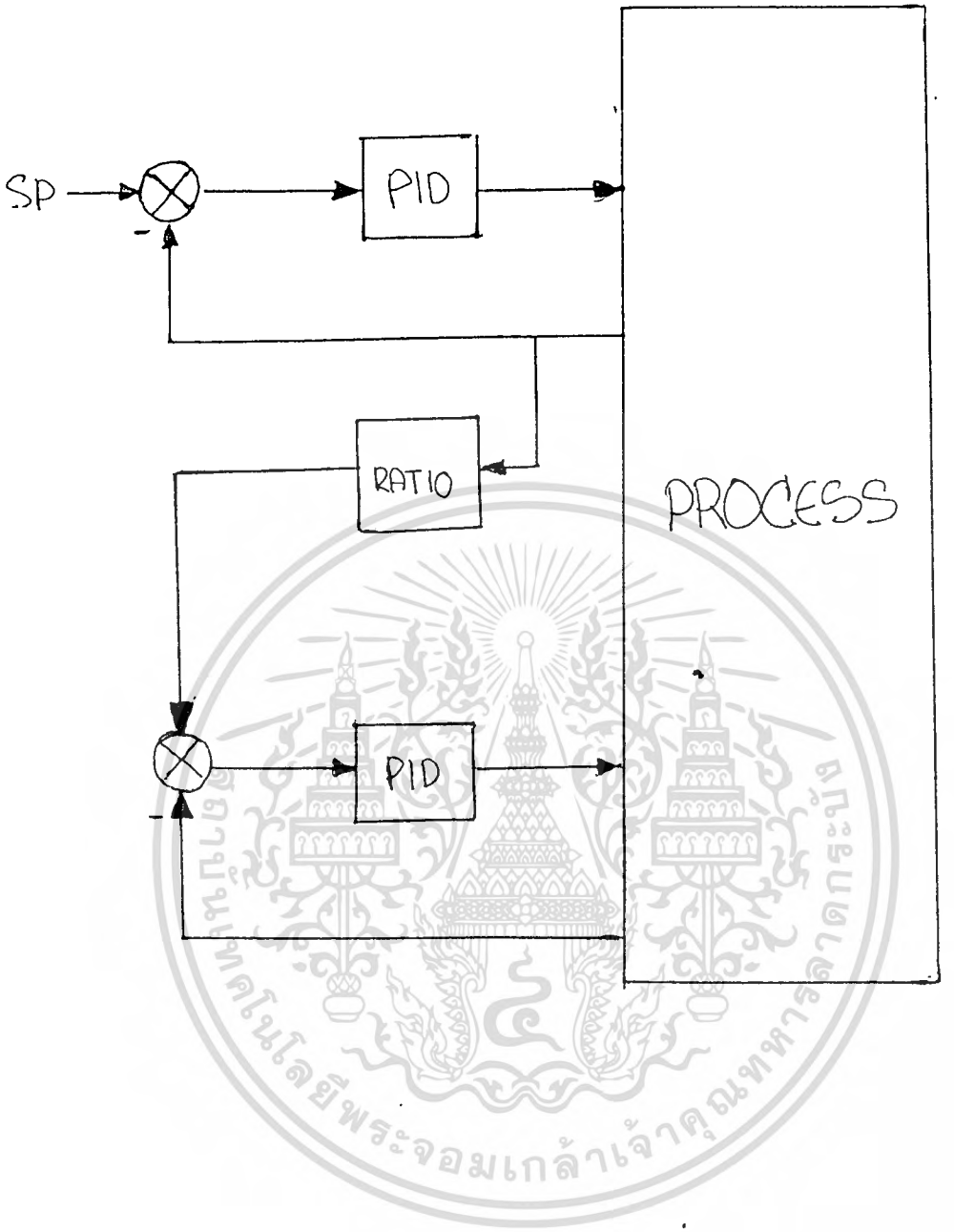
จะใช้การควบคุมแบบนี้เมื่อต้องการจะควบคุมตัวแปรของระบบตัวแปรเดียว แต่ต้องใช้ตัวแปรในการควบคุม 2 ตัว จึงไม่สามารถใช้ตัวควบคุมเพียงหนึ่งตัวควบคุมได้ ซึ่งตัวควบคุมแต่ละตัวนี้ สามารถจะกำหนดให้มีพารามิเตอร์ที่เหมาะสมกับลูปของตัวเอง ทำให้ระบบปรับตัวเองเมื่อมีการรบกวนได้ดี มีการตอบสนองที่ดีขึ้น แต่อย่างไรก็ตาม การควบคุมแบบนี้จะต้องได้รับการออกแบบที่เหมาะสม และต้องแบ่งเป็นลูปย่อยได้ จึงใช้ได้กับบางระบบเท่านั้น

เนื่องจากการควบคุมแบบนี้ใช้ตัวควบคุมถึง 2 ตัว ในปริญญานิพนธ์นี้จึงสร้างได้เพียงชุดเดียว ประกอบด้วยตัวควบคุมแบบดิจิตอล PID สองตัวซึ่งมีฟังก์ชันการคำนวณเช่นเดียวกับแบบลูปเดียว แต่นำค่าสัญญาณควบคุมของตัวควบคุมตัวแรกมาเป็นค่าที่กำหนด (SET POINT) ของตัวควบคุมชุดที่สอง ดังนั้นในการใช้งานจึงต้องกำหนดค่าที่ตัวควบคุมตัวแรกแล้วนำสัญญาณควบคุมจากตัวที่สองไปใช้งาน

1.5 การควบคุมอัตราส่วน (RATIO CONTROL)

การควบคุมอัตราส่วน เป็นระบบที่ประกอบด้วยตัวควบคุมสองชุดต่อกันอยู่โดยที่แต่ละตัวจะมีการวัดข้อมูลของตัวเอง และมีตัวควบคุมตัวแรกเท่านั้นที่จะเป็นอิสระเช่นเดียวกับการควบคุมแบบต่อเนื่อง แต่สัญญาณควบคุมของตัวแรกจะถูกนำไปคูณกับอัตราส่วนก่อนจะส่งไปยังตัวควบคุมที่สอง ดังนั้นการควบคุมในลักษณะนี้จะมีสัญญาณควบคุมจากตัวควบคุมทั้งสองไปควบคุมระบบ ดังรูปที่ 1.11

การควบคุมแบบอัตราส่วนนี้ ผลคือตัวควบคุมตัวที่สองจะควบคุมลูปที่สองให้มีอัตราส่วนคงที่เทียบกับลูปที่หนึ่ง เมื่อระบบแรกมีการเปลี่ยนแปลงไป ระบบที่สองก็จะเปลี่ยนแปลงตามเพื่อให้อัตราส่วนคงที่อยู่เสมอ ในปริญญานิพนธ์นี้จัดสร้างขึ้นโดยลักษณะคล้ายกับการควบคุมแบบต่อเนื่อง การใช้งานผู้ควบคุมจะกำหนดค่าที่ตัวควบคุมตัวแรกและอัตราส่วนที่ต้องการ แล้วนำสัญญาณควบคุมจากตัวควบคุมทั้งสองไปต่อเข้ากับระบบ

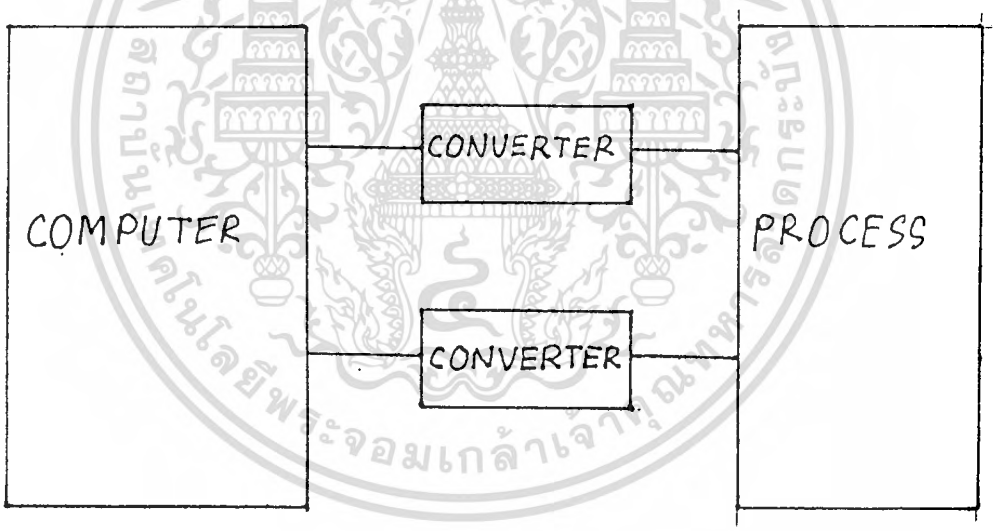


รูปที่ 1.11

บทที่ 2
ฮาร์ดแวร์ (Hardwere)

ฮาร์ดแวร์ คือส่วนของวงจรอิเลคทรอนิกส์ ซึ่งใช้เชื่อมโยง (interface) ระหว่างคอมพิวเตอร์และอุปกรณ์ที่เราต้องการจะควบคุม ในปริญญานิพนธ์นี้ ส่วนของฮาร์ดแวร์จะแบ่งได้เป็น 2 ส่วนใหญ่ ๆ คือ ส่วนรับข้อมูล (INPUT) สำหรับสัญญาณข้อมูลจากเครื่องวัด (SENSOR) เข้าสู่คอมพิวเตอร์เพื่อใช้ในการควบคุม และส่วนส่งข้อมูล (OUTPUT) สำหรับแปลงข้อมูลซึ่งคอมพิวเตอร์ประมวลผลแล้ว ให้เป็นสัญญาณควบคุมเพื่อส่งไปควบคุมเครื่องจักรหรือขบวนการ (Process) ดังรูปที่ 2.1

นอกจากนี้แล้ว ถ้าแบ่งตามหน้าที่ของวงจรอิเลคทรอนิกส์ ฮาร์ดแวร์จะแบ่งได้เป็น 5 ส่วน ดังรูปที่ 2.2



รูปที่ 2.1

1) ส่วนถอดรหัส (PORT DECODER) ทำให้ส่วนของฮาร์ดแวร์รู้ว่าคอมพิวเตอร์ต้องการจะติดต่อกับฮาร์ดแวร์ชุดใด ดังนั้นส่วนนี้ จึงทำให้คอมพิวเตอร์ติดต่อกับฮาร์ดแวร์ได้หลายชุด

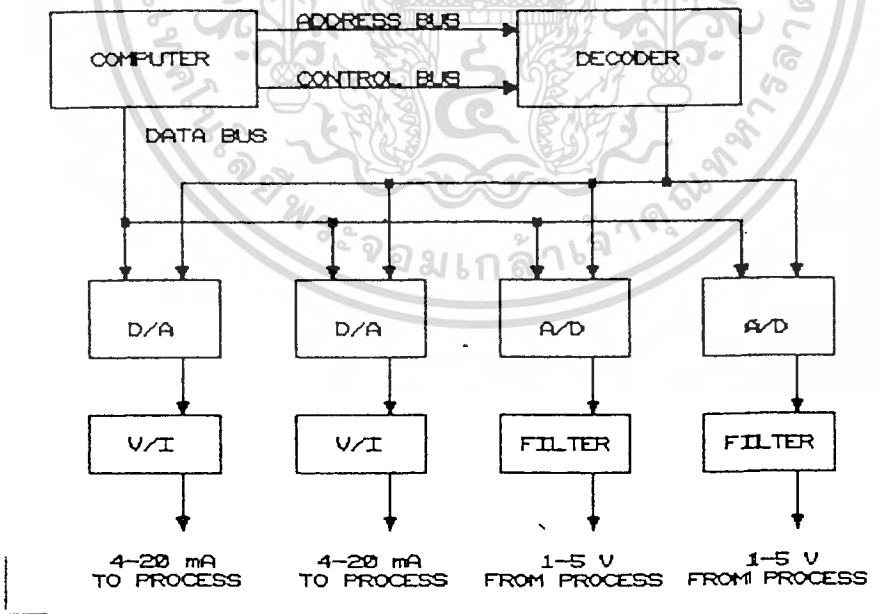
2) D/A คอนเวอร์ทเตอร์ (DIGITAL TO ANALOG CONVERTER) จะทำหน้าที่แปลงข้อมูล ซึ่งคอมพิวเตอร์คำนวณได้ ซึ่งเป็นตัวเลข 8 บิท (BIT) ให้เป็นความต่างศักย์ทางไฟฟ้า โดยจะแปลงจากตัวเลข 0 ถึง 255 ให้เป็นค่าความต่างศักย์ 1 ถึง 5

