โปรแกรม ระบบควบคุมและจำลองการทำงาน แบบ P I D ด้วย ภาษาซี

PROGRAM P I D CONTROLLER & SIMULATION

By

C - LANGUAGE

โดย

นาย บริคุณ ล้ำเลิศประเสริฐ

นาย อนุชา เอี่ยมไพจิตร์

นาย อัคนิต์ บุญยะไวโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมการวัดคุมทางอุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา ๒๕๓๖

King Mongkut 's Institute of Technology

.L A D K R A B A N G.

Instrumentation Department

PRESENT

P I D CONTROLLER & SIMULATION

BY

C -LANGUAGE

สร้างสรรค์โดย

บริคุณ ล้ำเลิศประเสริฐ	๓๓๑๐๐๑๘๓
อนุชา เอี่ยมไพจิตร์	๓๓๑๐๐๔๘๒
อัคนิต์ บุญยะไวโรจน์	๓๓๑๐๐๕๐๙

อาจารย์ที่ปรึกษา

อาจารย์ วิริยะ กองรัตน์

รายงานโครงงาน

สำหรับ

ปริญญาวิศวกรรมศาสตร์บัณฑิต.

ภาควิชา : วิศวกรรมการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ประจำปีการศึกษา ๒๕๓๖

เรื่อง : Program P I D Controller & Simulation By C-Language
(โปรแกรม ระบบควบคุมและจำลองการทำงาน แบบ P I D โดย ภาษาซี)

ผู้จัดทำ :

บริคุณ ล้ำเลิศประเสริฐ	รหัส	๓๓๑๐๐๑๘๓
อนุชา เอี่ยมไพจิตร์	รหัส	๓๓๑๐๐๔๘๒
อัคนิดิ์ บุญยะไวโรจน์	รหัส	๓๓๑๐๐๕๐๙

อาจารย์ที่ปรึกษา :

(อาจารย์ วิริยะ กองรัตน์)

สารบัญ

โปรแกรม ระบบควบคุม และจำลองการทำงาน แบบ P I D ด้วย ภาษา ซี

Program P I D Controller & Simulation By C-Language

โดย บริคุณ ล้ำเลิศประเสริฐ รหัส ๓๓๑๐๐๑๘๓
อนุชา เอี่ยมไพจิตร์ รหัส ๓๓๑๐๐๔๘๒
อัคนิต์ บุญยะไวโรจน์ รหัส ๓๓๑๐๐๕๐๙
อาจารย์ที่ปรึกษา อาจารย์ วิริยะ กองรัตน์

บทคัดย่อ

ปัจจุบันเทคโนโลยีการควบคุมกระบวนการผลิต ได้พัฒนาขึ้นมากกว่าสมัยก่อนมากส่งผลให้ไม่มีการปรับตัวของผู้ปฏิบัติการ ในการที่จะใช้งานตัวควบคุมต่าง ๆ ให้ได้เต็มความสามารถของอุปกรณ์นั้น ๆ ได้ 100 %

โครงงาน " โปรแกรมระบบควบคุม และ จำลองการทำงานแบบ PID " เป็นโครงงานที่นำทฤษฎีการควบคุม มาเขียนเป็นโปรแกรมที่สามารถทำงานได้บนเครื่อง Computer ทั่วไป เพื่อใช้ในการควบคุมกระบวนการผลิตแทนตัวอุปกรณ์จริงที่มีราคาแพง และมีการใช้งานที่ยุ่งยากกว่า

นอกจากจะสามารถควบคุมกระบวนการผลิตแล้ว โปรแกรมยังสามารถที่จะจำลองกระบวนการควบคุมง่าย ๆ เพื่อให้ผู้ใช้ได้สามารถศึกษาถึงธรรมชาติของกระบวนการควบคุมแต่ละชนิดได้ด้วย ซึ่งเป็นการพัฒนาความรู้ความสามารถของผู้ใช้งานอีกอย่างหนึ่ง.

ABSTRACT

Recently, Process control technology has developed from the last decade . Simultaneorsly the improvement of users has been cut down , the result is that workability of most of controllers not reach to 100 %.

" Program P I D Controller & Simulation By C - Language " is the project by applying the control theory in program computer which can be done in any Personal Computer , the purpose is to control the process instead of the controller , which is more expensive and difficult .

Besides controlling the process , the program can simulate a simple process in reason to studying about each controlling process which become the developing of users' knowledge.

คำนำ

ในโลกอุตสาหกรรมยุคโลกานุวัตรปัจจุบันนั้น ขั้นตอนการผลิตของกระบวนการผลิตต่าง ๆ อุปกรณ์อย่างหนึ่งที่จัดว่าเป็นส่วนสำคัญมากในการควบคุมแบบอัตโนมัตินั้นคือ ตัวควบคุม (CONTROLLER) ทางอุตสาหกรรม ซึ่งปัจจุบันมีการใช้อย่างแพร่หลายมาก

อุปกรณ์ควบคุมเหล่านี้ มีลักษณะเป็นฮาร์ดแวร์อิเลคโทรนิคส์ ที่มีราคาค่อนข้างแพง และยังมีการใช้ งานที่ค่อนข้างเข้าใจยากสำหรับผู้ปฏิบัติการทั่วไป นอกจากนี้ด้วยข้อจำกัดของขนาดทำให้การแสดงผลของ ค่าตัวแปรควบคุมต่าง ๆ สื่อความหมายได้ไม่ชัดเจน

สำหรับโครงงาน **Program P I D CONTROLLER & SIMULATION By C- Language** นี้ ได้นำเอาหลักการพื้นฐานของระบบควบคุมแบบป้อนกลับมาเขียนโปรแกรมให้สามารถทำงานได้บนเครื่อง คอมพิวเตอร์ทั่ว ๆ ไป ที่มีกันอย่างแพร่หลาย โดยการใช้ภาษา TURBO C++ Version 1.0 เป็นตัวแปล ภาษาซึ่งหลักการในการออกแบบโปรแกรมนั้น เป็นเช่นเดียวกับการออกแบบของตัวควบคุมที่เป็นฮาร์ดแวร์ เช่นกัน

ข้อได้เปรียบอีกอย่างของโปรแกรมนี้ก็คือ การใช้งานที่ง่ายด้วยรูปแบบ Pull-Down Menu และ การมีข้อความ Help อธิบายการใช้งานอย่างคร่าว ๆ การเปลี่ยนแปลงค่าตัวแปรควบคุมได้ง่าย รวมถึง การแสดงผลแบบ Graphics ที่มีสีสันชัดเจนสวยงาม และราคาที่ประหยัดกว่าตัวควบคุมจริง ๆ มาก

ส่วนสำคัญอีกส่วนหนึ่งที่ตัวควบคุมทั่วไปไม่มีคือ การจำลองสภาพกระบวน การทำงานของ Process ตัวอย่าง มาให้แก่ผู้ใช้งานได้ศึกษา โดยที่ไม่จำเป็นต้องมีการต่อเชื่อมกับระบบจริง ๆ ผู้ใช้สามารถศึกษา ถึงธรรมชาติการทำงานต่าง ๆ ของระบบควบคุมได้เป็นอย่างดี

หวังใจเป็นอย่างยิ่งว่าโครงงานเล็ก ๆ นี้จะเป็นตัวจุดประกายความรู้แก่ผู้ใช้งานและประชาชนชาวไทยโดยทั่วไป ให้สามารถสร้างสรรค์ประโยชน์อันยิ่งใหญ่แก่ชาติไทยของเราต่อไปเทอญ

คณะผู้จัดทำ

(มีนาคม ๒๕๓๗)

ภาค 1

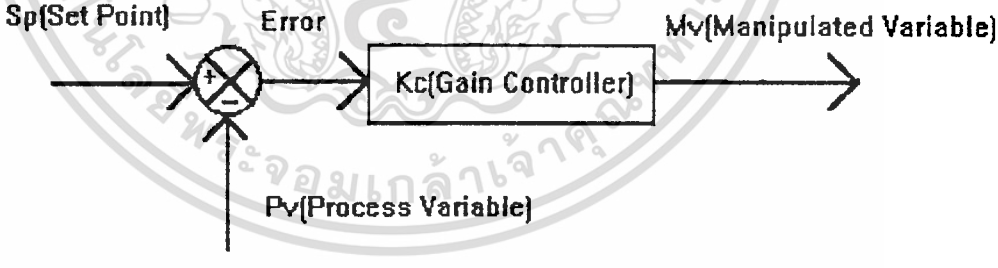
ปฐมบททฤษฎีการควบคุม

ระบบควบคุมกระบวนการ (Process Control System)

1. การควบคุมกระบวนการแบบอัตโนมัติ (Automatic Process Control)

การควบคุมกระบวนการแบบอัตโนมัติ เป็นเรื่องเกี่ยวข้องกับค่าตัวแปรต่าง ๆ ในกระบวนการ (Process Variables) เช่น อุณหภูมิ (temperature) ความดัน (pressure) อัตราการไหล (flows) และตำแหน่งต่างๆ (compositions)

ในการควบคุมกระบวนการให้เป็นผลสำเร็จลงได้นั้น เราต้องหาความคลาดเคลื่อน (error) โดยการวัดค่าตัวแปรต่างๆ (process variables) เหล่านี้ แล้วนำมาเปรียบเทียบกับค่าที่ตั้งไว้ (set point) แล้วตัดสินใจว่าจะทำอย่างไรกับค่าความแตกต่างนี้ เพื่อให้ได้ค่าที่เราต้องการ ซึ่งการทำงานเหล่านี้ สามารถทำได้โดยใช้ผู้ปฏิบัติการ (operator) แต่ในโรงงานส่วนใหญ่แล้ว จะมีตัวแปรต่างๆ เหล่านี้มากมาย ทำให้ต้องใช้คนมากมายตามไปด้วย ดังนั้นการใช้เครื่องมือที่มีความสามารถควบคุมกระบวนการได้โดยอัตโนมัติ จึงเป็นวิธีที่ดีกว่า



รูปที่ 1 การควบคุมกระบวนการ

ในการทำงานเช่นที่กล่าวนี้ ระบบควบคุมจะต้องได้รับการออกแบบมาให้มีส่วนประกอบที่สำคัญสำหรับการควบคุม 4 ตัวอย่าง ดังนี้

1. **ตัวตรวจจับสัญญาณ (sensor)** เป็นอุปกรณ์ตัวแรก (primary element)
2. **ตัวส่งผ่านสัญญาณ (transmitter)** เป็นอุปกรณ์ตัวที่สอง (secondary element)
3. **ตัวควบคุม (controller)** เป็น "สมอง" ของระบบควบคุม
4. **อุปกรณ์ควบคุมตัวสุดท้าย (final control element)** เป็นตัวส่งสัญญาณไปที่ระบบ

การปฏิบัติการ 3 อย่างที่ต้องมีในทุกๆ ระบบควบคุม คือ

1. **การวัด(Measurement : M)** : การวัดค่าตัวแปรที่ถูกควบคุม กระทำโดยตัวตรวจจับสัญญาณและตัวส่งผ่านสัญญาณ

2. **การตัดสินใจ (Decision : D)** : ขึ้นอยู่กับการวัด ตัวควบคุมต้องตัดสินใจว่าจะทำอย่างไรเพื่อให้ได้ค่าตัวแปรตามที่เราต้องการ

3. **การกระทำ (Action : A)** : เป็นผลงานของการตัดสินใจของตัวควบคุมระบบจะต้องถูกกระทำโดยอุปกรณ์ควบคุมตัวสุดท้าย

2. คำที่สำคัญในระบบควบคุมกระบวนการ

ตัวแปรที่ถูกควบคุมหรือตัวแปรกระบวนการ **(Controll Variable Process)** คือ ตัวแปรที่รับเข้ามาหรือถูกควบคุมตามค่าที่ต้องการ

ค่าเป้าหมายหรือค่าที่ตั้งไว้ **(Set Point)** : ค่าที่ต้องการของตัวแปรที่ถูกควบคุม

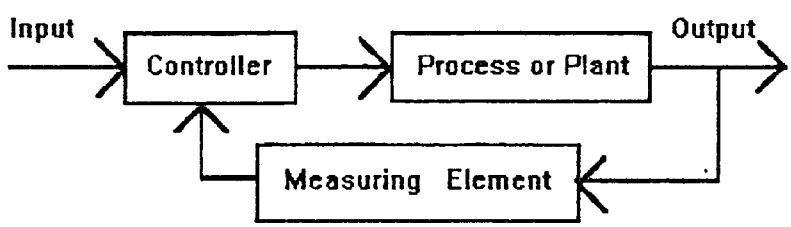
ตัวแปรที่ไปควบคุม **(Manipulated Variables)** คือ ค่าตัวแปรที่ทำให้ตัวแปรที่ถูกควบคุมเข้าหาค่าที่ตั้งไว้

การรบกวนกระบวนการ **(Disturbance Upset)** คือ ค่าตัวแปรที่ทำให้ค่าตัวแปรที่ถูกควบคุมเปลี่ยนแปลงไปจากเป้าหมาย

การควบคุมแบบเปิด **(Open-Loop Control)** คือ สภาพที่ตัวควบคุมไม่ได้ติดต่อกับกระบวนการ กล่าวคือ ตัวควบคุมไม่ต้องตัดสินใจว่าจะต้องทำอย่างไร ที่จะให้ตัวแปรที่ถูกควบ-คุมเข้าหาค่าที่ตั้งไว้

รูปที่ 2 ระบบควบคุมแบบ Open Loop

การควบคุมแบบปิด **(Close-Loop Control)** คือ สภาพที่ตัวควบคุมติดต่อกับกระบวนการ คือ มีการเปรียบเทียบค่าตัวแปรที่ถูกควบคุมกับค่าที่ตั้งไว้ แล้วพิจารณาการกระทำที่ถูกต้องที่สุด

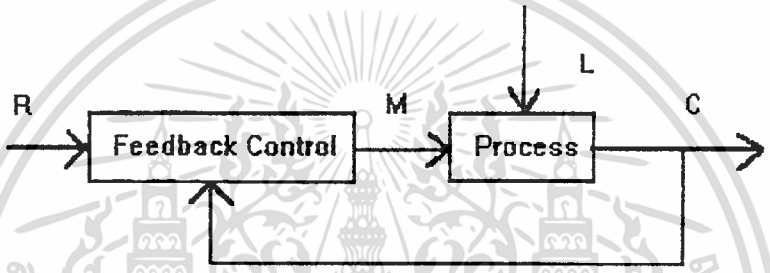


รูปที่ 3 ระบบควบคุมแบบ Close Loop

3. ประเภทของการควบคุม (Control Strategies)

การควบคุมแบบป้อนกลับ (Feedback Control)

วิธีการนี้ได้ถูกประยุกต์ใช้โดย James Watt เมื่อ 200 กว่าปีก่อน การที่จะเข้าใจถึงการทำงานของการควบคุมแบบป้อนกลับ เราสามารถดูได้จากข้อได้เปรียบและข้อเสียเปรียบของมัน ดังนี้



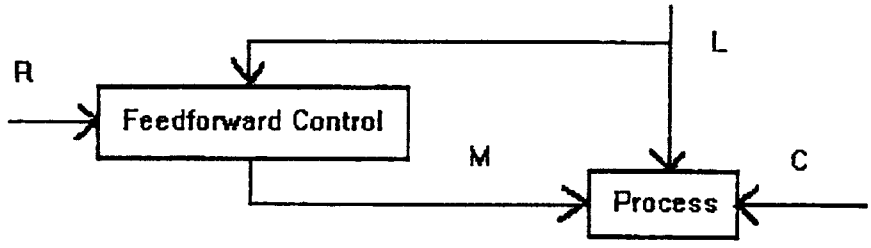
รูปที่ 4 การควบคุมแบบป้อนกลับ (Feedback Control)

ข้อได้เปรียบ (advantages) ของการควบคุมแบบป้อนกลับ คือ เป็นวิธีการที่ง่ายที่ชดเชยการรบกวนกระบวนการทุกชนิด การรบกวนกระบวนการทุกๆ ตัวจะมีผลต่อตัวแปรที่ถูกควบคุม และเมื่อค่าตัวแปรเบี่ยงเบนไปจากค่าที่ตั้งไว้ ตัวควบคุมจะเปลี่ยนแปลงค่า เอาท์พุทให้กลับเข้าสู่ค่าเป้าหมาย

ข้อเสียเปรียบ (disadvantages) ของการควบคุมแบบป้อนกลับ คือ มันสามารถชดเชยการรบกวนได้เพียงหลังจากที่ตัวแปรที่ถูกควบคุม เบี่ยงเบนไปจากค่าที่ตั้งไว้แล้วเท่านั้น นั่นคือ การรบกวนจะกระจายไปยังกระบวนการทั้งหมด ก่อนที่การควบคุมแบบป้อนกลับจะชดเชยให้มัน

การควบคุมแบบไปข้างหน้า (Feedforward Control)

จุดประสงค์ของการควบคุมแบบไปข้างหน้า คือ การวัดการรบกวนกระบวนการและชดเชยให้กับมัน ก่อนที่ค่าตัวแปรที่ถูกควบคุมจะเบี่ยงเบนไปจากค่าที่ตั้งไว้ ถ้าเราประยุกต์ใช้ได้อย่างถูกต้อง ค่าตัวแปรที่ถูกควบคุมจะไม่เบี่ยงเบนไปจากค่าเป้าหมาย



รูปที่ 5 การควบคุมแบบไปข้างหน้า (Feedforward Control)

ตัวควบคุมแบบป้อนกลับ (FEEDBACK CONTROLLERS)

Automatic / Manual Control Mode

Control ในอุดมคติจะแสดง automatic mode ขณะปฏิบัติงาน แต่ในบางสถานการณ์ที่ผู้ปฏิบัติงาน (operator) อาจจะต้องการลบ automatic mode ทิ้งไป และปรับแต่ง output ของ controller ด้วยมือ

Manual Mode ของการปฏิบัติงาน จะมีประโยชน์มาก เมื่อเริ่ม start up plant ,shut down plant หรือในสถานการณ์ฉุกเฉิน

Controller โดยปกติจะมีสวิทช์ manual/automatic เอาไว้เปลี่ยนแปลง Controller จาก automatic mode ไปเป็น manual mode และในทางกลับกันด้วย อย่างไรก็ดี output ของ controller อาจจะเปลี่ยนแปลงทันทีทันใด ระหว่างการเปลี่ยนแปลง mode นี้ และ "กระแทก" (bump) process controllers ปัจจุบันนี้จะมี bumper transfers ซึ่งจะทำให้ไม่กระทบกระเทือน process

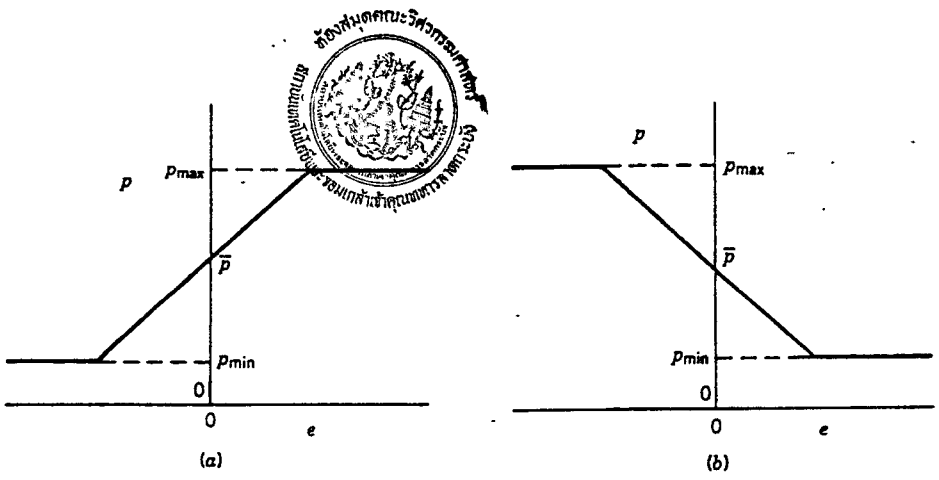
1. การกระทำของตัวควบคุม (Actions of controllers)

Reverse Action , Decrease คือ เมื่อสัญญาณ INPUT เข้าสู่ controller เพิ่มขึ้น (increase) จะทำให้สัญญาณของ OUTPUT controller ลดลง (decrease)

Direct Action , Increase คือ เมื่อสัญญาณ INPUT เข้าสู่ controller เพิ่มขึ้น (increase) จะทำให้สัญญาณ OUTPUT ของ controller เพิ่มขึ้น (increase)

แต่ในการพิจารณา action ของ controller เราต้องรู้ว่า

1. ความต้องการของ process สำหรับการควบคุม
2. action ของ control valve หรือ other final control element



รูปที่ 6 Reverse และ Direct Acting Proportional Controller
(a) Reverse Acting (K c >0) (b) Direct Action (K c<0)

2. รูปแบบของตัวควบคุมแบบป้อนกลับ (Types of Feedback Controllers)

รูปแบบของตัวควบคุมแบบป้อนกลับ มีดังนี้

Proportional Controller (P)

ตัวควบคุมแบบ P เป็นแบบที่ง่ายที่สุด หรือที่เรียกว่า on-off controller ซึ่งมีสมการที่อธิบายการทำงานดังนี้

$$m(t) = \bar{m} + K_c(r(t) - c(t)) \qquad (1)$$

หรือ

$$m(t) = \bar{m} + K_c e(t) \qquad (2)$$

เมื่อ

m(t) = เอาท์พุทจากตัวควบคุม,หน่วย psig or mA

r(t) = ค่าเป้าหมาย,หน่วย psig หรือ mA

c(t) = ตัวแปรที่ถูกควบคุม,หน่วย psig หรือ mA,นี่เป็นสัญญาณจากตัวส่งสัญญาณ

e(t) = สัญญาณที่ผิดพลาด,หน่วย psig หรือ mA.นี่เป็นความแตกต่างระหว่างค่าเป้าหมายและตัวแปรที่ถูกควบคุม

K_c = ค่า gain ของ controller , หน่วย psi/psi หรือ mA/mA

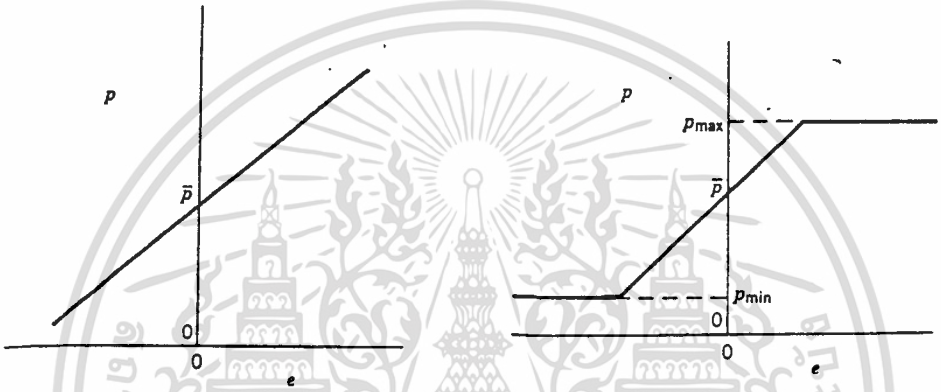
$\bar{m}$ = ค่า bias , หน่วย psig หรือ mA ความสำคัญของค่านี้คือ เอาท์พุดจาก controller เมื่อค่าผิดพลาดมีค่าเป็นศูนย์ ปกติแล้วค่านี้จะถูกตั้งไว้ที่กึ่งกลางสเกลในระหว่างการ calibration ของตัวควบคุม ,9 psig หรือ 12 mA

สมการที่ (1) เขียนขึ้นสำหรับ reverse acting controller *ถ้าตัวแปรที่ถูกควบคุม, C(t) เพิ่มขึ้น* เหนือจุด set point , r(t) error ค่าความคลาดเคลื่อนจะเป็นลบ และจากสมการจะได้ว่า ***output ของ controller , m(t) จะลดลง***

และการที่จะแสดงว่าเป็น direct-acting controller ก็โดยการให้ gain ของ controller,K_c

เป็นลบ อย่างไรก็ตาม controller ทั่วไปในโรงงานอุตสาหกรรมจะไม่มีค่า gain ที่เป็นลบ จะมีเพียงค่าที่เป็นบวก สวิทช์แบบ reverse/direct จะดูแลเรื่องนี้ ค่า K_C ที่เป็นลบจะใช้เมื่อทำการวิเคราะห์ทางคณิตศาสตร์ สำหรับระบบควบคุมที่ต้องการ direct-acting controller เท่านั้น

สมการที่ (1) และ (2) แสดงให้เห็นว่า เอาท์พุทของ controller เป็นสัดส่วนโดยตรงกับ error ระหว่างจุด set point และตัวแปรที่ถูกควบคุม การเป็นสัดส่วนโดยตรงนี้ เกิดขึ้นโดยค่า gain ของ controller ,K_C gainนี้ หรือที่เรียกว่า **controller sensitivity** ซึ่งเราจะพิจารณาว่าจะเปลี่ยนแปลง เอาท์พุทของ controller ไปเท่าไหรเมื่อมีการเปลี่ยนแปลงของ error เกิดขึ้น



รูปที่ 7 พฤติกรรมในอุดมคติของ **Propotional Control** (ความชันของเส้น=K c) ,ซ้าย

รูปที่ 8 พฤติกรรมตามความเป็นจริงของ **Propotional Control** ,ขวา

ข้อได้เปรียบของ controller แบบนี้คือ มี tuning parameter เพียงตัวเดียว คือ K_C อย่างไรก็ตามมันมีข้อเสียเปรียบที่สำคัญก็คือจะมี OFFSET หรือ steady state error กล่าวคือ ***error term,e(t) ไม่สามารถเป็นศูนย์ได้ที่ steady state*** และค่า steady state error นี้คือ offset

ผู้ผลิต controller บางรายจะไม่ใช้ term gain สำหรับ controller sensitivity แต่กลับใช้ term proportional band, PB แทน

ความสำคัญระหว่าง gain ,(Kc) และ PB คือ

$$\mathbf{PB = 100/Kc \quad (3)}$$

และจะได้

$$\mathbf{m(t) = \overline{m} + 100/PB\,(r(t) - c(t))}$$

หรือ

$$\mathbf{m(t) = \overline{m} + 100/PB\,e(t)}$$

term = 100 ใช้เพราะว่า PB มักจะหมายถึง "percent proportional band"

สมการที่ (3) แสดงให้เห็นว่า เมื่อ Kc เพิ่มขึ้นจะทำให้ PB แคบลง และเมื่อ Kc ลดลงจะพบว่า PB กว้างขึ้น ก่อนที่เราจะปรับค่าของ controller เราต้องทราบก่อนว่า controller นั้นใช้ K_C หรือ PB ส่วนในโครงงานนี้ใช้ค่า Kc

Proportional Integral Controller(PI)

มีหลายกระบวนการที่ไม่สามารถถูกควบคุมได้ด้วยค่า offset นั่นคือ ต้องถูกควบคุมไว้ที่ set -point ในกรณีนี้ เราต้องเพิ่มอีกเทอมของการควบคุมเข้าไป เราเรียกเทอมนี้ว่า integral หรือ reset action ดังสมการต่อไปนี้

$$m(t) = \bar{m} + Kc[r(t) - c(t) + (Kc/\tau_I) \int [r(t) - c(t)]dt \qquad (4)$$

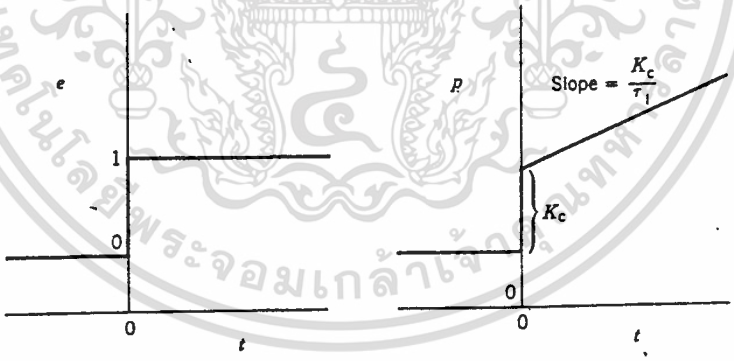
หรือ

$$m(t) = \bar{m} + K_c e(t) + (Kc/\tau_I) \int e(t)\, dt \qquad (5)$$

เมื่อ τ_I= integral หรือ reset time,หน่วย นาทีต่อรอบ

นั่นคือ PI Controller จะมีตัวแปรสองตัวคือ K_C และ τ_Iที่ใช้ปรับแต่งเพื่อให้ได้การควบคุมที่ต้องการ

จากสมการที่ (5) จะเห็นได้ว่า เมื่อค่า τ_Iลดลง เทอม K_C/τ_Iจะมีค่ามากขึ้น ฉะนั้นค่า gain จะมากขึ้น เมื่อเพิ่ม integral หรือ reset action และเมื่อใดก็ตามที่ error ยังคงมีอยู่ในสมการ controller จะเปลี่ยนแปลงเอาท์พุทของตัวมัน โดยทำการรวม error เพื่อกำจัด error



รูปที่ 9 ผลการตอบสนองของ PI Controller ที่มีผลต่อการเปลี่ยนแปลงแบบ Unit Step

ความจริงแล้ว การที่ค่า error เป็นศูนย์ ไม่ได้หมายความว่า เทอม integral จะเป็นศูนย์

$$(Kc/\tau_I) \int 0\, dt$$

แต่หมายความว่า controller รวม function ด้วยค่าศูนย์ หรือที่เรียกว่า " adding zero" (เติมให้เป็นศูนย์) ให้แก่ เอาท์พุทของตัวมัน

ผู้ผลิตบางรายไม่ได้ใช้เทอม reset time, แต่ใช้ reset rate ความสัมพันธ์ระหว่างตัวแปรสองตัวแปรนี้คือ สมการ

$$\tau_I^R = 1/\tau_I, \text{ repeats / min} \qquad (6)$$

นั่นคือ ก่อนที่จะทำการ tuning (การปรับแต่งค่า) ค่า integral เราต้องทราบก่อนว่า controller ที่เราใช้อยู่ ใช้ reset time หรือ reset rate ในโครงงานนี้ใช้ค่า reset rate (Ki)

Proportional-Integral-Derivative Controller (PID)

ในบางครั้งจะมีอีกเทอมเติมเข้าไปใน PI Controller นั่นคือ derivative action หรือที่เรียกว่า rate action หรือ **Preact** ก็เพื่อ ***คาดการณ์ล่วงหน้า*** สมการที่ใช้อธิบายมีดังนี้

$$m(t) = \bar{m} + Kce(t) + (Kc/\tau_1)\int e(t)dt + Kc\tau_D de(t)/dt \qquad (7)$$

เมื่อ

τ_D = derivative หรือ rate time (นาที)

นั่นคือ PID Controller มี 3 ตัวแปร คือ K_C หรือ PB, τ_I หรือ τ_I^R และ τ_D ที่ต้องปรับแต่งเพื่อให้ได้การควบคุมตามที่ต้องการ จะเห็นได้ว่าค่า τ_D จะมีหน่วยเดียว (นาที) ในทุกๆ ผู้ผลิต

อย่างที่ได้อธิบายมาแล้ว derivative action จะทำให้ controller สามารถทำนายล่วงหน้าได้ในขณะที่กระบวนการ (process) กำลังดำเนินอยู่ โดยการคำนวณการเกิดขึ้นของ error จำนวนที่คาดการณ์ล่วงหน้านี้ จะได้มาจาก ค่าจากการปรับแต่ง τ_D

PID Controller จะใช้กระบวนการคงที่เป็นระยะเวลานาน ตัวอย่างเช่น อุณหภูมิหรือ concentration loop กระบวนการที่คงที่เป็นเวลาสั้น (มีความจุน้อย) จะเร็วและไวต่อสัญญาณรบกวน process กระบวนการแบบนี้เช่น flow loop หรือ loop ที่ควบคุมความดันของไอของเหลว และกระบวนการที่คงที่กินเวลาเป็นระยะเวลานาน (มีความจุมาก) มักจะเกิดการแกว่ง (damped) แต่ความไวต่อสัญญาณรบกวนจะน้อย แต่ต้องระวังการรบกวนในสัญญาณ transmitter

Transfer Function ของ "ideal" PID Controller มีดังนี้

$$m(t) - \bar{m} = Kc(e(t) - 0) + (Kc/\tau_1)\int (e(t) - 0)dt + Kc\tau_D\, d(e(t) - 0)/dt$$

เราใช้คำจำกัดความของ deviation variable และ take Laplace จะได้

$$M(s)/E(s) = Kc(1 + 1/\tau_1 s + \tau_D s) \qquad (8)$$

Transfer Function ที่ได้นี้ เรียกว่าเป็น "ideal" เพราะในการใช้งานจริงการเพิ่มขึ้นของการคำนวณ derivative จะไม่มีผล derivative จะประมาณโดยการใช้ lead / lag เป็นผลให้เกิด "actual" (ใช้งานจริง) Transfer function แสดงได้ดังสมการที่ (9)

$$M(s)/E(s) = Kc\,(1 + 1/\tau_1)((\tau_D s + 1)/(\alpha\tau_D s + 1)) \qquad (9)$$

ค่า α จะประมาณอยู่ระหว่าง 0.05-0.1

Proportional-Derivative Controller(PD)

Controller ชนิดนี้ จะใช้กับ Process ที่ Proportional เท่านั้นที่ Controller ใช้ได้ แต่จำนวนของ " **การคาดการณ์ล่วงหน้า** " ก็ต้องการ