โวลท์ (VOLT) ความต่างศักย์นี้จะถูก ต่อไปยังส่วนที่สามต่อไป

3) V/I คอนเวอร์ทเตอร์ (VOLTAGE TO CURRENT CONVERTER) จะทำหน้าที่แปลงความต่างศักย์ที่ได้มาจากส่วนที่ 2 ให้เป็น กระแสไฟฟ้า เพื่อส่งไปควบคุมอุปกรณ์ต่าง ๆ โดยจะแปลงจากสัญญาร 1 ถึง 5 โวลท์ ให้เป็นกระแสไฟฟ้า ซึ่งตามมาตราฐานจะให้ค่ากระแส 4 ถึง 20 มิลลิแอมแปร์

4) ส่วนกรองสัญญาณรบกวน (LOW PASS FILTER) เป็นส่วนที่รับสัญญาณจากเครื่องมือวัด (SENSOR) ในรูปของความต่างศักย์ และ กรองสัญญาณรบกวนที่อาจจะเกิดขึ้น และส่งสัญญาณที่กรองแล้วในรูปของความต่างศักย์ไปยังส่วนที่ 5 ต่อไป นอกจากนี้แล้วส่วนนี้ยังทำหน้าที่ คอยป้องกันความเสียหายของคอมพิวเตอร์ เมื่อสัญญาณที่ส่งมาผิดพลาด หรือในกรณีที่มีอุปกรณ์มีการชำรุดเสียหาย

5) A/D คอนเวอร์ทเตอร์ (ANALOG TO DIGITAL CONVERTER) ทำหน้าที่แปลงสัญญาณจากส่วนที่ 4 เป็นตัวเลขและส่งให้กับคอมพิวเตอร์ เพื่อใช้ในการประมวลผลต่อไป สัญญาณที่รับมานี้จะอยู่ในรูปของความต่างศักย์มาตรฐาน 1 ถึง 5 โวลท์ จะถูกแปลงเป็น ตัวเลข 0 ถึง 255

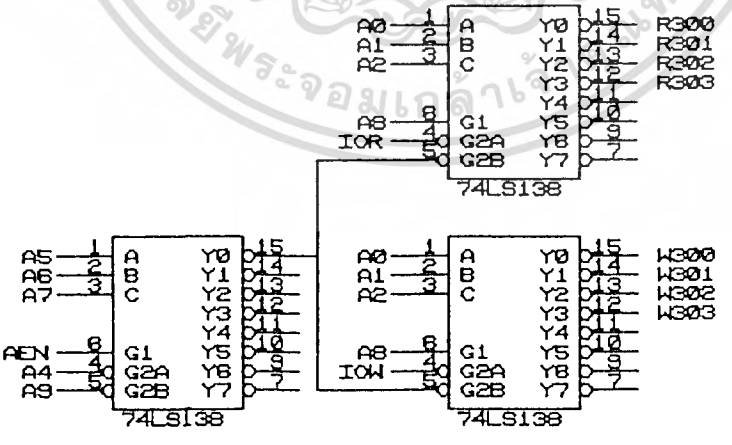


รูปที่ 2.2

ส่วนของฮาร์ดแวร์ทั้งหมดนี้ได้ออกแบบสร้างให้เป็นแผ่นวงจรซึ่งสามารพเสียบเข้ากับส่วนขยาย (EXPANSION SLOT) ของเครื่องคอมพิวเตอร์ ไอบีเอ็ม พีซี ในหัวข้อต่อไปจะกล่าวถึงฮาร์ดแวร์ทั้ง 5 ส่วนนี้ โดยละเอียด

2.1 ส่วนถอดรหัส

ในส่วนนี้ต้องการจะถอดรหัสสัญญาณอ้างอิงเบอร์พอร์ต (ADDRESS BUS) ของคอมพิวเตอร์และสัญญาณควบคุม (CONTROL BUS) โดยใช้ ไอซีเบอร์ 74138 เมื่อ คอมพิวเตอร์ต้องการจะส่งข้อมูลหรือรับข้อมูลจากพอร์ตใด คอมพิวเตอร์จะส่งสัญญาณต่างๆออกมา วงจรดังรูปที่2.3 จะทำการถอดรหัสเป็นสัญญาณไปควบคุมการทำงานของ ฮาร์ดแวร์ส่วนอื่น ๆ ที่เกี่ยวข้องด้วย คือ เมื่อคอมพิวเตอร์ ส่งหรือรับข้อมูล ทาง พอร์ท 300 ถึง 307 (ฐานสิบหก) วงจรจะทำการถอดรหัสเป็นสัญญาณศูนย์ ทำให้ ฮาร์ดแวร์ส่วนที่เกี่ยวข้องด้วย คอมพิวเตอร์ ต้องการจะติดต่อด้วย



รูปที่ 2.3

2.2 D/A คอนเวอร์ทเตอร์

ทำหน้าที่แปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก ในส่วนนี้ใช้ไอซีเบอร์ DAC 5018 ของบริษัท SIGNATIC เป็นหลัก ไอซีนี้จะรับตัวเลขได้ 8 บิท (BIT) ซึ่งภายในตัวไอซีนี้ประกอบด้วยส่วนสำคัญ 4 ส่วน คือ

1 ส่วนคงสถานะของตัวเลข (LATCH) ทำหน้าที่จำตัวเลขที่ต้องการจะเปลี่ยนไว้ เพื่อส่งไปเปลี่ยนเป็นสัญญาณอนาลอก

2 ส่วนสร้างความต่างศักย์ ใช้สร้างความต่างศักย์สำหรับอ้างอิงค่าตัวเลข 1 000 0000 จะถูกเปลี่ยนเป็นความต่างศักย์ค่านี้ ส่วนค่าตัวเลขอื่นจะลดลงหรือเพิ่มขึ้นด้วยอัตราส่วนที่คงที่

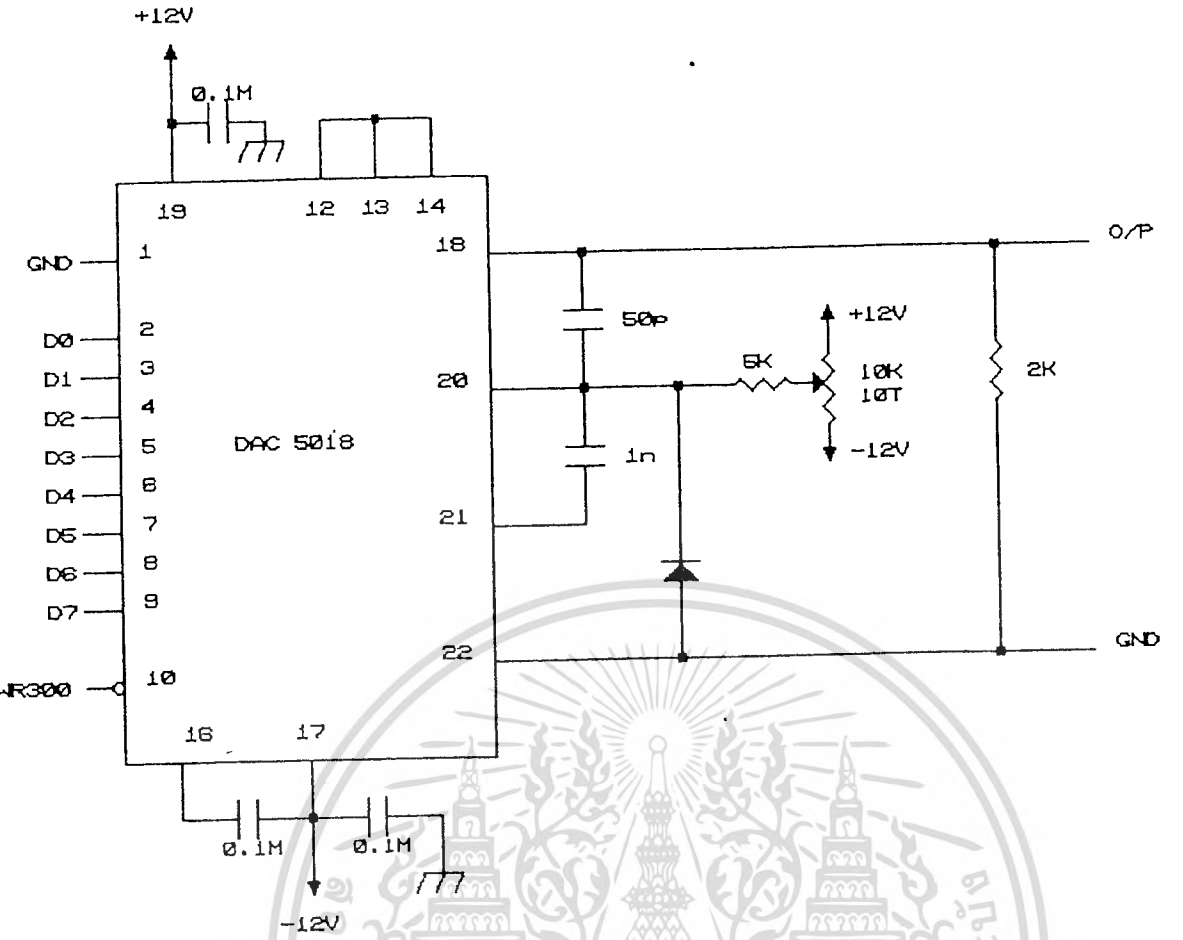
3 ส่วนแปลงสัญญาณ ทำหน้าที่แปลงสัญญาณดิจิตอลจากส่วนที่ 1 ให้เป็นสัญญาณ อนาลอค ในรูปของกระแสไฟฟ้า

4 ส่วนปรับแต่งสัญญาณ ทำหน้าที่เปลี่ยนรูปกระแสไฟฟ้าให้เป็นความต่างศักย์ และเป็นจุดที่ใช้ปรับค่าชดเชย (OFSET ADJUST)

การออกแบบวงจร

เราต้องการจะเปลี่ยนตัวเลข 0 ถึง 255 ให้เป็นความต่างศักย์ 1 ถึง 5 โวลท์ ดังนั้นจะต้องให้ส่วนสร้างความต่างศักย์ สร้างความต่างศักย์สำหรับอ้างอิง เท่ากับ 2 โวลท์ ซึ่งจะทำให้เปลี่ยนจากตัวเลข 0 ถึง 255 เป็นความต่างศักย์ 0 ถึง 4 โวลท์ จากนั้นจึงปรับที่ส่วนปรับแต่งสัญญาณให้ปรับค่าชดเชย (OFSET) เพิ่มขึ้นอีก 1 โวลท์ ดังนั้นความต่างศักย์ที่ได้ (OUTPUT VOLTAGE) จะมีค่าเป็น 1 ถึง 5 โวลท์

การต่อวงจรจะประกอบด้วย วงจรนี้ 2 ชุด โดยชุดแรกจะต่อ ขา LE (LATCH ENABLE) เข้ากับสัญญาณ WRITE 0300 ส่วนชุดที่สองจะต่อกับ สัญญาณ WRITE 0301 ทำให้คอมพิวเตอร์สามารถเลือกชุดทำงานได้ และจากการออกแบบวงจรดังที่กล่าวทำให้เราได้วงจรของชุดแรก ดังรูปที่ 2.4 (ส่วนวงจรของชุดที่สองจะเหมือนกัน ต่างกันเพียงสัญญาณที่ขา LE เท่านั้น)



รูปที่ 2.4

2.3 V/I คอนเวอร์ทเตอร์

ทำหน้าที่แปลงจากความต่างศักย์ให้เป็นกระแสไฟฟ้า โดยใช้หลักการของวงจรสะท้อนกระแส (CURRENT MIRROR) สร้างเป็นแหล่งจ่ายกระแส (CURRENT SOURCE) ส่งสัญญาณควบคุมไปยังระบบ เพื่อป้องกันมิให้ D/A คอนเวอร์ทเตอร์ จ่ายกระแสมากเกินกำลัง นอกจากนี้เพราะว่าเป็นลักษณะของแหล่งจ่ายกระแส ทำให้เราสามารถจ่ายกระแสได้คงที่เสมอ ไม่ขึ้นอยู่กับความต้านทาน ของระบบ

ในการออกแบบเราต้องการให้เปลี่ยนจากความต่างศักย์ 1 – 5 โวลท์ เป็นกระแส 4-20 มิลลิแอมแปร รูปที่ 2.5 แสดงวงจรที่ออกแบบใช้ในโครงงานนี้ หลักการทำ

งานคือ เมื่อรับความต่างศักย์ (INPUT VOLTAGE) เข้ามา ความต่างศักย์นี้จะไปตกคร่อมความต้านทาน R1,R2 และ ขา BE ทรานซิสเตอร์ Q2 ทำให้เกิดแระแสไฟฟ้าไหลลงมาจาก Q1 ผ่าน Q2,R1 และ R2 ลงมา กระแสนี้จะเป็นไปตามสมการ

$$I = V/R$$

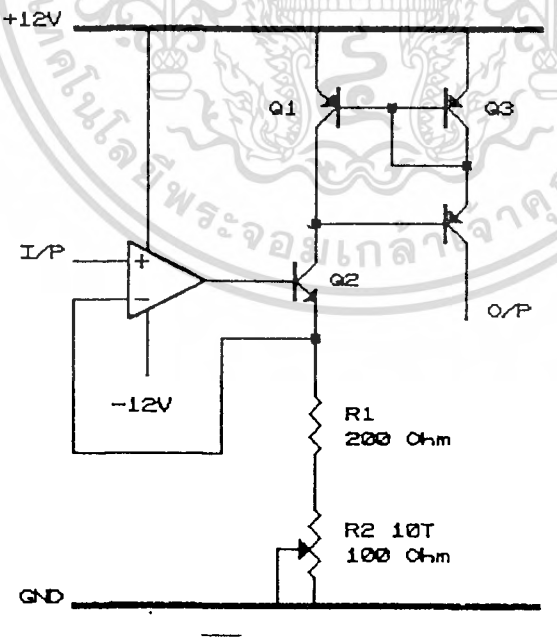
เมื่อ I คือ กระแสที่ไหล

V คือ ความต่างศักย์

R คือ ความต้านทานของ R1 + R2

เนื่องจาก Q1 และ Q3 มีขา BASE ต่อถึงกันดังนั้น Q3 จึงมีกระแสไหลผ่านเท่ากับ Q1 ทำให้มีกระแสไหลผ่าน Q3 ลงมายัง Q4 และจ่ายไปยังอุปกรณ์ตามต้องการ

ในวงจรนี้เราจะต้องชดเชยความต่างศักย์ที่ไปตกคร่อมขา BE ของ Q2 โดยการเพิ่มค่าชดเชย (OFSET) ของไอซี 5Ø18 อีกเท่ากับความต่างศักย์คร่อม ขา BE ของ Q2 อีกจุดหนึ่งที่ต้องชดเชยคือ อัตราขยายของ Q1 และ Q3 มีไม่เท่ากันทำให้กระแสไหลผ่าน Q1 และ Q3 มีไม่เท่ากัน ซึ่งชดเชยได้โดยการปรับค่า R2 ซึ่งจะทำให้กระแสที่ผ่าน Q1 เปลี่ยนและมีผลทำให้เราปรับค่าแระแสผ่าน Q3 ให้เป็นไปตามต้องการได้



รูปที่ 2.5

2.4 ส่วนกรองสัญญาณรบกวน

ในส่วนนี้ใช้ไอซี ออบแอมป์ (OP AMP) เบอร์ 741 มาสร้างขึ้นเป็นวงจรกรองสัญญาณรบกวน (ACTIVE LOW PASS FILTER) นอกจากนั้น วงจรนี้ยังสามารถจะป้องกันความเสียหายให้กับอุปกรณ์ส่วนอื่น เมื่อ สัญญาณที่รับมา ผิดพลาด

ในการออกแบบวงจรนี้ ต้องการให้กรองสัญญาณความถี่สูงออก จึงออกแบบให้มีค่า FREQUENCY ประมาณ 10 KHz ทำให้สามารถคำนวณ ค่าอุปกรณ์ต่าง ๆ ได้ดังรูปที่ 2.6 และได้วงจรดังรูปที่ 2.7

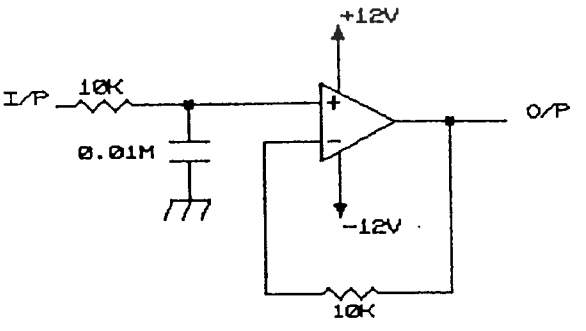
$$\omega_c = 1 / RC$$

$$C = 1 / 2\pi R f_c$$

เลือก $R = 10\ K$ และ $f_c = 1.5\ K$

$$C = 0.01\ \mu F$$

รูปที่ 2.6



รูปที่ 2.7

2.5 A/D คอนเวอร์ทเตอร์

ทำหน้าที่แปลงสัญญาณอนาลอคเป็นสัญญาณดิจิตอล ในส่วนนี้ใช้ไอซีเบอร์ ADC 5034 ของบริษัท SIGNATIC เป็นหลัก ไอซีนี้จะส่งตัวเลขได้ 8 บิท (BIT) ซึ่งภายในตัวไอซีนี้ประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

1 ส่วนสร้างสัญญาณคล๊อก (CLOCK) เพื่อใช้อ้างอิงในการทำงานทั้งหมดของวงจร เราสามารถกำหนดความถี่ของสัญญาณนี้ได้

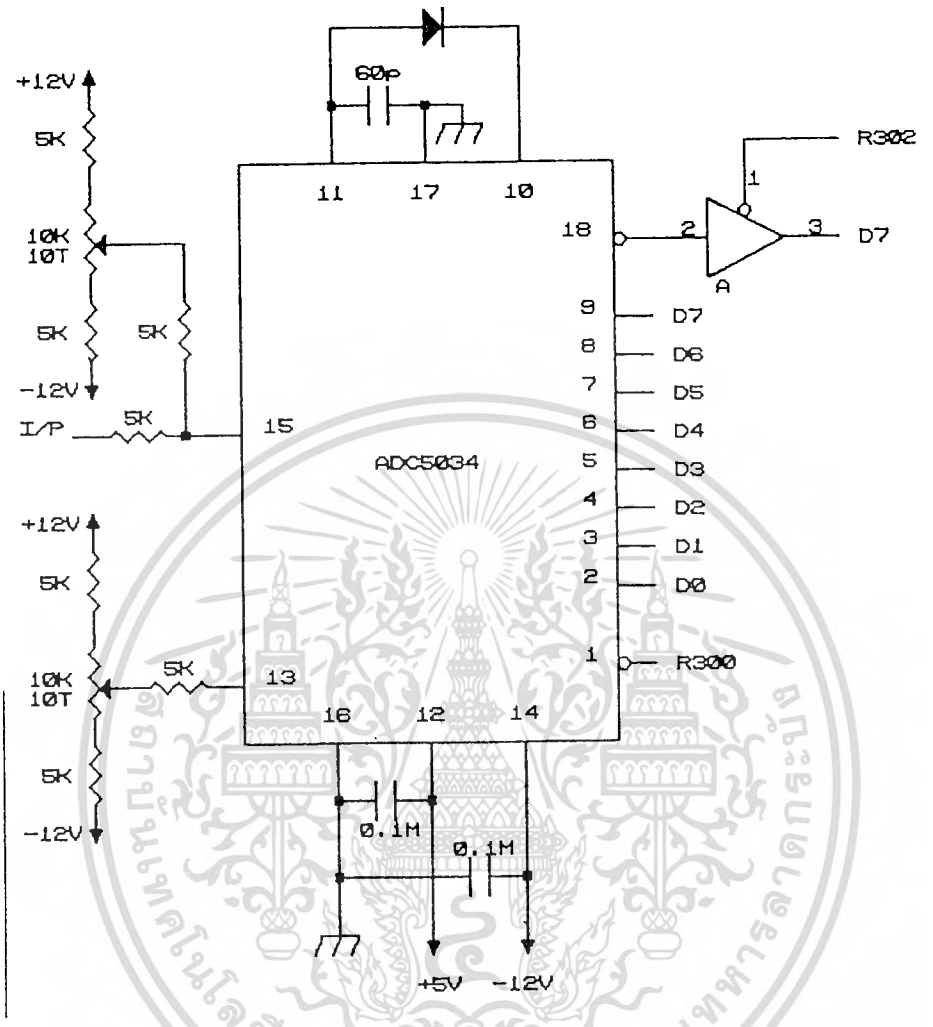
2 ส่วนแปลงสัญญาณ ทำหน้าที่แปลงสัญญาณอนาลอคให้เป็นดิจิตอล โดยใช้เวลา 8 คาบสัญญาณคล๊อก

3 ส่วนเก็บสัญญาณ (OUTPUT BUFFER) จะเก็บสัญญาณดิจิตอลที่ได้ไว้รอจนกว่าคอมพิวเตอร์ ต้องการจึงจะส่งออกไป

การออกแบบ

ในปริญญานิพนธ์นี้ เลือกใช้ สัญญาณคล๊อก ความถี่ 500 KHz จึงใช้เวลาแปลงสัญญาณ 8/[500KHz] เท่ากับ 16 ไมโครวินาที ในการแปลงเราต้องแปลงสัญญาณ 1 ถึง 5 โวลท์ ให้เป็นกระแสไฟฟ้า 0.2 ถึง 1 มิลลิแอมป์ และลดระดับลงเหลือ 0 ถึง 0.8 มิลลิแอมป์ เพื่อส่งให้ 5034 เพื่อใช้ในการแปลงสัญญาณต่อไป นอกจากนี้ความต่างศักย์ที่ใช้อ้างอิง จะต้องปรับให้เหมาะสม จึงต้องมีตัวต้านทานปรับค่าได้อย่างละเอียด สองตัวสำหรับปรับลดระดับกระแส และ ปรับสัญญาณอ้างอิง

การต่อวงจรจะประกอบด้วย วงจรนี้ 2 ชุด โดยชุดแรกจะต่อ ขา OE (OUTPUT ENABLE) เข้ากับสัญญาณ READ 0300 ขา START กับ WRITE 0302 และ DR (DATA READY) กับ READ 0302 เมื่อต้องการจะแปลงสัญญาณ จะต้องส่งสัญญาณให้เริ่มทำงานและรอจนกระทั่ง 5034 ส่งสัญญาณ DR กลับไป คอมพิวเตอร์จึงจะอ่านข้อมูลได้ทำให้คอมพิวเตอร์สามารถเลือกชุดทำงานได้ และจากการออกแบบวงจรดังที่กล่าวทำให้เราได้วงจรของชุดแรก ดังรูปที่ 2.8 (ส่วนวงจรของชุดที่สองจะเหมือนกันแต่ใช้ READ0301 แทน READ0300, READ0303 แทน READ0302 และ WRITE 0303 แทน WRITE0302



รูปที่ 2.8

บทที่3

ซอฟแวร์ (SOFTWARE)

ในปริญญานิพนธ์นี้โปรแกรมทั้งหมดเขียนด้วยภาษาซี (C LANGUAGE) โดยใช้โปรแกรม TURBOC VERSION 1.0 ของบริษัท BORLAND INTERNATIONAL เป็นตัวแปรภาษา ในโปรแกรมมีการควบคุมให้เลือก 4 ชนิด คือ

1) การควบคุมแบบลูปเดียว เป็นตัวควบคุมแบบดิจิตอล PID สองชุด แยกเป็นอิสระกันสามารถเลือกใช้ชุดใดชุดหนึ่งได้ด้วย