สมการที่ใช้อธิบายคือ

$$m(t) = \bar{m} + Kce(t) + Kc\tau_D\, de(t)/dt \qquad (10)$$

และ ideal transfer function คือ

$$M(s)/E(s) = Kc\,(1 + \tau_D S) \qquad (11)$$

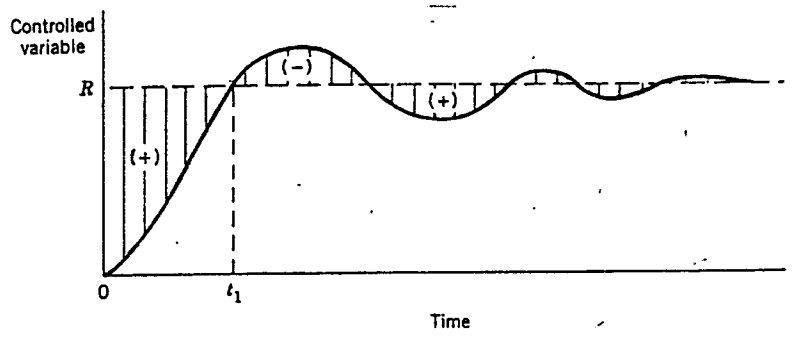
ข้อเสียเปรียบของ PD Controller คือการทำงานด้วยค่า offset ในตัวแปรที่ถูกควบคุม อย่างไรก็ดี PD Controller จะมี gain รวมที่สูงกว่า ส่งผลทำให้มี offset ที่ต่ำกว่าใน P Controller ใน loop เดียวกัน

Reset Wind Up

Reset Wind Up เป็นปัญหาที่สำคัญอันหนึ่งของการควบคุมกระบวนการ ซึ่งเกิดขึ้นเมื่อ controller ประกอบด้วยเทอม integral

ลองนึกถึงว่า mode integral จะทำให้ output ของ controller เปลี่ยนแปลงตลอดเวลา ตราบใดที่ e(t) ยังไม่เท่ากับศูนย์ เมื่อ error ยังคงเกิดขึ้นอยู่เรื่อยๆ เทอม integral ก็จะเริ่มมากขึ้น และทำให้ output ของ controller **อิ่มตัว (saturate)** ได้ การเพิ่มขึ้นของเทอม integral ขณะที่ controller อิ่มตัวอยู่นี้ เรียกว่า reset wind up หรือ integral wind up

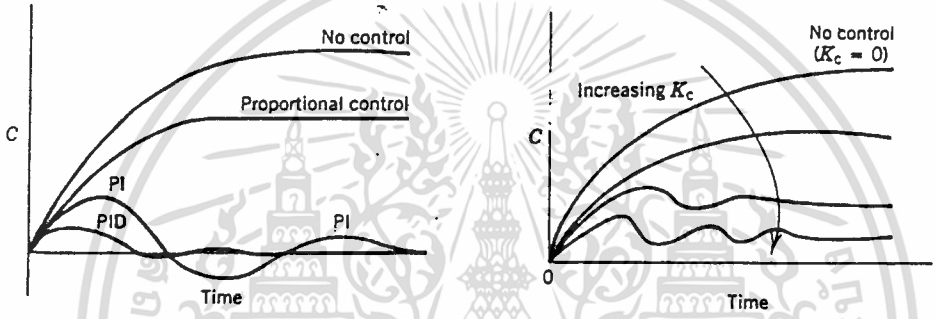
Reset wind up จะเกิดขึ้นเมื่อ PI หรือ PID Controller เพิ่มค่าความผิดพลาดที่ยังคงอยู่ ยกตัวอย่างเช่น ระหว่างการ start up กระบวนการแบบ batch หรือ หลังจากที่มีการเปลี่ยนแปลง SP ไปมากๆ และ reset wind up ยังคงสามารถเกิดขึ้นเนื่องมาจาก การรบกวน load ที่ยังคงเกิดขึ้นอย่างมากมาย ซึ่งอยู่นอกเหนือ range ของ manipulated variable อีกด้วย



รูปที่ 10 **Reset Wind Up ขณะที่เปลี่ยนแปลง Set Point**

ในกรณีนี้ การจำกัดขอบเขตของ value (เปิดสุด หรือปิดสุด) จะป้องกัน controller จากการลดสัญญาณ การผิดพลาดลงสู่ศูนย์ได้ พูดให้ชัดเจนก็คือ มันไม่มีความจำเป็นที่จะตัดเทอม integral หลังจากที่ output ของ controller เกิดอิ่มตัว ซึ่งถ้าเราทำเช่นนี้ได้จะสามารถลดค่าความผิดพลาดลงได้ controller เชิงพาณิชย์จะจัดเตรียมส่วนนี้ไว้ให้เรียกว่า ***anti reset wind up***

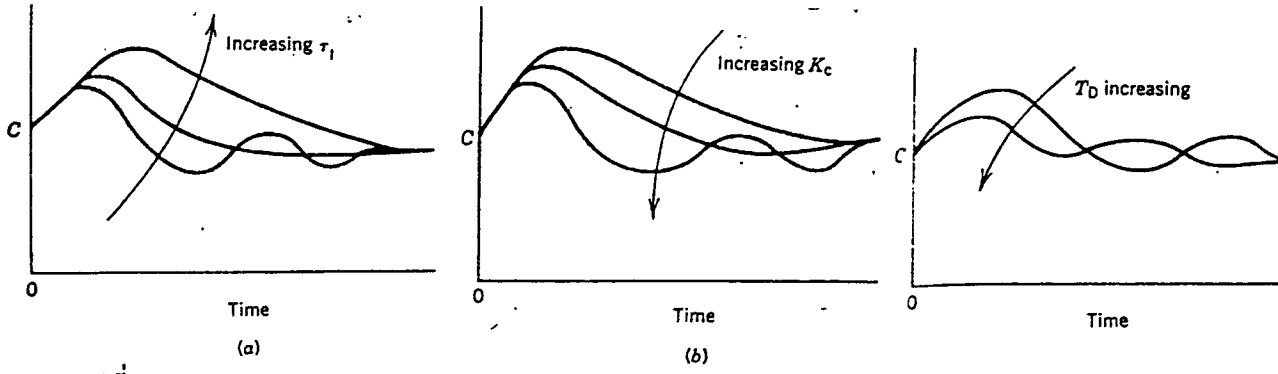
ผลตอบสนองของระบบควบคุมแบบป้อนกลับแต่ละชนิด
(Typical Response of Feedback Control System)



รูปที่ 11 ผลตอบสนองของ Process แต่ละชนิดใน Feedback Control ,ซ้าย
รูปที่ 12 ผลของ Gain Controller ใน P Control ,ขวา

รูปที่11 แสดงพฤติกรรมของกระบวนการที่ถูกควบคุมหลังจากเกิดการเปลี่ยนแปลงโหลด (load change) แบบขั้นบันได (step) ตัวแปรที่ถูกควบคุม C แสดงค่าเบี่ยงเบนจากค่าสภาวะเริ่มต้น

ถ้าไม่มีการควบคุมแบบ ป้อนกลับ กระบวนการจะค่อยๆ ขึ้นไปสู่สภาวะคงที่ค่าใหม่ การควบคุมแบบ Proportional จะเร่งการตอบสนองกระบวนการและลด offset การเพิ่มการกระทำการควบคุมแบบ integral จะกำจัด offset แต่มีแนวโน้มที่จะแกว่ง (oscillate)มากขึ้น การเพิ่มการกระทำ derivative จะลดทั้งขนาดของการ oscillation และเวลาในการตอบสนอง การใช้ P, PI, PID Controller ใช่ว่าจะตอบสนองแบบแกว่งเสมอไป ขึ้นอยู่กับการตั้งค่าของ Controller (K_C ,τ_I, τ_D) และการเปลี่ยนแปลงของกระบวนการด้วย



รูปที่ 13 (a) ผลของ Reset time (b) ผลของ Gain controler ใน PI Controniler
รูปที่ 14 ผลของ derivative time ใน PID Controniler

รูปที่ 12-14 แสดงผลที่เกิดขึ้นเนื่องจากการเปลี่ยนแปลงค่าต่างๆ ที่จัดตั้งไว้ของ controller แต่ละค่า โดยปกติแล้วการเพิ่ม gain มีแนวโน้มที่จะทำให้ผลการตอบสนองของกระบวนการเร็วขึ้น แต่ถ้าค่า K_c มากเกินไป ผลตอบสนองจะแสดงขนาดของการ oscillation ที่ไม่ต้องการ หรืออาจจะไม่เสถียร (unstable) เลยก็ได้

การเพิ่ม reset time τ_I จะทำให้การควบคุมแบบ PI , PID ช้าลง ในการทฤษฎีค่า offset จะหายไป สำหรับค่า τ_I ตั้งแต่ 0 ถึง ∞ แต่สำหรับค่า τ_I ที่มีมากเกินไป ค่าตัวแปรที่ถูกควบคุมจะเข้าสู่ SP ช้าลงมากๆ หลังจากการเปลี่ยนแปลง load หรือ SP

เป็นการยากที่จะอธิบายเกี่ยวกับผลของ derivative τ_D สำหรับค่า τ_D น้อยๆ การเพิ่มมีแนวโน้มที่จะปรับปรุง ผลตอบสนองโดยการลดการเบี่ยงเบนสูดสุด (maximum derivaion) เวลาการตอบสนอง (response time) และขนาดของการแกว่ง (degree of oscillation) อย่างไรก็ตาม ถ้า τ_D มากเกินไป สัญญาณรบกวนจากการวัดมีแนวโน้มที่จะขยาย (amplify) และการตอบสนองจะเริ่ม oscillate นั่นคือ ค่ากลางๆ ของ τ_D จะดีที่สุด

ตัวควบคุม PID แบบ Digital

เมื่อมีการควบคุมแบบ Digital input และ output ของ controller จะต้องเป็นสัญญาณ digital หรือ สัญญาณที่สุ่มรับค่าเข้า (sampled) มากกว่าที่จะเป็นสัญญาณ analog หรือสัญญาณต่อเนื่อง

สัญญาณต่อเนื่องที่ถูกส่งมาจาก transmitter จะถูกสุ่มรับค่าเข้ามา และแปลงเป็นสัญญาณ digital ด้วย analog-to-digital converter (ADC) ด้วยระยะเวลาที่คงที่

Algorithm ของ digital control จะถูกนำมาใช้เพื่อคำนวณ output ของ controller เป็นสัญญาณ digital ก่อนที่ output ของ controller จะถูกส่งออกไปยังอุปกรณ์ควบคุมตัวสุดท้าย (เช่น control valve) สัญญาณ digital จะถูกแปลงไปเป็นสัญญาณต่เนื่องโดย digital-to-analog converter (DAC)

ความก้าวหน้าของ digital version จากสมการ PID Controller ดังสมการที่ (5) คือ การแปลงเทอม integral และ derivative เป็นแบบ discreat โดยประมาณ การ integral ด้วยการรวม(summation) และการ derivative โดยหาผลแตกต่างย้อนหลังแบบ first-order ดังสมการ

$$m_n = \overline{m} + Kc\,[\,e_n + \Delta t/\tau_1 \sum e_k + \tau_D/\Delta t(e_n - e_{n-1})] \quad (12)$$

เมื่อ Δt = เวลาที่ใช้ในการสุ่มรับค่าเข้ามา (sampling period) เป็นเวลาที่อยู่ระหว่างการเก็บค่าตัวอย่างที่ตามๆ กันมาของตัวแปรที่ถูกควบคุม

m_n = output ของ controller ขณะที่มีการสุ่มรับค่าข้อมูลครั้งที่ n

n = 1,2....

e_n = error ครั้งที่ n ของการสุ่มรับค่าข้อมูล

สมการที่ (12)นี้เรียกว่า position form ของ PID algorithm เมื่อ output ของ controller จริงๆ ถูกคำนวณ

ทางเลือกใหม่ที่ดีกว่า คือ การใช้ velocity form ของ algorithm ซึ่งใช้การเปลี่ยนแปลง (change) ของ output ของ controller ในการคำนวณ

ทำสมการที่ (12) สำหรับการสุ่มรับค่าข้อมูลครั้งที่ (n-1)

$$m_{n-1} = \bar{m} + Kc\,[e_{n-1} + (\Delta t/\tau 1)\sum e_k + \tau_d/\Delta t(e_{n-1} - e_{n-2})] \quad (13).$$

การรวม (summation) จะเริ่มต้นที่ k = 1 นั่นคือ ระบบมีสภาวะคงที่ตามที่ต้องการ เมื่อ k <= 0 หรือ e_k = 0 สำหรับ k <= 0

สมการที่ (12) กับ สมการที่ (13)

$$\Delta m_n = m_n - m_{n-1} = Kc\,[e_n - e_{n-1}) + \Delta t/\tau_1 e_n + \tau_D/\Delta t\,(e_n - e_{n-1} + e_{n-2})] \quad (14)$$

และ output ของ controller คำนวณได้โดย

$$m_n = p_{n-1} + Kc[(e_n - e_{n-1}) + (\Delta t/\tau_1)e_n + \tau_D/\Delta t(e_n - 2e_{n-1} + e_{n-2})] \quad (15)$$

$$\Delta m_n = m_n - m_{n-1}$$

$$= Kc((1 + \Delta t/\tau_1 + \tau_D/\Delta t)\, e_n + (-1-2\,\tau_D/\Delta t)e_{n-1} + \tau_D/\Delta t\, e_{n-2} \quad (16)$$

$$m_n = m_{n-1} + [k_1 e_n + k_2 e_{n-1} + k_3 e_{n-2}] \quad (17)$$

$k_1 = Kc(1 + K_I dt + K_d/dt)$, $k_2 = -Kc(1+2K_d/dt)$, $k_3 = KcK_d/dt$

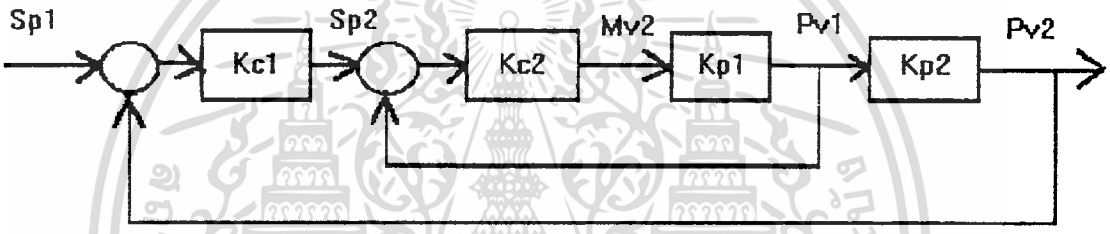
การเปรียบเทียบระหว่าง Position และ Velocity Algorithms (Comparision of Position and Velocity Algorithms)

1. position form ของ PID Controller ต้องการค่า m ในขณะที่แน่นอนเลยว่า velocity form ไม่ต้องการค่าที่สภาวะคงที่ สำหรับ output ของ controller อย่างไรก็ดี เป็นการง่ายที่จะเริ่มต้นให้แต่ละ algorithm มีค่าเท่ากัน เมื่อเปลี่ยนระบบควบคุมจาก manual ไปเป็นแบบ automatic อันจะเป็น bumpless transfer ในการลด bump ของ process

2. เมื่อ $\overline{m}$ (หรือ m_{n-1} สำหรับ velocity algorithm) ทำสัญญาณที่ส่งไปยังอุปกรณ์ควบคุมตัวสุดท้าย ให้เท่ากันได้ง่าย ขณะเปลี่ยน mode และ velocity form หลีกเลี่ยงการคำนวณ $\int e(t)dt$ ได้ ทำให้ velocity form มีแนวโน้มในการเกิดปัญหา reset wind up น้อยกว่า

การควบคุมแบบ Cascade (Cascade Control)

การควบคุมแบบ Cascade หรือ การควบคุมแบบลดหลั่นกันไปเป็นการควบคุมที่มีประโยชน์มากในโรงงานอุตสาหกรรม หลักการทำงานคือการต่อ controller 2 ตัวเข้าด้วยกัน โดยที่ output หรือค่า manipulated variables ของ controller ตัวแรกจะเป็น set point ให้กับ controller ตัววที่ 2 ดังรูปที่ 15



รูปที่ 15 การควบคุมแบบ Cascade

โดยทั่วไปแล้ว controller ที่ควบคุมตัวแปรที่ถูกควบคุมตัวแรก จะเรียกว่า master controller, outer controller หรือ primary controller และ controller ที่ตัวควบคุมตัวแปรที่ถูกที่ถูกควบคุมตัวที่ 2 จะเรียกว่า slave controller, inner controller หรือ secondary controller โดเทอมของ primary/secondary จะน่าสนใจกว่า เพราะสามารถเผื่อไว้สำหรับระบบหลายๆ loop

ในการออกแบบควบคุมแบบ cascade สิ่งที่ข้อควรพิจารณาที่สำคัญที่สุด คือ inner หรือ secondary loop ต้องเร็วกว่า outer หรือ primary loop ข้อพิจารณานี้เป็นสิ่งที่รู้กันดีทั่วไป และข้อพิจารณานี้สามารถจะขยายไปในหลายๆ loop ได้ คือใน ระบบควบคุมที่มี 3 loop cascade, tertiary loop จะต้องเร็วกว่า secondary loop และ primary loop

โดยปกติแล้ว เมื่อเราประยุกต์ใช้อย่างถูกต้อง การควบคุมแบบ cascade จะทำให้ loop ทั้งหมดเสถียรมากกว่า และมีการตอบสนองที่เร็วกว่า

มีอยู่ 2 คำถามที่ยังคงอยู่ คือ เราจะทำอย่างไรในการเอาการควบคุมแบบ cascade ใส่ลงไปในการปฏิบัติงานแบบ automatic และ เราจะ tune controller อย่างไร คำถามคือ loop ที่อยู่ในสุด จะถูก tune ก่อน และถูกเปลี่ยนเป็น automatic ขณะที่ loop ที่อยู่ข้างนอกยังเป็น manual อยู่ ทำให้เหมือนอย่างนี้เรื่อยไป จน loop นอกสุดเป็น automatic และสิ่งที่สำคัญที่ต้องระวังคือ Action ของ controller โดยดูความต้องการของ process และ action ของ

control valve โดยจะดูตัวอย่างการควบคุมแบบนี้ได้ในกรณีศึกษา

First-Order

ผลตอบสนองของระบบ First-Order

จากความสัมพันธ์สำหรับผลตอบสนอง (response) แบบไดนามิคของ stirred-tank heater แบบง่าย เพื่อหาว่าอุณหภูมิที่ออกมา จะเปลี่ยนแปลงไปอย่างไร เมื่ออินพุทตัวใดตัวหนึ่งคือ $Q'_{(s)}$ หรือ $T'_i(s)$ ถูกเปลี่ยนไป เราจะใช้ transfer function แบบ first-order ทั่วๆไปเป็น

$$PV(s)/MV(s) = K/\tau s + 1 \qquad (18)$$

เมื่อค่า K เป็นค่า gain ของ process และ τ เป็นค่า time constant

ในการตรวจสอบรูปแบบเฉพาะของอินพุท X(s) บางรูปแบบ จะได้สมการของ PV(s) และผลตอบสนอง PV(s) ดังนี้

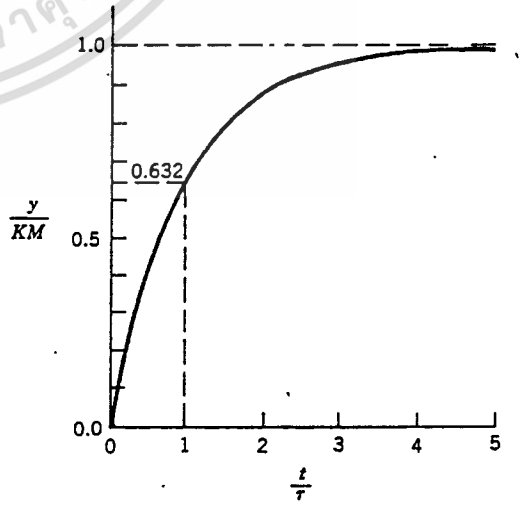
ผลตอบสนองแบบ step (Step Response)

สำหรับอินพุทแบบ 1 step ที่มีขนาด M, MV(s) = M/s และ

$$PV(s) = MK/s(\tau s + 1) \qquad (19)$$

จะได้ผลตอบสนองที่ขึ้นอยู่กับเวลาเป็น

$$PV(t) = MK(1-e^{-t/\tau}) \qquad (20)$$



รูปที่ 16 ผลตอบสนองแบบ Step สำหรับ First-Order Process

จากรูป curve ของความสัมพันธ์นี้แสดงว่า process แบบ first-order ไม่ได้ตอบสนองอย่างทันทีทันใด เมื่อมีการเปลี่ยนแปลงอินพุทอย่างรวดเร็ว ในความเป็นจริงแล้ว หลังที่ช่วงเวลาหนึ่งเท่ากับ time constant ของ process (กล่าวคือ $t = \tau$) เราจะพบว่าผลตอบ

สนองของ process จะไม่เท่ากับค่าใหม่ที่สภาวะคงที่ จะประมาณให้มีค่าเป็นค่าใหม่ เมื่อ t เท่ากับ 3 ถึง 5 เท่าของ time constant ดังตาราง 1 จะสังเกตได้ว่า รูปที่ 1 ถูกแสดงให้อยู่ในรูปปกติที่มีค่าเวลาถูกหารโดยค่า time constant และ การเปลี่ยนแปลงของ เอาท์พุทถูกหารโดยผลคูณของค่า gain และขนาดการเปลี่ยนแปลงของอินพุท

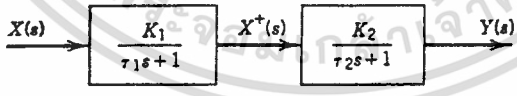
t	$y(t)/KM = 1 - e^{-t/\tau}$
0	0
τ	0.6321
2τ	0.8647
3τ	0.9502
4τ	0.9817
5τ	0.9933

ตารางที่ 1 ผลตอบสนองของ First Order Process ที่มีต่อ Input แบบ Step

Second-Order

ผลตอบสนองของระบบแบบ second-order

transferfunction แบบ second-order สามารถเกิดขึ้นได้ตามลักษณะของกายภาพ เมื่อไรก็ตามที่ process แบบ first-order 2 processes ถูกต่อเข้าด้วยกันแบบอนุกรม ยกตัวอย่างเช่น stirred-tank heater 2 ชุด ถูกต่อเข้าด้วยกัน ซึ่งแต่ละชุดจะมี transfer function แบบ first-order ที่แสดงถึงความสัมพันธ์ของอุณหภูมิที่เข้ากับอุณหภูมิที่ออก ดังนั้น ของไหลออกจาก heater ตัวแรกจะถูกใช้เป็นของไหลที่ไหลเข้า heater ตัวที่ 2 ดังแสดงความสัมพันธ์ได้ในรูปที่ 17



รูปที่ 17 ระบบ First Oder 2 ตัวที่ต่อแบบ Series ในระบบแบบ First Oder

Bandpass Filter Circuits

ซึ่งจะได้สมการเป็น

$$G(s) = PV(s)/MV(s) = K_1K_2/(\tau_1 s + 1)(\tau_2 s + 1)$$

$$= K/(\tau_1 s + 1)(\tau_2 s + 1) \qquad (21)$$

เมื่อ $K = k_1K_2$ และจะพิจารณา transfer function แบบ second-order ในกรณีที่มีมีรูปแบบมาตรฐานเป็น

$$G(s) = K/\tau^2 s^2 + 2\zeta\tau s + 1 \qquad (22)$$

เราสามารถหน่วงให้การเกิดขึ้นของกรณีดังกล่าวช้าลงด้วยเทอมของ time-delay

transfer function แบบ first-order K เป็นค่า gain ของ process , τ กำหนดความเร็วของการตอบสนอง (หรือเปรียบได้กับเวลาในการตอบสนอง) ของระบบ ถ้า parameter ใหม่ ζ(zeta) จะไม่มีขนาด ζ เป็นตัววัดจำนวนของ damping ในระบบ นั่นคือ ระดับของการแกว่งในผลตอบสนองของระบบหลังจากที่มีการรบกวนเกิดขึ้น ค่าของ ζ ที่มีค่าน้อยแสดงว่าเกิด damping เพียงเล็กน้อย ในตำราบางเล่ม สมการที่ (22) จะถูกเขียนอยู่ในเทอมของ $W_n = 1/\tau$ เรียกว่า ***ความถี่ธรรมชาติที่ไม่เกิด damp*** ของระบบ ซึ่งชื่อนี้เกิดขึ้นเพราะใช้แทนความถี่ของการแกว่งของระบบสำหรับกรณีที่ไม่มีการ damp(ζ=0)

มีกรณีย่อยๆ ที่สำคัญ 3 กรณี ดังแสดงในตาราง 2.2 กรณีที่ ζ < 0 ค่าจะถูกตัดทิ้งไปเนื่องจากเป็นการตอบสนองต่อระบบ second-order ที่ไม่เสถียรภาพ ซึ่งจะมีผลตอบสนองต่ออินพุทใดๆ อย่างไม่มีขอบเขตจำกัด transfer function แบบ second-order ส่วนใหญ่ที่ปรากฏในตารางบ่อยๆ จะเป็นแบบ overdamped และ critically damped เมื่อระบบแบบ first-order 2 ชุดต่ออนุกรมกัน จะได้ transfer function ดังสมการที่ (21) และ (22) ซึ่งแตกต่างตรงส่วนที่เป็นตัวหารเท่านั้น การทำให้สมการทั้งสองเท่ากันเพื่อให้เกิดความสัมพันธ์ระหว่างรูปแบบที่เลือกได้ สำหรับกรณี Second-order แบบ overdamped สังเกตได้ว่า เมื่อ ζ >=1 แล้วตัวหารของสมการที่(22) สามารถถูกทำให้เป็น factor ได้ ดังสมการ

$$\tau^2 s^2 + 2\zeta\tau s + 1 = (\tau_1 s + 1)(\tau_2 s + 1) \qquad (23)$$

กระจายนิพจน์ทางขวามือของสมการ และให้สัมประสิทธิ์ของ s เท่ากัน จะได้

$$\tau^2 = \tau_1\tau_2$$

$$2\zeta\tau = \tau_1 + \tau_2$$

ซึ่งจะได้

$$\tau = \sqrt{\tau_1\tau_2} \qquad (24)$$

$$\zeta = (\tau_1 + \tau_2)/2\sqrt{\tau_1\tau_2} \qquad (25)$$

หรืออีกทางหนึ่ง เมื่อด้านซ้ายของสมการ (23) ถูกทำให้เป็น factor ดังนี้

$$\tau^2 s^2 + 2\zeta\tau + 1 = (\tau s/\zeta - \sqrt{\zeta_2 - 1}) \qquad (26)$$

ซึ่งจากสมการจะได้ τ_1 และ τ_2 เป็น

$$\tau_1 = \tau/(\zeta - \sqrt{\zeta^2 - 1}) \qquad (\zeta \geq 1) \qquad (27)$$

$$\tau 2 = \tau/(\zeta + \sqrt{\zeta 2 - 1}) \qquad (\zeta \geq 1) \qquad (28)$$

Case	*Range of Damping Coefficient*	*Characterization of Response*	*Roots of Characteristic Equation*
a	$\zeta > 1$	Overdamped	Real and unequal
b	$\zeta = 1$	Critically damped	Real and equal
c	$0 \leq \zeta < 1$	Underdamped	Complex conjugates (of the form $a + jb$ and $a - jb$)

ตารางที่ 2 ฟังก์ชันในการเปลี่ยนแปลง Second-Oder 3 รูปแบบ

จากของไหลหรือ process อื่นๆ เช่น ท่อเครื่องมือวัดแบบนิวแมติค (ลม) ที่มีความจุน้อยมากๆ หรือจาก มาโนมิเตอร์แบบปรอท ซึ่งผลของแรงเฉื่อยมีความสำคัญ

ผลตอบสนองแบบ 1 step

สำหรับอินพุทแบบ 1 step ที่มีรูปแบบดังสมการที่ (12) คือ

$$MV(s) = M/S \qquad (29)$$

จะได้

$$PV(s) = MK/S(\tau^2 s^2 + 2\zeta\tau + 1) \qquad (30)$$

หลังจากการจัดการและแปลงให้อยู่ในรูปของ time domain แล้ว จะได้ผลตอบสนอง 3 แบบ คือ

case a($\zeta > 1$)

ถ้าตัวหารของสมการที่ (13) เป็นค่า factor ที่ใช้ดังสมการ (10) และ (11) แล้วผลตอบสนองจะเป็น

$$PV(t) = MK((1-\tau_1 e^{-t/\tau 1} - \tau_2 e^{-t/\tau 2})/(\tau_1 - \tau_2)) \qquad (31)$$

ถ้าตัวหารของสมการที่ (13) ไม่ใช่ค่า factor ทางซ้ายแล้ว ผลตอบสนองจะเขียนเป็น

$$PV(t) = MK\{ 1 - e^{-t/\tau} [\cosh(\sqrt{\zeta^2 - 1}\, t/\tau) + (\zeta/\sqrt{\zeta^2 - 1}) \sinh(\sqrt{\zeta^2 - 1}\, t/\tau) \qquad (32)$$

case b($\zeta = 1$) จะได้

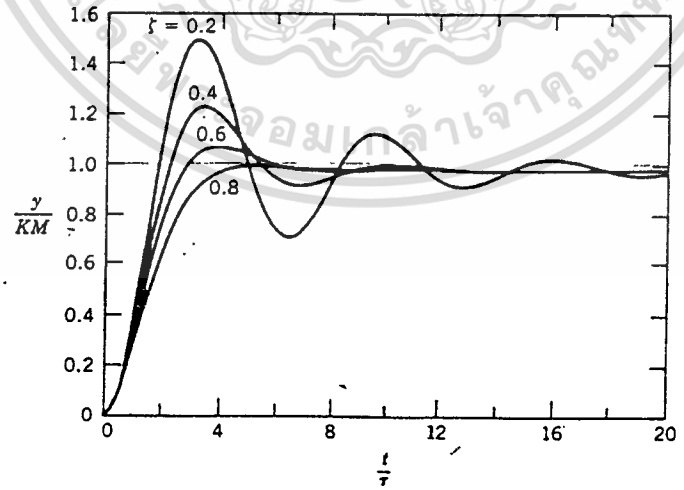
$$PV(t) = MK\,[\,1 - (\,1 + t/\tau)e^{-t/\tau}\,] \qquad (33)$$

case c($0 <= \zeta >= 1$) จะได้

$$PV(t) = MK\,\{\,1 - e^{-\zeta t/\tau}\,[\,\cos(\sqrt{1-\zeta^2}\,t/\tau) + \zeta/\sqrt{1-\zeta^2}\,\sin(\sqrt{1-\zeta^2}\,t/\tau)]\} \qquad (34)$$

curve ของผลตอบสนองแบบ step สำหรับค่า ζ ต่างๆ กัน แสดงได้ดังรูป 2.3 และ 2.4 abscissas ถูกทำให้เป็นปกติในส่วนของ t เมื่อ τ มีค่าน้อยๆ แสดงว่าผลตอบสนองเป็นไปอย่างรวดเร็ว หมายความว่า ค่าความถี่ธรรมชาติที่ไม่มี damp , $W_n = 1/\tau$ จะมีค่ามาก ข้อสังเกตทั่วๆ ไป ที่มีส่วนเกี่ยวข้องกับ ผลตอบสนองดังแสดงในรูป 2.3 และ 2.4

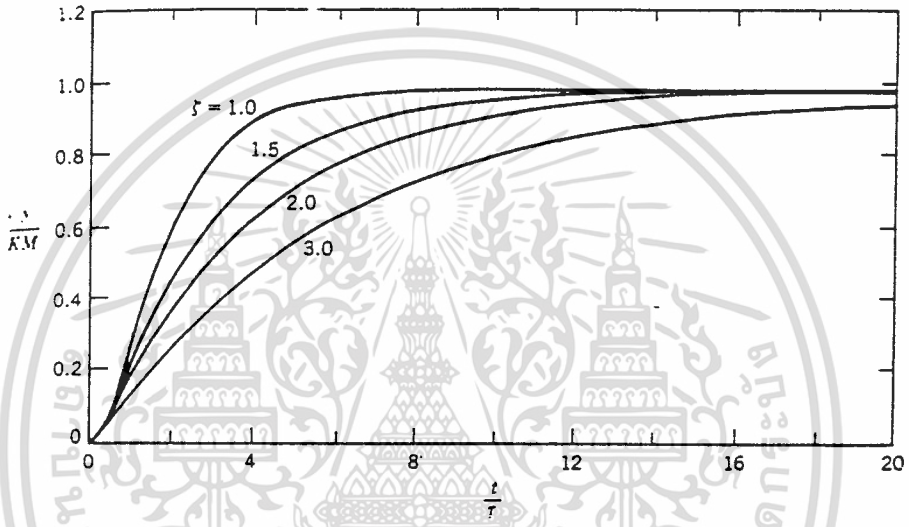
1. ผลตอบสนองแสดงให้เห็นการแกว่งและการ overshoot ($PV/KM > 1$) จะเกิดขึ้นเมื่อค่าของ $\zeta < 1$ เท่านั้น
2. ค่า ζ ที่มีค่ามากทำให้ผลตอบสนองช้าลง
3. ผลตอบสนองที่เร็วที่สุด โดยไม่เกิดการ overshoot จะเกิดขึ้นเมื่อเป็นกรณี critically damped($\zeta = 1$)



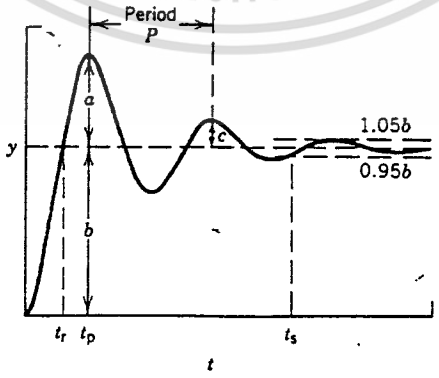
รูปที่ 18 ผลตอบสนอง Step สำหรับ Underdamped Second-Order Process

ผู้ออกแบบระบบควบคุมจะพยายามทำให้ ผลตอบสนองแบบ stepที่ set point

ของตัวแปรที่ถูกควบคุมใกล้เคียงกับผลตอบสนองแบบ step ของระบบ second-order แบบ underdamped นั่นคือ ทำให้มันแสดงให้เห็นถึงจำนวนที่ถูกกำหนดไว้ของการ overshoot และการแกว่งขณะอยู่ที่ operating point ใหม่ ค่าของ ζ ในช่วง 0.4-0.8 ซึ่งเหมาะสำหรับการระบุถึงผลตอบสนองของระบบควบคุมที่ต้องการ สมมุติว่ามันถูกประมาณให้เป็นระบบ second-order แบบ underdamped ในช่วงนี้ๆ ตัวแปรที่ถูกควบคุม y จะเข้าใกล้ operatint point ใหม่ซึ่งเร็วกว่าที่ค่า ζ = 1.0 หรือ 1.4 แต่ผลตอบสนองจะมีการแกว่ง (ตั้งค่าได้เร็วกว่า) น้อยกว่าที่ค่า ζ = 0.2