2) การควบคุมแบบต่อเนื่อง จะมีเพียงชุดเดียว โดยกำหนดค่าที่ตัวควบคุมตัวแรก แต่ทั้งสองตัวสามารถกำหนดพารามิเตอร์อื่นได้โดยอิสระ

3) การควบคุมอัตราส่วน จะมีชุดเดียวเช่นกัน กำหนดค่าที่ตัวควบคุมตัวแรกและอัตราส่วน ทั้งสองสามารถกำหนดพารามิเตอร์อื่นโดยอิสระเช่นเดียวกัน

4) การควบคุมโดยผู้ควบคุม (MANUAL) ผู้ควบคุมสามารถกำหนดค่าของสัญญาณควบคุมได้โดยตรง

นอกจากนี้ยังสามารถเลือกฟังก์ชันของตัวควบคุมได้อีกคือ

1) แบบต่อกลับค่า (REVERSE) เมื่อเลือกฟังชันก์นี้ค่าสัญญาณควบคุมที่ได้จะกลับกับขนาดของสัญญาณควบคุม คือค่า 0 จะเป็นขนาดกระแส 20 มิลลิแอมป์ (mA) ส่วนค่า 255 จะเป็นกระะแส 4 มิลลิแอมป์

2) ส่วนถอดรากที่สอง (SQUARE ROOT EXTRACTION) เมื่อเลือกฟังก์ชันนี้สัญญาณที่รับเข้ามายังตัวควบคุมจะถูกแปลงเป็นตัวเลข จากนั้นค่านี้จะถูกถอดรากที่สองก่อนจึงจะนำมาใช้ในการคำนวณ

โปรแกรมนี้เขียนขึ้นมาใช้กับจอภาพแบบ COLOUR GRAPHIC (CGA) เนื่องจากโปรแกรมนี้เขียนในลักษณะของกราฟฟิก (GRAPHIC MODE) ดังนั้นถ้าหากใช้กับจอภาพแบบ HERCULES GRAPHIC (HCG) จะต้องใช้โปรแกรม DRIVER เพื่อทำการจำลองการทำงานแบบของจอ CGA ก่อนจึงสามารถใช้โปรแกรมนี้ได้ การทำงานทั้งหมดของโปรแกรมสามารถอธิบายได้ดังรูปที่ 3.1 ส่วนการคำนวณต่างๆแสดงในรูปที่ 3.2 ถึง 3.5

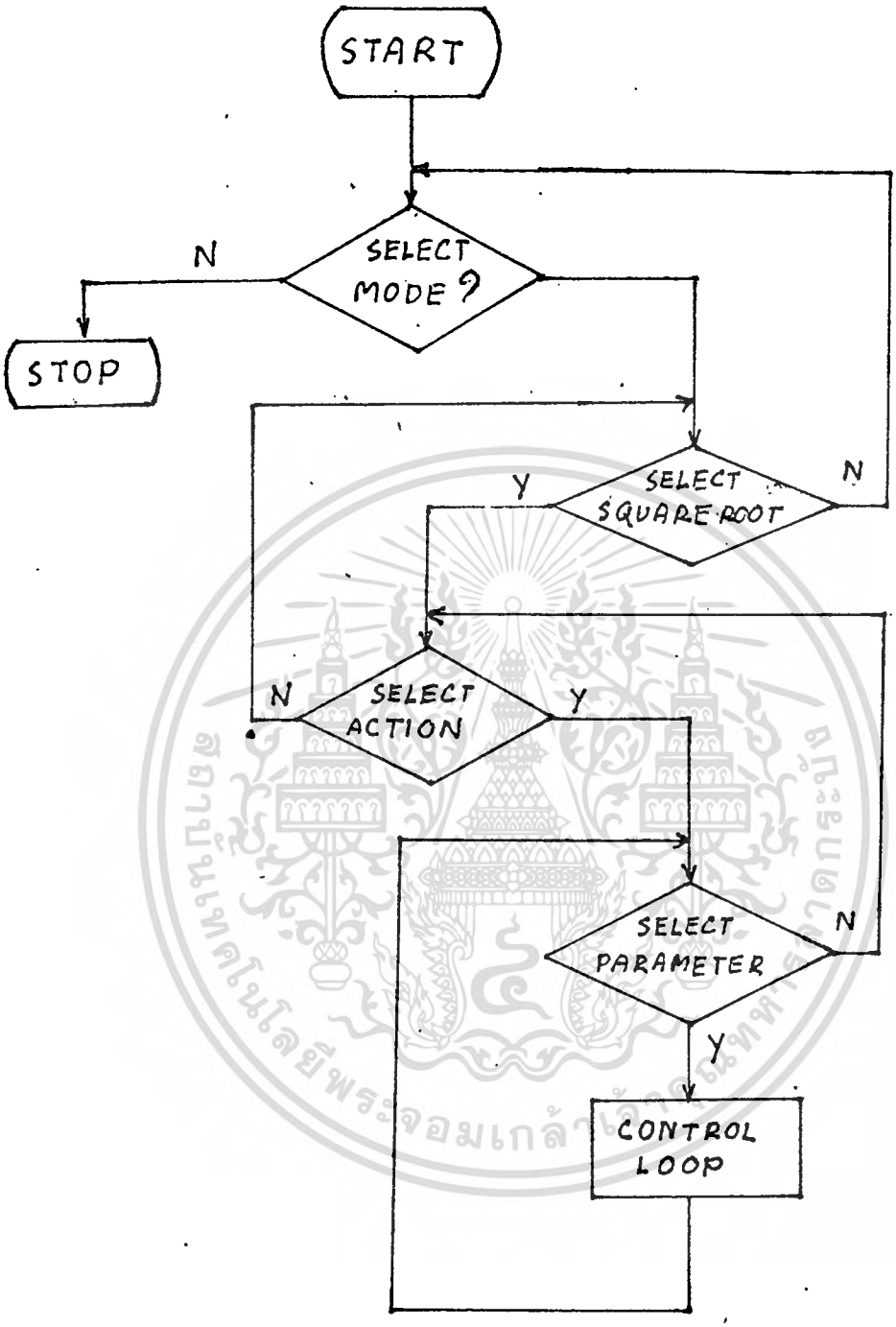
ขั้นตอนการใช้โปรแกรม

1. เลือกลักษณะการทำงาน นอกจากการควบคุมทั้งสี่ชนิดที่กล่าวมาแล้ว ยังสามารถเลือก ทดสอบ (TEST) เพื่อใช้ปรับการทำงานของฮาร์ดแวร์ส่วนรับข้อมูลให้ถูกต้อง (CELEBRATE) ส่วนการปรับส่วนส่งข้อมูลจะปรับโดยใช้การควบคุมโดยผู้ควบคุม

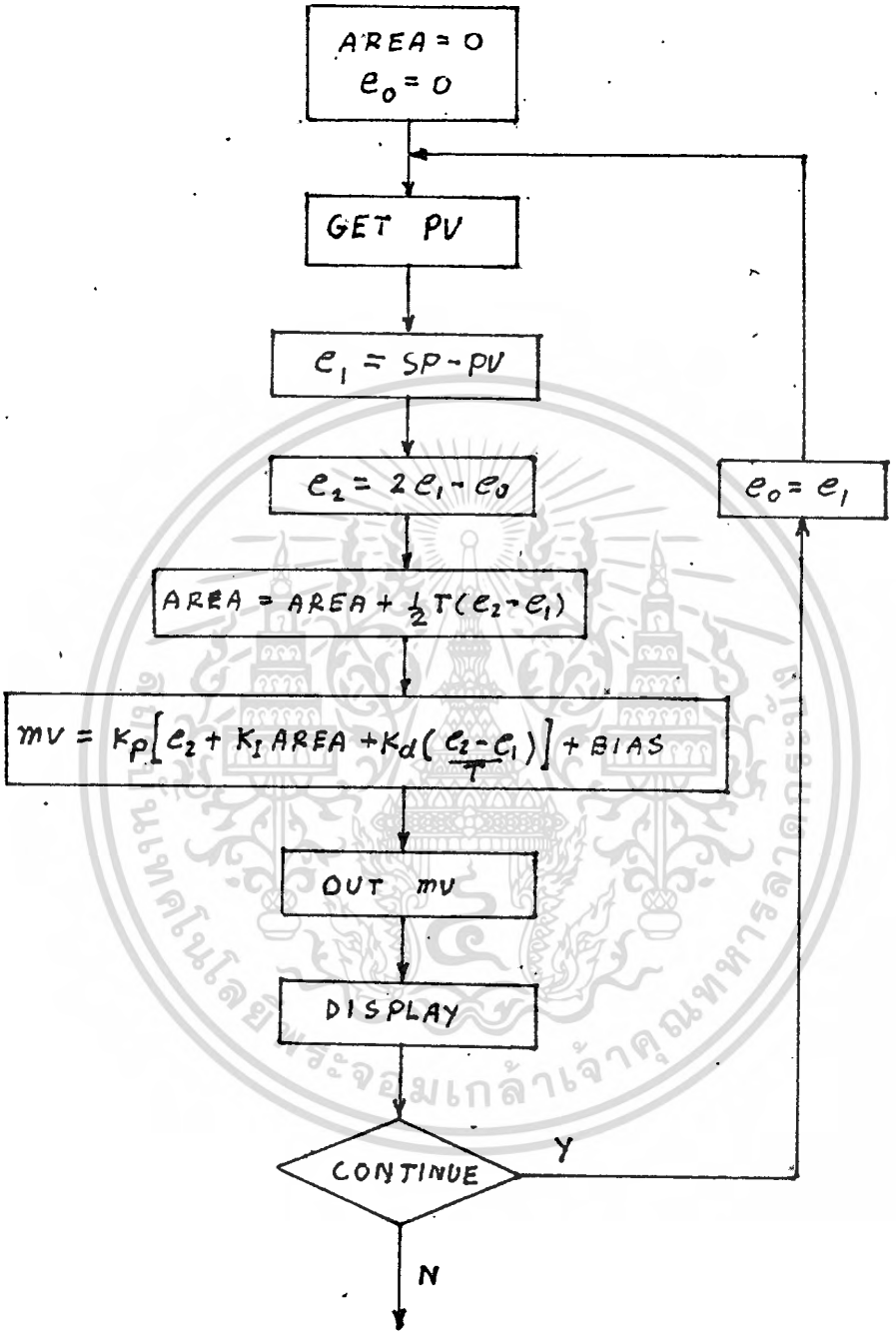
2. เลือกฟังก์ชั่นถอดราก ใช้เลือกว่าต้องการให้คอมพิวเตอร์ ถอดรากที่สองของสัญญาณที่รับมา ก่อนที่จะนำมาคำนวณหรือไม่ ซึ่งตัวควบคุมทั้งสองนี้สามารถเลือกได้โดยอิสระ

3. เลือกลักษณะการแปลงข้อมูล สามารถเลือกได้ว่า ให้กระแส แปรตาม (NORMAL) หรือแปรผกผัน (REVERSE) กับค่าที่คำนวณได้ (คือเลือกให้ค่าสูงสุดเป็น 4 หรือ 20 มิลลิแอมป์) ตัวควบคุมทั้งสองจะเลือกได้โดยอิสระเช่นกัน