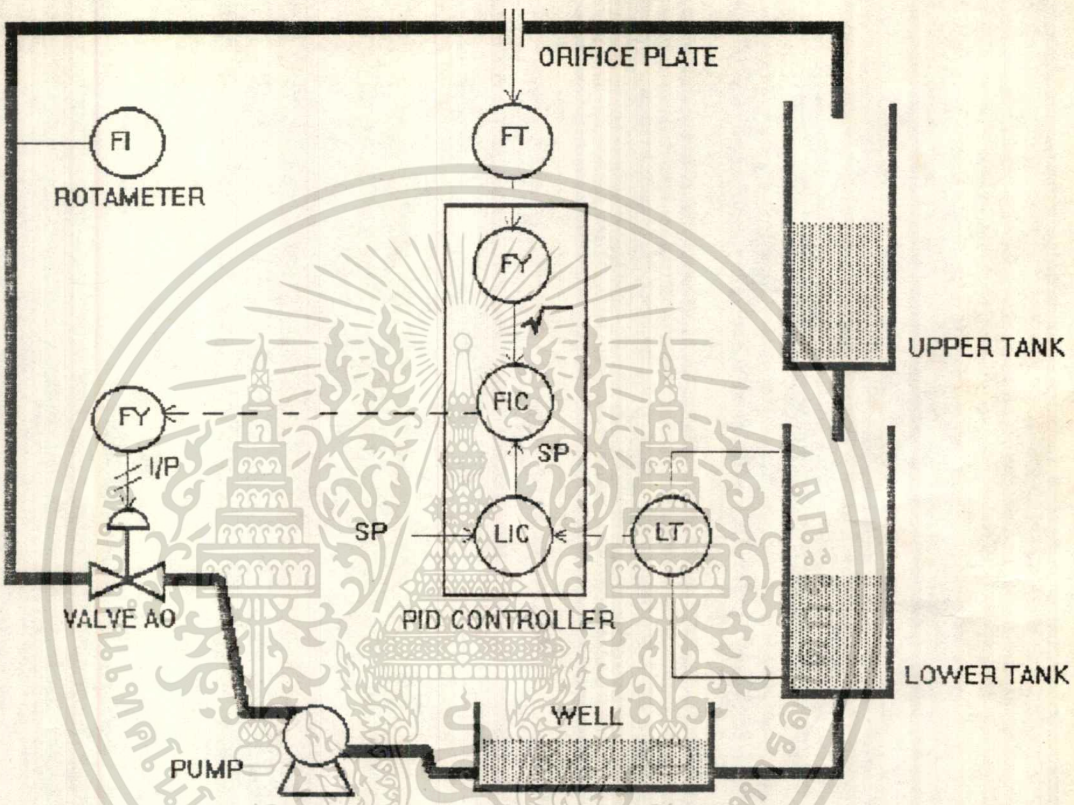
รูปที่ 19 ผลตอบสนองแบบ Step สำหรับ Critically-damped และ Overdamped Second Order Process



รูปที่ 20 คุณลักษณะที่แสดงสำหรับผลตอบสนองแบบ Step สำหรับ Underdamped Processes

กรณีศึกษา (Case Study)

การควบคุมและจำลองกระบวนการควบคุมแบบ cascade ของระดับน้ำและอัตราการไหล



รูปที่ 21 กรณีศึกษาของการควบคุมและจำลองกระบวนการควบคุมแบบ **Cascade** ของระดับน้ำ และอัตราการไหล

อุปกรณ์ที่ใช้

1. PID controller & simulation

   2 Loop cascade

   Loop 1 : Primary Controller (Master) = Level

   Reverse Action : Level เพิ่มขึ้น Flow ลดลง รับมาจาก A/D Channel 5

   Loop 2 : Secondary Controller (Slave) = Flow

   (Square Root Extraction)

Reverse Action : Flow เพิ่มขึ้น Valve Position ลดลง Air to Open ต้องลด Air ลง รับมาจาก A/D Channel 6

2. I/P (current to pressure converter)

Input Signals 4-20 mA รับมาจาก Controller Channel 8

Output Signals 3-15 psi

3. Valve Size 1 CV 3.9

Rating Ansi 150 Fail Close (Air to Open)

4. Flow Indicator : Rotametor

5. Flow Transmitter Calibration range 0-100 mmH2O

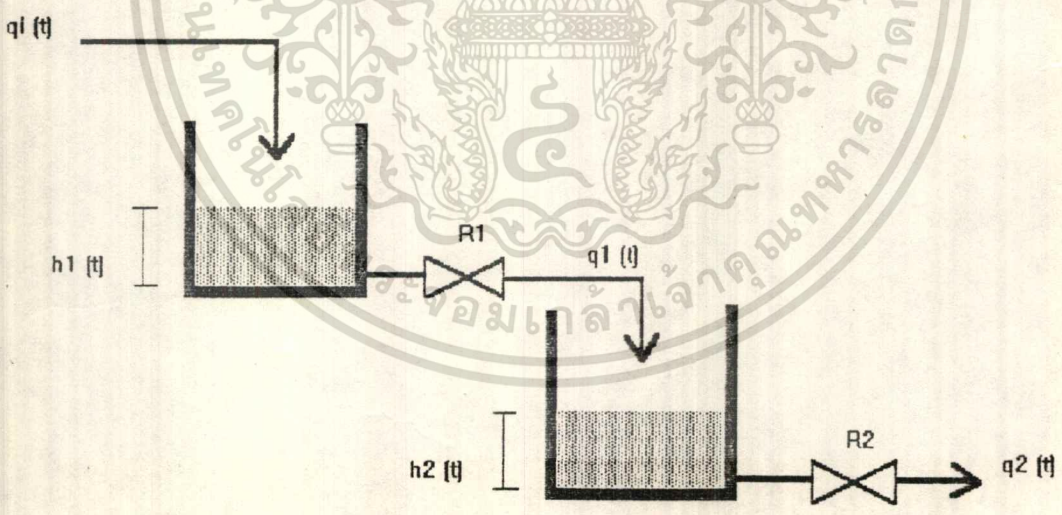
Supply Dc 24 V Output DC 4-20 mA

6. Level Transmitter Calibration range 0-1000 mmH2O

Supply DC 24 V Output DC 4-20 mA

7. Others tank, well, pump, pipe etc.

การ Simulate Process ของ Level และ Flow



รูปที่ 22 การ Simulate Process Loop 1 (Master) โดยเป็นการควบคุม Level

พิจารณา tank1 , $q_i(t) - q_1(t) = A_1 \, dh_1(t)/dt$ (35)

$$q_1(t) = (1/R_1) \, h_1(t) \quad (36)$$

$$q_i(t) - (1/R_1) \, h_1(t) = A_1 \, dh_1/dt \quad (37)$$

สมการ differential equation จาก steady state

$$(q_i(t) - q_i) - (1/R_1)(h_1(t) - h) = A_1\, d(h_1(t) - h)$$

ใช้ตัวแปร derivation

$$Q_i(t) = q_i(t) - \overline{q_i}$$

$$H_1(t) = h_1(t) - \overline{h}$$

ใช้ $Q_i(t) - (1/R_1)H_1(t) = A_1\, dH_1(t)/dt$

$$R_1A_1\, dH_1(t)/dt + H_1(t) = R_1Q_i(t)$$

take laplace

$$H_1(s)/Q_i(s) = R_1/R_1A_1s + 1 \qquad (38)$$

$$= K_1/\tau_1 s + 1$$

เมื่อ $K_1 = R_1$, $\tau_1 = R_1A_1$

และจากสมการ (2) ทำแบบเดียวกันได้

$$Q_1(s)/H_1(s) = (1/R_1) = (1/K_1) \qquad (39)$$

ในทำนองเดียวกันกับ tank 2

$$H_2(s)/Q_1(s) = R_2/R_2A_2s + 1 \qquad (40)$$

$$= K_2/\tau_2 s + 1$$

เมื่อ $K_1 = R_2$

$\tau_2 = R_2A_2$

จากสมการ (4), (5) , (6) เปรียบเทียบ $H_2(s)/Q_i(s)$

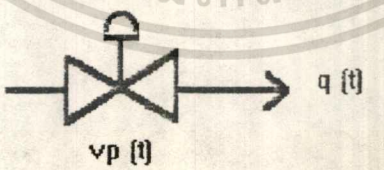
$$(H_2(s)/Q_1(s))(Q_1(s)/H_1(s))(H_1(s)/Q_i(s))$$

$$= (K_2/\tau_2 s + 1)(1/K_1)(K_1/\tau_1 s + 1)$$

$$= K_2/(\tau_2 s + 1)(\tau_1 s + 1) \qquad (41)$$

จากสมการของไหลบน valve ได้

รูปที่ 23 การ Simulate Process Loop 2 (Slave) โดยเป็นการควบคุม Flow



$$q(t) = Cv(vp(t))\sqrt{\Delta P/G} \qquad (42)$$

สมการ differential equation

$$q(t) - q = Cv\sqrt{\Delta P/G}\,(vp(t) - vp)$$

ตัวแปร deviation

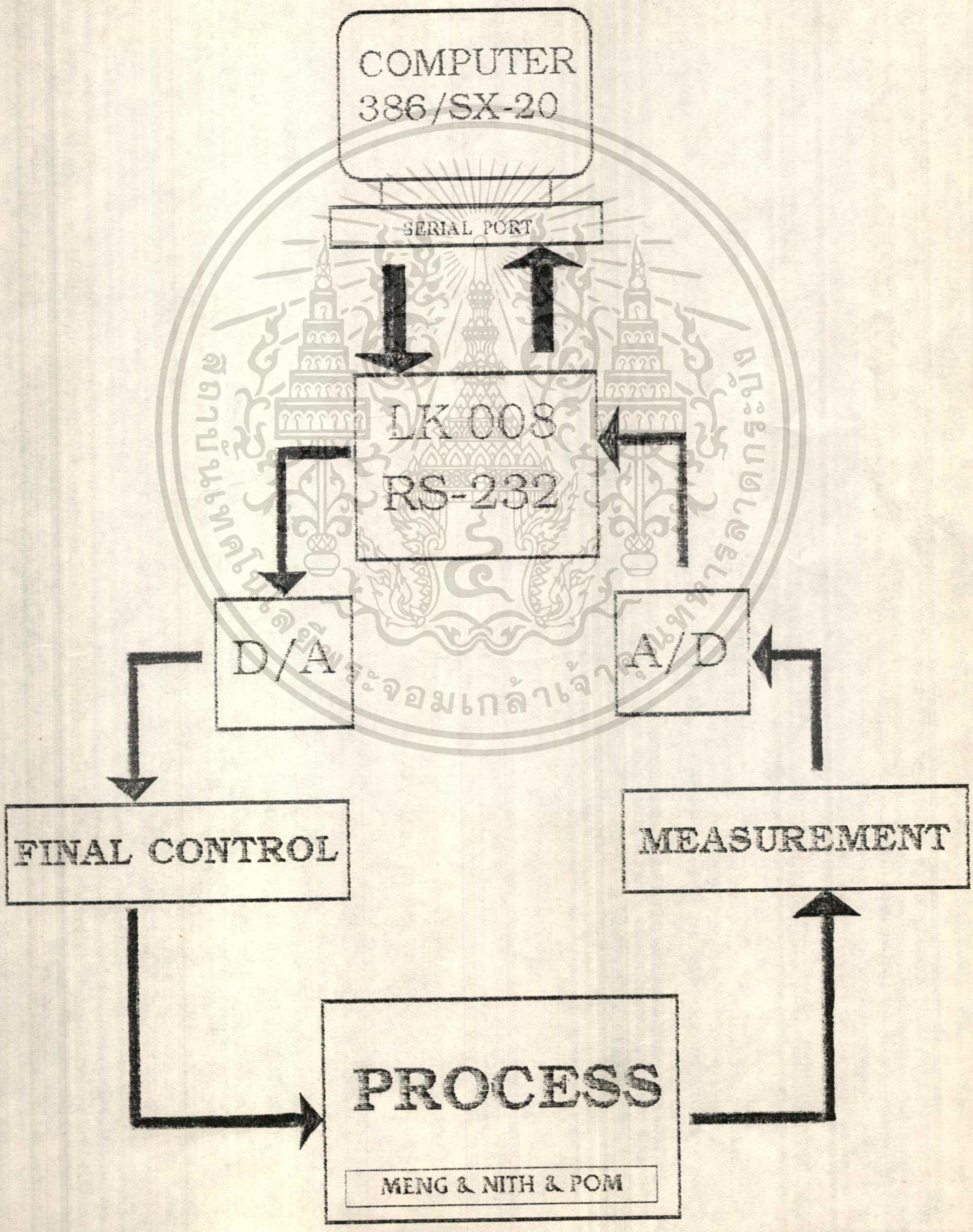
$$Q(t) = Cv\sqrt{\Delta P/G}\; vp(t)$$

take laplace $Q(s)/vp(s) = Cv\sqrt{\Delta P/G} = K_3 \qquad (43)$

ภาค 2

การจัดการและติดต่อ Hardware.

BLOCK DIAGRAM



HARDWARE

ส่วนประกอบที่สำคัญของโปรแกรมระบบควบคุม และจำลองการทำงาน แบบ PID โดยใช้ภาษาซี ประกอบไปด้วย

1. ส่วนประมวลผล (CPU)
2. ส่วนเชื่อมต่อกับอุปกรณ์ภายนอก (I/O INTERFACE)
3. อุปกรณ์แปลงสัญญาณอนาลอกเป็นดิจิตอล (A/D)
4. อุปกรณ์แปลงสัญญาณดิจิตอลเป็นอนาลอก (D/A)

ส่วนประมวลผล

โครงงานชิ้นนี้เราได้เลือกใช้ personal computer (PC) ที่มี CPU 386sx-20 เป็นตัวประมวลผล มีจอภาพ (monitor)SVGA และ Keyboard เป็นอุปกรณ์ติดต่อกับผู้ใช้งาน (User Interface) เพื่อสะดวกในการใช้งาน

ในการควบคุมกระบวนการทั่วไป จำเป็นต้องมีการติดต่อกับอุปกรณ์ภายนอกโดยผ่านพอร์ตอนุกรม (serial port)

ส่วนเชื่อมต่อกับอุปกรณ์ภายนอก

ทางด้านที่เชื่อมต่อกับเครื่องคอมพิวเตอร์ เราใช้ serial port com-1 ในการติดต่อ ส่วนอีกด้านหนึ่งจะเชื่อมต่อกับอุปกรณ์ชุด LK-008 ของ OMRON ซึ่งเป็นอุปกรณ์ที่ทำหน้าที่แปลงสัญญาณที่ได้รับจากเครื่องคอมพิวเตอร์ผ่าน RS-232 ให้เป็นสัญญาณที่ PLC สามารถเข้าใจได้

PLC ที่ใช้เป็นรุ่น SYSMAC C-500 ของ OMRON ซึ่งการติดต่อระหว่างเครื่องคอมพิวเตอร์และ PLC จำเป็นต้องมีการส่งข้อมูลที่ PLC สามารถเข้าใจได้ว่าเครื่องคอมพิวเตอร์สั่งงาน (commands) อะไรให้ PLC และPLC มีการตอบสนอง (Response) อย่างไรต่อคำสั่งที่ได้รับ

อุปกรณ์แปลงสัญญาณอนาลอกเป็นดิจิตอล

อุปกรณ์ที่ใช้งานเป็นรุ่น SYSMAC-C series ขนาดความละเอียด 12-bit ซึ่งเราจะสามารถเลือกสัญญาณอินพุท (input signal) ได้ว่า จะเป็นกระแสหรือโวลท์เตจ โดยมีช่วงการใช้งานดังนี้

สัญญาณอินพุทที่เป็นกระแส มีช่วงการใช้งาน 4~20mA

สัญญาณอินพุทที่เป็นโวลท์เตจ มีช่วงการใช้งาน 0~10V, 0~5V, +1~+5V, -5~+5V หรือ -10~+10V

อุปกรณ์แปลงสัญญาณนี้สามารถเลือกใช้งานได้ 2 ช่อง (channels) โดยอุปกรณ์แปลงสัญญาณนี้จะทำงานโดยรับสัญญาณอนาลอกที่ได้รับจาก transmitter มาแปลงเป็นสัญญาณดิจิตอลที่ PLC สามารถเข้าใจได้เพื่อจะนำค่าที่ได้รับนี้ส่งไปให้เครื่องคอมพิวเตอร์ทำการประมวลผลต่อไป

อุปกรณ์แปลงสัญญาณดิจิตอลเป็นอนาลอก

อุปกรณ์ที่ใช้งานเป็นรุ่น SYSMAC-C series ขนาดความละเอียด 12-bit ซึ่งเราสามารถเลือกสัญญาณเอาท์พุท (output signal) ได้ว่าจะเป็นกระแสหรือโวลท์เตจ โดยมีช่วงการใช้งานดังนี้

สัญญาณเอาท์พุทที่เป็นกระแสมีช่วงการใช้งาน 4~20 mA

สัญญาณเอาท์พุทที่เป็นโวลท์เตจมีช่วงการใช้งาน 0~10V, 0~5V, +1~+5V, -15~+15V, -10~+10V สามารถใช้งานได้ 2 ช่อง (channels)

อุปกรณ์แปลงสัญญาณนี้จะทำงานโดยรับสัญญาณดิจิตอลที่ได้รับจาก PLC มาแปลงเป็นสัญญาณอนาลอกที่สามารถนำไปควบคุม FINAL ELEMENT ได้

Bios Com

ในการสื่อสารแบบอะซิงโครนัสนั้น ข้อมูลจะถูกส่งผ่านพอร์ตอนุกรม ครั้งละ 1 บิต ซึ่งจะแตกต่างจากการส่งแบบขนานที่จะส่งครั้งละ 1 ไบต์

การส่งข้อมูลผ่านพอร์ตอนุกรมนั้น ข้อมูลแต่ละไบต์ จะประกอบด้วย

1. บิตเริ่มต้น 1 บิต
2. บิตข้อมูล 7 หรือ 8 บิต
3. พาริตี้บิต จะมีหรือไม่มีก็ได้
4. บิตสิ้นสุด 1 หรือ 2 บิต

สถานะของสายส่งในขณะที่ไม่มีข้อมูล จะมีสถานะเป็นสูง ข้อมูลบิตใดมีค่า 0

จะทำให้สายส่งมีสถานะต่ำ ข้อมูลใดมีค่า 1 ก็จะทำให้สายส่งมีสถานะสูงอยู่เช่นเดิม

ส่วนบิตเริ่มต้นใช้สำหรับบอกจุดเริ่มต้นของไบต์ข้อมูล โดยการทำให้สถานะของสายส่งมีค่าต่ำเป็นเวลา 1 รอบ (cycle) จากนั้นจะเป็นบิตของข้อมูล

และตามด้วยพาริตี้บิตซึ่งจะมี หรือไม่ก็ได้ สุดท้ายคือบิตสิ้นสุด ซึ่งจะมี 1 หรือ 2 บิตก็ได้ ขึ้นกับว่าจะใช้เท่าใด

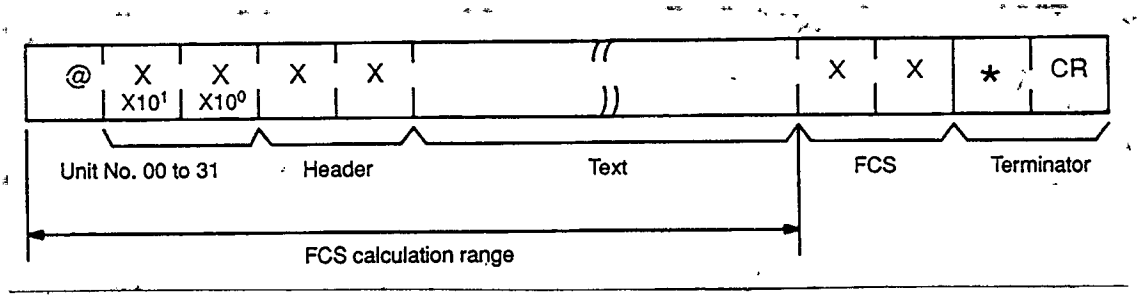
พาริตี้บิต ถ้าหากมีในไบต์ข้อมูล ก็จะทำหน้าที่ตรวจเช็คความผิดพลาดของข้อมูล พาริตี้บิตมีค่า 2 อย่างคือ เป็นคู่ หรือ คี่ (even or odd) ถ้าเป็นคู่ หมายความว่า เมื่อรวมค่าพาริตี้บิตแล้ว จำนวนของบิตข้อมูลที่มีค่า 1 จะเป็นจำนวนคู่ และถ้าพาริตี้บิตเป็นคี่ หมายความว่าเมื่อรวมพาริตี้แล้ว จำนวนของบิตข้อมูลที่มีค่าเป็น 1 จะเป็นจำนวนคี่

อัตราการส่งข้อมูลมีหน่วยเป็น baud (bit per second) ค่า baud rate ที่ต่ำที่สุดที่มีใช้กัน คือ 300 baud ซึ่งจะใช้กับโมเด็มรุ่นเก่า (โมเด็มรุ่นใหม่มักจะใช้ 1200-2400 baud) ส่วนเครื่องคอมพิวเตอร์ระดับ IBM PC สามารถใช้ค่า baud rate ได้สูงถึง 9600 baud rate baud rate มีความสำคัญอย่างมากในการรับส่งข้อมูล ระหว่าง PLC และเครื่องคอมพิวเตอร์ เพราะทำให้การประมวลผลรวดเร็วและการควบคุมได้ดียิ่งขึ้น ซึ่งในโครงงานนี้ใช้ hand raate 9600 bps. พาริตี้คู่, จำนวนบิตในข้อมูล 1 ไบต์ เท่ากับ 7 บิต,จำนวนของบิตสั้นสุดเท่ากับ 2 บิต

band rate มีความสำคัญ เนื่องจาก การทำงานของพอร์ตอนุกรม เมื่อพอร์ตได้รับบิตเริ่มต้น ก็จะสุ่มอ่านค่าจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบเพื่ออ่านบิตต่อไป ถ้าหากเครื่องคอม พิวเตอร์และ PLC มี band rate ที่ต่างกันจะเป็นผลให้การรับสัญญาณข้อมูลผิดพลาดได้ ซึ่งเรียกว่า **framing error**

เมื่อเราทำการตั้งค่า band rate ของเครื่องคอมพิวเตอร์ให้เท่ากันกับ PLC แล้ว อุปกรณ์ทั้งสองจะทำ Hardware Handshaking ซึ่งเป็นการตรวจสอบสถานะของอุปกรณ์ตัวรับข้อมูลว่า พร้อมที่จะรับข้อมูลหรือไม่ เมื่ออุปกรณ์ ตัวส่งข้อมูลต้องการข้อมูล ดังนั้นข้อมูลจะต้องไม่ถูกส่งออกไป จนกว่าสัญญาณพร้อมรับข้อมูลจะถูกส่งกลับมาจากอุปกรณ์ตัวรับข้อมูล

PLC จะมีโปรโตคอลในการรับส่งข้อมูลดังนี้



Unit no. จะถูกกำหนดให้เป็น 00

Header สำหรับการอ่านข้อมูลจาก A/D จะเป็น RR

Header สำหรับการเขียนข้อมูลไปยัง D/A จะเป็น WR

FCS คือ ข้อมูลขนาด 8 บิตที่ถูกแปลงเป็นอักขระ ASCII 2 อักขระ ซึ่งได้จากการ Exclusive OR อักขระของโปรโตคอลทีละ 1 อักขระเรียบตามลำดับจากอักขระตัวแรกจนสิ้นสุด

อักขระตัวสุดท้ายของอักขระของข้อความ (Text) (*) เป็นอักขระ (*) และ CR เป็นรหัส ASCII ของ Enter

ภาค 3

โปรแกรมการควบคุมทั้งหมด.

ในการทำงานของโปรแกรมระบบควบคุม และจำลองการทำงานแบบ P I D ด้วยภาษาซีนั้น เราเลือกใช้ compiler ของ TURBO C++ version 1.00 การออกแบบโปรแกรมให้มีการจัดการทางด้าน graphics หน้าจอในส่วนของ user's interface และมีการทำงานตามทฤษฎีของระบบควบคุมแบบ PID โปรแกรมประกอบไปด้วย function ต่างๆ ที่สำคัญ แบ่งเป็น 3 ส่วนใหญ่ๆ ดังนี้

I. Function ส่วนการจัดการหน้าจอผู้ใช้

1) **Initialize-Graphics-Mode ( )** เป็น function ที่ตรวจสอบสถานะของเครื่องคอมพิวเตอร์ว่าสามารถจะแสดงผลในลักษณะของ Graphics Mode ได้หรือไม่ ถ้าไม่ได้ function ก็จะส่งค่า error ออกมา

2) **CLS ( )** เป็น function ที่ใช้ในการลบหน้าจอ

3) **Save-Screen ( )** เป็น function ที่ใช้เก็บหน้าจอบางส่วนไปไว้ในส่วนความจำที่จัดไว้ให้

4) **Restore-Screen ()** เป็น function คืนหน้าจอที่เก็บไว้ให้แก่โปรแกรมโดยจะคืนหน้าจอ ตั้งแต่จุดที่เก็บค่าเริ่มต้นตำแหน่งไว้จะหมด

5) **Menu-Assignment ( )** เป็น function กำหนดค่าต่างๆ ในการสร้าง Menu

6) **Display-Main-Menu ( )** แสดง menu หลัก

7) **Display-Menu ( )** แสดงเมนูย่อยๆ แต่ละเมนู

8) **Select-Menu ( ) และ Select-Submenu ( )** เป็น function รับการเลือกค่าจากผู้ใช้ในการทำงาน

9) **Draw-Fill-Rectangle ( ) , Inverse ( ) และ Normal ( )** เป็น function ประกอบในการจัดการกับเมนู

10) **P-Error ( )** เป็น function แสดงค่า error ที่ตรวจพบในการทำงานของโปรแกรม

11) **Time ( )** เป็น function แสดงเวลาที่เริ่มการทำงานของโปรแกรม

12) **Example-Screen ( )** เป็น function แสดงหน้าจอการทำงาน

13) **Bar-Graph ( )** เป็น function นำค่าของตัวแปรมาแสดงผลเป็นลักษณะ graph เพื่อให้ผู้ใช้สามารถเข้าใจได้ง่าย

14) **Display-Data ( )** เป็น function แสดงค่าตัวแปรต่างๆ ที่ใช้ในการทำงานบนหน้าจอ

15) **Title ( )** เป็น function นำผู้ใช้เข้าสู่โปรแกรม

II. FUNCTION จัดการเกี่ยวกับ File

การออกแบบโปรแกรมถูกออกแบบให้มีการเก็บค่าตัวแปรควบคุมต่าง ๆ ที่ใช้ในการทำงาน ลงบน file ที่ชื่อว่า Data.Fil ซึ่งเมื่อเริ่มการทำงานโปรแกรมจะทำการตรวจสอบ File นี้ก่อนเพื่อนำค่าของข้อมูลเดิมมาให้แก่ User ทำให้ไม่เสียเวลาในการคำนวณค่าเริ่มต้นต่าง ๆ ใหม่อีก

ซึ่ง Function การจัดการเกี่ยวกับ File ประกอบด้วย

1) **File-Write ()** นำค่าตัวแปรที่ใช้ในการควบคุมครั้งล่าสุด (ก่อนเปิดเครื่อง) บันทึกลง File Data.Fil

2) **File-Read ()** อ่านค่าตัวแปรเดิมจาก File Data.Fil มาเก็บเป็นตัวแปรของโปรแกรม เพื่อรอการทำงาน

3) **Waiting ()** เป็น function ให้ผู้ใช้รอขณะที่กำลังอ่าน File มีสีสันสวยงาม

4) **Ini-Data()** กรณีที่โปรแกรมไม่สามารถหา File Data.Fil พบ ก็จะอ่านค่าจาก function Ini-Data ( ) ที่ทางผู้ผลิตได้บันทึกทั่วไปของโปรแกรมไว้แล้ว เมื่อเริ่มต้นในการทำงาน

III. FUNCTION การทำงานเกี่ยวกับการ Control และ Simulation

เป็นการเขียน Function ขึ้นมาเพื่อให้มีการควบคุมการทำงาน ตามทฤษฎีของการควบคุม มี Function ดังนี้

1) **Read-Key ( )** เป็น Function รับค่าการเคาะ Keyboard เพื่อดูว่าค่าที่รับมาเป็นค่าเท่าไร เพื่อนำมาประมวลผลอีกครั้ง

2) **R-D ( )** เป็น function อ่านค่าจากส่วนอุปกรณ์ Analog to digital converter รับค่า process variable (Pv.) จากกระบวนการควบคุม

3) **W-R ( )** เป็น function ส่งค่าที่มีการประมวลผลแล้วจาก function PID ( ) ออกไปยัง port COM 1 เพื่อเป็นค่าควบคุม process ผ่านทางอุปกรณ์ DAC (Digital to Analog Converter) โดยค่าที่ส่งออกไปเป็นค่า Manipulated Variable

4) **PID ( )** เป็น function หลักที่ใช้ในการประมวลผลการควบคุมแบบ PID Controller

5) **Run ( )** เป็น function ในการจัดการการประมวลผลทั้งหมดให้มีลักษณะวน loop

6) **Control-Loop ( )** เป็น function จัดการควบคุมการทำงานในกรณีที่ผู้ใช้เลือกการทำงานแบบ controller ทั้งแบบ 1 และ 2 loop

7) **Simulate-Loop ( )** เป็น function จัดการควบคุมการจำลองกระบวนการ เพื่อให้ผู้ใช้ได้ศึกษาระบบตัวอย่าง ของการควบคุม

8) **Simulate ( )** เป็น function หลักในการจำลองกระบวนการ

9) **Check-Keyboard ( )** เป็น function ที่คอยตรวจสอบการกด keyboard เพื่อขัดจังหวะระหว่างที่มีการควบคุมการทำงานแบบ Loop อยู่เพื่อให้โปรแกรมสามารถทำงานอย่างอื่นขณะที่กำลัง Control หรือ Simulate ได้

10) **Change-Data ( )** เป็น function ใช้ในการเปลี่ยนแปลงค่าตัวแปรต่างๆ ใน process ทั้งระหว่างการทำงานของโปรแกรมหรือก่อนการทำงานของโปรแกรม

11) **Change-Mode ( )** เป็น function ใช้ในการเปลี่ยน mode การควบคุมจากแบบอัตโนมัติ (Auto) เป็นแบบปรับด้วยตนเอง (Manual)

12) **Change-Action ( )** เป็น function ใช้ในการเปลี่ยนลักษณะการทำงานแบบ Direct Action เป็น Reverse Action หรือกลับกัน

ขั้นตอนการทำงานของโปรแกรม

การทำงานของโปรแกรมเริ่มจาก เมื่อเริ่มทำงานแล้ว โปรแกรมจะตรวจสอบลักษณะของจอภาพว่าสามารถใช้กับโปรแกรมนี้ได้หรือไม่ แล้วก็เป็นการสร้าง menu ให้ผู้ใช้ได้เลือกว่าต้องการทำงานในลักษณะใด ซึ่งแบ่งเป็น การทำงานแบบ controller และแบบ simulation ในแต่ละแบบจะสามารถทำงานได้ทั้งแบบ 1 loop หรือ 2 loop ซึ่งแบบ 2 loop สามารถเลือกได้อีกว่าเป็น 2 loop แบบ CASCADE หรือ แบบ SEPERATE ว่าจะให้ทั้ง 2 loop ทำงานสัมพันธ์กันหรือแยกอิสระต่อกัน

หลังจากนั้นก็เป็นการเลือกลักษณะของการควบคุมว่าจะเป็นแบบ direct action หรือ reverse action

เมื่อผู้ใช้เลือกค่าต่างๆ แล้วโปรแกรมก็จะเริ่มการทำงานตามที่ผู้ใช้เลือก โดยจะมีการอ่านค่าเริ่มต้นของตัวแปรต่างๆ จาก file Data.Fil ถ้าโปรแกรมไม่พบ file ข้อมูลนี้ก็จะทำการสร้างfileข้อมูล Data.Fil และนำค่าตัวแปรต่างๆจากตัวโปรแกรม set เป็นค่าเริ่มต้นเอง

ขณะที่โปรแกรมทำงานควบคุมตามรูปแบบที่ผู้ใช้ได้เลือกแล้วนั้น ผู้ใช้งานสามารถจะเปลี่ยนแปลงค่าต่างๆ ในโปรแกรมดังนี้

ปุ่ม **<F1>** เป็นการเลือกเมนูหลักว่าต้องการเปลี่ยนแปลงการทำงานหรือเปลี่ยนแปลงการทำงานหรือเปลี่ยนลักษณะการควบคุมตามตัวเลือกที่มีไว้ใช้

ปุ่ม **<F2>** ในกรณีที่เลือกการทำงานแบบ 2 loop เราสามารถกดปุ่ม F2 เพื่อเลือกพิจารณา แต่ละloop ได้ว่าจะเลือก loop ไหน ซึ่งแสดงผลโดยเป็นลักษณะกรอบสีขาวสว่าง

ปุ่ม **<F3>** เป็นปุ่มที่ใช้ในการเปลี่ยนค่าตัวแปรควบคุมในการทำงาน โดยเมื่อกดปุ่ม F3 แล้ว โปรแกรมจะแสดงค่าตัวแปรที่ต้องการเปลี่ยนโดยใช้ปุ่ม **<Page-Up>** หรือปุ่ม **<Page-Down>** เลือกตัวแปรและใช้ปุ่มลูกศรขึ้น **(Up-Arrow)** หรือปุ่มลูกศรลง **(Down-Arrow)**

ในการเพิ่มหรือลดค่าตัวแปร และเมื่อกด **<Enter>** แล้วโปรแกรมจะเริ่มทำงานตามค่าที่เปลี่ยนแปลงไป

ปุ่ม **<F4>** เป็นปุ่มการเลือกเปลี่ยน mode การควบคุมว่าเป็นแบบอัตโนมัติ **(auto mode)** หรือแบบกึ่งอัตโนมัติ **(manual mode)** โดยใช้ร่วมกับปุ่ม **<Left-Arrow>** และ **<Right-Arrow>** ในการเพิ่มหรือลดค่าของค่า manipulate variable (Mv.)

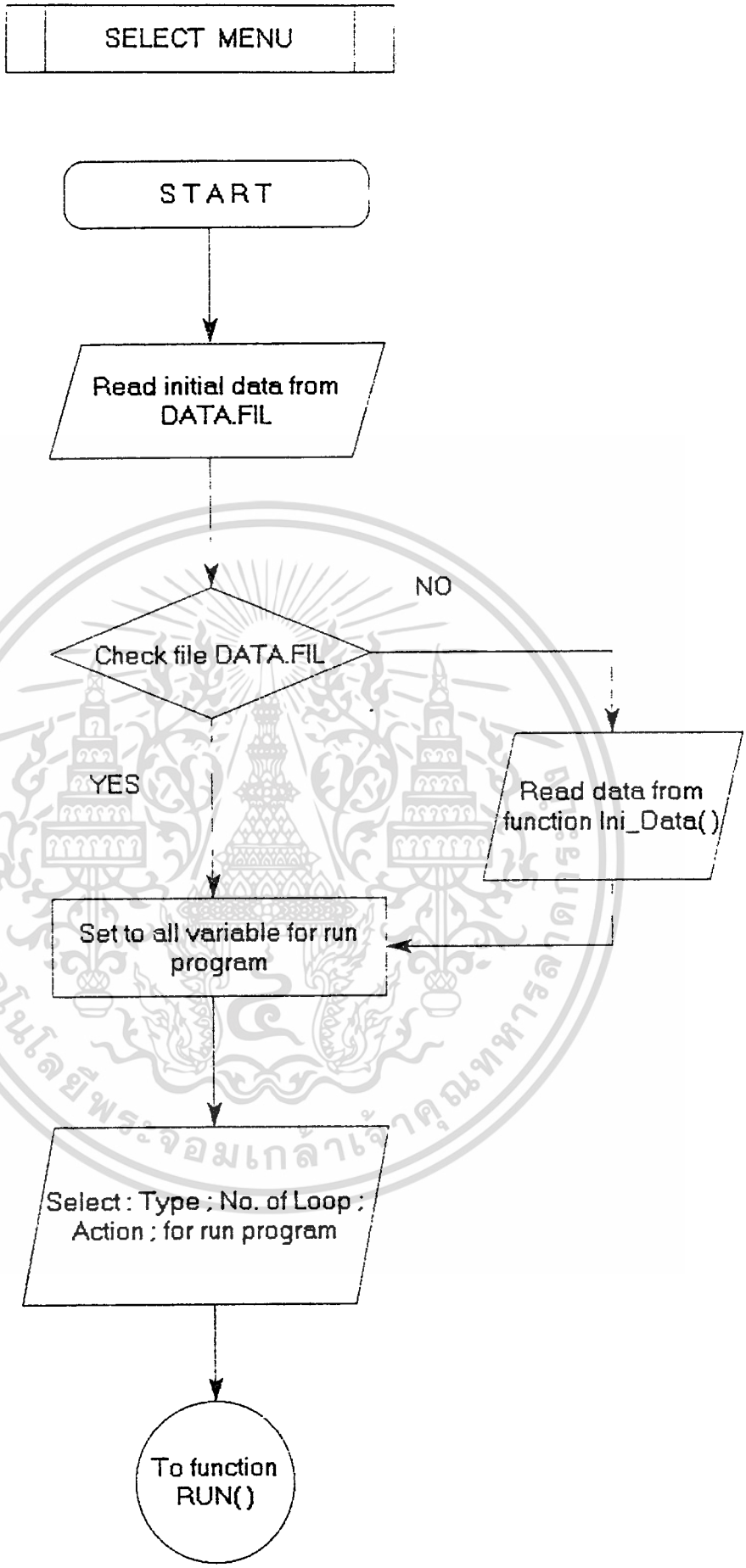
ปุ่ม **<F5>** เป็นปุ่มการเปลี่ยนลักษณะของการควบคุมว่าเป็นแบบ **direct action** ให้เป็นแบบ **reverse action** หรือกลับกัน

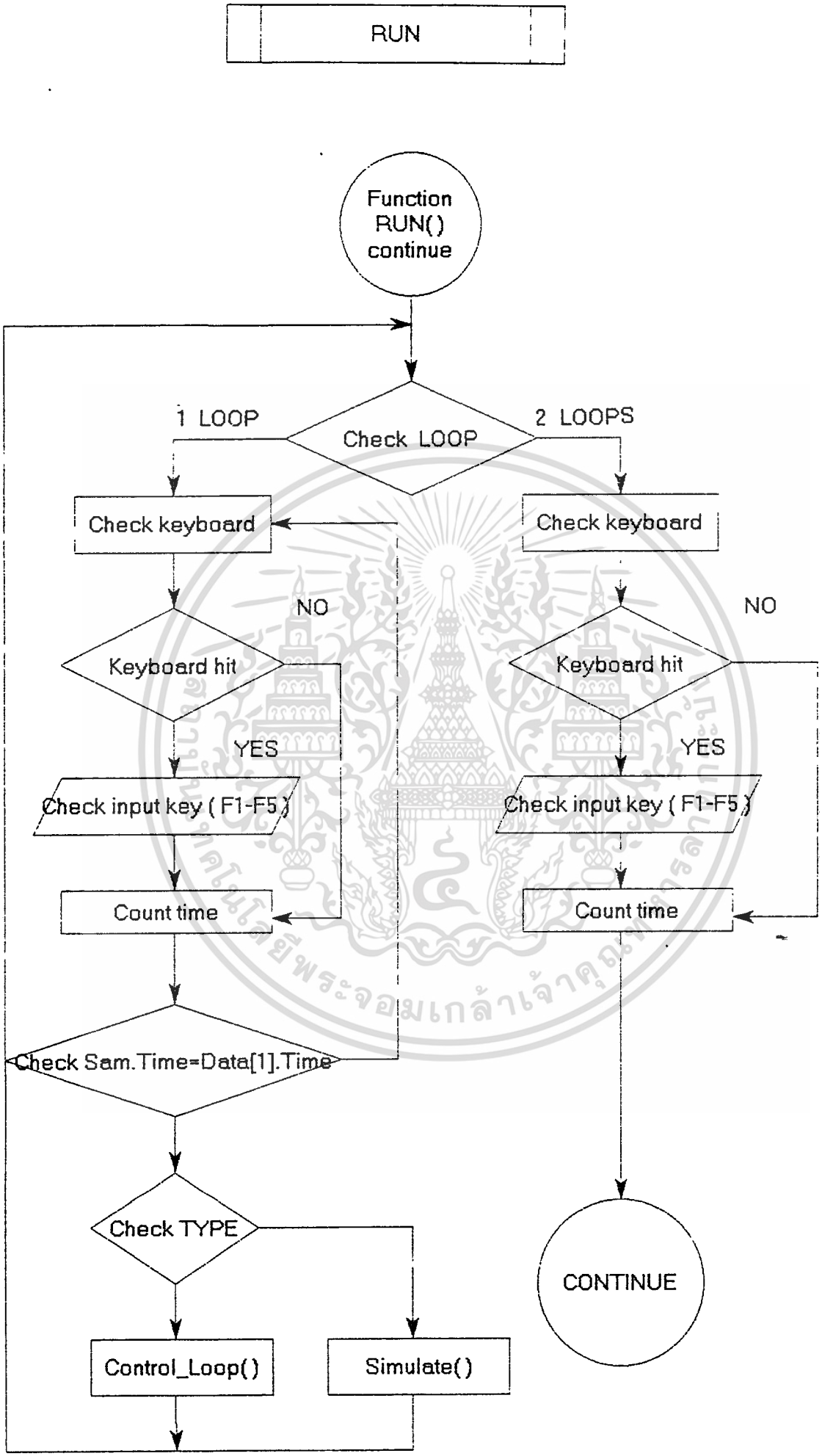
ปุ่ม **<ESC>** ใช้ในกรณีที่ยกเลิกการทำงาน

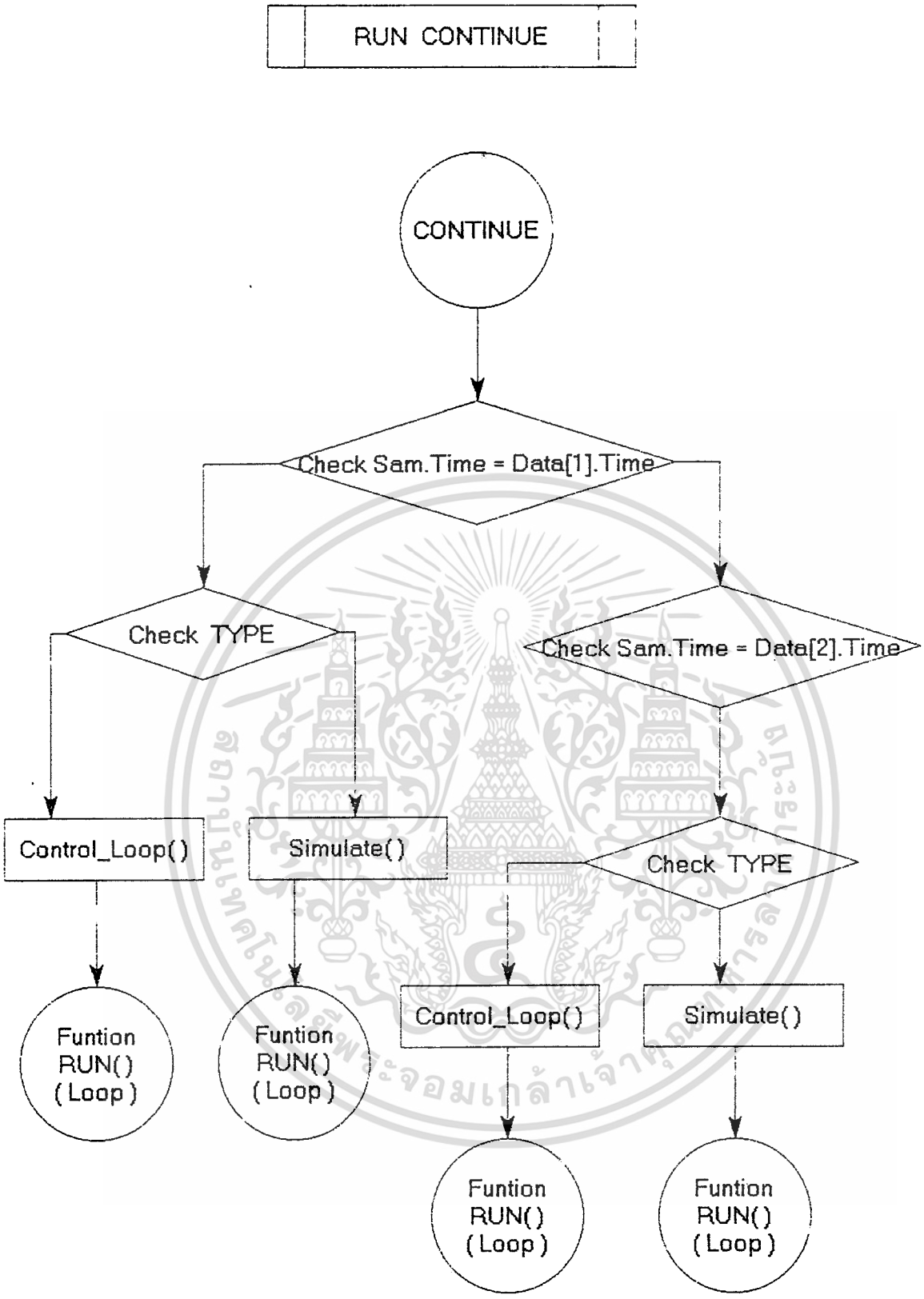
เมื่อผู้ใช้งานต้องการจะออกจากโปรแกรมทำได้โดยการเลือกเมนูหลักที่หัวข้อ **QUIT** โปรแกรมจะทำการบันทึกค่าตัวแปรต่างๆ ลง file Data.Fil แล้วออกจากการทำงาน

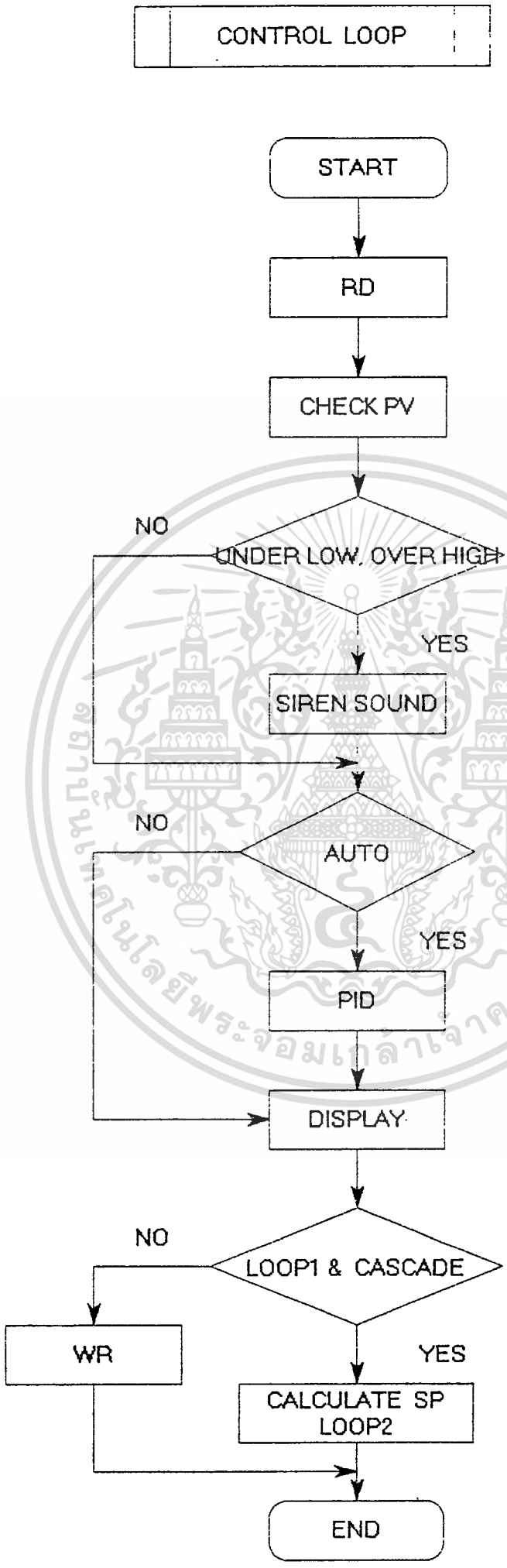
ตัวอย่างของโปรแกรม และ FLOW CHART การทำงาน

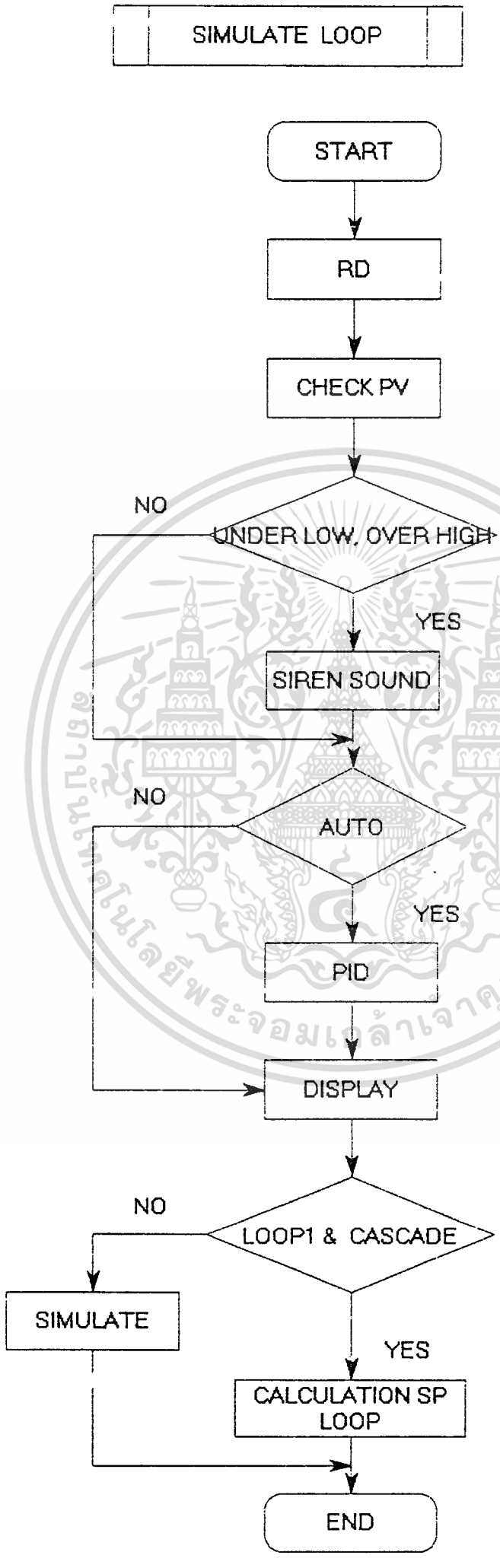
ในโปรแกรมนี้ได้รวม FUNCTION การทำงานทุกส่วนมารวมในโปรแกรม โดยเขียนแยกเป็นส่วน ๆ ให้สามารถเข้าใจได้ง่าย โดยโปรแกรมแสดงได้ดังนี้

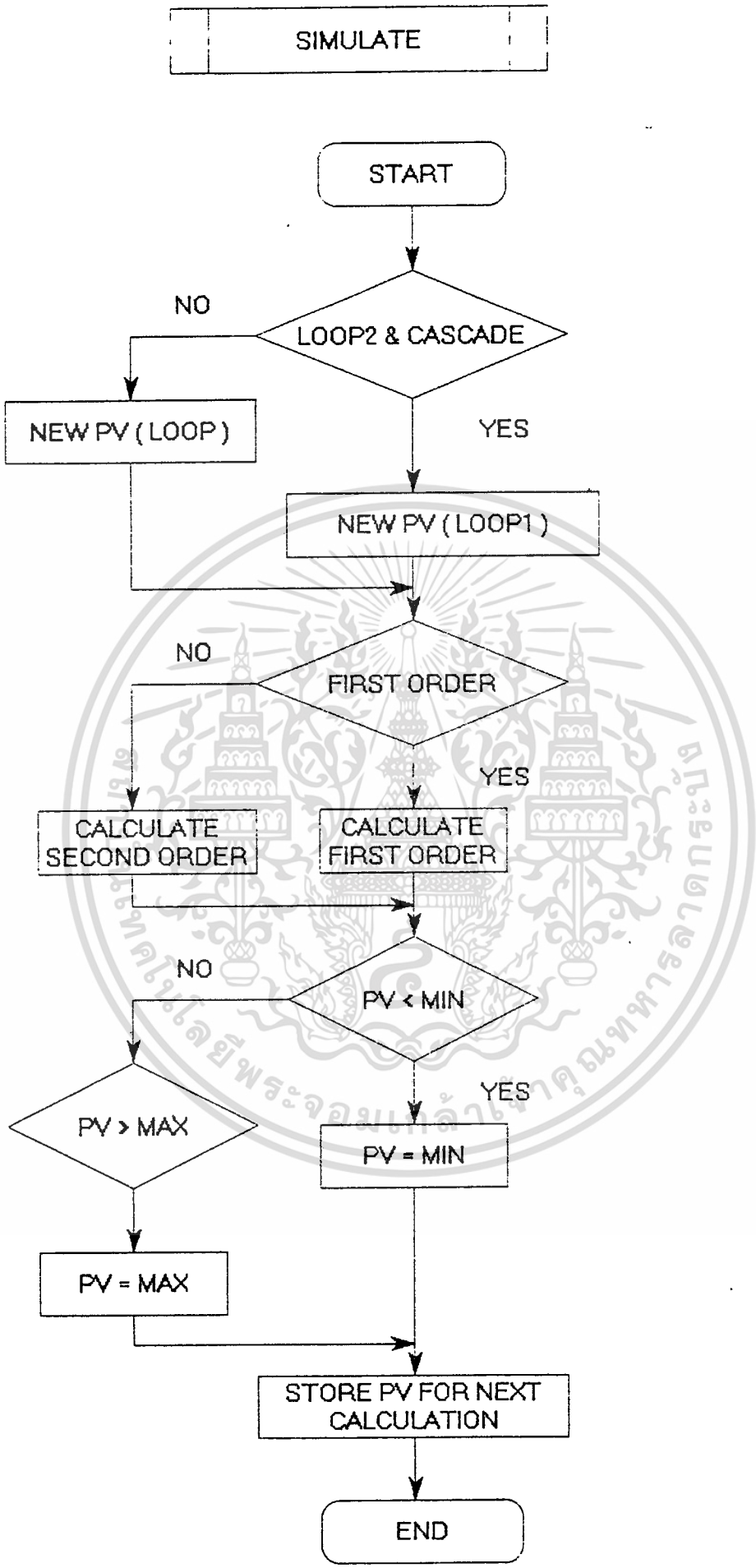

SELECT MENU
START
Read initial data from DATA.FIL
NO
Check file DATA.FIL
YES
Read data from function Ini_Data( )
Set to all variable for run program
Select : Type ; No. of Loop ; Action ; for run program
To function RUN( )


RUN
Function RUN( ) continue
Check LOOP
1 LOOP
2 LOOPS
Check keyboard
Check keyboard
NO
NO
Keyboard hit
Keyboard hit
YES
YES
Check input key (F1-F5)
Check input key (F1-F5)
Count time
Count time
Check Sam.Time=Data[1].Time
Check TYPE
CONTINUE
Control_Loop( )
Simulate( )


RUN CONTINUE
CONTINUE
Check Sam.Time = Data[1].Time
Check TYPE
Check Sam.Time = Data[2].Time
Control_Loop( )
Simulate( )
Check TYPE
Funtion RUN( ) (Loop)
Funtion RUN( ) (Loop)
Control_Loop( )
Simulate( )
Funtion RUN( ) (Loop)
Funtion RUN( ) (Loop)


CONTROL LOOP
START
RD
CHECK PV
NO
UNDER LOW, OVER HIGH
YES
SIREN SOUND
NO
AUTO
YES
PID
DISPLAY
NO
LOOP1 & CASCADE
WR
YES
CALCULATE SP
LOOP2
END


SIMULATE LOOP
START
RD
CHECK PV
NO
UNDER LOW, OVER HIGH
YES
SIREN SOUND
NO
AUTO
YES
PID
DISPLAY
NO
LOOP1 & CASCADE
SIMULATE
YES
CALCULATION SP LOOP
END


SIMULATE
START
LOOP2 & CASCADE
NO
YES
NEW PV (LOOP)
NEW PV (LOOP1)
FIRST ORDER
NO
YES
CALCULATE SECOND ORDER
CALCULATE FIRST ORDER
PV < MIN
NO
YES
PV > MAX
PV = MIN
PV = MAX
STORE PV FOR NEXT CALCULATION
END