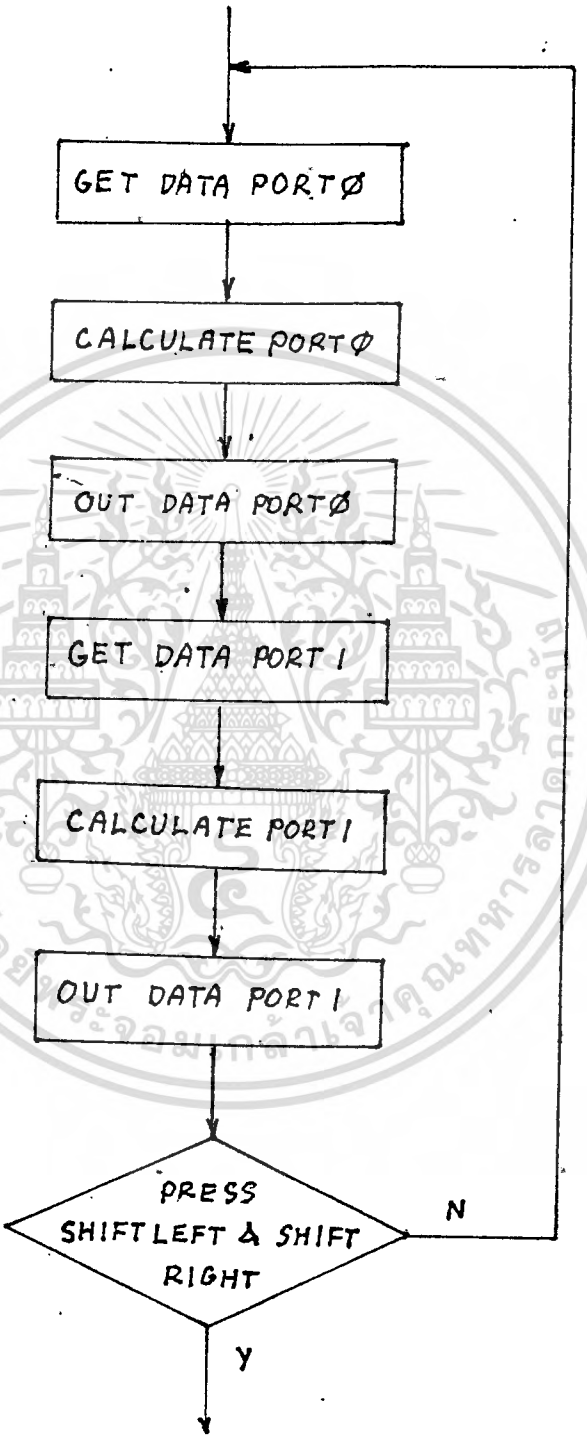
4. กำหนดพารามิเตอร์ต่าง ๆ ในการควบคุม จากนั้นจึงสั่งให้โปรแกรมเริ่มทำการควบคุมด้วยพารามิเตอร์ที่กำหนด ขณะที่โปรแกรมกำลังทำงานอยู่นั้น สามารถหยุดโปรแกรมได้โดยกด [SHIFT LEFT] และ [SHIFT RIGHT] พร้อมกัน



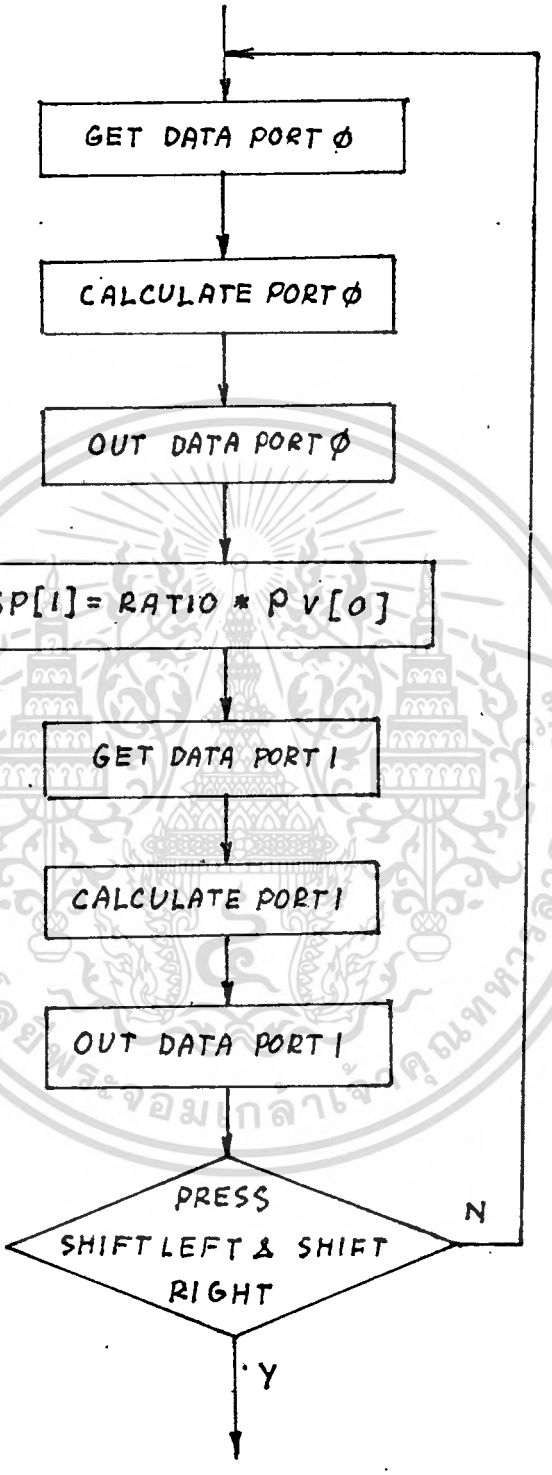
รูปที่ ๓



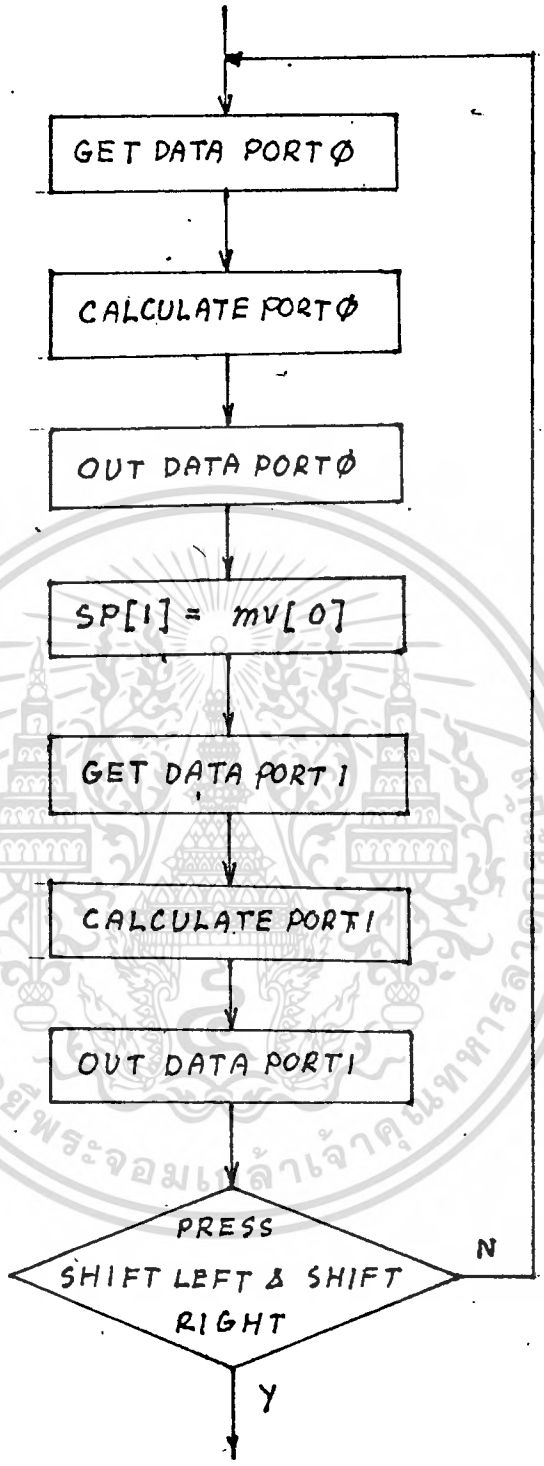
รูปที่ 3.2



รูปที่ 3.3



รูปที่ 3.4



รูปที่ 3.5

บทที่ 4
การทดลอง

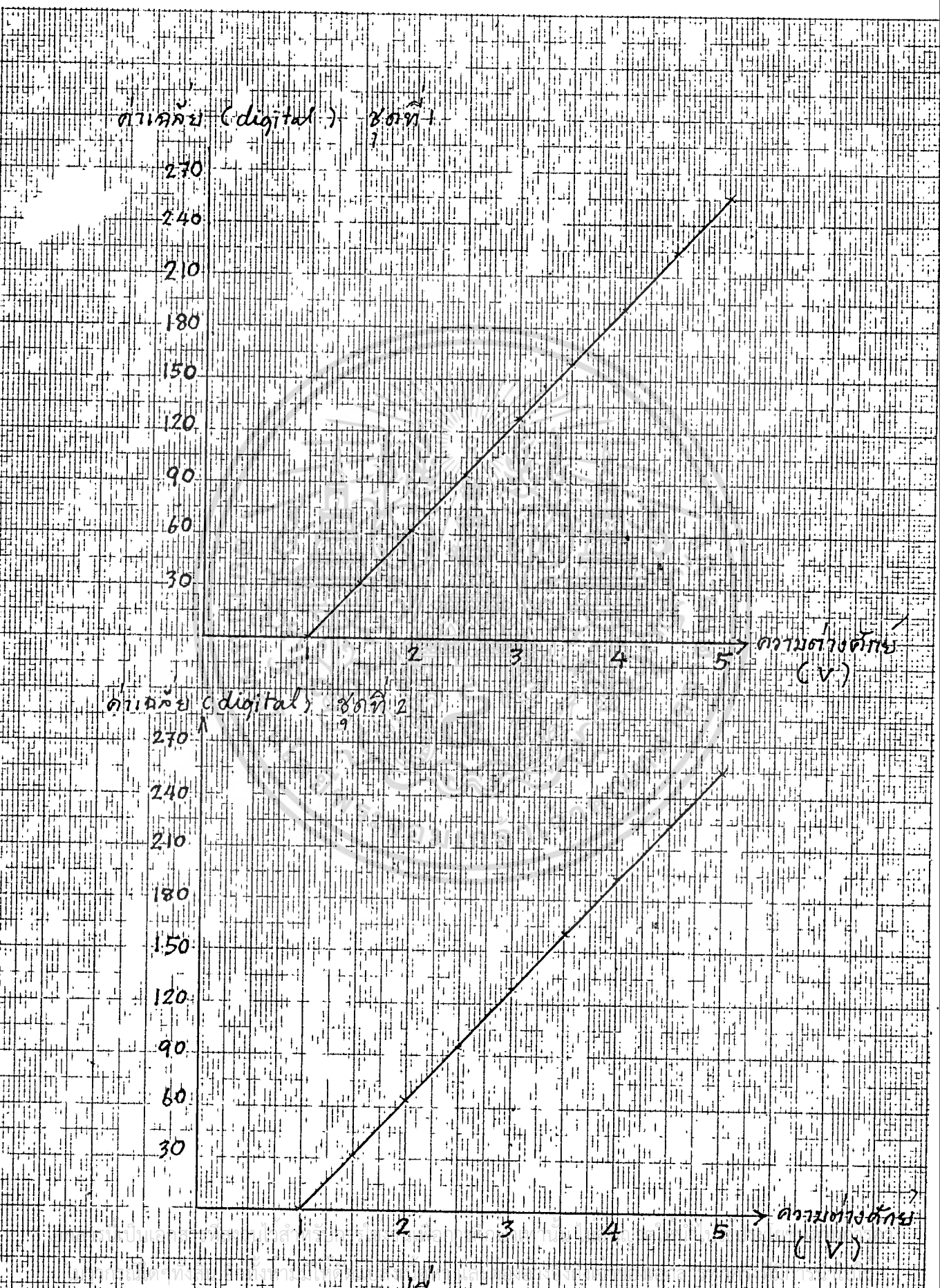
การทดลองการทำงานของฮาร์ดแวร์ แบ่งเป็นสองส่วนคือ ทางด้านรับและส่งข้อมูล ทางด้านรับข้อมูลทดลองแปลงสัญญาณจากอนาลอคเป็นดิจิตอล ทำโดยใช้แหล่งจ่ายพลังงาน (POWER SUPPLY) จ่ายความต่างศักย์เพื่อจำลองสัญญาณจากเครื่องมือวัดของระบบ ผลจากการอ่านค่าดิจิตอลที่ความต่างศักย์ต่างๆ แสดงในตารางที่ 4.1 และกราฟแสดงความสัมพันธ์ของค่าเฉลี่ย แสดงในรูปที่ 4.1

ทางด้านส่งข้อมูล ทดลองแปลงสัญญาณจากดิจิตอลเป็นอนาลอค โดยทดลองจ่ายกระแสไฟฟ้าให้กับความต้านทาน 250 โอห์ม ซึ่งเป็นค่าความต้านทานมาตรฐานของระบบ (STANDARD IMPEDANCE) ผลจากการอ่านค่ากระแสไฟฟ้าที่ค่าดิจิตอลต่างๆ แสดงในตารางที่ 4.2 และ กราฟแสดงความสัมพันธ์ของค่าเฉลี่ย แสดงในรูปที่ 4.2