/* C program to create PID Controller & Simulation */

```
#include <alloc.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define DELAY 100
#define RATE 10
#define NO_CHOICE 4  /* number of choices in each menu */
#define NO_MENU 12   /* number of menus */
#define UP_ARROW 0x4800       /*  scan code(16 bits) for up arrow */
#define DOWN_ARROW 0x5000     /*                down arrow       */
#define LEFT_ARROW 0x4B00     /*                left arrow       */
#define RIGHT_ARROW 0x4D00    /*               right arrow       */
#define ENTER 0x1C0D          /*                 return key      */
#define ESCAPE 0x011B         /*                 escape key      */
#define Page_Up 0x4900        /*                 page up key     */
#define Page_Down 0x5100      /*               page down key     */
#define F1   0x3B00           /*                 F1 key */
#define F2   0x3C00           /*                 F2 key */
#define F3   0x3D00           /*                 F3 key */
#define F4   0x3E00           /*                 F4 key */
#define F5   0x3F00           /*                 F5 key */
#define COM1         0
#define DATA_READY   0x100
#define FALSE        0
#define TRUE         1
#define SETTINGS     (0xE0 | 0x18 | 0x00 | 0x02)        /* 9600 E 7 1 */
#define CONTROL      0
#define SIMULATION   1
#define FIRST        1
#define SECOND       2
#define SEPARATION   0
#define CASCADE      1
```

```
#define AUTO          0
#define MANUAL        1
#define DIRECT        0
#define REVERSE       1
#define FLOW          0
#define LEVEL         1
#define CHANNEL       2

void siren(void);
void Initialize_Graphics_Mode(void);
void Draw_Fill_Rectangle(int lt,int tp,int rt,int bt);
void cls(void);
void Menu_Assignment(void);
void Display_Main_Menu(void);
void Display_Menu(int menu_no);
void Select_Menu(int menu_no,int choice_no);
     Select_Submenu(int menu_no,int choice_no);
int  Read_Key(int key);
void Inverse(int menu_no,int choice_no);
void Normal(int menu_no,int choice_no);
void Save_Screen(int X1,int Y1,int X2,int Y2);
void Restore_Screen(int X1,int Y1);
/************************************************************** VARIABLE */
typedef struct heading {     /*  data structure of choice heading */
   char *choice;
   };
 /* data structure of menu */
typedef struct menu_struct{
        int frame[4];                    /* 4 parameters for drawing menu frame */
        int row[NO_CHOICE];              /* x-value of each choice */
        int col;                         /* y-value of each choice */
        struct heading item[NO_CHOICE]; /*  index of each item */
        struct heading help[NO_CHOICE]; /* index of each help */
        int last_choice;                 /* the last choice of each menu */
        };
 /* structure of menus */
struct  menu_struct menus[NO_MENU];
struct  cpid   { int Process;
                 float Max,Min,Set_Point,Low_Alarm,High_Alarm;
                 float Kp,Ki,Kd,Time;   }   data[CHANNEL];
```

```
struct  modact { int Mode,Action; } status[CHANNEL];
struct  spare  { int ORDER;
                 float Ks,T1,T2; } sim[CHANNEL];
struct show    { float Pv,Mv; } remote[CHANNEL];
struct set     { int Switch_On,Choice,Change,Dum_int;
                 float Dummy; } change_data[CHANNEL];
struct store   { float Mv;
                    float Error1,Error2;
                    float Sp_old,Pv_old,Mv_old;
                    float s_old,p_old,m_old; } merror[CHANNEL];
struct hmsm    { int hours,minutes,seconds,milliseconds; } tm;
char *savemon;                          /* Function Save_Screen & Restore_Screen */
int start = 2,one_choice_width = 40;    /* Main_Menu */
int Switch_new=0;                       /* Protect Press For The First Time */
int in,out,STATUS,DONE = FALSE;         /* bioscom */
int CTSM=0,SEPCAS=0;
int No_Loop=0,Loop=0,Z=0,m=0,d=0;
float Sp_Trend[2][282];
float Pv_Trend[2][282];
float Time_1=0,Time_2=0;                /* clock to cal. pro. */
float ONT1=0,ONT2=0;                    /* switch pro. loop 1-2 */
FILE *fp;

/************************************************************** FUNCTION */

main(void)
{
  nosound();
  cls();
  Initialize_Graphics_Mode();
  Title();
  Menu_Assignment();

  if((fp=fopen("a:\data.fil","r"))==NULL)
    {
     if((fp=fopen("b:\data.fil","r"))==NULL)
      {
      if((fp=fopen("c:\data.fil","r"))==NULL)
         {
         Ini_Data();
```

```
            File_Write();
            File_Read();
            fclose(fp);
            }
        }
      }

    else Waiting();
         File_Read();
    Display_Main_Menu();
    Inverse(0,0);
    Select_Menu(0,1);
    closegraph();
    return(0);
}

int Title()
{
    int i;
    setfillstyle(1,4);
    bar(0,0,639,479);
    rectangle(10,10,629,469);
    setcolor(12);
    setfillstyle(1,7);
    bar(11,11,628,468);
    rectangle(15,15,624,464);
    rectangle(17,17,622,462);
    setcolor(1);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
    outtextxy(22,25,"King Mongkut Institute of Technology");
    outtextxy(150,65,"L A D K R A B A N G");
    setfillstyle(1,15);
    bar(40,160,599,165);
    bar(40,160,45,220);
    bar(160,390,479,400);
    bar(469,240,479,400);
    setfillstyle(1,8);
    bar(40,215,599,220);
    bar(594,160,599,220);
    bar(160,240,479,250);
```

```
bar(160,240,170,400);
setcolor(8);
outtextxy(88,172,"P I D Controller & Simulation");
rectangle(158,238,482,403);
setcolor(15);
outtextxy(85,170,"P I D Controller & Simulation");
line(40,220,45,215);
line(594,165,599,160);
for(i=0;i<5;i++)
  {
   line(40,215+i,45-i,215+i);
   line(594,160+i,599-i,160+i);
  }
for(i=0;i<10;i++)
  {
   line(160+i,400-i,170,400-i);
   line(469+i,250-i,479,250-i);
  }

setfillstyle(9,8);
bar(40,115,599,140);
settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
outtextxy(90,120,"Department of Instrumentation");
setfillstyle(1,4);
bar(169,249,469,390);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(236,289,"AKANIT  BOONYAWAIROJ");
outtextxy(246,322,"ANUCHA  IAMPAIJIT");
outtextxy(226,355,"BORRI.  LAMLERT-PRASERT");
setcolor(0);
rectangle(39,159,600,221);
rectangle(38,113,601,142);
outtextxy(250,260,"..Presented By..");
outtextxy(225,420,"COPYRIGHT (C) April 1994");
if(getch());
 {
  setfillstyle(9,7);
  bar(39,159,600,221);
  setfillstyle(1,8);
  bar(39,159,600,165);
```

```
    bar(39,159,45,221);
    setfillstyle(1,15);
    bar(39,215,600,221);
    bar(594,160,600,221);
    setcolor(8);
    for(i=0;i<7;i++)
        {
          line(39,215+i,45-i,215+i);
          line(594,159+i,600-i,159+i);
        }
    setcolor(15);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,4);
    outtextxy(55,172,"Switch-On to Start Program Now");
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    delay(1000);
  }
  cls();
  return(0);
}

void Initialize_Graphics_Mode(void)
{
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
  errorcode = graphresult();
  if (errorcode != grOk)
  {
      printf("Graphics error : %s \n", grapherrormsg(errorcode));
      printf("Press any key to halt : ");
      exit(1);    /* return with error code */
  }
}

void Draw_Fill_Rectangle(int lt,int tp,int rt,int bt)
{
   setfillstyle(1,11);
   bar(lt,tp,rt,bt);
}

void Inverse(int menu_no,int choice_no)
```

```
{
  Draw_Fill_Rectangle(menus[menu_no].col+2,
                      menus[menu_no].row[choice_no]-2,
                      menus[menu_no].col+155,
                      menus[menu_no].row[choice_no]+10);
  setcolor(4);
  outtextxy(menus[menu_no].col,menus[menu_no].row[choice_no],
            menus[menu_no].item[choice_no].choice);
  setcolor(15);
  setfillstyle(1,4);
  bar(75,460,635,479);
  outtextxy(76,465,menus[menu_no].help[choice_no].choice);
}

void Normal(int menu_no,int choice_no)
{
  setfillstyle(1,0);
  bar(menus[menu_no].col+2,
      menus[menu_no].row[choice_no]-2,
      menus[menu_no].col+155,
      menus[menu_no].row[choice_no]+10);
  outtextxy(menus[menu_no].col,menus[menu_no].row[choice_no],
                    menus[menu_no].item[choice_no].choice);
  setfillstyle(1,4);
  bar(159,460,635,479);
}

int Read_Key(int key)
{
  return bioskey(key);
}

void Save_Screen(int X1,int Y1,int X2,int Y2)
{
   unsigned size;
   size = imagesize(X1,Y1,X2,Y2);    /* get byte size of image */
   if ((savemon = malloc(size)) == NULL)
    {
      closegraph();
      printf("Error: not enough heap space in save_screen().\n");
```

```
        exit(1);
      }
      getimage(X1,Y1,X2,Y2,savemon);
}

void Restore_Screen(int X1,int Y1)
{
   putimage(X1,Y1,savemon, COPY_PUT);
   free(savemon);
}

void cls(void)   /* Clear the screen. */
{
  union REGS r;
  r.h.ah   = 6;
  r.h.al   = 0;
  r.h.ch   = 0;
  r.h.cl   = 0;
  r.h.dh   = 29;
  r.h.dl   = 79;
  r.h.bh   = 0;
  int86(0x10, &r, &r);
}

Ini_Data()
{
      data[0].Process      = LEVEL;
      data[0].Max          = 100;
      data[0].Min          = 0;
      data[0].Set_Point    = 50;
      data[0].Low_Alarm    = 10;
      data[0].High_Alarm   = 90;
      data[0].Kp           = 3;
      data[0].Ki           = 0.5;
      data[0].Kd           = 0.5;
      data[0].Time         = 0.1;
      data[1].Process      = FLOW;
      data[1].Max          = 200;
      data[1].Min          = 100;
      data[1].Set_Point    = 150;
```

```
    data[1].Low_Alarm   = 110;
    data[1].High_Alarm  = 190;
    data[1].Kp          = 2;
    data[1].Ki          = 0.5;
    data[1].Kd          = 0.1;
    sim[0].ORDER        = FIRST;
    sim[0].Ks           = 3;
    sim[0].T1           = 1;
    sim[0].T2           = 1;
    sim[1].ORDER        = SECOND;
    sim[1].Ks           = 2;
    sim[1].T1           = 1;
    sim[1].T2           = 1;
return(0);
}

void Menu_Assignment(void)
{
   int one_part;
   one_part = (getmaxx()/4);
   menus[0].frame[0] = start;
   menus[0].frame[1] = one_choice_width;
   menus[0].frame[2] = start+one_part;
   menus[0].frame[3] = (one_choice_width*3)-20;
   menus[0].row[0]   = 24;
   menus[0].row[1]   = 44;
   menus[0].row[2]   = 64;
   menus[0].row[3]   = 84;
   menus[0].col      = start;
   start +           = one_part;
   menus[0].item[0].choice = " About";
   menus[0].item[1].choice = "  Clear desktop";
   menus[0].item[2].choice = "  Ini._Data_1  ";
   menus[0].item[3].choice = "  Ini._Data_2  ";
   menus[0].help[1].choice =  "Clear and Repaint desktop";
   menus[0].help[2].choice =  "Display initial data for loop_1";
   menus[0].help[3].choice =  "Display initial data for loop_2";
   menus[0].last_choice    = 3;
   menus[1].frame[0]       = start;
   menus[1].frame[1]       = one_choice_width;
```

```
menus[1].frame[2]        = start+one_part;
menus[1].frame[3]        = one_choice_width+20;
menus[1].row[0]          = 24;
menus[1].row[1]          = 44;
menus[1].col             = start;
start +                  = one_part;
menus[1].item[0].choice = " Controller     ";
menus[1].item[1].choice = "  Number of loop";
menus[1].help[1].choice ="Select the number of loops that you want ";
menus[1].last_choice = 1;
menus[2].frame[0] = start;
menus[2].frame[1] = one_choice_width;
menus[2].frame[2] = start+one_part;
menus[2].frame[3] = one_choice_width+20;
menus[2].row[0]   = 24;
menus[2].row[1]   = 44;
menus[2].col      = start;
start +           = one_part;
menus[2].item[0].choice = " Simulation    ";
menus[2].item[1].choice = "  Number of loop";
menus[2].help[1].choice = "Select the number of loops that you want";
menus[2].last_choice = 1;
menus[3].frame[0] = start;
menus[3].frame[1] = one_choice_width;
menus[3].frame[2] = start+one_part;
menus[3].frame[3] = one_choice_width+20;
menus[3].row[0]   = 24;
menus[3].row[1]   = 44;
menus[3].col      = start;
start +           = one_part;
menus[3].item[0].choice = " Quit ";
menus[3].item[1].choice = "  eXit";
menus[3].help[1].choice = "Quit program and exit to DOS ..( BYE.! )..";
menus[3].last_choice = 1;
menus[4].frame[0] = 2+one_part;
menus[4].frame[1] = one_choice_width;
menus[4].frame[2] = 2+(one_part*2);
menus[4].frame[3] = one_choice_width*2;
menus[4].row[0] = 24;
menus[4].row[1] = 44;
```

```
menus[4].row[2] = 64;
menus[4].col    = 2+ one_part;
menus[4].item[0].choice = " No._of_loop  ";
menus[4].item[1].choice = "  1_LOOP      ";
menus[4].item[2].choice = "  2_LOOP      ";
menus[4].help[1].choice =  "You.will choose 1 loop to controll process";
menus[4].help[2].choice =  "You will choose 2 loops to controll process";
menus[4].last_choice = 2;
menus[10].frame[0] = 2+one_part;
menus[10].frame[1] = one_choice_width;
menus[10].frame[2] = 2+(one_part*2);
menus[10].frame[3] = one_choice_width*2;
menus[10].row[0] = 24;
menus[10].row[1] = 44;
menus[10].row[2] = 64;
menus[10].col    = 2+one_part;
menus[10].item[0].choice = " Loop_Relation  ";
menus[10].item[1].choice = "  Cascade  ";
menus[10].item[2].choice = "  Separation";
menus[10].last_choice    = 2;
menus[5].frame[0] = 2+one_part;
menus[5].frame[1] = one_choice_width;
menus[5].frame[2] = 2+(one_part*2);
menus[5].frame[3] = one_choice_width*2;
menus[5].row[0] = 24;
menus[5].row[1] = 44;
menus[5].row[2] = 64;
menus[5].col    = 2+one_part;
menus[5].item[0].choice = " Control Action_1";
menus[5].item[1].choice = "  Direct Act.  ";
menus[5].item[2].choice = "  Reverse Act.";
menus[5].help[1].choice = "You will choose mode simulate DIRECT ACTION ";
menus[5].help[2].choice = "You will choose mode simulate REVERSE ACTION ";
menus[5].last_choice    = 2;
menus[8].frame[0] = 2+one_part;
menus[8].frame[1] = one_choice_width;
menus[8].frame[2] = 2+(one_part*2);
menus[8].frame[3] = one_choice_width*2;
menus[8].row[0] = 24;
menus[8].row[1] = 44;
```

```
menus[8].row[2] = 64;
menus[8].col = 2+one_part;
menus[8].item[0].choice = " Control Action_2";
menus[8].item[1].choice = "  Direct Act.  ";
menus[8].item[2].choice = "  Reverse Act.";
menus[8].help[1].choice =  "You will choose mode control DIRECT ACTION ";
menus[8].help[2].choice =  "You will choose mode control REVERSE ACTION ";
menus[8].last_choice = 2;
menus[6].frame[0] = 2+(one_part*2);
menus[6].frame[1] = one_choice_width;
menus[6].frame[2] = 2+(one_part*3);
menus[6].frame[3] = one_choice_width*2;
menus[6].row[0] = 24;
menus[6].row[1] = 44;
menus[6].row[2] = 64;
menus[6].col = 2+(one_part*2);
menus[6].item[0].choice = " No._of_loop  ";
menus[6].item[1].choice = "  1_LOOP      ";
menus[6].item[2].choice = "  2_LOOP      ";
menus[6].help[1].choice =  "You will choose 1 loop to simulate process";
menus[6].help[2].choice =  "You will choose 2 loops to simulate process";
menus[6].last_choice = 2;
menus[11].frame[0] = 2+(one_part*2);
menus[11].frame[1] = one_choice_width;
menus[11].frame[2] = 2+(one_part*3);
menus[11].frame[3] = one_choice_width*2;
menus[11].row[0] = 24;
menus[11].row[1] = 44;
menus[11].row[2] = 64;
menus[11].col = 2+(one_part*2);
menus[11].item[0].choice = " Loop_Relation  ";
menus[11].item[1].choice = "  Cascade  ";
menus[11].item[2].choice = "  Separation";
menus[11].last_choice = 2;
menus[7].frame[0] = 2+(one_part*2);
menus[7].frame[1] = one_choice_width;
menus[7].frame[2] = 2+(one_part*3);
menus[7].frame[3] = one_choice_width*2;
menus[7].row[0] = 24;
menus[7].row[1] = 44;
```

```
    menus[7].row[2] = 64;
    menus[7].col = 2+(one_part*2);
    menus[7].item[0].choice = " Control Action_1";
    menus[7].item[1].choice = "  Direct Act.  ";
    menus[7].item[2].choice = "  Reverse Act.";
    menus[7].help[1].choice = "You will choose mode simulate DIRECT ACTION ";
    menus[7].help[2].choice = "You will choose mode simulate REVERSE ACTION ";
    menus[7].last_choice = 2;
    menus[9].frame[0] = 2+(one_part*2);
    menus[9].frame[1] = one_choice_width;
    menus[9].frame[2] = 2+(one_part*3);
    menus[9].frame[3] = one_choice_width*2;
    menus[9].row[0] = 24;
    menus[9].row[1] = 44;
    menus[9].row[2] = 64;
    menus[9].col = 2+(one_part*2);
    menus[9].item[0].choice = " Control Action_2";
    menus[9].item[1].choice = "  Direct Act.  ";
    menus[9].item[2].choice = "  Reverse Act.";
    menus[9].help[1].choice = "You will choose mode simulate DIRECT ACTION ";
    menus[9].help[2].choice = "You will choose mode simulate REVERSE ACTION ";
    menus[9].last_choice = 2;
}

void Display_Main_Menu(void)
{
    int i;
   setfillstyle(1,4);
   bar(0,0,639,19);
   setcolor(14);
   outtextxy(152,5," P  I  D   Controller  &  Simulation ");
   bar(0,460,639,479);
   outtextxy(5,465," Help =>");
   setcolor(15);
   setfillstyle(1,0);
   bar(0,20,639,39);
   rectangle(0,20,639,39);
    for(i=0;i<=3;i++)
    {
    outtextxy(menus[i].col,menus[i].row[0],menus[i].item[0].choice);
```

```
    }
}

void Display_Menu(int menu_no)
{
    int i;
    setfillstyle(1,0);
    bar(menus[menu_no].frame[0]-1,menus[menu_no].frame[1]+1,
        menus[menu_no].frame[2]-1,menus[menu_no].frame[3]);
    rectangle(menus[menu_no].frame[0]-1,menus[menu_no].frame[1]+1,
            menus[menu_no].frame[2]-1,menus[menu_no].frame[3]);
    for(i=1;i<4;i++)
        outtextxy(menus[menu_no].col,menus[menu_no].row[i],
                menus[menu_no].item[i].choice);
}

R_D()
{
  int  n=0,xr=0,i=0,j=3,S_Rd=0;
  char r1[18] = "@00RR0005000144*\x0D";
  char r2[18] = "@00RR0006000147*\x0D";
  char r[18]  = "@00RR0000000000*\x0D";
  int  p[14];
  if(Z==0) {
    for(n=0;n<18;n++)
    r[n]=r1[n]; }
  if(Z==1) {
    for(n=0;n<18;n++)
    r[n]=r2[n]; }
  DONE=FALSE;
  bioscom(0, SETTINGS, COM1);
  while (!DONE) {
    STATUS = bioscom(3, 0, COM1);
    if(STATUS & DATA_READY)
      if((in = bioscom(2, 0, COM1) & 0x7F) != 0) {
          p[i] = in;
          i++;
          if (in == '\x0D')   DONE = TRUE; }
    if(xr<17) {
        for (xr=0;xr<17;xr++)
```

```
        bioscom(1, r[xr], COM1); } }
  for (i=7,j=3;i<11;i++,j--) {
    if((p[i]>47)&&(p[i]<58))    p[i] -= 48;
    if((p[i]>64)&&(p[i]<71))    p[i] -= 55;
    S_Rd += p[i]*pow(16,j); }
  if(data[Z].Process==FLOW) {
     S_Rd = sqrt(S_Rd);
     remote[Z].Pv=((data[Z].Max-data[Z].Min)/64*S_Rd)+data[Z].Min; }
  else    remote[Z].Pv=((data[Z].Max-data[Z].Min)/4095*S_Rd)+data[Z].Min;
  return 0;
}

PID()
{
  float  k1=0,k2=0,k3=0;
  float Error=0;
  k1= data[Z].Kp*(1+(data[Z].Ki*data[Z].Time)+(data[Z].Kd/data[Z].Time));
  k2=-data[Z].Kp*(1+(2*(data[Z].Kd/data[Z].Time)));
  k3= data[Z].Kp*(data[Z].Kd/data[Z].Time);
  Error= 100*(data[Z].Set_Point-remote[Z].Pv)/(data[Z].Max-data[Z].Min);
  remote[Z].Mv+=(k1*Error)+(k2*merror[Z].Error1)+(k3*merror[Z].Error2);
  merror[Z].Error2=merror[Z].Error1;
  merror[Z].Error1=Error;
  if(remote[Z].Mv>100)      remote[Z].Mv=100;
  else if(remote[Z].Mv<0)    remote[Z].Mv=0;
  if(status[Z].Action==DIRECT)
     remote[Z].Mv=100-remote[Z].Mv;
  return 0;
}

W_R()
{
  div_t a,b;
  int   n=0, s=0, i=9, j=3, k=0, xw=0;
  char  ww1[18] ="@00WR0007000000*\x0D";
  char  ww2[18] ="@00WR0008000000*\x0D";
  char  ww[18]  ="@00WR0007000000*\x0D";
  s=(remote[Z].Mv/100)*4095;
  if(Z==0) {
     for(n=0;n<18;n++)
```

```
    ww[n]=ww1[n]; }
  if(Z==1) {
    for(n=0;n<18;n++)
    ww[n]=ww2[n]; }
  for(i=9,j=3;i<13;i++,j--) {
    a=div(s,pow(16,j));
    ww[i]=a.quot;
    if((ww[i]>='\x0')&&(ww[i]<='\x9'))  ww[i] += 48;
    if((ww[i]>='\xA')&&(ww[i]<='\xF'))  ww[i] += 55;
    s=a.rem; }
  for(j=0;j<13;j++)
    k ^= ww[j];
  b=div(k,16);
  ww[13]=b.quot;
  ww[14]=b.rem;
  for(i=13;i<15;i++) {
    if((ww[i]>='\x0')&&(ww[i]<='\x9'))    ww[i] += 48;
    if((ww[i]>='\xA')&&(ww[i]<='\xF'))    ww[i] += 55; }
  DONE=FALSE;
  bioscom(0, SETTINGS, COM1);
  while(!DONE) {
    STATUS=bioscom(3, 0, COM1);
    if(STATUS & DATA_READY)
        if((out=bioscom(2, 0, COM1) & 0x7F) != 0) {
          if(out=='\x0D')  DONE = TRUE; }
  if(xw<17)
    for(xw=0;xw<17;xw++)
    bioscom(1, ww[xw], COM1); }
return 0;
}

File_Write()
{
  int i;
  if((fp=fopen("a:\data.fil","w"))==NULL)
  {
   if((fp=fopen("b:\data.fil","w"))==NULL)
   {
    if((fp=fopen("c:\data.fil","w"))==NULL)
    {
```

```
    P_Error();
   }
  }
 }
 for(i=0;i<2;++i) {
 fprintf(fp,"%d                    \n",data[i].Process);
 fprintf(fp,"%f \n%f \n%f \n%f \n%f \n",data[i].Max,data[i].Min,
                                    data[i].Low_Alarm,data[i].High_Alarm,
                                    data[i].Set_Point);
 fprintf(fp,"%f \n%f \n%f \n%f      \n",data[i].Kp,data[i].Ki,
                                    data[i].Kd,data[i].Time);
 fprintf(fp,"%d                    \n",sim[i].ORDER);
 fprintf(fp,"%f \n%f \n%f          \n",sim[i].Ks,sim[i].T1,sim[i].T2); }
 fclose(fp);
 return 0;
}

File_Read()
{
 int i;
 if((fp=fopen("a:\data.fil","r"))==NULL)
 {
  if((fp=fopen("b:\data.fil","r"))==NULL)
  {
   if((fp=fopen("c:\data.fil","r"))==NULL)
   {
    P_Error();
   }
  }
 }
 for(i=0;i<2;++i) {
 fscanf(fp,"%d \n",&data[i].Process);
 fscanf(fp,"%f%f%f%f%f \n",&data[i].Max,&data[i].Min,
                           &data[i].Low_Alarm,&data[i].High_Alarm,
                           &data[i].Set_Point);
 fscanf(fp,"%f%f%f%f   \n" ,&data[i].Kp,&data[i].Ki,
                           &data[i].Kd,&data[i].Time);
 fscanf(fp,"%d         \n",&sim[i].ORDER);
 fscanf(fp,"%f%f%f     \n",&sim[i].Ks,&sim[i].T1,&sim[i].T2); }
 fclose(fp);
```

```
 return 0;
}
Run()
{
   for(;;)
    {
    if(No_Loop==1)
     {
         Time();
         delay(10);
         tm.milliseconds+=10;
         Check_Keyboard();
         Time_1+=0.01;
         ONT1=(data[0].Time-0.9)-Time_1;
         if((ONT1>=-0.005)&&(ONT1<=0.005))
         {
           Time_1=0; Z=0; d=0;
           if(CTSM==CONTROL)
             Control_Loop();
           else
             Simulation_Loop();
          tm.milliseconds+=900;
         }
     }
    if(No_Loop==2)
     {
         Time();
         delay(10);
         tm.milliseconds+=10;
         Check_Keyboard();
         Time_1+=0.01;
         Time_2+=0.01;
         ONT1=(data[0].Time-0.9)-Time_1;
         ONT2=(data[1].Time-0.9)-Time_2;
         /* delay for R_D W_R only when Time1!=Time2 and ONT1=ONT2=0 */
         if(((ONT1>=-0.005)&&(ONT1<=0.005))&&((ONT2>=-0.005)&&(ONT2<=0.005)))
           m=50;
         if((ONT1>=-0.005)&&(ONT1<=0.005))
         {
           Time_1=0; Z=0; d=0;
```

```
            if(CTSM==CONTROL)
              Control_Loop();
            else
              Simulation_Loop();
            tm.milliseconds+=900;
          }
          if((ONT2>=-0.005)&&(ONT2<=0.005))
          {
            Time_2=0; Z=1; d=321;
            delay(m); m=0;
            if(CTSM==CONTROL)
              Control_Loop();
            else
              Simulation_Loop();
          }
      }
    } /* for */
}

Control_Loop()
{
  float Sp_new;
  char  Sp[8];
  R_D();
  if((remote[Z].Pv>=data[Z].High_Alarm)||(remote[Z].Pv<=data[Z].Low_Alarm))
     siren();
  if(status[Z].Mode==AUTO)                    /* auto mode */
     PID();
  Bar_Graph();
  if((Z==0)&&(SEPCAS==CASCADE))
    {
     data[1].Set_Point=((remote[0].Mv/100)*(data[1].Max-data[1].Min))
                                          +data[1].Min;
     if(Sp_new!=data[1].Set_Point)
      {
          setfillstyle(1,1);
          bar(401,200,471,210);
          sprintf(Sp," %6.2f ",data[1].Set_Point);
          setcolor(12);
          outtextxy(401,200,Sp);
```

```
            Sp_new=data[1].Set_Point;
          }
        }
       else   W_R();
       return 0;
}
Simulation_Loop()
{
 float Sp_new;
 char  Sp[8];
  if((remote[Z].Pv>=data[Z].High_Alarm)||(remote[Z].Pv<=data[Z].Low_Alarm))
      siren();
  if(status[Z].Mode==AUTO)
     PID();
  Bar_Graph();
  if((Z==0)&&(SEPCAS==CASCADE))
   {
    data[1].Set_Point=((remote[0].Mv/100)*(data[1].Max-data[1].Min))
                                          +data[1].Min;
    if(Sp_new!=data[1].Set_Point)
     {
          setfillstyle(1,1);
          bar(401,200,471,210);
          sprintf(Sp," %6.2f ",data[1].Set_Point);
          setcolor(12);
          outtextxy(401,200,Sp);
          Sp_new=data[1].Set_Point;
     }
   }
  else   Simulate();                    /* Z=0 && CASCADE ==> do not wr */
  return 0;
}
  Simulate()
{
  int   S=0,srt=0,lp=0;
  float tou=0,damp=0;
  float x=0,y=0;
                                        /* Cascade loop1 ==> Z=0  do not be done here */
  if((SEPCAS==CASCADE)&&(Z==1))       /* for Cascade loop2 ==> Z=1 */
   { srt=0; lp=2; } /* PVloop1(Z=0)&&loop2(Z=1) get value from MVloop2(Z=1) */
```

```
  else   { srt=Z; lp=Z+1; }                    /* for Separate & One Loop */
  for(S=srt;S<lp;++S)
  {
   if(sim[S].ORDER==FIRST)                              /* First Order */
     remote[S].Pv+=(remote[Z].Mv-merror[Z].Mv)*sim[S].Ks*
                   (1-(exp(-data[S].Time/sim[S].T1)));
  else  { tou=sqrt(sim[S].T1*sim[S].T2);              /* Second Order */
          damp=(sim[S].T1*sim[S].T2)/(2*tou);
          if(damp>1) {
            x=sqrt(pow(damp,2)-1);
            y=x*data[S].Time/tou;
            remote[S].Pv+=(remote[Z].Mv-merror[Z].Mv)*sim[S].Ks*
                          (1-((exp(-(damp*data[S].Time/tou)))*
                          (cosh(y)+(damp/x*sinh(y)))));}
          else if(damp==1) {
                 remote[S].Pv+=(remote[Z].Mv-merror[Z].Mv)*sim[S].Ks*
                               (1-((1+(data[S].Time/tou))*
                               exp(-(data[S].Time/tou))));}
                else if((damp>=0)&&(damp<1)) {
                 x=sqrt(1-pow(damp,2));
                 y=x*data[S].Time/tou;
                 remote[S].Pv+=(remote[Z].Mv-merror[Z].Mv)*sim[S].Ks*
                               (1-((exp(-(damp*data[S].Time/tou)))*
                               (cosh(y)+(damp/x*sinh(y)))));}}
    if(remote[S].Pv<data[S].Min)    remote[S].Pv=data[S].Min;
    if(remote[S].Pv>data[S].Max)    remote[S].Pv=data[S].Max;}
 Display_Data()
{
 char Max[8],Min[8],Low[8],Hi[8],Kp[8],Ki[8],Kd[8],Sp[8],Time[8],Process[8];
 int d=0,i;
 for(i=0;i<No_Loop;i++,d=321)
    {
    setfillstyle(1,1);
    bar(80+d,80,150+d,210);
    bar(95+d,215,150+d,225);
    bar(110+d,230,150+d,240);
    setcolor(12);
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    outtextxy(80+d,80,Max);
    sprintf(Min," %6.2f ",data[i].Min);
```

```
        outtextxy(80+d,95,Min);
        sprintf(Low,"  %6.2f ",data[i].Low_Alarm);
        outtextxy(80+d,115,Low);
        sprintf(Hi,"  %6.2f ",data[i].High_Alarm);
        outtextxy(80+d,130,Hi);
        sprintf(Kp,"  %6.2f ",data[i].Kp);
        outtextxy(80+d,150,Kp);
        sprintf(Ki,"  %6.2f ",data[i].Ki);
        outtextxy(80+d,165,Ki);
        sprintf(Kd,"  %6.2f ",data[i].Kd);
        outtextxy(80+d,180,Kd);
        sprintf(Sp,"  %6.2f ",data[i].Set_Point);
        outtextxy(80+d,200,Sp);
        sprintf(Time,"  %5.2f ",data[i].Time);
        outtextxy(95+d,215,Time);
        if(data[i].Process==1)
            outtextxy(110+d,230,"LEVEL");
        else
            outtextxy(110+d,230,"FLOW");
        }
    return(0);
}

void Select_Menu(int menu_no,int choice_no)
{
    int new_menu_no,new_choice_no =0;
    int r;
    int i,j;
/* initial data */
    Loop=0;
    SEPCAS=0;
    if(data[0].Time==data[1].Time)
      Time_1=(data[0].Time-0.9)/2;
    Time_2=0;
                                /* clear for time() */
    setfillstyle(1,4);
    bar(555,460,639,479);
    tm.milliseconds=0;
    tm.seconds=0;
    tm.minutes=0;
```

```
    tm.hours=0;
    for(i=0;i<2;i++)
    {
     change_data[i].Choice=1;
     status[i].Mode=MANUAL;
     remote[i].Pv=0;
     remote[i].Mv=0;
     merror[i].Mv=0;
     merror[i].Error1=0;
     merror[i].Error2=0;
     merror[i].Sp_old=0;
     merror[i].Pv_old=0;
     merror[i].Mv_old=0;
     merror[i].s_old=0;
     merror[i].p_old=0;
     merror[i].m_old=0;
     for(j=0;j<282;j++)
      {  Sp_Trend[i][j]=5;
         Pv_Trend[i][j]=5;
      }
    }

   Save_Screen(menus[menu_no].frame[0]-1,menus[menu_no].frame[1]+1,
                menus[menu_no].frame[2]-1,menus[menu_no].frame[3]);
/* defult menu */
     Display_Menu(menu_no);
     Inverse(menu_no,choice_no);
   for(;;)
   {
    r = Read_Key(0);                          /* read keyboard */
     switch (r)
     {
        case LEFT_ARROW : if (menu_no == 0)   new_menu_no = 4-1;
                          else                new_menu_no = menu_no-1;
                          Restore_Screen(menus[menu_no].frame[0]-1,
                                         menus[menu_no].frame[1]+1);
                          Normal(menu_no,0);
                          Inverse(new_menu_no,0);
                          Select_Menu(new_menu_no,1);
        break;
```

```
case RIGHT_ARROW: if (menu_no == 4-1) new_menu_no = 0;
                  else              new_menu_no = menu_no+1;
                  Restore_Screen(menus[menu_no].frame[0]-1,
                                 menus[menu_no].frame[1]+1);
                  Normal(menu_no,0);
                  Inverse(new_menu_no,0);
                  Select_Menu(new_menu_no,1);
break;
case UP_ARROW   : if (choice_no == 1)
                     new_choice_no = menus[menu_no].last_choice;
                  else new_choice_no = choice_no-1;
                  Normal(menu_no,choice_no);
                  Inverse(menu_no,new_choice_no);
                  choice_no = new_choice_no;
break;
case DOWN_ARROW : if (choice_no == menus[menu_no].last_choice)
                     new_choice_no = 1;
                  else new_choice_no = choice_no+1;
                  Normal(menu_no,choice_no);
                  Inverse(menu_no,new_choice_no);
                  choice_no = new_choice_no;
break;
case ENTER : switch (menu_no)
             {
              case 0 : switch (choice_no)
                       {
                        case 1:
                        break;
                        case 2:
                        break;
                        case 3:
                        break;
                       }
             break;
             case 1 : Restore_Screen(menus[menu_no].frame[0]-1,
                                     menus[menu_no].frame[1]+1);
                      Inverse(4,0);
                      Select_Submenu(4,1);
             break;
             case 2 : Restore_Screen(menus[menu_no].frame[0]-1,
```

```
                                        menus[menu_no].frame[1]+1);
                    Inverse(6,0);
                    Select_Submenu(6,1);
                break;
                case 3 : File_Write();
                    closegraph();
                    exit(0);
                break;
                } /* return */
        break;
        case ESCAPE : if(Switch_new==1)
                {
                Restore_Screen(menus[menu_no].frame[0]-1,
                                menus[menu_no].frame[1]+1);
                Display_Main_Menu();
                Run();
                }
        break;
    }           /* switch */
  }             /* for */
}

Select_Submenu(int menu_no,int choice_no)
{
  int new_choice_no =0;
  int r;
  Save_Screen(menus[menu_no].frame[0]-1,menus[menu_no].frame[1]+1,
              menus[menu_no].frame[2]-1,menus[menu_no].frame[3]);
   /* defult menu */
    Display_Menu(menu_no);
    Inverse(menu_no,choice_no);
  for(;;)
  {
   r = Read_Key(0);     /* read keyboard */
    switch (r)
      {
       case UP_ARROW  : if (choice_no == 1)
                          new_choice_no = menus[menu_no].last_choice;
                        else new_choice_no = choice_no-1;
                        Normal(menu_no,choice_no);
```

```
                    Inverse(menu_no,new_choice_no);
                    choice_no = new_choice_no;
break;
case DOWN_ARROW: if (choice_no == menus[menu_no].last_choice)
                        new_choice_no = 1;
                    else new_choice_no = choice_no+1;
                    Normal(menu_no,choice_no);
                    Inverse(menu_no,new_choice_no);
                    choice_no = new_choice_no;
break;
case ENTER : switch (menu_no)
            {
                case 4 : switch (choice_no)
                        {
                        case 1: No_Loop=1;
                                CTSM=CONTROL;
                                Restore_Screen
                                (menus[menu_no].frame[0]-1,
                                 menus[menu_no].frame[1]+1);
                                Example_Screen();
                                Inverse(5,0);
                                Select_Submenu(5,1);
                        break;
                        case 2: No_Loop=2;
                                CTSM=CONTROL;
                                Restore_Screen
                                (menus[menu_no].frame[0]-1,
                                 menus[menu_no].frame[1]+1);
                                Example_Screen();
                                Inverse(10,0);
                                Select_Submenu(10,1);
                        break;
                        }
                break;
                case 5 : switch (choice_no)
                        {
                        case 1: status[0].Action=DIRECT;
                                Display_Main_Menu();
                                Restore_Screen(menus[5].frame[0]-1,
                                               menus[5].frame[1]+1);
```

```
                    settextstyle(SMALL_FONT,HORIZ_DIR,4);
                    outtextxy(175,203,"CONTROLLER Type");
                    outtextxy(175,227,
                              "DIRECT_ACTION Pro.");
                    settextstyle(DEFAULT_FONT,
                                 HORIZ_DIR,1);
                     if (No_Loop!=1) {
                       Inverse(8,0);
                       Select_Submenu(8,1); }
                      else {
                        Display_Data();
                        Run(); }
                  break;
                  case 2: status[0].Action=REVERSE;
                    Display_Main_Menu();
                    Restore_Screen(menus[5].frame[0]-1,
                                   menus[5].frame[1]+1);
                    settextstyle(SMALL_FONT,HORIZ_DIR,4);
                    outtextxy(175,203,"CONTROLLER Type");
                    outtextxy(175,227,
                              "REVERSE_ACTION Pro.");
                    settextstyle(DEFAULT_FONT,
                                 HORIZ_DIR,1);
                     if (No_Loop!=1) {
                       Inverse(8,0);
                       Select_Submenu(8,1); }
                      else {
                       Display_Data();
                       Run(); }
                   break;
                  }
          break;
case 6 : switch (choice_no)
                  {
                   case 1: No_Loop=1;
                           CTSM=SIMULATION;
                           Restore_Screen
                            (menus[menu_no].frame[0]-1,
                             menus[menu_no].frame[1]+1);
                           Example_Screen();
```

```
                    Inverse(7,0);
                    Select_Submenu(7,1);
                break;
                case 2: No_Loop=2;
                    CTSM=SIMULATION;
                    Restore_Screen
                     (menus[menu_no].frame[0]-1,
                      menus[menu_no].frame[1]+1);
                    Example_Screen();
                    Inverse(11,0);
                    Select_Submenu(11,1);
                 break;
               }
        break;
        case 7 : switch (choice_no)
                 {
                 case 1: status[0].Action=DIRECT;
                    Display_Main_Menu();
                    Restore_Screen
                     (menus[7].frame[0]-1,
                      menus[7].frame[1]+1);
                    settextstyle(SMALL_FONT,HORIZ_DIR,4);
                    outtextxy(175,203,"SIMULATION Type");
                    outtextxy(175,227,
                              "DIRECT_ACTION Pro.");
                    settextstyle(DEFAULT_FONT,
                                 HORIZ_DIR,1);
                     if(No_Loop!=1) {
                      Inverse(9,0);
                      Select_Submenu(9,1); }
                     else {
                      Display_Data();
                      Run(); }
                break;
                case 2: status[0].Action=REVERSE;
                    Display_Main_Menu();
                    Restore_Screen
                     (menus[7].frame[0]-1,
                      menus[7].frame[1]+1);
                    settextstyle(SMALL_FONT,
```

```
                    HORIZ_DIR,4);
                outtextxy(175,203,"SIMULATION Type");
                outtextxy(175,227,
                        "REVERSE_ACTION Pro.");
                settextstyle(DEFAULT_FONT,
                        HORIZ_DIR,1);
                 if(No_Loop!=1) {
                   Inverse(9,0);
                   Select_Submenu(9,1); }
                  else {
                   Display_Data();
                   Run(); }
            break;
           }
    break;
    case 8 : switch (choice_no)
           {
           case 1: status[1].Action=DIRECT;
                   Display_Main_Menu();
                   Restore_Screen(menus[8].frame[0]-1,
                                menus[8].frame[1]+1);
                   settextstyle(SMALL_FONT,HORIZ_DIR,4);
                   outtextxy(496,203,"CONTROLLER Type");
                   outtextxy(496,227,
                           "DIRECT_ACTION Pro.");
                   settextstyle(DEFAULT_FONT,
                           HORIZ_DIR,1);
                   Display_Data();
                   Run();
           break;
           case 2: status[1].Action=REVERSE;
                   Display_Main_Menu();
                   Restore_Screen(menus[8].frame[0]-1,
                                menus[8].frame[1]+1);
                   settextstyle(SMALL_FONT,HORIZ_DIR,4);
                   outtextxy(496,203,"CONTROLLER Type");
                   outtextxy(496,227,
                           "REVERSE_ACTION Pro.");
                   settextstyle(DEFAULT_FONT,
                           HORIZ_DIR,1);
```

```
                    Display_Data();
                    Run();
                break;
               }
break;
case 9 : switch (choice_no)
               {
                case 1: status[1].Action=DIRECT;
                        Display_Main_Menu();
                        Restore_Screen(menus[9].frame[0]-1,
                                       menus[9].frame[1]+1);
                        settextstyle(SMALL_FONT,HORIZ_DIR,4);
                        outtextxy(496,203,"SIMULATION Type");
                        outtextxy(496,227,
                                  "DIRECT_ACTION Pro.");
                        settextstyle(DEFAULT_FONT,
                                     HORIZ_DIR,1);
                        Display_Data();
                        Run();
                break;
                case 2: status[1].Action=REVERSE;
                        Display_Main_Menu();
                        Restore_Screen(menus[9].frame[0]-1,
                                       menus[9].frame[1]+1);
                        settextstyle(SMALL_FONT,HORIZ_DIR,4);
                        outtextxy(496,203,"SIMULATION Type");
                        outtextxy(496,227,
                                  "REVERSE_ACTION Pro.");
                        settextstyle(DEFAULT_FONT,
                                     HORIZ_DIR,1);
                        Display_Data();
                        Run();
                break;
               }
break;
case 10: switch (choice_no)
               {
                case 1: SEPCAS=CASCADE;
                        Restore_Screen
                         (menus[menu_no].frame[0]-1,
```

```
                    menus[menu_no].frame[1]+1);
                outtextxy(172,171,"CASCADE Relat.");
                outtextxy(493,171,"CASCADE Relat.");
                Inverse(5,0);
                Select_Submenu(5,1);
            break;
            case 2: SEPCAS=SEPARATION;
                Restore_Screen
                  (menus[menu_no].frame[0]-1,
                   menus[menu_no].frame[1]+1);
                outtextxy(172,171,"SEPARATE Relat.");
                outtextxy(493,171,"SEPARATE Relat.");
                Inverse(5,0);
                Select_Submenu(5,1);
            break;
            }
        break;
        case 11: switch (choice_no)
            {
            case 1: SEPCAS=CASCADE;
                Restore_Screen
                  (menus[menu_no].frame[0]-1,
                   menus[menu_no].frame[1]+1);
                outtextxy(172,171,"CASCADE Relat.");
                outtextxy(493,171,"CASCADE Relat.");
                Inverse(7,0);
                Select_Submenu(7,1);
            break;
            case 2: SEPCAS=SEPARATION;
                Restore_Screen
                  (menus[menu_no].frame[0]-1,
                   menus[menu_no].frame[1]+1);
                outtextxy(172,171,"SEPARATE Relat.");
                outtextxy(493,171,"SEPARATE Relat.");
                Inverse(7,0);
                Select_Submenu(7,1);
            break;
            }
        break;
    } /* switch(menu_num)*/
```

```
            break;   /* case ENTER */
            case ESCAPE : if(Switch_new==1)
                          {
                          if((menu_no==4)||(menu_no==6))
                          {
                          Restore_Screen(menus[menu_no].frame[0]-1,
                                         menus[menu_no].frame[1]+1);
                          Display_Main_Menu();
                          Run();
                          }
                          }
            break;
          }         /* switch(i) */
    }           /* for */
}

P_Error(int Massage_Error)
{
  int h;

   Save_Screen(159,159,481,331);
   setfillstyle(1,4);
   bar(160,160,480,330);
   setfillstyle(9,0);      /* shadow */
   bar(205,175,445,200);
   bar(293,298,353,320);
   setfillstyle(1,7);      /* box */
   bar(200,170,440,195);
   bar(290,295,350,317);
   settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
   setcolor(14);
   rectangle(162,162,478,328);
   outtextxy(220,175,"E R R O R..!! ");
   outtextxy(300,300,"OK.");
   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

   h = Massage_Error;
   switch(h)
    {
    case 1: outtextxy(195,250,"Min_Range is less than Max_Range");
```

```
            break;
    case 2: outtextxy(195,230,"Set_Point is more than Max_Range");
            outtextxy(310,250,"or");
            outtextxy(195,270,"Set_Point is less than Min_Range");
            break;
    case 3: outtextxy(195,230,"Low_Alarm is more than Set_Point");
            outtextxy(310,250,"or");
            outtextxy(195,270,"Low_Alarm is less than Min_Range");
            break;
    case 4: outtextxy(191,230,"High_Alarm is more than Max_Range");
            outtextxy(310,250,"or");
            outtextxy(191,270,"High_Alarm is less than Set_Point");
            break;
    case 5: outtextxy(170,250,"Proportional Gain (Kp) not equal zero");
            break;
    case 6: outtextxy(170,250,"Must select Control Action [0] or [1]");
            break;
    case 7: outtextxy(193,230,"Process_Var. more than High_Alarm");
            outtextxy(310,250,"or");
            outtextxy(195,270,"Process_Var. less than Low_Alarm");
            break;
   }
 if(getch())
   {
   setfillstyle(1,4);
   bar(290,295,350,317);
   setfillstyle(1,7);      /* box */
   bar(293,298,353,320);
   settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
   outtextxy(303,305,"OK.");
   Restore_Screen(159,159);
   }
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

 return(0);
}

Example_Screen()
{
  int d=0,i;
```

```
char Sp_d[8],Pv_d[8],Mv_d[8];
setfillstyle(1,0);                    /* clear display screen */
bar(0,41,639,458);
setfillstyle(1,0);                    /* clear sp loop2 cascade */
bar(401,200,471,210);
or(i=0;i<No_Loop;i++,d=321)
{
 setfillstyle(8,3);
 bar(0+d,41,318+d,458);                    /* background zone */
 setfillstyle(1,7);
 setcolor(4);
 bar(5+d,253,313+d,453);                   /* bargraph  zone */
 rectangle(7+d,255,311+d,451);
 setfillstyle(1,1);
 setcolor(14);
 bar(5+d,46,159+d,248);                    /* data zone */
 rectangle(7+d,48,157+d,246);
 setfillstyle(1,6);                  /* lower menu zone */
 bar(165+d,194,308+d,248);
 rectangle(167+d,196,306+d,246);
 bar(165+d,46,308+d,184);                  /* upper menu zone */
 rectangle(167+d,48,306+d,182);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
 outtextxy(192+d,51,"Menu Key");
 outtextxy(25,50,"DATA LOOP-1");
 if(No_Loop==2)  { outtextxy(346,50,"DATA LOOP-2"); }
 settextstyle(SMALL_FONT,HORIZ_DIR,4);
 setcolor(15);
 outtextxy(175+d,215,"MANUAL MODE");
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
 setcolor(14);
 outtextxy(15+d,80,  "<MAX> =");
 outtextxy(15+d,95,  "<MIN> =");
 outtextxy(15+d,115, "<Low> =");
 outtextxy(15+d,130, "<Hi.> =");
 outtextxy(15+d,150, "<Kp.> =");
 outtextxy(15+d,165, "<Ki.> =");
 outtextxy(15+d,180, "<Kd.> =");
 outtextxy(15+d,200, "<SP.> =");
 outtextxy(15+d,215, "<Time.> =");
```

```
outtextxy(15+d,230, "<Process> = ");
setcolor(15);
outtextxy(172+d,81,"<F1> Main Menu");
outtextxy(172+d,96,"<F2> Change Loop");
outtextxy(172+d,111,"<F3> Change Data");
outtextxy(172+d,126,"<F4> Auto/Manual");
outtextxy(172+d,141,"<F5> Dirt./Rev.");
outtextxy(172+d,156,"");
/* bargraph area */
setfillstyle(1,15);
bar(15+d,260,303+d,350);
rectangle(13+d,258,305+d,352);
setfillstyle(1,15);
bar(15+d,355,303+d,445);
rectangle(13+d,353,305+d,447);
/* draw block */
setfillstyle(9,15);
bar(89+d,265,289+d,285);
bar(89+d,295,289+d,315);
bar(89+d,325,289+d,345);
setcolor(8);
rectangle(88+d,264,290+d,286);
rectangle(87+d,263,291+d,287);
rectangle(88+d,294,290+d,316);
rectangle(87+d,293,291+d,317);
rectangle(88+d,324,290+d,346);
rectangle(87+d,323,291+d,347);
/* bargraph 's name */
setcolor(12);
outtextxy(25+d,265,"SP.");
setcolor(2);
outtextxy(25+d,295,"PV.");
setcolor(9);
outtextxy(25+d,325,"MV.");
/* first initial show value for bar graph */
setcolor(12);
sprintf(Sp_d," %.2f ",data[i].Set_Point);
outtextxy(15+d,280,Sp_d);
setcolor(2);
sprintf(Pv_d," %.2f ",remote[i].Pv);
```

```
      outtextxy(15+d,310,Pv_d);
      setcolor(9);
      sprintf(Mv_d," %.2f ",remote[i].Mv);
      outtextxy(15+d,340,Mv_d);
    }

    setcolor(15);
    rectangle(0,41,318,458);
    rectangle(1,42,317,457);
    rectangle(2,43,316,456);
    return(0);
}

Bar_Graph()
{
    int s=0,p=0,m=0,delta=0;
    int j=0;
    char Sp_d[8],Pv_d[8],Mv_d[8];
    /* show value .*/
      if(merror[Z].Sp_old!=data[Z].Set_Point)
      {
       setfillstyle(1,15);
       bar(15+d,280,85+d,286);
       sprintf(Sp_d," %5.2f ",data[Z].Set_Point);
       setcolor(12);
       outtextxy(15+d,280,Sp_d);
       merror[Z].Sp_old=data[Z].Set_Point;
      }
      if(merror[Z].Pv_old!=remote[Z].Pv)
      {
       setfillstyle(1,15);
       bar(15+d,310,85+d,316);
       sprintf(Pv_d," %5.2f ",remote[Z].Pv);
       setcolor(2);
       outtextxy(15+d,310,Pv_d);
       merror[Z].Pv_old=remote[Z].Pv;
      }
      if(merror[Z].Mv_old!=remote[Z].Mv)
      {
       setfillstyle(1,15);
```

```
 bar(15+d,340,85+d,346);
 sprintf(Mv_d," %5.2f ",remote[Z].Mv);
 setcolor(9);
 outtextxy(15+d,340,Mv_d);
 merror[Z].Mv_old=remote[Z].Mv;
}

/* BARGRAPH SP */
s=((data[Z].Set_Point-data[Z].Min)/(data[Z].Max-data[Z].Min))*200;
if (s<0)        s=0;
else if (s>200) s=200;
delta=s-merror[Z].s_old;
if(delta<0)
  {
  setfillstyle(9,15);
  bar(89+d+s,265,89+d+merror[Z].s_old,285);
  }
else if(delta>0)
        {
        setfillstyle(1,12);
        bar(89+d+merror[Z].s_old,268,89+d+s,282);
        }
merror[Z].s_old=s
                                                /* shift */
for(j=0;j<281;j++)
      Sp_Trend[Z][j]=Sp_Trend[Z][j+1];
if(s>190) s=190;
else if(s<10) s=10;
                                                /* select new data */
Sp_Trend[Z][281]=s/2;
                                                /* clear trend graph */
setfillstyle(1,15);
bar(15+d,355,295+d,445);
                                                /* display */
for(j=281;j>1;j--)
  {
  setcolor(12);
  line(15+d+j-2,(450-Sp_Trend[Z][j-1]),15+d+j-1,(450-Sp_Trend[Z][j]));
  }
```

```
/* BARGRAPH PV */
p=((remote[Z].Pv-data[Z].Min)/(data[Z].Max-data[Z].Min))*200;
if (p<0) p=0;
else if (p>200) p=200;
delta=p-merror[Z].p_old;
if(delta<0)
  {
  setfillstyle(9,15);
  bar(89+d+p,295,89+d+merror[Z].p_old,315);
  }
else if(delta>0)
  {
  setfillstyle(1,2);
  bar(89+d+merror[Z].p_old,298,89+d+p,312);
  }
merror[Z].p_old=
                                        /* shift */
for(j=0;j<281;j++)
      Pv_Trend[Z][j]=Pv_Trend[Z][j+1];
if(p>190) p=190;
else if(p<10) p=10;
                                        /* select new data */
Pv_Trend[Z][281]=p/2;
                                        /* delta check for display or clear */
delta=Pv_Trend[Z][281]-Pv_Trend[Z][280];
                                        /* display bar */
if(delta>0)
{
setfillstyle(1,2);
bar(296+d,(450-Pv_Trend[Z][281]),303+d,(450-Pv_Trend[Z][280]));
}
                                        /* display */
for(j=281;j>1;j--)
  {
  setcolor(2);
  line(15+d+j-2,(450-Pv_Trend[Z][j-1]),15+d+j-1,(450-Pv_Trend[Z][j]));
  }
                                        /* clear bar */
if(delta<0)
      {
```

```
            setfillstyle(1,15);
            bar(296+d,(450-Pv_Trend[Z][280]),303+d,(450-Pv_Trend[Z][281]));
            }

     /* BARGRAPH MV */
     m=((remote[Z].Mv)/100)*200;
     if (m<0)   m=0;
     else if (m>200) m=200;
     delta=m-merror[Z].m_old;
     if(delta<0)
       {
       setfillstyle(9,15);
       bar(89+d+m,325,89+d+merror[Z].m_old,345);
       }
     else if(delta>0)
            {
            setfillstyle(1,9);
            bar(89+d+merror[Z].m_old,328,89+d+m,342);
            }
     merror[Z].m_old=m;
return(0);
}

Check_Keyboard()
 {
   int r,d=0;
   while(kbhit())
   {
    r=Read_Key(0);
    switch(r)
    {
     case F1: Switch_new=1;/* Swich for return Select_menu ==> Protect ESC */
              Inverse(0,0);                              /* Main Menu */
              Select_Menu(0,1);
     break;
     case F2: if((No_Loop==2)&&(change_data[Loop].Switch_On==0))
              {                                                  /* Change Loop */
              if(Loop==0)   { Loop=1;d=0; }
              else          { Loop=0;d=321; }
              setcolor(8);
```

```
            rectangle(0+d,41,318+d,458);
            rectangle(1+d,42,317+d,457);
            rectangle(2+d,43,316+d,456);
            setcolor(15);
            rectangle(321-d,41,639-d,458);
            rectangle(322-d,42,638-d,457);
            rectangle(323-d,43,637-d,456);
          }
break;
case F3: if(change_data[Loop].Switch_On==0)      /* Change Data */
          {
           change_data[Loop].Switch_On=1;
           change_data[Loop].Change=1;
           if(Loop==0)   d=0;                    /* Change loop 1 */
           else          d=321;                  /* Change loop 2 */
           Save_Screen(165+d,46,308+d,184);
           setfillstyle(1,9);
           bar(165+d,46,308+d,184);
           rectangle(167+d,48,306+d,182);
           settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
           setcolor(15);
           outtextxy(180+d,51,"Change Data");
           settextstyle(SMALL_FONT,HORIZ_DIR,4);
           setcolor(10);
           outtextxy(172+d,81,"<Pg_up>");
           outtextxy(172+d,96,"<Pg_dn>");
           outtextxy(172+d,111,"<Up_arrow>");
           outtextxy(172+d,126,"<Dn_arrow>");
           outtextxy(172+d,141,"<Enter>");
           setcolor(13);
           outtextxy(240+d,81,"Upper Data");
           outtextxy(240+d,96,"Lower Data");
           outtextxy(248+d,111,"Increment");
           outtextxy(248+d,126,"Decrement");
           outtextxy(225+d,141,"Data Transfer");
           setfillstyle(1,15);
           bar(171+d,155,302+d,179);
           settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
          }
          else
```

```
        {
         change_data[Loop].Switch_On=0;
         if(Loop==0) Restore_Screen(165,46);  /* Change loop 1 */
         else        Restore_Screen(486,46);  /* Change loop 2 */
        }
break;
case F4: Change_Mode();                  /* Change Auto - Manual */
break;
case F5: Change_Action();
break;
case RIGHT_ARROW: if(status[Loop].Mode==MANUAL)
                    remote[Loop].Mv+=0.5;
                  if(remote[Loop].Mv>100) remote[Loop].Mv=100;
break;
case LEFT_ARROW : if(status[Loop].Mode==MANUAL)
                    remote[Loop].Mv-=0.5;
                  if(remote[Loop].Mv<0)   remote[Loop].Mv=0;
break;
case UP_ARROW   : if(change_data[Loop].Switch_On==1)
                  {
                   if(((change_data[Loop].Choice>=1)
                    &&(change_data[Loop].Choice<=9))
                    ||((change_data[Loop].Choice>=12)
                    &&(change_data[Loop].Choice<=14)))
                   change_data[Loop].Dummy+=0.1;
                  else if(change_data[Loop].Choice==10)
                   change_data[Loop].Dum_int=LEVEL;
                  else if(change_data[Loop].Choice==11)
                   change_data[Loop].Dum_int=SECOND;
                  }

break;
case DOWN_ARROW : if(change_data[Loop].Switch_On==1)
                  {
                   if(((change_data[Loop].Choice>=1)
                    &&(change_data[Loop].Choice<=9))
                    ||((change_data[Loop].Choice>=12)
                    &&(change_data[Loop].Choice<=14)))
                   change_data[Loop].Dummy-=0.1;
                  else if(change_data[Loop].Choice==10)
```

```
                        change_data[Loop].Dum_int=FLOW;
                      else if(change_data[Loop].Choice==11)
                        change_data[Loop].Dum_int=FIRST;
                    }
    break;
    case Page_Up    : if(change_data[Loop].Switch_On==1)
                    {
                      change_data[Loop].Change=1;
                      change_data[Loop].Choice--=1;
                    }
    break;
    case Page_Down  : if(change_data[Loop].Switch_On==1)
                    {
                      change_data[Loop].Change=1;
                      change_data[Loop].Choice+=1;
                    }
    break;
   }
   Change_Data(r);
  }
  return(0);
}

Change_Data(r)
{
 int d,i;
 char Max[8],Min[8],Low[8],Hi[8],Kp[8],Ki[8],Kd[8],Sp[8],Time[8];
 char Ks[8],T1[8],T2[8];
 if(CTSM==CONTROL)
 {
  if(change_data[Loop].Choice<1)   change_data[Loop].Choice=10;
  if(change_data[Loop].Choice>10)  change_data[Loop].Choice=1;
 }

 if(sim[Loop].ORDER==FIRST)
 {
  if(change_data[Loop].Choice<1)    change_data[Loop].Choice=13;
  if(change_data[Loop].Choice>13)   change_data[Loop].Choice=1;
 }
```

```
else if(sim[Loop].ORDER==SECOND)
    {
    if(change_data[Loop].Choice<1)    change_data[Loop].Choice=14;
    if(change_data[Loop].Choice>14)   change_data[Loop].Choice=1;
    }

if(change_data[Loop].Switch_On==1)
{
 setcolor(12);
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    if(Loop==0)   {i=0,d=0;}              /* Change loop 1 */
    else          {i=1,d=321;}            /* Change loop 2 */
switch(change_data[Loop].Choice)
 {
  case 1: setfillstyle(1,14);
          bar(171+d,155,302+d,179);
          setcolor(8);
          rectangle(172+d,156,301+d,178);
          settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
          setcolor(2);
          outtextxy(172+d,165,"<MAX> =");
          if(change_data[Loop].Change)
          {
           change_data[Loop].Dummy=data[Loop].Max;
           change_data[Loop].Change=0;
          }
          if(change_data[Loop].Dummy<data[Loop].High_Alarm)
            change_data[Loop].Dummy=data[Loop].High_Alarm;
          sprintf(Max,"%6.2f ",change_data[Loop].Dummy);
          setcolor(12);
          outtextxy(241+d,165,Max);
          switch(r)
          {
           case ENTER: data[Loop].Max=change_data[Loop].Dummy;
                       change_data[Loop].Switch_On=0;
                       if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                       else        Restore_Screen(486,46); /* Change loop 2 */
                      setfillstyle(1,1);
                       bar(80+d,80,150+d,90);
                       sprintf(Max," %6.2f ",data[Loop].Max);
```

```
                    setcolor(12);
                    outtextxy(80+d,80,Max);
                break;
               }
        break;
        case 2: setfillstyle(1,14);
                bar(171+d,155,302+d,179);
                setcolor(8);
                rectangle(172+d,156,301+d,178);
                settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                setcolor(2);
                outtextxy(172+d,165,"<MIN> =");
                if(change_data[Loop].Change)
                {
                 change_data[Loop].Dummy=data[Loop].Min;
                 change_data[Loop].Change=0;
                }

              if(change_data[Loop].Dummy>data[Loop].Low_Alarm)
                 change_data[Loop].Dummy=data[Loop].Low_Alarm;
                sprintf(Min,"%6.2f ",change_data[Loop].Dummy);
                setcolor(12);
                outtextxy(241+d,165,Min);
                switch(r)
                {
                 case ENTER: data[Loop].Min=change_data[Loop].Dummy;
                             change_data[Loop].Switch_On=0;
                             if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                             else        Restore_Screen(486,46); /* Change loop 2 */
                            setfillstyle(1,1);
                             bar(80+d,95,150+d,105);
                             sprintf(Min,"  %6.2f ",data[i].Min);
                             setcolor(12);
                             outtextxy(80+d,95,Min);
                 break;
                }
        break;
        case 3: setfillstyle(1,14);
                bar(171+d,155,302+d,179);
                setcolor(8);
```

```
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<Low> =");
        if(change_data[Loop].Change)
        {
         change_data[Loop].Dummy=data[Loop].Low_Alarm;
         change_data[Loop].Change=0;
        }

        if(change_data[Loop].Dummy<data[Loop].Min)
          change_data[Loop].Dummy=data[Loop].Min;
        if(change_data[Loop].Dummy>(data[Loop].Set_Point-0.1))
          change_data[Loop].Dummy=data[Loop].Set_Point-0.1;
        sprintf(Low,"%6.2f ",change_data[Loop].Dummy);
        setcolor(12);
        outtextxy(241+d,165,Low);
        switch(r)
        {
         case ENTER: data[Loop].Low_Alarm=change_data[Loop].Dummy;
                     change_data[Loop].Switch_On=0;
                     if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                     else        Restore_Screen(486,46); /* Change loop 2 */
                    setfillstyle(1,1);
                     bar(80+d,115,150+d,125);
                     sprintf(Low,"  %6.2f ",data[i].Low_Alarm);
                     setcolor(12);
                     outtextxy(80+d,115,Low);
         break;
        }
break;
case 4: setfillstyle(1,14);
        bar(171+d,155,302+d,179);
        setcolor(8);
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<Hi.> =");
        if(change_data[Loop].Change)
        {
```

```
          change_data[Loop].Dummy=data[Loop].High_Alarm;
          change_data[Loop].Change=0;
         }

       if(change_data[Loop].Dummy>data[Loop].Max)
          change_data[Loop].Dummy=data[Loop].Max;
         if(change_data[Loop].Dummy<(data[Loop].Set_Point+0.1))
          change_data[Loop].Dummy=data[Loop].Set_Point+0.1;
         sprintf(Hi,"%6.2f ",change_data[Loop].Dummy);
         setcolor(12);
         outtextxy(241+d,165,Hi);
         switch(r)
         {
          case ENTER: data[Loop].High_Alarm=change_data[Loop].Dummy;
                      change_data[Loop].Switch_On=0;
                      if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                      else        Restore_Screen(486,46); /* Change loop 2 */
                      setfillstyle(1,1);
                      bar(80+d,130,150+d,140);
                      sprintf(Hi," %6.2f ",data[i].High_Alarm);
                      setcolor(12);
                      outtextxy(80+d,130,Hi);
          break;
         }
break;
case 5: setfillstyle(1,14);
         bar(171+d,155,302+d,179);
         setcolor(8);
         rectangle(172+d,156,301+d,178);
         settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
         setcolor(2);
         outtextxy(172+d,165,"<Kp.> =");
         if(change_data[Loop].Change)
         {
          change_data[Loop].Dummy=data[Loop].Kp;
          change_data[Loop].Change=0;
         }

       if(change_data[Loop].Dummy<0.1)
          change_data[Loop].Dummy=0.1;
```

```
        sprintf(Kp,"%6.2f ",change_data[Loop].Dummy);
        setcolor(12);
        outtextxy(241+d,165,Kp);
        case ENTER: data[Loop].Kp=change_data[Loop].Dummy;
                    change_data[Loop].Switch_On=0;
                    if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                    else        Restore_Screen(486,46); /* Change loop 2 */
                    setfillstyle(1,1);
                    bar(80+d,150,150+d,160);
                    sprintf(Kp," %6.2f ",data[i].Kp);
                    setcolor(12);
                    outtextxy(80+d,150,Kp);
         break;
        }
break;
case 6: setfillstyle(1,14);
        bar(171+d,155,302+d,179);
        setcolor(8);
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<Ki.> =");
        if(change_data[Loop].Change)
        {
         change_data[Loop].Dummy=data[Loop].Ki;
         change_data[Loop].Change=0;
        }

      if(change_data[Loop].Dummy<0)
        change_data[Loop].Dummy=0;
        sprintf(Ki,"%6.2f ",change_data[Loop].Dummy);
        setcolor(12);
        outtextxy(241+d,165,Ki);
        case ENTER: data[Loop].Ki=change_data[Loop].Dummy;
                    change_data[Loop].Switch_On=0;
                    if(Loop==0) Restore_Screen(165,46); /* Change loop 1 *
                    else        Restore_Screen(486,46); /* Change loop 2 */
                    setfillstyle(1,1);
                    bar(80+d,165,150+d,175);
                    sprintf(Ki," %6.2f ",data[i].Ki);
```

```
                    setcolor(12);
                    outtextxy(80+d,165,Ki);
          break;
         }
break;
case 7: setfillstyle(1,14);
         bar(171+d,155,302+d,179);
         setcolor(8);
         rectangle(172+d,156,301+d,178);
         settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
         setcolor(2);
         outtextxy(172+d,165,"<Kd.> =");
         if(change_data[Loop].Change)
         {
          change_data[Loop].Dummy=data[Loop].Kd;
          change_data[Loop].Change=0;
         }

         if(change_data[Loop].Dummy<0)
           change_data[Loop].Dummy=0;
         sprintf(Kd,"%6.2f ",change_data[Loop].Dummy);
         setcolor(12);
         outtextxy(241+d,165,Kd);

         switch(r)
         {
          case ENTER: data[Loop].Kd=change_data[Loop].Dummy;
                    change_data[Loop].Switch_On=0;
                    if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                    else        Restore_Screen(486,46); /* Change loop 2 */
                    setfillstyle(1,1);
                    bar(80+d,180,150+d,190);
                    sprintf(Kd," %6.2f ",data[i].Kd);
                    setcolor(12);
                    outtextxy(80+d,180,Kd);
          break;
         }
break;
case 8: setfillstyle(1,14);
         bar(171+d,155,302+d,179);
```

```
        setcolor(8);
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<Sp.> =");
        if(change_data[Loop].Change)
        {
         change_data[Loop].Dummy=data[Loop].Set_Point;
         change_data[Loop].Change=0;
        }

        if(change_data[Loop].Dummy>(data[Loop].High_Alarm-0.1))
          change_data[Loop].Dummy=data[Loop].High_Alarm-0.1;
        if(change_data[Loop].Dummy<(data[Loop].Low_Alarm+0.1))
          change_data[Loop].Dummy=data[Loop].Low_Alarm+0.1;
        sprintf(Sp,"%6.2f ",change_data[Loop].Dummy);
        setcolor(12);
        outtextxy(241+d,165,Sp);

        switch(r)
        {
         case ENTER: data[Loop].Set_Point=change_data[Loop].Dummy;
                     change_data[Loop].Switch_On=0;
                     if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                     else        Restore_Screen(486,46); /* Change loop 2 */
                     setfillstyle(1,1);
                     bar(80+d,200,150+d,210);
                     sprintf(Sp," %6.2f ",data[i].Set_Point);
                     setcolor(12);
                     outtextxy(80+d,200,Sp);
         break;
        }
break;
case 9: setfillstyle(1,14);
        bar(171+d,155,302+d,179);
        setcolor(8);
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<Time.> =");
```

```
        if(change_data[Loop].Change)
        {
         change_data[Loop].Dummy=data[Loop].Time;
         change_data[Loop].Change=0;
        }

        if(change_data[Loop].Dummy<=1)
           change_data[Loop].Dummy=1;
        sprintf(Time,"%5.2f ",change_data[Loop].Dummy);
        setcolor(12);
        outtextxy(251+d,165,Time);

        switch(r)
        {
         case ENTER: data[Loop].Time=change_data[Loop].Dummy;
                     change_data[Loop].Switch_On=0;
                     if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                     else        Restore_Screen(486,46); /* Change loop 2 */
                     setfillstyle(1,1);
                     bar(95+d,215,150+d,225);
                     sprintf(Time," %5.2f ",data[i].Time);
                     setcolor(12);
                     outtextxy(95+d,215,Time);
         break;
        }
break;
case 10:setfillstyle(1,14);
        bar(171+d,155,302+d,179);
        setcolor(8);
        rectangle(172+d,156,301+d,178);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        setcolor(2);
        outtextxy(172+d,165,"<PROCESS>=");
        if(change_data[Loop].Change)
         {
          change_data[Loop].Dum_int=data[Loop].Process;
          change_data[Loop].Change=0;
         }
        setcolor(12);
        if(change_data[Loop].Dum_int==1)
```

```
            outtextxy(256+d,165,"LEVEL");
          else
            outtextxy(256+d,165,"FLOW");

          switch(r)
          {
           case ENTER: data[Loop].Process=change_data[Loop].Dum_int;
                       change_data[Loop].Switch_On=0;
                       if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                       else        Restore_Screen(486,46); /* Change loop 2 */
                       setfillstyle(1,1);
                       bar(110+d,230,150+d,240);
                       setcolor(12);
                       if(data[i].Process==1)
                         outtextxy(110+d,230,"LEVEL");
                       else
                         outtextxy(110+d,230,"FLOW");
           break;
          }
break;
case 11: setfillstyle(1,14);
          bar(171+d,155,302+d,179);
          setcolor(8);
          rectangle(172+d,156,301+d,178);
          settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
          setcolor(2);
          outtextxy(172+d,165,"<Order> =");
          if(change_data[Loop].Change)
          {
           change_data[Loop].Dum_int=sim[Loop].ORDER;
           change_data[Loop].Change=0;
          }
          setcolor(12);
          if(change_data[Loop].Dum_int==1)
            outtextxy(251+d,165,"FIRST");
          else
            outtextxy(251+d,165,"SECOND");

          switch(r)
          {
```

```
            case ENTER: sim[Loop].ORDER=change_data[Loop].Dum_int;
                        change_data[Loop].Switch_On=0;
                        if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                        else        Restore_Screen(486,46); /* Change loop 2 */
             break;
            }
  break;
  case 12: setfillstyle(1,14);
           bar(171+d,155,302+d,179);
           setcolor(8);
           rectangle(172+d,156,301+d,178);
           settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
           setcolor(2);
           outtextxy(172+d,165,"<Ks.> =");
           if(change_data[Loop].Change)
           {
            change_data[Loop].Dummy=sim[Loop].Ks;
            change_data[Loop].Change=0;
           }

           sprintf(Ks,"%6.2f ",change_data[Loop].Dummy);
           setcolor(12);
           outtextxy(241+d,165,Ks);

           switch(r)
           {
            case ENTER: sim[Loop].Ks=change_data[Loop].Dummy;
                        change_data[Loop].Switch_On=0;
                        if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                        else        Restore_Screen(486,46); /* Change loop 2 */
             break;
           }
 break;
 case 13: setfillstyle(1,14);
          bar(171+d,155,302+d,179);
          setcolor(8);
          rectangle(172+d,156,301+d,178);
          settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
          setcolor(2);
          outtextxy(172+d,165,"<T1> =");
```

```
            if(change_data[Loop].Change)
            {
             change_data[Loop].Dummy=sim[Loop].T1;
             change_data[Loop].Change=0;
            }

            if(change_data[Loop].Dummy<=0.1)
               change_data[Loop].Dummy=0.1;
            sprintf(T1,"%6.2f ",change_data[Loop].Dummy);
            setcolor(12);
            outtextxy(241+d,165,T1);

            switch(r)
            {
             case ENTER: sim[Loop].T1=change_data[Loop].Dummy;
                         change_data[Loop].Switch_On=0;
                         if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                         else        Restore_Screen(486,46); /* Change loop 2 */
             break;
            }
break;
case 14: setfillstyle(1,14);
            bar(171+d,155,302+d,179);
            setcolor(8);
            rectangle(172+d,156,301+d,178);
            settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
            setcolor(2);
            outtextxy(172+d,165,"<T2> =");
            if(change_data[Loop].Change)
            {
             change_data[Loop].Dummy=sim[Loop].T2;
             change_data[Loop].Change=0;
            }

            if(change_data[Loop].Dummy<=0.1)
               change_data[Loop].Dummy=0.1;
            sprintf(T2,"%6.2f ",change_data[Loop].Dummy);
            setcolor(12);
            outtextxy(241+d,165,T2);
```

```
            switch(r)
            {
             case ENTER: sim[Loop].T2=change_data[Loop].Dummy;
                         change_data[Loop].Switch_On=0;
                         if(Loop==0) Restore_Screen(165,46); /* Change loop 1 */
                         else        Restore_Screen(486,46); /* Change loop 2 */
                      //  Display_Data();
             break;
            }
   break;
  }
  }
  return(0);
}

Change_Mode()
{
   int d;

   if(Loop==0)  d=0;
   else         d=321;

   setfillstyle(1,6);
   bar(174+d,215,260+d,225);
   settextstyle(SMALL_FONT,HORIZ_DIR,4);
   setcolor(15);

   if(status[Loop].Mode==MANUAL)
   {
    outtextxy(175+d,215,"AUTO MODE");
    status[Loop].Mode=AUTO;
    merror[Loop].Error1=0;
    merror[Loop].Error2=0;
   }
   else
   {
    outtextxy(175+d,215,"MANUAL MODE");
    status[Loop].Mode=MANUAL;
   }
   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
```

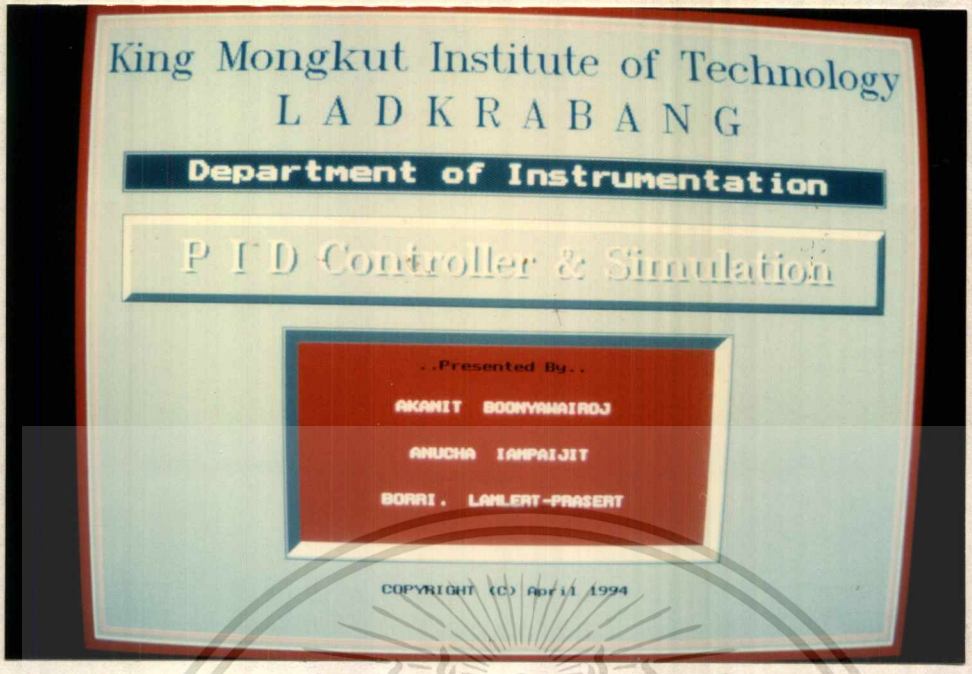
```
    return(0);
}

Change_Action()
{
    int d;

    if(Loop==0)  d=0;
    else         d=321;

    setfillstyle(1,6);
    bar(174+d,227,285+d,237);
    settextstyle(SMALL_FONT,HORIZ_DIR,4);
    setcolor(15);

    if(status[Loop].Action==DIRECT)
    {
     outtextxy(175+d,227,"REVERSE_ACTION Pro.");
     status[Loop].Action=REVERSE;
    }
    else
    {
     outtextxy(175+d,227,"DIRECT_ACTION  Pro.");
     status[Loop].Action=DIRECT;
    }
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    return(0);
}

Waiting()
{
   setfillstyle(1,5);
   bar(160,160,480,330);
   setfillstyle(1,8);      /* shadow */
   bar(205,175,445,200);
   setfillstyle(1,7);      /* box */
   bar(200,170,440,195);
   settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
   setcolor(14);
   rectangle(162,162,478,328);
```

```
    outtextxy(220,175,"W A I T I N G.");
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    outtextxy(195,250,"...PROGRAM READ FILE DATA.FIL...");
    delay(2000);
    cls();
    return(0);
}
Time()
{
    char Tm[8];

    if(tm.milliseconds==1000) {
       tm.milliseconds=0;
       tm.seconds++;          }
    if(tm.seconds==60) {
       tm.seconds=0;
       tm.minutes++;  }
    if(tm.minutes==60) {
       tm.minutes=0;
       tm.hours++;    }
    if(tm.hours==24)
       tm.hours=0;
    if(tm.milliseconds==0)
      {
       sprintf(Tm,"%2d:%2d:%2d",tm.hours,tm.minutes,tm.seconds);
       setfillstyle(1,11);
       bar(555,460,639,479);
       setcolor(0);
       outtextxy(565,465,Tm);
      }
    return 0;
}
/* Create a siren effect. */
void siren(void)
{
  unsigned freq;
  unsigned long i;
  union {
    long divisor;
    unsigned char c[2];
```

```
} count;

unsigned char p;

p = inportb(97); /* get existing bit pattern */
outportb(97, p | 3); /* turn on bits 0 and 1 */
/* ascending siren */
for(freq=1000; freq<3000; freq+=RATE) {
  count.divisor = 1193280 / freq;  /* compute the proper count */
  outportb(67, 182); /* tell 8253 that a count is coming */
  outportb(66, count.c[0]); /* send low-order byte */
  outportb(66, count.c[1]); /* send high-order byte */

  for(i=0; i<DELAY; ++i) ;
}

/* descending siren */
for( ; freq>1000; freq-=RATE) {
  count.divisor = 1193280 / freq;  /* compute the proper count */
  outportb(67, 182); /* tell 8253 that a count is coming */
  outportb(66, count.c[0]); /* send low-order byte */
  outportb(66, count.c[1]); /* send high-order byte */

  for(i=0; i<DELAY; ++i) ;
}
outportb(97, p); /* restore original bits to turn off speaker */
}
```

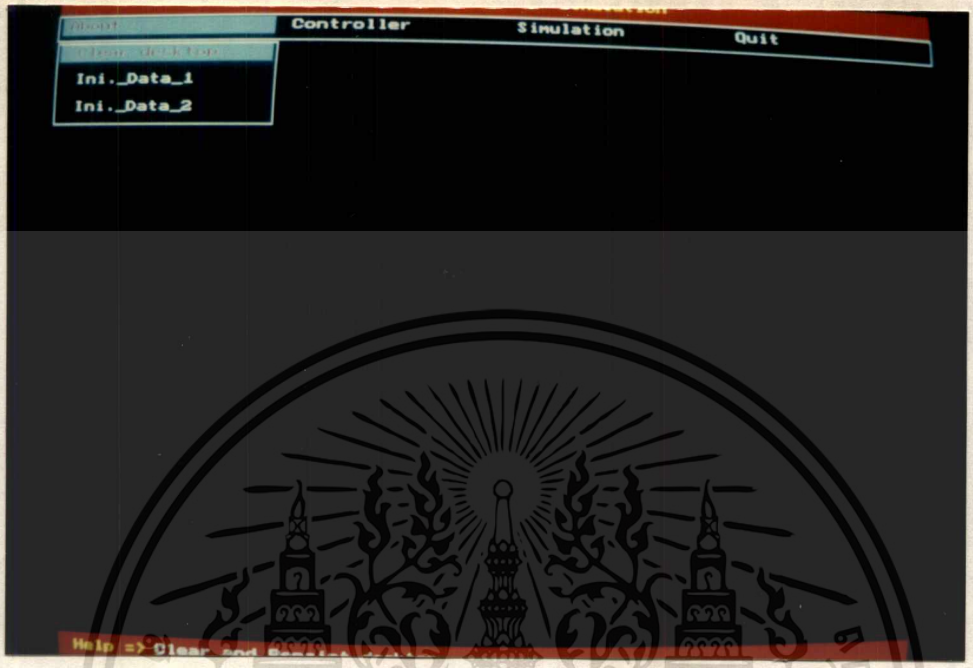
ภาค 4

ผลการทดลอง.