การทดลองเมื่อใช้โปรแกรมควบคุม จะมีความถี่ในการอ่านข้อมูล ประมาณ 14 Hz สำหรับการใช้งานชุดเดียว และ 7 Hz สำหรับการใช้งานสองเดียว

ตารางที่ 4.1 แสดงการทดสอบส่วนรับข้อมูล

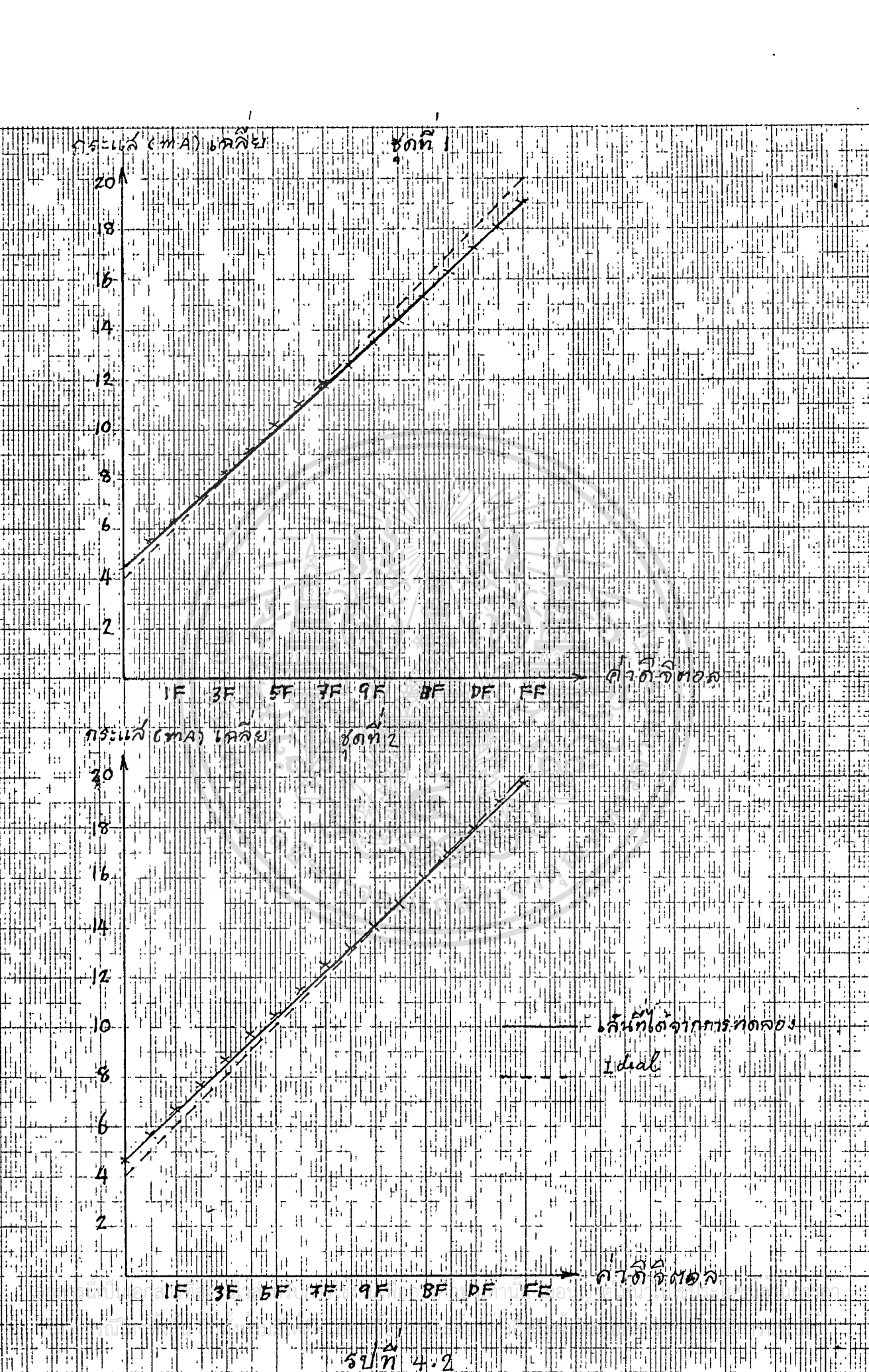
โวลท์	ชุดที่ 1			ชุดที่ 2		
	อ่านขึ้น	อ่านลง	เฉลี่ย	อ่านขึ้น	อ่านลง	เฉลี่ย
1.0	0	0	0.0	0	0	0.0
1.5	31	31	31.0	31	31	31.0
2.0	62	61	61.5	64	64	64.0
2.5	93	94	93.5	95	95	95.0
3.0	127	127	127.0	128	128	128.0
3.5	159	159	159.0	161	159	160.0
4.0	191	192	191.5	193	191	192.0
4.5	225	223	224.0	225	224	224.5
5.0	255	255	255.0	255	255	255.0


ค่าเฉลี่ย (digital) ของที่ 1
270
240
210
180
150
120
90
60
30
1
2
3
4
5
ความต่างศักย์ (V)
ค่าเฉลี่ย (digital) ของที่ 2
270
240
210
180
150
120
90
60
30
1
2
3
4
5
ความต่างศักย์ (V)

รูปที่ 4.1

ตารางที่ 4.2 แสดงการทดสอบส่วนส่งข้อมูล

ดิจิตอล	ชุดที่ 1			ชุดที่ 2		
	อ่านขึ้น	อ่านลง	เฉลี่ย	อ่านขึ้น	อ่านลง	เฉลี่ย
00	4.5	4.5	4.50	4.5	5.0	4.75
0F	5.5	5.5	5.50	5.5	6.0	5.75
1F	6.5	6.0	6.25	6.5	7.0	6.75
2F	7.5	7.0	7.25	7.5	8.0	7.75
3F	8.5	8.0	8.25	8.5	9.0	8.75
4F	9.5	9.0	9.25	9.5	10.0	9.75
5F	10.5	10.0	10.25	10.5	10.5	10.50
6F	11.2	10.8	11.00	11.5	11.5	11.50
7F	12.0	11.5	11.75	12.5	12.5	12.50
8F	13.0	12.2	12.60	13.2	13.0	13.20
9F	14.0	13.0	13.50	14.0	14.0	14.00
AF	14.8	14.0	14.40	15.0	15.0	15.00
BF	15.5	15.0	15.25	16.0	16.0	16.00
CF	16.5	16.0	16.25	17.0	17.0	17.00
DF	17.5	17.0	17.25	18.0	18.0	18.00
EF	18.0	18.0	18.00	19.0	19.0	19.00
FF	19.0	19.0	19.00	19.5	20.0	19.75


กระแส (mA) เฉลี่ย
ชุดที่ 1
20
18
16
14
12
10
8
6
4
2
1F 3F 5F 7F 9F BF DF FF
ค่าดิจิตอล
กระแส (mA) เฉลี่ย
ชุดที่ 2
เส้นที่ได้จากการทดลอง
Ideal
ค่าดิจิตอล

รูปที่ 4.2

บทที่ 5
สรุป

จากการทดลองฮาร์ดแวร์ จะเห็นได้ว่า ได้ผลใกล้เคียงกับทางทฤษฎีมาก แต่อย่างไรก็ตามยังมีข้อผิดพลาดเกิดขึ้น ซึ่งคาดว่าเกิดมาจากสาเหตุสำคัญ คือ

1 เครื่องมือวัดและการอ่านค่าข้อมูล เครื่องมือที่ใช้วัดทั้งกระแสและความต่างศักย์ ใช้ มัลติมิเตอร์ชนิดเข็ม ของ SANWA ซึ่งอ่านค่าได้ยากและต้องใช้การประมาณค่าในส่วนทศนิยม ทำให้ผิดพลาดได้ง่าย

2 การปรับค่าต่างๆของฮาร์ดแวร์ ทำได้ยากและผิดพลาดได้ง่าย จึงยังมีความคลาดเคลื่อนอยู่เล็กน้อย

อย่างไรก็ตาม ข้อผิดพลาดขนาดนี้สามารถยอมรับได้ ดังจากการทดลองควบคุมระดับน้ำสามารถควบคุมได้ตามทฤษฎี ซึ่งถ้าหากใช้ เพิ่มส่วนช่วยคำนวนทางคณิตศาสตร์ (MATH COPROCESSOR) ทำให้ความถี่ในการอ่านข้อมูลเพิ่มขึ้น คาดว่าสามารถจะใช้ควบคุมระบบที่มีการเปลี่ยนแปลงรวดเร็วกว่านี้ได้

ภาคผนวก

```
/*
Program   Programmable Controller
By        Kanokchat Liawannata
Date      March 1988
Compiler  Turbo C
Hardware  - IBM PC-XT with color graphic card
          - AD/DA card

Overview
        This program is base on digital PID controller algorithm,
mv = K*{ e(t) + Ki*Ae + Kd*{e(t)-e(t-1)}/T }. there are four control
mode i.e. single loop, cascade, ratio and manual with option
        - Input signal pass Square root extractor or not
        - Reverse or normal operation

        This program is operate on graphic mode (CGA) but can use
driver to simulate on monochrome monitor (HGC). While running the
control loop, user can press (shift left)and(shift right) to quit
the loop to change command or reset parameter.                  */



#define F1      0x3B00
#define F2      0x3C00
#define F3      0x3D00
#define F4      0x3E00
#define F5      0x3F00
#define F6      0x4000
#define F7      0x4100
#define F8      0x4200
#define F9      0x4300
#define F10     0x4400
#define ESC     0x011B
#define SPACE   0x3920
#define ENTER   0x1C0D

#define SLOOP0  0
#define SLOOP1  1
#define SLOOP2  2
#define CASCADE 3
#define RATIO   4
#define TEST    5

#define MANUAL  1
#define AUTO    0
#define REVERSE 1
#define NORMAL  0
#define ON      1
#define OFF     0
#define YES     1
#define NO      0
#define TEXT    2
#define GRAPHIC 6
#define EXIT    3
```

```
void    outportb(int iport,char byte);
int     inportb(int port);
void    CurPos(char x,char y);
void    Cl_Window(char x1, char y1, char x2, char y2);
void    Screen(char S_Mode);
void    Hline(int Left, char Length);
void    Vline(int Bottom, char Hight, char Data);
void    BarGraph(int Bottom, char Hight);
char    Get_Data(int Port);
void    Table(void);
char    Select_Mode(void);
char    Select_Sroot(void);
char    Select_Action(void);
char    Select_Parameter(void);
void    Single_Loop(void);
void    Cascade_Loop(void);
void    Ratio_Loop(void);
void    Test_Loop(char Port);
void    Manual(char Port);

#define MK_FP(seg,ofs)  ((void far *)(((unsigned long)(seg) << 16) | (ofs)))
#define Out_Data(port,byte)     outportb(port,byte)



float   T[2]            = {  0.5,  0.5 };
float   SP[2]           = { 50.0, 50.0 };
float   Kp[2]           = {  5.0,  5.0 };
float   Ki[2]           = { 10.0, 10.0 };
float   Kd[2]           = {  0.0,  0.0 };
float   Bias[2]         = { 10.0, 10.0 };
float   Ratio;
char    Mode;
char    Action[2];
char    Sroot[2];



main()
{
char    Exit_M,Exit_S,Exit_A,Exit_P;

    Screen(GRAPHIC);
    Table();
    for(Exit_M=NO; Exit_M==NO;)
    {
        Exit_M = Select_Mode();
        if(Exit_M == NO)
        for(Exit_S=NO; Exit_S==NO;)
        {
            Exit_S = Select_Sroot();
            if(Exit_S == NO)
            for(Exit_A=NO; Exit_A==NO;)
```

```
            {
                Exit_A = Select_Action();
                if(Exit_A == NO)
                for(Exit_P=NO; Exit_P==NO;)
                {
                    Exit_P = Select_Parameter();
                    if(Exit_P == NO)
                    switch(Mode)
                    {
                        case SLOOP0     :
                        case SLOOP1     :
                        case SLOOP2     : Single_Loop();  break;
                        case CASCADE    : Cascade_Loop(); break;
                        case RATIO      : Ratio_Loop();   break;
                    }
                }
            }
        }
    }
    Screen(TEXT);
}



char Select_Mode(void)
{
int     Cmd;
char    OK;

Cl_Window(28,6,77,22);
CurPos(37,8);  printf("    OPERATION MODE      ");
CurPos(37,10); printf("F1  Single  loop port 0 ");
CurPos(37,11); printf("F2  Single  loop port 1 ");
CurPos(37,12); printf("F3  Single  loop 2 ports");
CurPos(37,13); printf("F4  Cascade control     ");
CurPos(37,14); printf("F5  Ratio   control     ");
CurPos(37,15); printf("F7  Manual  port 0      ");
CurPos(37,16); printf("F8  Manual  port 1      ");
CurPos(37,17); printf("F9  Test    port 0      ");
CurPos(37,18); printf("F10 Test    port 1      ");
CurPos(37,20); printf("ESC EXIT                ");

for(OK=NO;OK==NO;)
{
    Cmd = bioskey(0);
    switch(Cmd)
    {
        case F1  : Mode = SLOOP0; OK = YES;
                   CurPos(5,5); printf("Single loop port0   "); break;
        case F2  : Mode = SLOOP1; OK = YES;
                   CurPos(5,5); printf("Single loop port1   "); break;
        case F3  : Mode = SLOOP2; OK = YES;
                   CurPos(5,5); printf("Single loop 2ports  ");break;
        case F4  : Mode = CASCADE;OK = YES;
```

```
                CurPos(5,5); printf("Cascade control     ");break;
        case F5  : Mode = RATIO;  OK = YES;
                CurPos(37,17);printf("Ratio : "); scanf("%f",&Ratio);
                CurPos(5,5); printf("Ratio control %5.2f",Ratio); break;
        case F7  : Manual(0x300); break;
        case F8  : Manual(0x301); break;
        case F9  : Test_Loop(0x300); break;
        case F10 : Test_Loop(0x301); break;
        case ESC : OK = YES; break;
    }
}
return((Cmd == ESC) ? YES : NO);
}


char Select_Sroot(void)
{
int     Cmd;
char    OK;

Cl_Window(28,6,77,22);
CurPos(37,8);  printf("        SQUARE ROOT        ");
CurPos(37,10); printf("F1    Port0 OFF   Port1 OFF");
CurPos(37,11); printf("F2    Port0 ON    Port1 OFF");
CurPos(37,12); printf("F3    Port0 OFF   Port1 ON ");
CurPos(37,13); printf("F4    Port0 ON    Port1 ON ");
CurPos(37,20); printf("ESC   EXIT                 ");

for(OK=NO;OK==NO;)
{
    Cmd = bioskey(0);
    switch(Cmd)
    {
        case F1  : Sroot[0]=OFF; Sroot[1]=OFF; OK = YES;
                CurPos(5,8);  printf("Square Root  OFF  ");
                CurPos(5,17); printf("Square Root  OFF  ");break;
        case F2  : Sroot[0]=ON;  Sroot[1]=OFF; OK = YES;
                CurPos(5,8);  printf("Square Root  ON   ");
                CurPos(5,17); printf("Square Root  OFF  ");break;
        case F3  : Sroot[0]=OFF; Sroot[1]=ON;  OK = YES;
                CurPos(5,8);  printf("Square Root  OFF  ");
                CurPos(5,17); printf("Square Root  ON   ");break;
        case F4  : Sroot[0]=ON;  Sroot[1]=ON;  OK = YES;
                CurPos(5,8);  printf("Square Root  ON   ");
                CurPos(5,17); printf("Square Root  ON   ");break;
        case ESC : OK = YES;     break;
    }
}
CurPos(5,7);  printf("Port 0 ");
CurPos(5,16); printf("Port 1 ");
return((Cmd == ESC) ? YES : NO);
}


char Select_Action(void)
```

```
{
int     Cmd;
char    OK;

Cl_Window(28,6,77,22);
CurPos(37,8);  printf("             PORT ACTION          ");
CurPos(37,10); printf("F1    Port0 Normal     Port1 Normal ");
CurPos(37,11); printf("F2    Port0 Reverse    Port1 Normal ");
CurPos(37,12); printf("F3    Port0 Normal     Port1 Reverse");
CurPos(37,13); printf("F4    Port0 Reverse    Port1 Reverse");
CurPos(37,20); printf("ESC   EXIT                         ");

for(OK=NO;OK==NO;)
{
    Cmd = bioskey(0);
    switch(Cmd)
    {
        case F1  : Action[0]=NORMAL; Action[1]=NORMAL; OK = YES;
                   CurPos(5,9);  printf("Action      Normal ");
                   CurPos(5,18); printf("Action      Normal ");break;
        case F2  : Action[0]=REVERSE;Action[1]=NORMAL; OK = YES;
                   CurPos(5,9);  printf("Action      Reverse");
                   CurPos(5,18); printf("Action      Normal ");break;
        case F3  : Action[0]=NORMAL; Action[1]=REVERSE;OK = YES;
                   CurPos(5,9);  printf("Action      Reverse");
                   CurPos(5,18); printf("Action      Normal ");break;
        case F4  : Action[0]=REVERSE;Action[1]=REVERSE;OK = YES;
                   CurPos(5,9);  printf("Action      Reverse");
                   CurPos(5,18); printf("Action      Reverse");break;
        case ESC : OK = YES; break;
    }
}
return((Cmd == ESC) ? YES : NO);
}


char Select_Parameter(void)
{
int     Cmd;
char    OK;

Cl_Window(28,6,77,22);
CurPos(5,10);   printf("SP         :%6.2f ",SP[0]);
CurPos(5,19);   printf("SP         :%6.2f ",SP[0]);
CurPos(5,11);   printf("Kp         :%6.2f ",Kp[0]);
CurPos(5,20);   printf("Kp         :%6.2f ",Kp[0]);
CurPos(5,12);   printf("Ki         :%6.2f ",Ki[0]);
CurPos(5,21);   printf("Ki         :%6.2f ",Ki[0]);
CurPos(5,13);   printf("Kd         :%6.2f ",Kd[0]);
CurPos(5,22);   printf("Kd         :%6.2f ",Kd[0]);
CurPos(5,14);   printf("Bias       :%6.2f ",Bias[0]);
CurPos(5,23);   printf("Bias       :%6.2f ",Bias[0]);
CurPos(37,8);   printf("          PORT ACTION           ");
CurPos(37,10);  printf("F1   Port0 SP      F2  Port1 SP  ");
CurPos(37,11);  printf("F3   Port0 Kp      F4  Port1 Kp  ");
```

```
CurPos(37,12); printf("F5   Port0 Ki      F6  Port1 Ki  ");
CurPos(37,13); printf("F7   Port0 Kd      F8  Port1 Kd  ");
CurPos(37,14); printf("F9   Port0 Bias    F10 Port1 Bias");
CurPos(37,19); printf("ENTER START CONTROLLER          ");
CurPos(37,20); printf("ESC   EXIT                      ");

for(OK=NO;OK==NO;)
{
    Cmd = bioskey(0);
    switch(Cmd)
    {
        case F1    : CurPos(17,10);printf("       ");
                     CurPos(17,10);scanf("%f",&SP[0]);
                     CurPos(17,10);printf("%6.2f",SP[0]);break;
        case F2    : CurPos(17,19);printf("       ");
                     CurPos(17,19);scanf("%f",&SP[1]);
                     CurPos(17,19);printf("%6.2f",SP[1]);break;
        case F3    : CurPos(17,11);printf("       ");
                     CurPos(17,11);scanf("%f",&Kp[0]);
                     CurPos(17,11);printf("%6.2f",Kp[0]);break;
        case F4    : CurPos(17,20);printf("       ");
                     CurPos(17,20);scanf("%f",&Kp[1]);
                     CurPos(17,20);printf("%6.2f",Kp[1]);break;
        case F5    : CurPos(17,12);printf("       ");
                     CurPos(17,12);scanf("%f",&Ki[0]);
                     CurPos(17,12);printf("%6.2f",Ki[0]);break;
        case F6    : CurPos(17,21);printf("       ");
                     CurPos(17,21);scanf("%f",&Ki[1]);
                     CurPos(17,21);printf("%6.2f",Ki[1]);break;
        case F7    : CurPos(17,13);printf("       ");
                     CurPos(17,13);scanf("%f",&Kd[0]);
                     CurPos(17,13);printf("%6.2f",Kd[0]);break;
        case F8    : CurPos(17,22);printf("       ");
                     CurPos(17,22);scanf("%f",&Kd[1]);
                     CurPos(17,22);printf("%6.2f",Kd[1]);break;
        case F9    : CurPos(17,14);printf("       ");
                     CurPos(17,14);scanf("%f",&Bias[0]);
                     CurPos(17,14);printf("%6.2f",Bias[0]);break;
       case F10    : CurPos(17,23);printf("       ");
                     CurPos(17,23);scanf("%f",&Bias[1]);
                     CurPos(17,23);printf("%6.2f",Bias[1]);break;
        case ENTER :
        case ESC   : OK = YES;
    }
}
return((Cmd == ESC) ? YES : NO);
}


void Table(void)
{
        Vline(7920,196,192);
        Vline(7999,196,3);
        Vline(7946,169,3);
        Hline(80,80);
```

```
        Hline(1200,80);
        Hline(7920,80);
}


void Test_Loop(void)
{
char Loop = YES;
char Data;

while( Loop != EXIT)
{
        Data = Get_Data(Port);