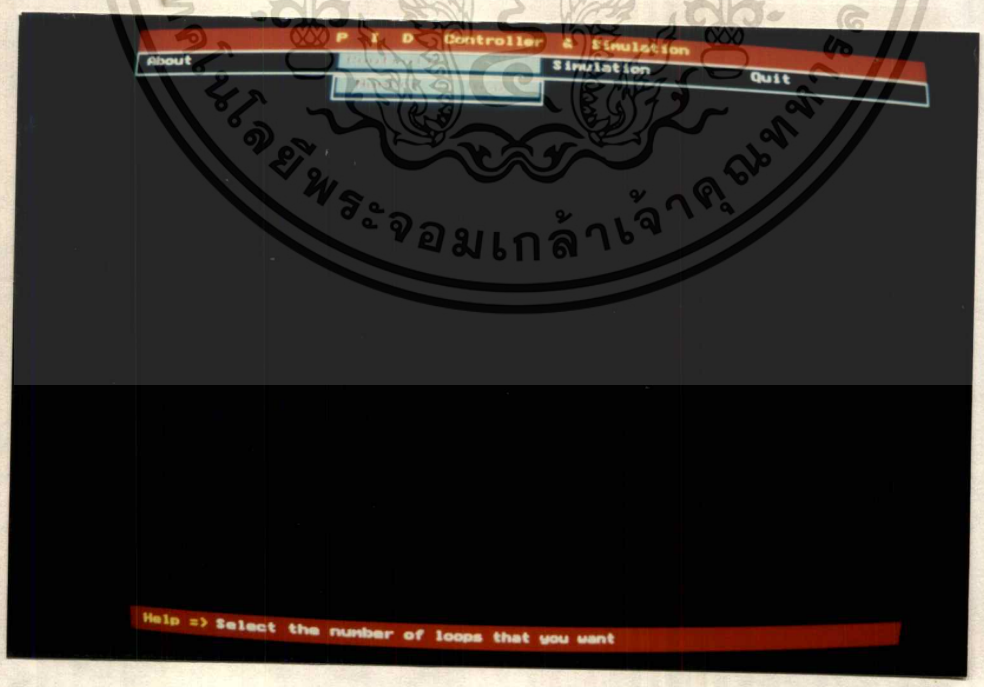


รูปที่ 1 หน้าจอขณะเข้าสู่การทำงานของโปรแกรม**PID Controller & Simulation**

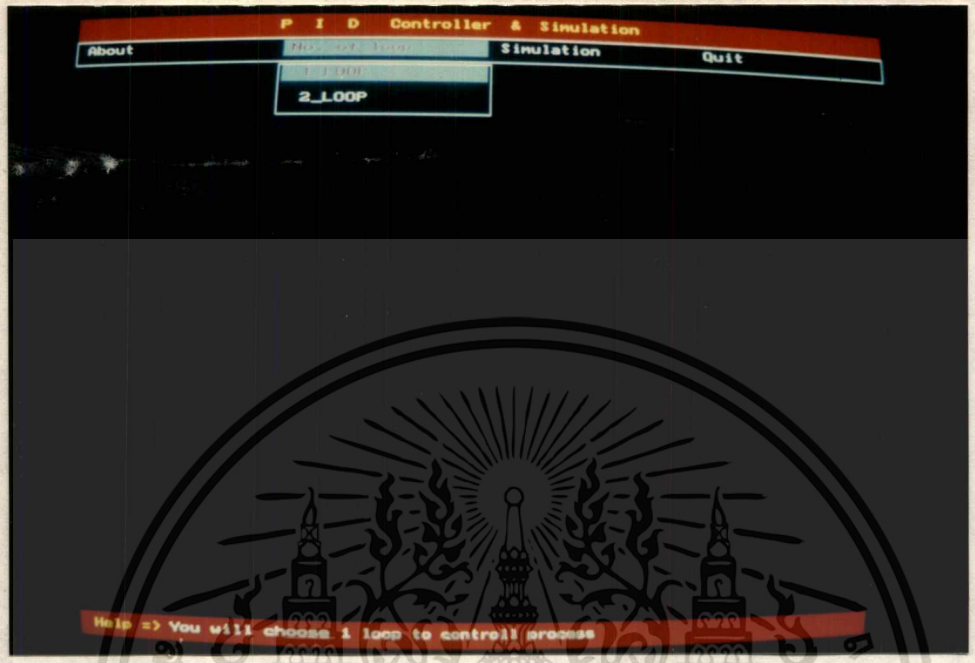
รูปที่ 2 เมื่อมีการกดปุ่มใดๆ ขณะที่หน้าจอรูปที่ 1 แสดงผลอยู่จะทำให้ได้หน้าจอตามรูปที่ 2 และมีการอ่านข้อมูลจากไฟล์ที่เก็บข้อมูลของค่าเริ่มต้นของกระบวนการ



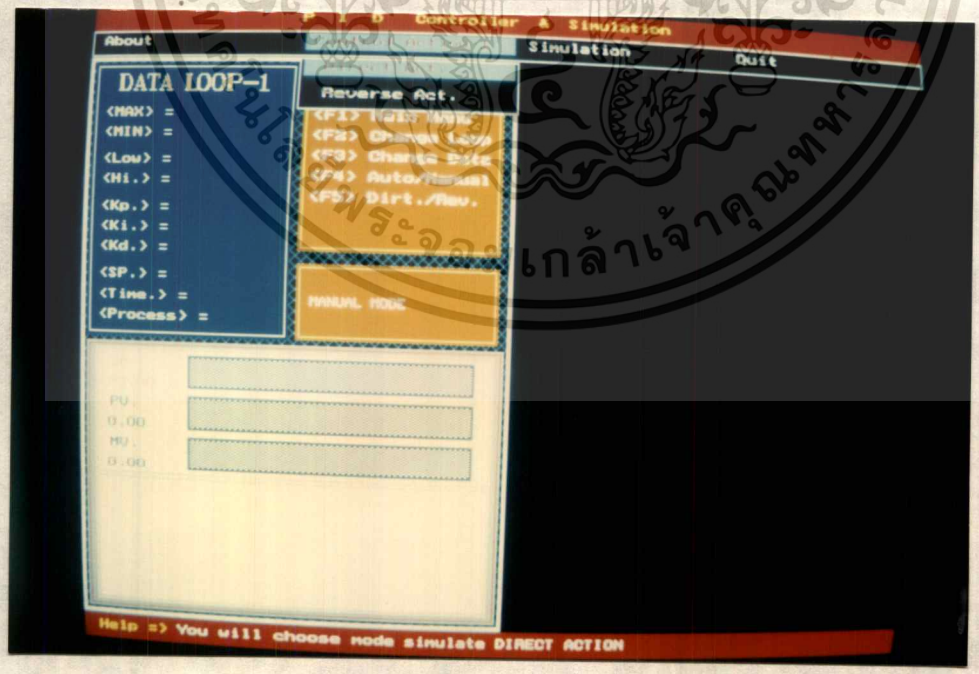
รูปที่ 3 เมื่ออ่านไฟล์ได้เรียบร้อยแล้ว จะขึ้นหน้าจอตามรูปที่ 3 เพื่อให้ผู้ใช้งานเลือกการทำงาน



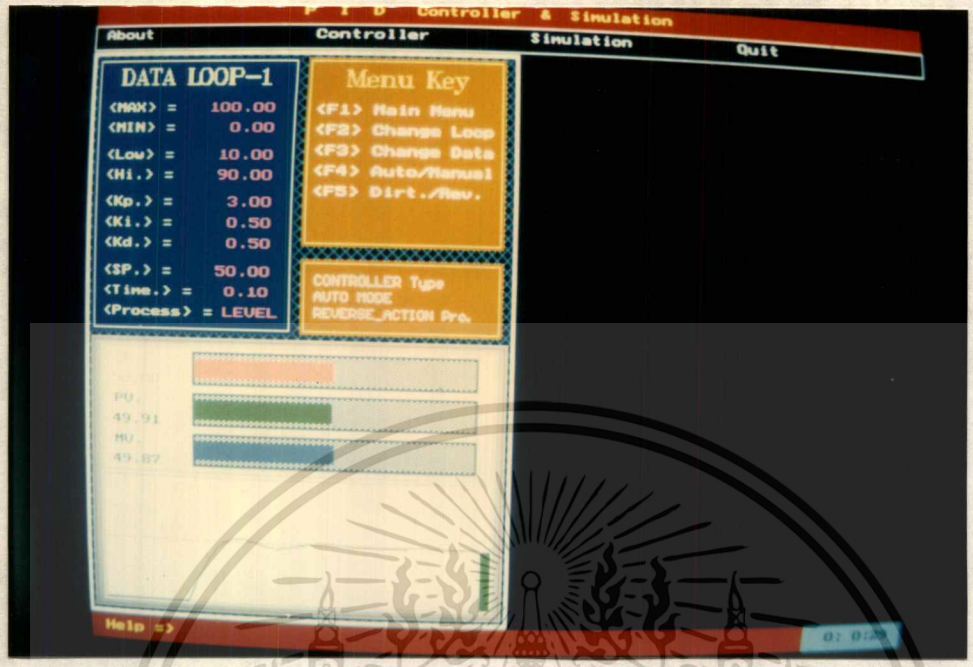
รูปที่ 4 เมื่อผู้ใช้งานกด key → (Right Arrow)1 ครั้ง จะเลือกการทำงานของโปรแกรมเป็นแบบ **Controller**



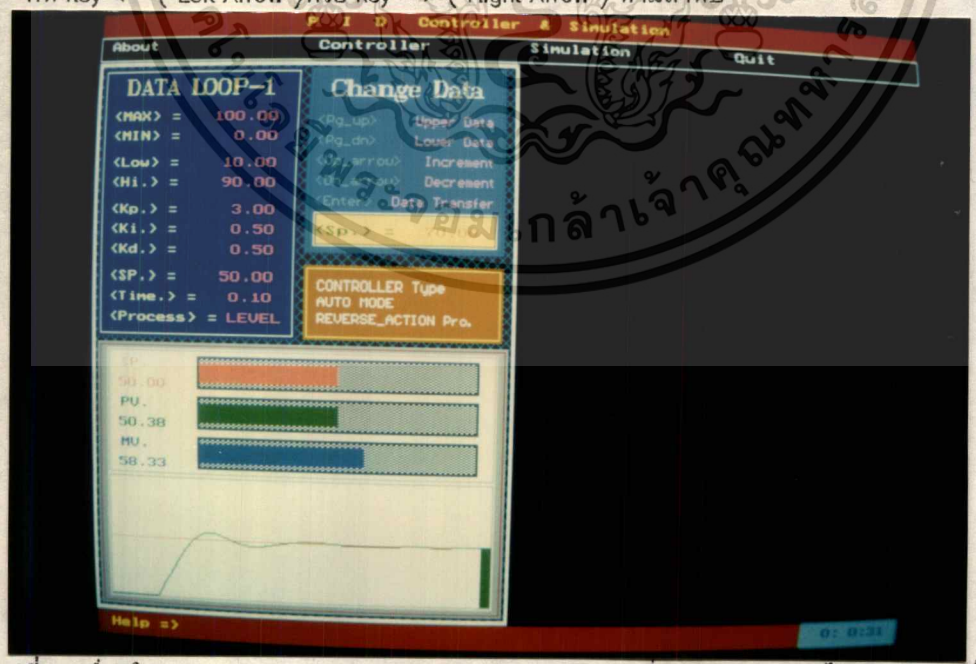
รูปที่ 5 เมื่อผู้ใช้งานกด key ↵ (Enter) เลือกการทำงานของโปรแกรมเป็นแบบ controller แล้ว โปรแกรมจะให้ผู้ใช้งานเลือกการทำงานว่าต้องการควบคุมกระบวนการจำนวน เท่าใด



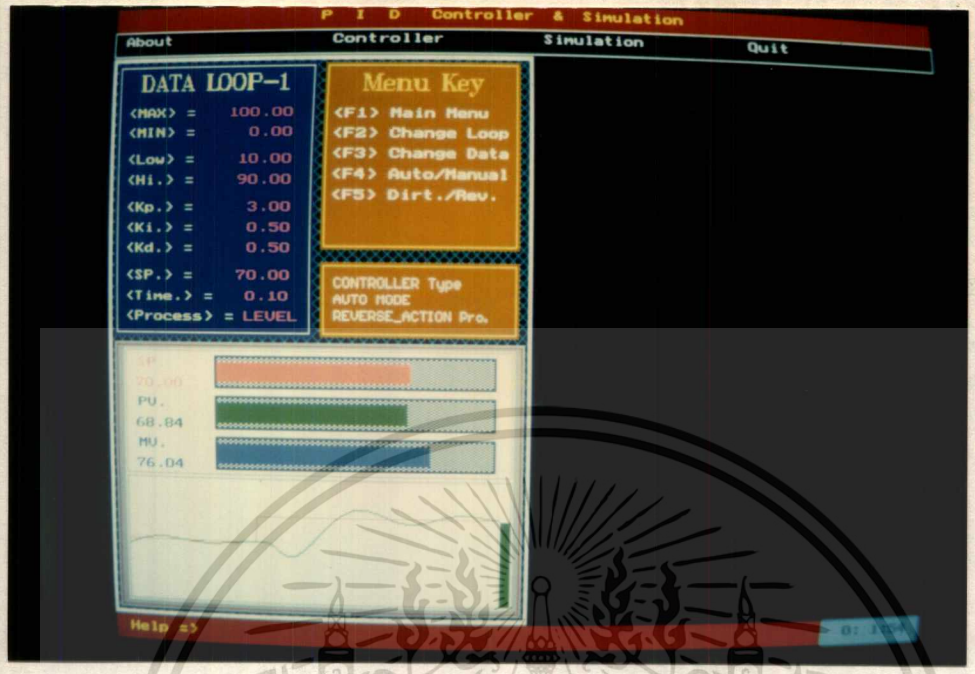
รูปที่ 6 เมื่อผู้ใช้งานกด key ↵ (Enter) เลือกการทำงานของโปรแกรมเข้าสู่การทำงานเพื่อควบคุมกระบวนการ ซึ่งจะเป็นการควบคุมแบบ controller และมีการทำงานควบคุมกระบวนการจำนวน 1 กระบวนการ



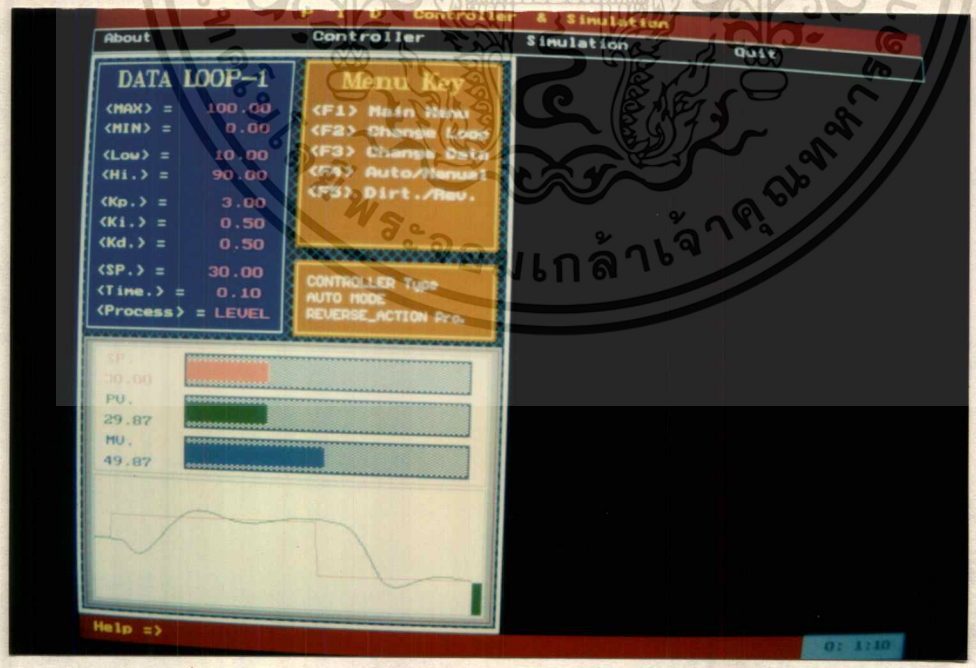
รูปที่ 7 เมื่อผู้ใช้งานเลือกการควบคุมแบบ **Reverse action** โดยการกด key ↓ (Down Arrow) และ key ↵ (Enter)จะทำให้โปรแกรมเข้าสู่กระบวนการซึ่งจะเป็นการควบคุมแบบ**Manual** ขณะที่มีการควบคุมโหมด Manual อยู่นี้ จะสามารถเพิ่มหรือลดค่า **MV** ด้วยการกด key ← (Left Arrow)หรือ key → (Right Arrow) ตามลำดับ



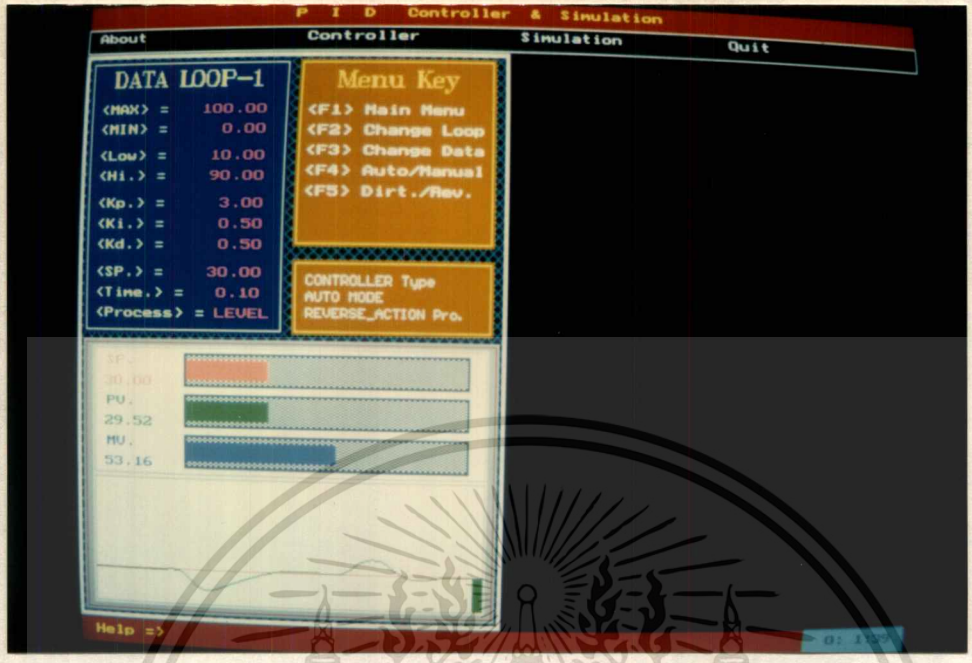
รูปที่ 8 เมื่อผู้ใช้งานกด **key F4** การควบคุมกระบวนการจะเปลี่ยนจาก Manual ไปเป็น **Auto** ซึ่งขณะที่ทำการควบคุมกระบวนการอยู่นี้ไม่ว่าจะอยู่ โหมดการทำงาน Manual หรือ Auto ก็ตาม ผู้ใช้งานสามารถเปลี่ยนค่าข้อมูลในส่วนพื้นสีน้ำเงินด้วยการกด**key F3** เพื่อให้ได้หน้าจอส่วน **Change Data** ปรากฏขึ้นมา



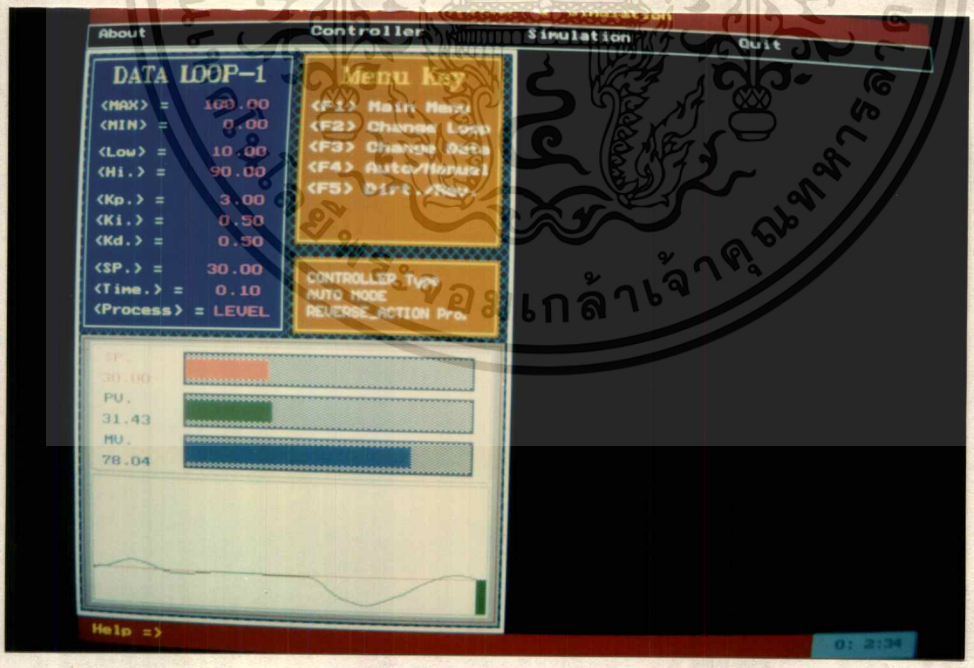
รูปที่ 9 เมื่อระบบเข้าสู่สภาวะเสถียรแล้ว ลองทำการเปลี่ยนค่า **Set Point** โดยการกด **key F3** เพื่อให้ได้หน้าจอตามรูปที่ 8 แล้วกด **key Pg-up** หรือ **key Pg-dw** เพื่อเลือกค่าตัวแปรที่ต้องการเปลี่ยนแปลงค่า เมื่อได้ตัวแปรที่ต้องการแล้วสามารถเพิ่มหรือลดค่าตัวแปรได้ด้วยการกด **key** ↓ **(Down** Arrow)ตามลำดับ จากรูปที่ 9 แสดงให้เห็นถึงผลการเปลี่ยนที่การเพิ่มค่า Set Point จาก 50 ไปเป็น 70



รูปที่ 10 แสดงให้เห็นการเปลี่ยนแปลงที่มีผลต่อการลดค่า Set Point จาก 70 ไปเป็น 30

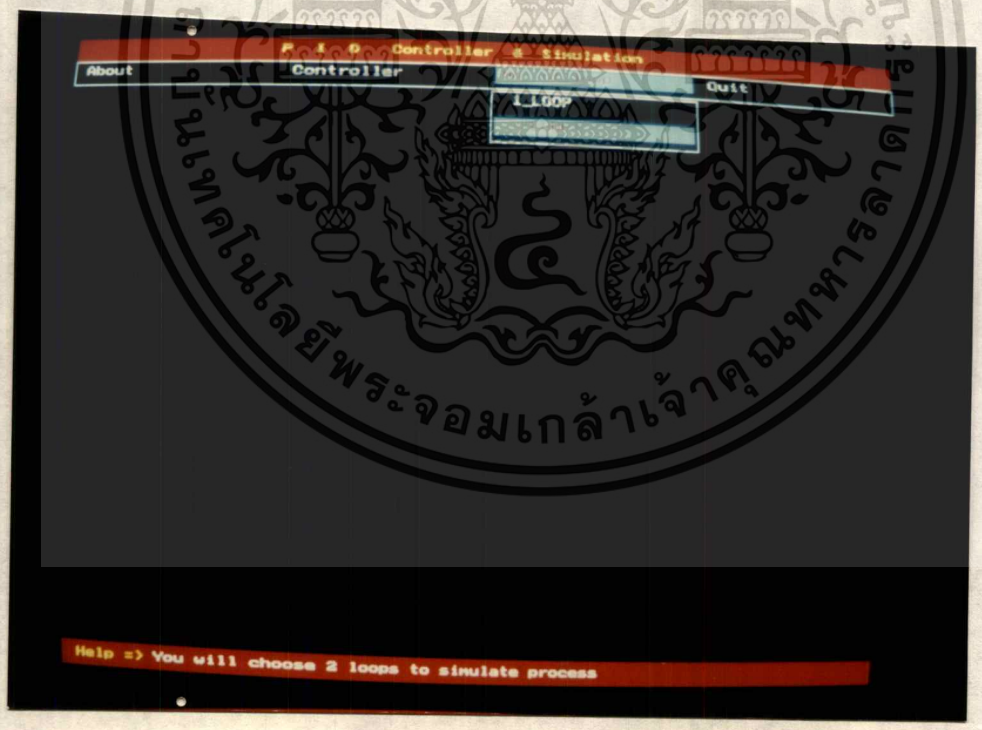


รูปที่ 11 จากรูปที่ 10 เมื่อระบบเข้าสู่สภาวะเสถียรแล้วทำการทดสองให้สิ่งรบกวนกับระบบ (disturbance) โดยลดกำลังและเพิ่มกำลังของปั๊มน้ำจะมีผลให้เกิดการตอบสนองของการควบคุมตามลำดับ

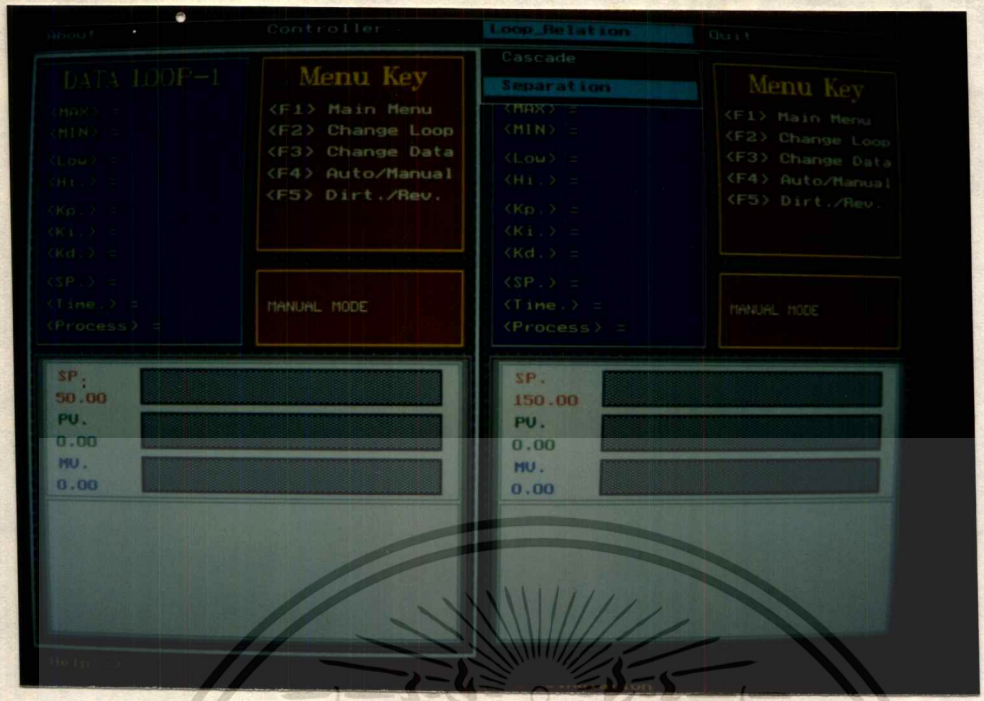


รูปที่ 12 จากรูปที่ 11 เมื่อระบบเข้าสู่สภาวะเสถียรแล้วทำการทดลองให้สิ่งรบกวนกับระบบ (disturbance) โดยเพิ่มกำลังและลดกำลังของปั๊มน้ำจะมีผลให้เกิดการตอบสนองของการควบคุมตามรูป

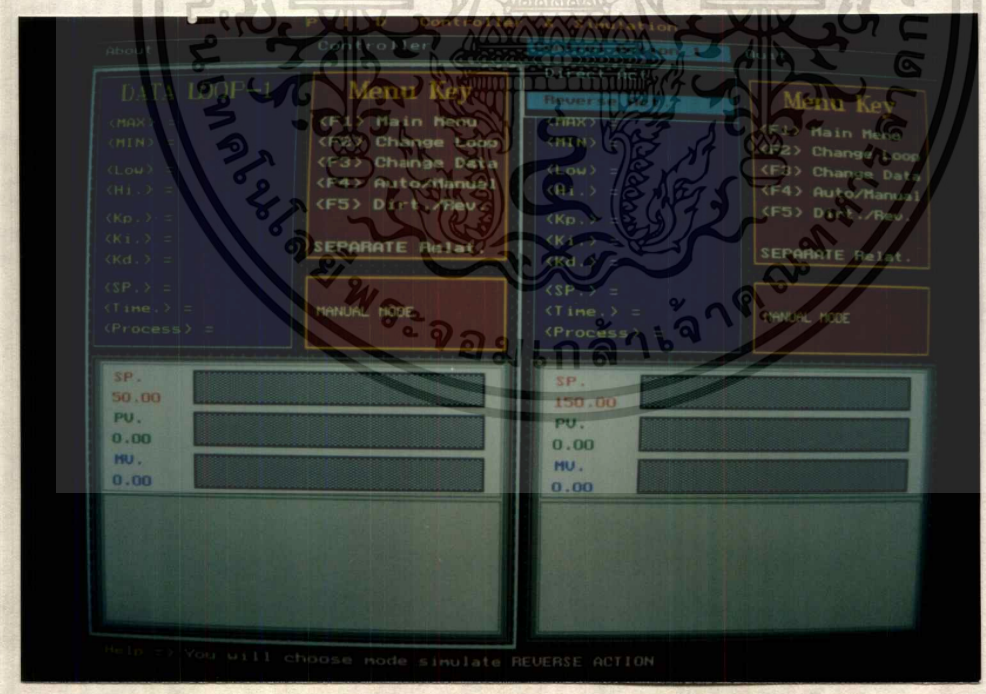
รูปที่ 13 จากรูปที่ 4 เมื่อผู้ใช้งานกด **key** → **(Right Arrow)** 1 ครั้ง จะเลือกการทำงานของโปรแกรมเป็นแบบ **Simulation**