        CurPos(37,17);printf("Data : ,%x  ",Data);
        Loop = bioskey(2);
        Loop &= 3;
}
}


void Manual(char Port)
{
char Data;

        CurPos(37,17);printf("Data : "); scanf("%d",&Data);
        Out_Data(Port,Data);
        CurPos(37,17);printf("              ");
}

void Single_Loop(void)
{
char    Loop=0;
char    display;
char    indata[2],outdata[2];
float   e0[2],e1[2],e2[2],xi[2],mv[2],pv[2];


Cl_Window(28,6,77,22);

if(Mode != SLOOP1)
{
        e0[0]    = 0.0;
        display  = SP[0];
        BarGraph(6760,display);
        CurPos(40,22); printf("SP  PV  MV");
        CurPos(9,9); printf("SP %6.2f",SP[0]);
        Loop = YES;
}

if(Mode != SLOOP0)
{
        e0[1]    =0.0;
        display  = SP[1];
        BarGraph(6778,display);
        CurPos(58,22); printf("SP  PV  MV");
```

```
        CurPos(9,14); printf("SP %6.2f",SP[1]);
        Loop = YES;
}

while(Loop != EXIT)
{

    if(Mode != SLOOP1)
    {
        indata[0]       = Get_Data(0x300);
        pv[0]           = indata[0] / 2.56;
        if(Sroot[0]==ON)
            e1[0]       = SP[0] - sqrt(pv[0]);
        else
            e1[0]       = SP[0] - pv[0];
        e2[0]           = e1[0]+e1[0]-e0[0];

        if(Ki[0]>0.0)
        {
            xi[0] = xi[0]+(e1[0]+e2[0])*T[0]/2.0*Ki[0];
            if (xi[0]>100.0) xi[0]=100.0;
            if (xi[0]<0.0)   xi[0]=0.0;
        }
        else xi[0]=0.0;

        mv[0]=Kp[0]*(e2[0]+xi[0]+Kd[0]/T[0]*(e2[0]-e1[0]))+Bias[0];
        if(mv[0] > 100.0) mv[0] = 100.0;
        if(mv[0] < 0.0  ) mv[0] = 0.0;

        display         = pv[0];
        outdata[0]      = mv[0];
        BarGraph(6764,display);
        BarGraph(6768,outdata[0]);
        CurPos(9,10); printf("PV %6.2f",pv[0]);
        CurPos(9,11); printf("MV %6.2f",mv[0]);

        outdata[0] *= 2.56;
        if(Action[0]==REVERSE) outdata[0] = 255 - outdata[0];
        Out_Data(0x300,outdata[0]);
        e0[0] = e1[0];
    }



    if(Mode != SLOOP0)
    {
        indata[1]       = Get_Data(0x301);
        pv[1]           = indata[1] / 2.56;
        if(Sroot[1]==ON)
            e1[1]       = SP[1] - sqrt(pv[1]);
        else
            e1[1]       = SP[1] - pv[1];
        e2[1]           = e1[1]+e1[1]-e0[1];

        if (Ki[1]>0.0)
```

```
        {
            xi[1] = xi[1]+(e1[1]+e2[1])*T[1]/2*Ki[1];
            if (xi[1]>100.0) xi[1]=100.0;
            if (xi[1]<0.0)   xi[1]=0.0;
        }
        else xi[1]=0.0;

        mv[1]=Kp[1]*(e2[1]+xi[1]+Kd[1]/T[1]*(e2[1]-e1[1]))+Bias[1];
        if(mv[1] > 100.0) mv[1] = 100.0;
        if(mv[1] < 0.0  ) mv[1] = 0.0;

        display = pv[1];
        outdata[1] = mv[1];
        BarGraph(6782,display);
        BarGraph(6786,outdata[1]);
        CurPos(9,15); printf("PV %6.2f",pv[1]);
        CurPos(9,16); printf("MV %6.2f",mv[1]);

        outdata[1] *= 2.56;
        if(Action[1]==REVERSE) outdata[1] = 255 - outdata[1];
        Out_Data(0x301,outdata[1]);
        e0[1] = e1[1];
    }
    Loop = bioskey(2);
    Loop &= 3;
}

if(Ki[0]!=0) Bias[0]=mv[0];
if(Ki[1]!=0) Bias[1]=mv[1];
}


void Cascade_Loop(void)
{
char    Loop=0;
char    display;
char    indata[2],outdata[2];
float   e0[2],e1[2],e2[2],xi[2],mv[2],pv[2];


Cl_Window(28,6,77,22);

        e0[0]    = 0.0;
        display  = SP[0];
        BarGraph(6760,display);
        CurPos(40,22); printf("SP  PV  MV");
        CurPos(9,9); printf("SP %6.2f",SP[0]);
        Loop = YES;
        e0[1]    =0.0;
        display  = SP[1];
        BarGraph(6778,display);
        CurPos(58,22); printf("SP  PV  MV");
        CurPos(9,14); printf("SP %6.2f",SP[1]);

    while(Loop != EXIT)
```

```
{
        indata[0]       = Get_Data(0x300);
        pv[0]           = indata[0] / 2.56;
        if(Sroot[0]==ON)
            e1[0]       = SP[0] - sqrt(pv[0]);
        else
            e1[0]       = SP[0] - pv[0];
        e2[0]           = e1[0]+e1[0]-e0[0];

        if(Ki[0]>0.0)
        {
            xi[0] = xi[0]+(e1[0]+e2[0])*T[0]/2.0*Ki[0];
            if (xi[0]>100.0) xi[0]=100.0;
            if (xi[0]<0.0)   xi[0]=0.0;
        }
        else xi[0]=0.0;

        mv[0]=Kp[0]*(e2[0]+xi[0]+Kd[0]/T[0]*(e2[0]-e1[0]))+Bias[0];
        if(mv[0] > 100.0) mv[0] = 100.0;
        if(mv[0] < 0.0  ) mv[0] = 0.0;

        display         = pv[0];
        outdata[0]      = mv[0];
        BarGraph(6764,display);
        BarGraph(6768,outdata[0]);
        CurPos(9,10); printf("PV %6.2f",pv[0]);
        CurPos(9,11); printf("MV %6.2f",mv[0]);

        outdata[0]      *= 2.56;
        SP[1]           = pv[0];
        if(Action[0]==REVERSE) outdata[0] = 255 - outdata[0];
        Out_Data(0x300,outdata[0]);
        e0[0] = e1[0];

        indata[1]       = Get_Data(0x301);
        pv[1]           = indata[1] / 2.56;
        if(Sroot[1]==ON)
            e1[1]       = SP[1] - sqrt(pv[1]);
        else
            e1[1]       = SP[1] - pv[1];
        e2[1]           = e1[1]+e1[1]-e0[1];

        if (Ki[1]>0.0)
        {
            xi[1] = xi[1]+(e1[1]+e2[1])*T[1]/2*Ki[1];
            if (xi[1]>100.0) xi[1]=100.0;
            if (xi[1]<0.0)   xi[1]=0.0;
        }
        else xi[1]=0.0;

        mv[1]=Kp[1]*(e2[1]+xi[1]+Kd[1]/T[1]*(e2[1]-e1[1]))+Bias[ :
        if(mv[1] > 100.0) mv[1] = 100.0;
        if(mv[1] < 0.0  ) mv[1] = 0.0;
```

```
        display = pv[1];
        outdata[1] = mv[1];
        BarGraph(6782,display);
        BarGraph(6786,outdata[1]);
        CurPos(9,15); printf("PV %6.2f",pv[1]);
        CurPos(9,16); printf("MV %6.2f",mv[1]);

        outdata[1] *= 2.56;
        if(Action[1]==REVERSE) outdata[1] = 255 - outdata[1];
        Out_Data(0x301,outdata[1]);
        e0[1] = e1[1];

        Loop = bioskey(2);
        Loop &= 3;
}

if(Ki[0]!=0) Bias[0]=mv[0];
if(Ki[1]!=0) Bias[1]=mv[1];
}

void Ratio_Loop(void)
{
char    Loop=0;
char    display;
char    indata[2],outdata[2];
float   e0[2],e1[2],e2[2],xi[2],mv[2],pv[2];


Cl_Window(28,6,77,22);

        e0[0]    = 0.0;
        display  = SP[0];
        BarGraph(6760,display);
        CurPos(40,22); printf("SP  PV  MV");
        CurPos(9,9); printf("SP %6.2f",SP[0]);
        Loop = YES;
        e0[1]    =0.0;
        display  = SP[1];
        BarGraph(6778,display);
        CurPos(58,22); printf("SP  PV  MV");
        CurPos(9,14); printf("SP %6.2f",SP[1]);

while(Loop != EXIT)
.

        indata[0].      = Get_Data(0x300);
        pv[0]           = indata[0] / 2.56;
        if(Sroot[0]==ON)
            e1[0]       = SP[0] - sqrt(pv[0]);
        else
            e1[0]       = SP[0] - pv[0];
        e2[0]           = e1[0]+e1[0]-e0[0];

        if(Ki[0]>0.0)
        {
```

```
    xi[0] = xi[0]+(e1[0]+e2[0])*T[0]/2.0*Ki[0];
    if (xi[0]>100.0) xi[0]=100.0;
    if (xi[0]<0.0)   xi[0]=0.0;
}
else xi[0]=0.0;

mv[0]=Kp[0]*(e2[0]+xi[0]+Kd[0]/T[0]*(e2[0]-e1[0]))+Bias[0];
if(mv[0] > 100.0) mv[0] = 100.0;
if(mv[0] < 0.0  ) mv[0] = 0.0;

display         = pv[0];
outdata[0]      = mv[0];
BarGraph(6764,display);
BarGraph(6768,outdata[0]);
CurPos(9,10); printf("PV %6.2f",pv[0]);
CurPos(9,11); printf("MV %6.2f",mv[0]);

outdata[0]      *= 2.56;
if(Action[0]==REVERSE) outdata[0] = 255 - outdata[0];
Out_Data(0x300,outdata[0]);
e0[0] = e1[0];

SP[1]           = pv[0] * Ratio;
indata[1]       = Get_Data(0x301);
pv[1]           = indata[1] / 2.56;
if(Sroot[1]==ON)
    e1[1]       = SP[1] - sqrt(pv[1]);
else
    e1[1]       = SP[1] - pv[1];
e2[1]           = e1[1]+e1[1]-e0[1];

if (Ki[1]>0.0)
{
    xi[1] = xi[1]+(e1[1]+e2[1])*T[1]/2*Ki[1];
    if (xi[1]>100.0) xi[1]=100.0;
    if (xi[1]<0.0)   xi[1]=0.0;
}
else xi[1]=0.0;

mv[1]=Kp[1]*(e2[1]+xi[1]+Kd[1]/T[1]*(e2[1]-e1[1]))+Bias[1];
if(mv[1] > 100.0) mv[1] = 100.0;
if(mv[1] < 0.0  ) mv[1] = 0.0;

display = pv[1];
outdata[1] = mv[1];
BarGraph(6782,display);
BarGraph(6786,outdata[1]);
CurPos(9,15); printf("PV %6.2f",pv[1]);
CurPos(9,16); printf("MV %6.2f",mv[1]);

outdata[1] *= 2.56;
if(Action[1]==REVERSE) outdata[1] = 255 - outdata[1];
Out_Data(0x301,outdata[1]);
e0[1] = e1[1];
```

```
        Loop = bioskey(2);
        Loop &= 3;
}

if(Ki[0]!=0) Bias[0]=mv[0];
if(Ki[1]!=0) Bias[1]=mv[1];
}

void CurPos(char x,char y)
{
        _AH = 2;
        _BH = 0;
        _DH = y;
        _DL = x;
        __int__(0x10);
}

void Cl_Window(char x1, char y1, char x2, char y2)
{
        _AH = 6;
        _AL = 0;
        _BH = 0;
        _CH = y1;
        _CL = x1;
        _DH = y2;
        _DL = x2;
        __int__(0x10);
}


void Screen(char S_Mode)
{
char    Delay;

        Cl_Window(0,0,79,24);
        _AH = 0;
        _AL = S_Mode;
        __int__(0x10);
        for(Delay=1; Delay<=200; Delay++);
}


void Hline(int Left, char Length)
{
register char far       *Pointer;
register char           Counter;

        Pointer = MK_FP(0xB800,Left);
        for(Counter = 1;Counter <= Length;Counter++)
        {
                *Pointer = 0xFF;
                Pointer++;
        }
}
```

```
void Vline(int Bottom, char Hight, char Data)
{
register char far       *Pointer;
register char           Counter =0;

        Pointer = MK_FP(0xB800,Bottom);
        for(;;)
        {
                *Pointer  = Data;
                Pointer  += 8112;
                Counter++;
                if(Counter == Hight) break;
                *Pointer  = Data;
                Pointer  -= 8192;
                Counter++;
                if(Counter == Hight) break;
        }
}


void BarGraph(int Bottom, char Hight)
{
register char far       *Pointer;
register char           Counter =0;

        Pointer = MK_FP(0xB800,Bottom);
        for(;;)
        {
                *Pointer  = 0xFF;
                Pointer  += 8112;
                Counter++;
                if(Counter == Hight) break;
                *Pointer  = 0xFF;
                Pointer  -= 8192;
                Counter++;
                if(Counter == Hight) break;
        }
        for(;;)
        {
                *Pointer  = 0;
                Pointer  += 8112;
                Counter++;
                if(Counter >= 100) break;
                *Pointer  = 0;
                Pointer  -= 8192;
                Counter++;
                if(Counter >= 100) break;
        }
}


char Get_Data(int Port)
{
```

```
char Data=0;

        Port++;
        Port++;
        outportb(Port,0);
        while(Data == 0)
        {
                Data  = inportb(Port);
                Data &= 1;
        }
        Port -= 2;
        Data  = inportb(Port);
return (Data);
}
```

กิตติกรรมประกาศ

กลุ่มผู้ทำปริญญานิพนธ์ ขอขอบพระคุณ อาจารย์ วันชัย ริ้วรุจา ที่ได้ให้คำแนะนำในการออกแบบและจัดสร้างฮาร์ดแวร์ และขอขอบพระคุณ อาจารย์ สุเชียร เกียรติสุนทร ที่ได้ให้คำแนะนำในการเขียนโปรแกรมคอมพิวเตอร์ ซึ่งส่งผลให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

หนังสืออ้างอิง

1. KATSUHIKO OGATA, "Discrete-time Control Systems",Prentice-Hall, 1987
2. KATSUHIKO OGATA, "Modern Control Engineering",Prentice-Hall ,1970
4. CURTIS D. JOHNSON, "Microprocessor-Based Process Control", Prentice-Hall, 1984
5. F. G. SHINSKEY, "Process Control System " McGraw-Hill Book Company
6. JOHN P. GERRY, " A Comparision of PID Control Algorithms" , Control Engineering, Vol 34, number 3, march 1987 p102-105