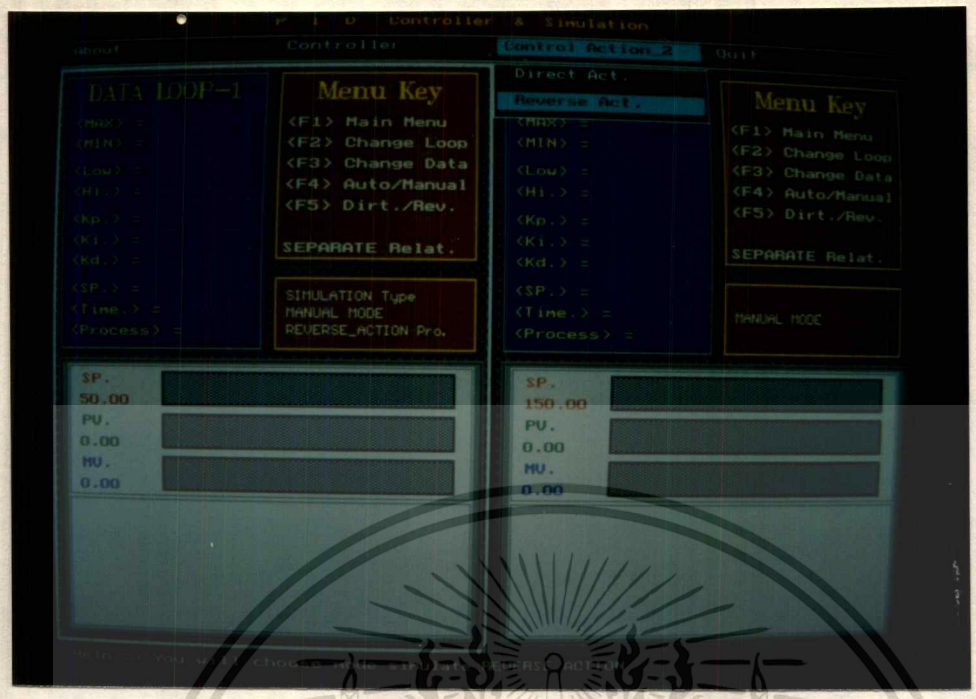
รูปที่ **14** จากรูปที่ 13 เมื่อผู้ใช้งานกด **key** ↵ **(Enter)** และ **key** ↓ **(Down Arrow)** จะได้หน้าจอดังรูปที่ 14 เป็นการเลือกการทำงานแบบ Simulation และควบคุมกระบวนการ 2 กระบวนการ



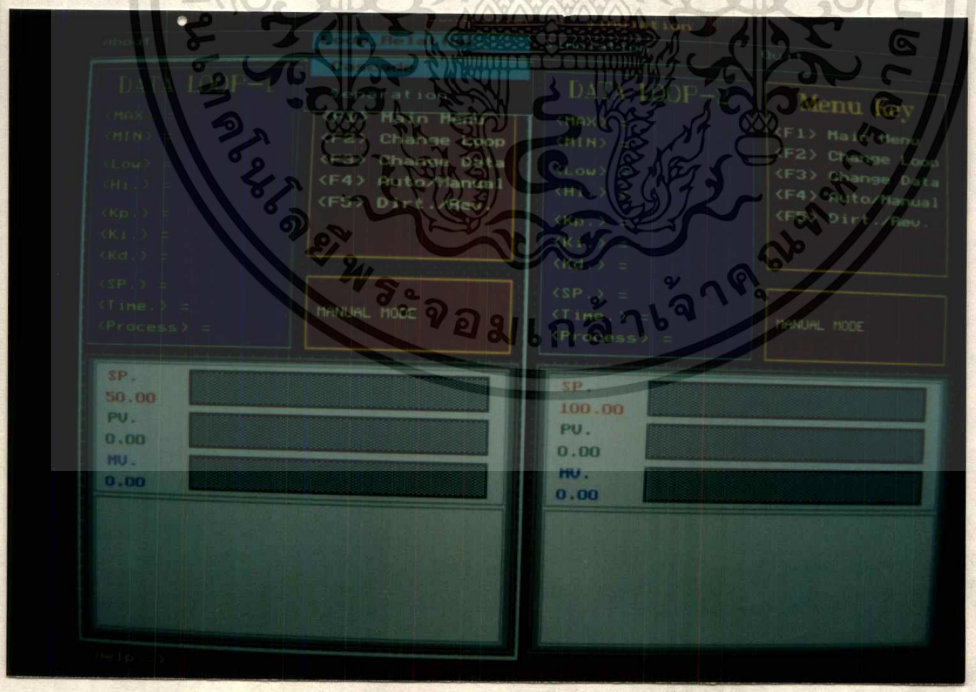
รูปที่ 15 จากรูปที่ 14 เมื่อผู้ใช้งานกด **key** ↵ **(Enter)** และ **key** ↓ **(Down Arrow)** จะขึ้นหน้าจอดังรูปที่ 15 เพื่อให้ผู้ใช้งานเลือกว่าในการควบคุมกระบวนการทั้ง 2 กระบวนการนั้นต้องการให้ควบคุมแบบต่อเนื่องกัน หรือควบคุมแบบแยกแต่ละกระบวนการ สามารถเลือกได้ด้วยการกด **key** ↑ **(Up Arrow)** หรือ **key** ↓ **(Down Arrow)**



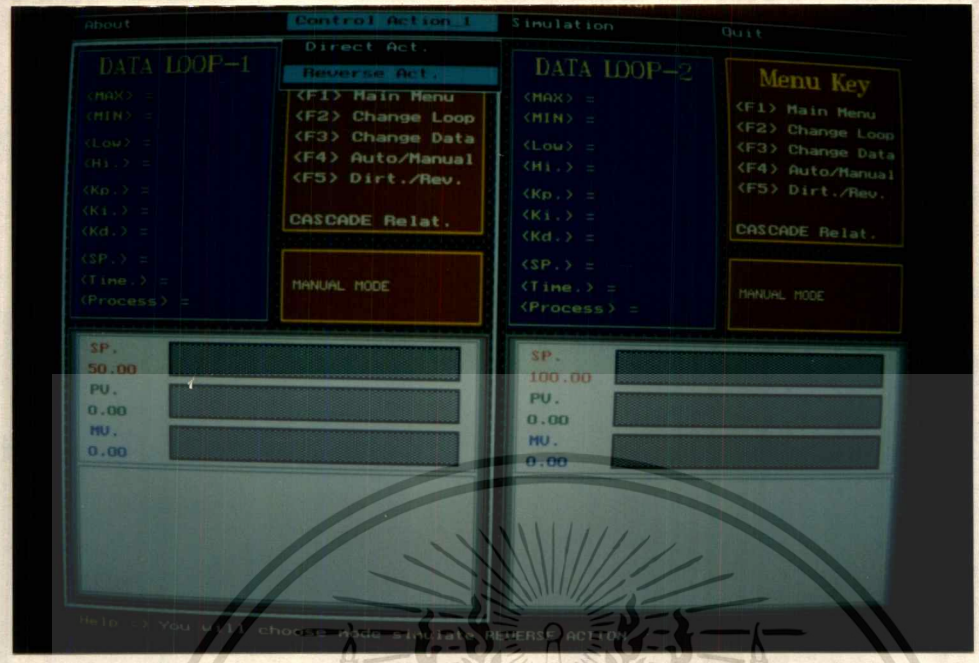
รูปที่ 16 จากรูปที่ 15 เมื่อผู้ใช้งานกด **key** ↵ **(Enter)** จะขึ้นหน้าจอดังรูปที่ 16 เพื่อให้ผู้ใช้งานเลือกว่าในการควบคุมกระบวนการที่ 1 ต้องการให้มีการควบคุมแบบ **Direct Action** หรือแบบ **Reverse Action** ด้วยการกด **key** ↑ **(Up Arrow)** หรือ **key** ↓ **(Down Arrow)**



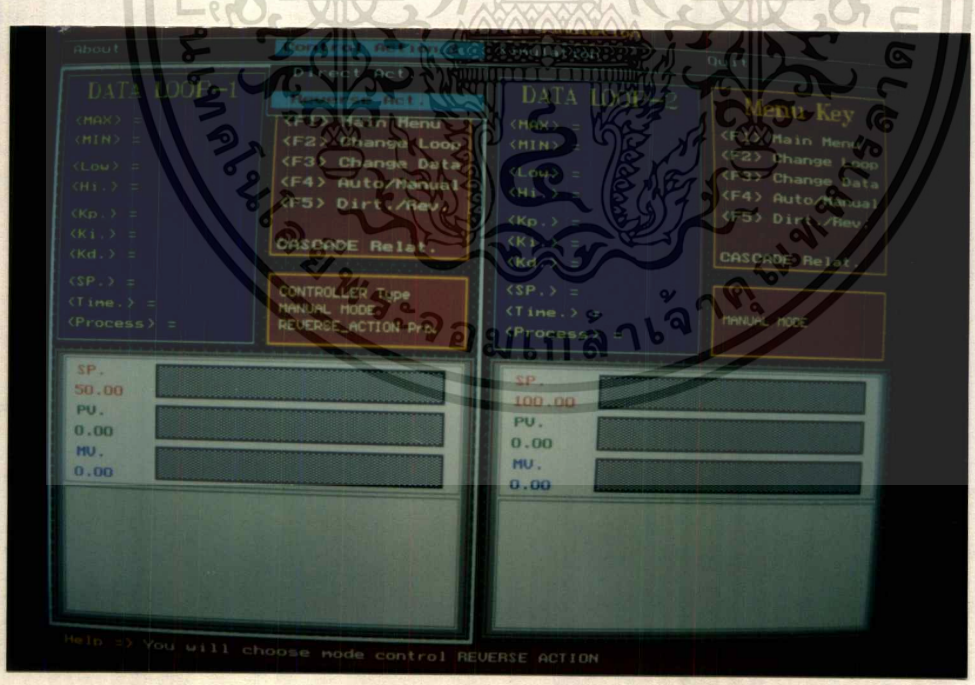
รูปที่ 17 จากรูปที่ 16 เมื่อผู้ใช้งานเลือกการทำงานของกระบวนการที่ 1 เสร็จแล้วจะขึ้นหน้าจอตามรูปที่ 17 เพื่อให้ผู้ใช้งานเลือกการทำงานของกระบวนการที่ 2 ด้วยการกด**key** ↑ **(Up Arrow)** หรือ **key** ↓ **(Down Arrow)**



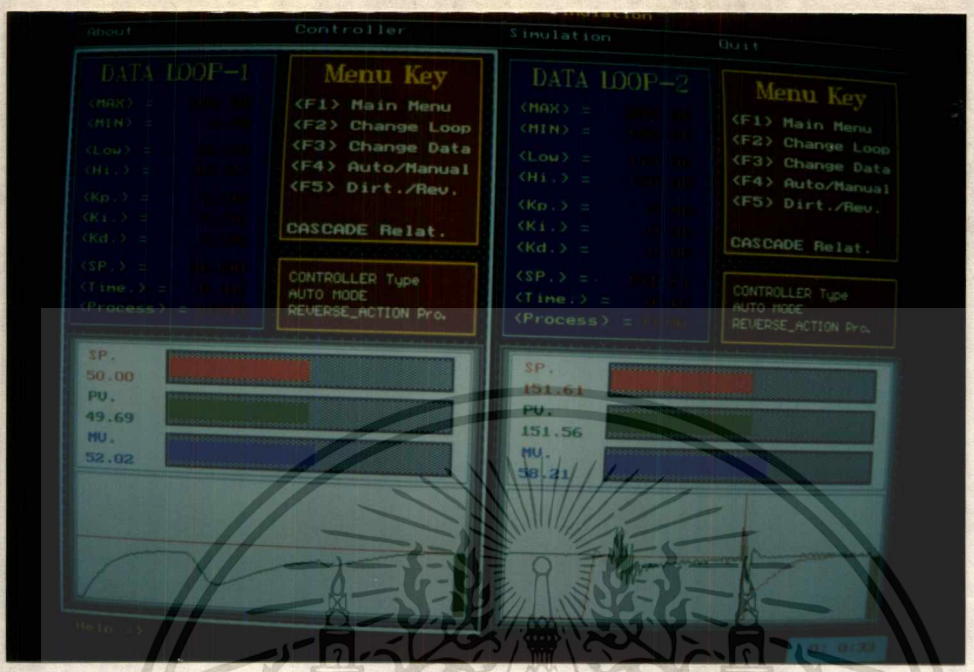
รูปที่ 18 จากรูปที่ 5 เมื่อผู้ใช้งานเลือกการควบคุมกระบวนการ 2 กระบวนการ จะได้หน้าจอดังรูปที่ 18 เพื่อให้ผู้ใช้งานเลือกความสัมพันธ์ในการควบคุมกระบวนการของทั้ง 2 กระบวนการ



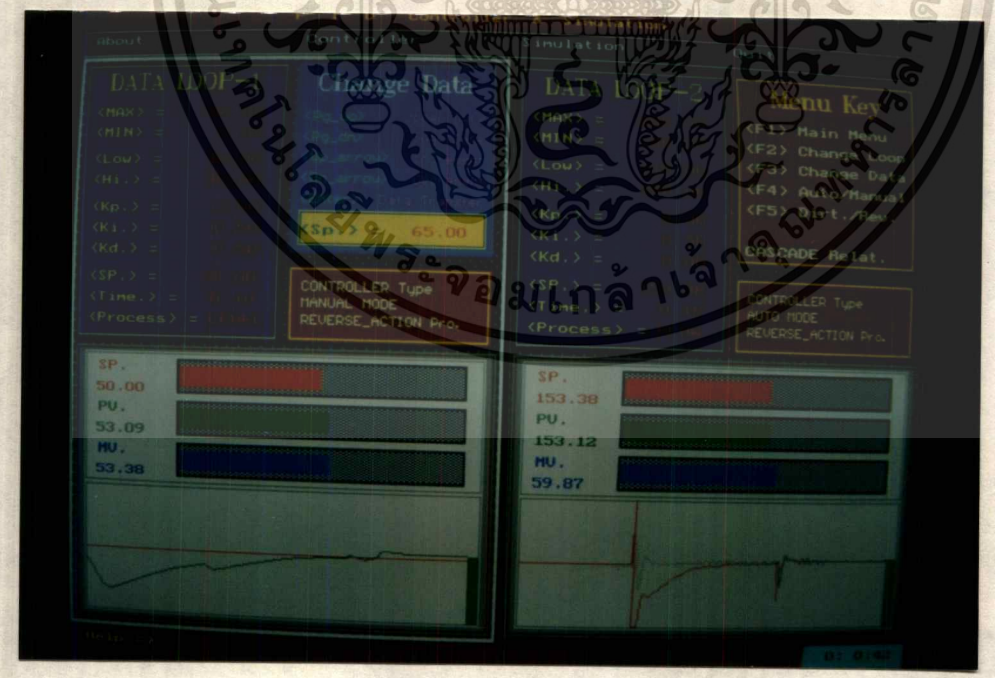
รูปที่ 19 จากรูปที่ 18 เมื่อผู้ใช้งานกด **key** ↵ **(Enter)** จะได้หน้าจอดังรูปที่ 19 เพื่อให้ผู้ใช้งานเลือกว่าในการควบคุมกระบวนการที่ 1 ต้องการให้มีการควบคุมแบบ **Direct Action** หรือ **Reverse Action** ด้วยการกด **key** ↑ **(Up Arrow)** หรือ **key** ↓ **(Down Arrow)**



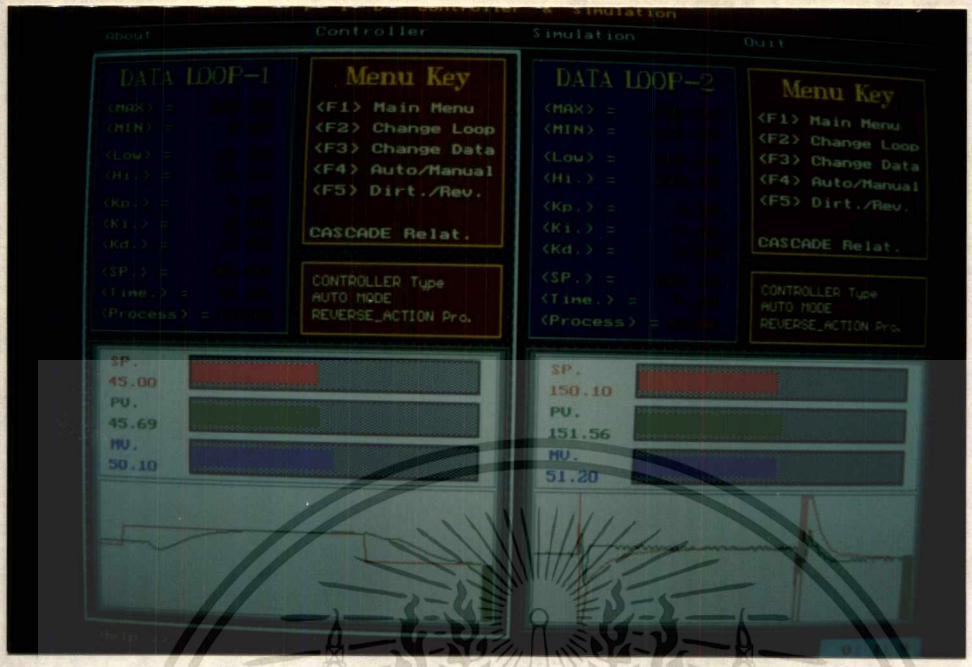
รูปที่ 20 จากรูปที่ 19 เมื่อผู้ใช้งานเลือกการทำงานของกระบวนการที่ 1 เสร็จแล้ว จะขึ้นหน้าจอตามรูปที่ 20 เพื่อให้ผู้ใช้งานเลือกการทำงานของกระบวนการที่ 2 ด้วยการกด**key** ↑**(Up Arrow)** หรือ **key** ↓ **(Down Arrow)**



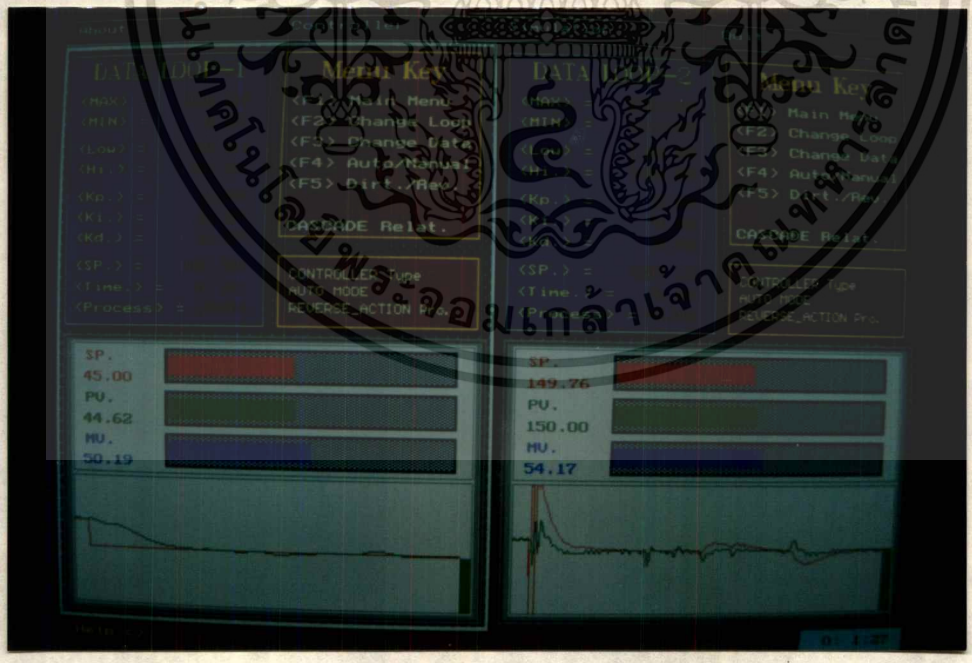
รูปที่ 21 จากรูปที่ 20 เมื่อมีการเลือกการทำงานของทั้งสองกระบวนการโดยเลือกให้เป็นการควบคุมแบบต่อเนื่องเรียบร้อยแล้วโดย Set Point ของกระบวนการที่ 1 เป็น **50** จะได้ลักษณะการควบคุมดังรูป



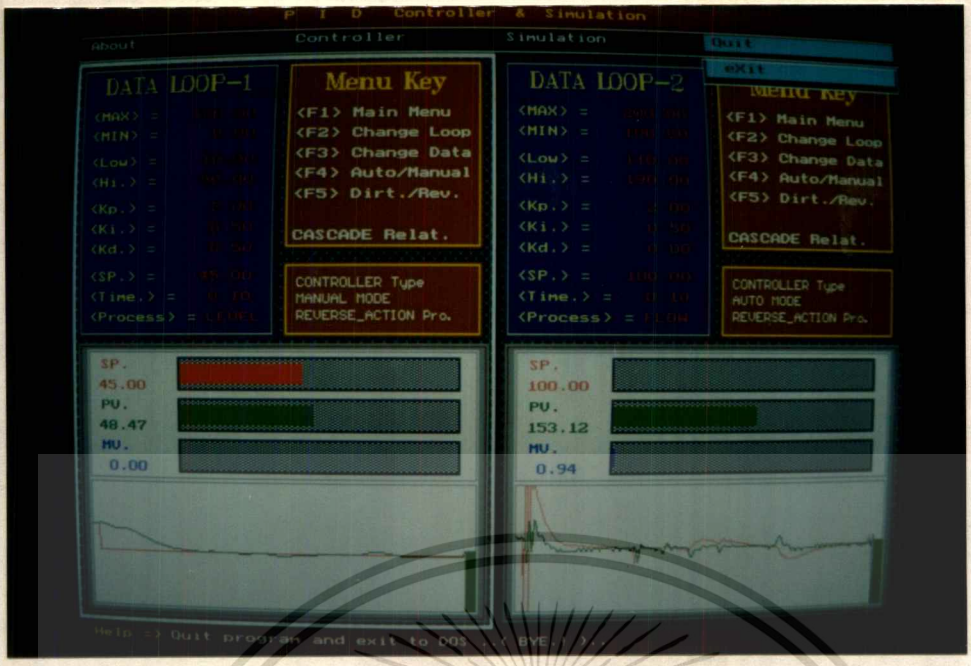
รูปที่ 22 จากรูปที่ 21 เมื่อระบบเข้าสู่สภาวะเสถียรแล้วมีการเปลี่ยนค่า Set Point ของกระบวนการที่ 1 เป็น **65**



รูปที่ 23 จากรูปที่ 22 เมื่อเปลี่ยนค่า Set Point ของกระบวนการที่ 1 เป็น 65 ด้วยการกด key ↵ (Enter) แล้วเมื่อระบบเข้าสู่สภาวะเสถียรแล้วมีการเปลี่ยนค่า Set Point ของกระบวนการที่ 1 เป็น **65** จะทำให้ได้ผลการตอบสนองกับการควบคุมดังรูปที่ 23

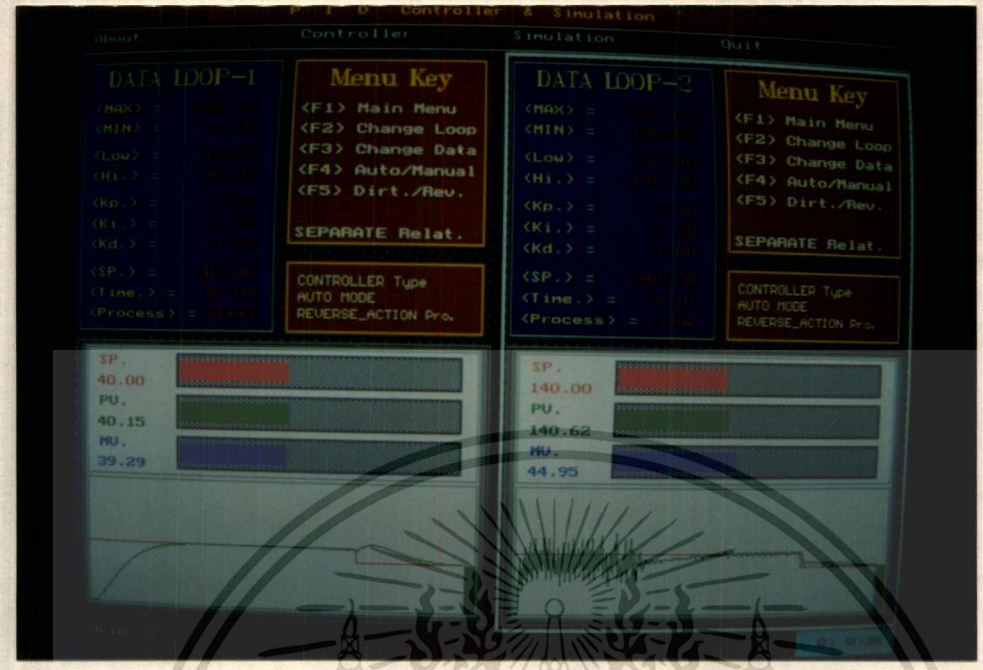


รูปที่ 24 จากรูปที่ 23 เมื่อระบบเข้าสู่สภาวะเสถียรแล้วทำการทดสองให้สิ่งรบกวนกับระบบ (disturbance) โดยลดกำลังและเพิ่มกำลังของปั๊มน้ำลงเล็กน้อยจะมีผลการตอบสนองของการควบคุมตามรูป

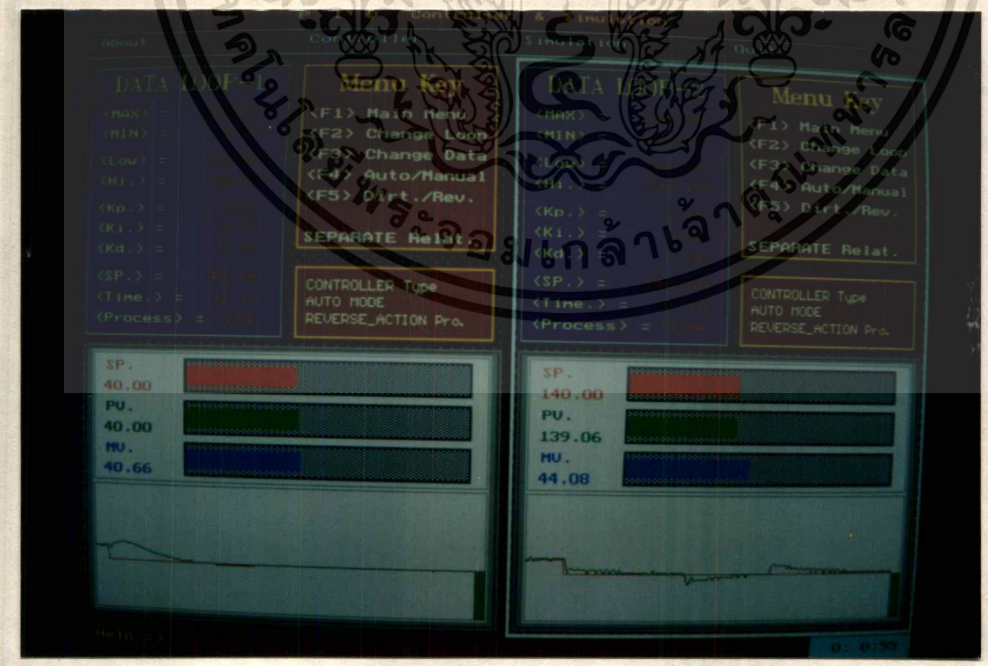


รูปที่ 25 เมื่อต้องการหยุดการทำงาน ให้ผู้ใช้งานกด**key F1** และกด **key** → **(Right Arrow)** หรือกด **key** ← **(Left Arrow)** จนกระทั่งได้หน้าจอดังรูปที่ 25 จากนั้นกด **key** ↵ **(Enter)** จะมีผลให้ออกจากโปรแกรม

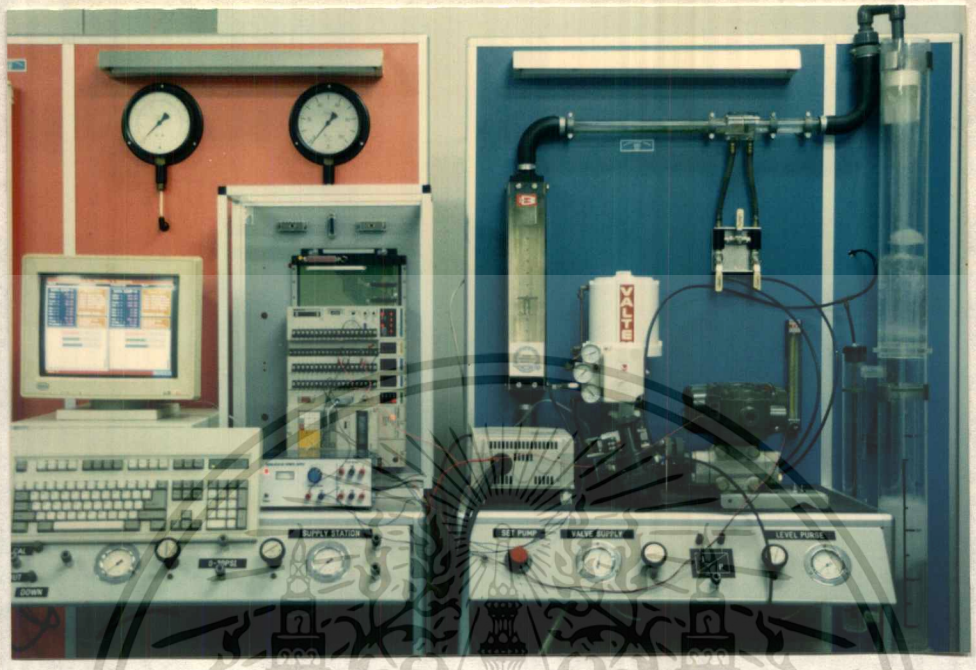
รูปที่ 26 จากรูปที่ 18 เมื่อมีการเลือกการทำงานของทั้ง 2 กระบวนการโดยเลือกให้เป็นการควบคุมแบบแยกแต่ละกระบวนการเรียบร้อยแล้ว โปรแกรมจะเริ่มทำการควบคุมกระบวนการตามที่ต้องการ จากรูปที่ 26 เป็นการแสดงผลของการควบคุมต่อระบบจากที่มีค่า **PV** เริ่มต้นของกระบวนการทั้ง 2 เป็น 0 และ 100 มีค่า Set Point ของกระบวนการที่ 1 เป็น 50 กระบวนการที่ 2 เป็น 150 โดยกระบวนการที่ 1 เป็นการควบคุมระดับน้ำ (Level process) กระบวนการที่ 2 เป็นการควบคุมอัตราการไหล (Flow process)



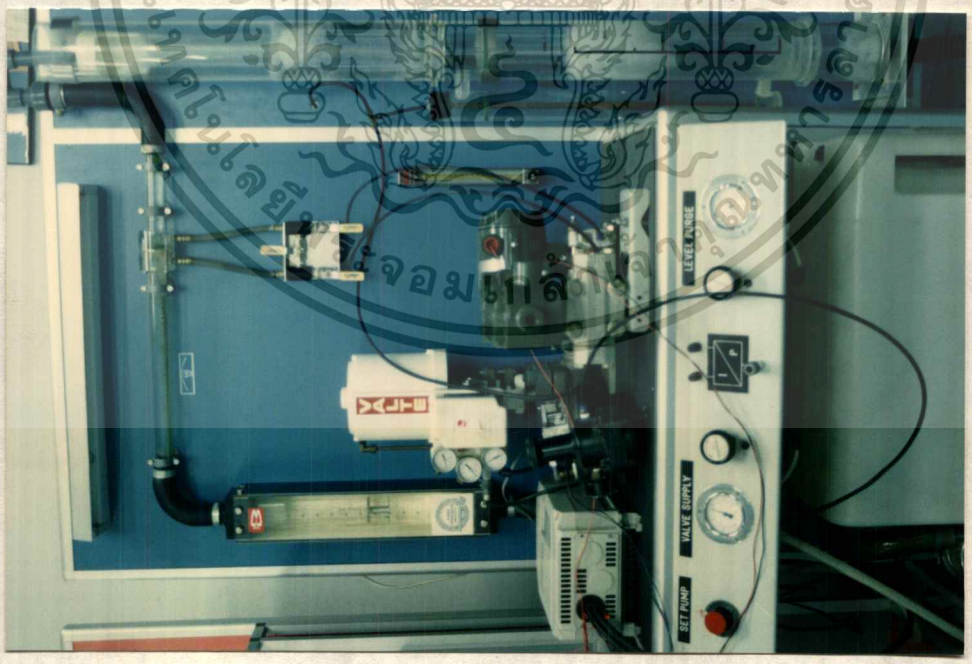
รูปที่ 27 จากรูปที่ 26 ในช่วงต้นของการควบคุมกระบวนการที่ 2 จะเห็นว่าไม่สามารถทำการควบคุมให้ระบบเสถียรได้ จึงได้มีการปรับค่า **Kd** ให้เป็น 0 จากนั้นเมื่อระบบเข้าสู่สภาวะเสถียรทั้ง 2 กระบวนการแล้ว มีการปรับค่า Set Point ของทั้ง 2 กระบวนการโดยให้ของกระบวนการที่ 1 เป็น 40 และกระบวนการที่ 2 เป็น 140 ตามลำดับ เพื่อดูผลตอบสนองของการควบคุมกระบวนการ



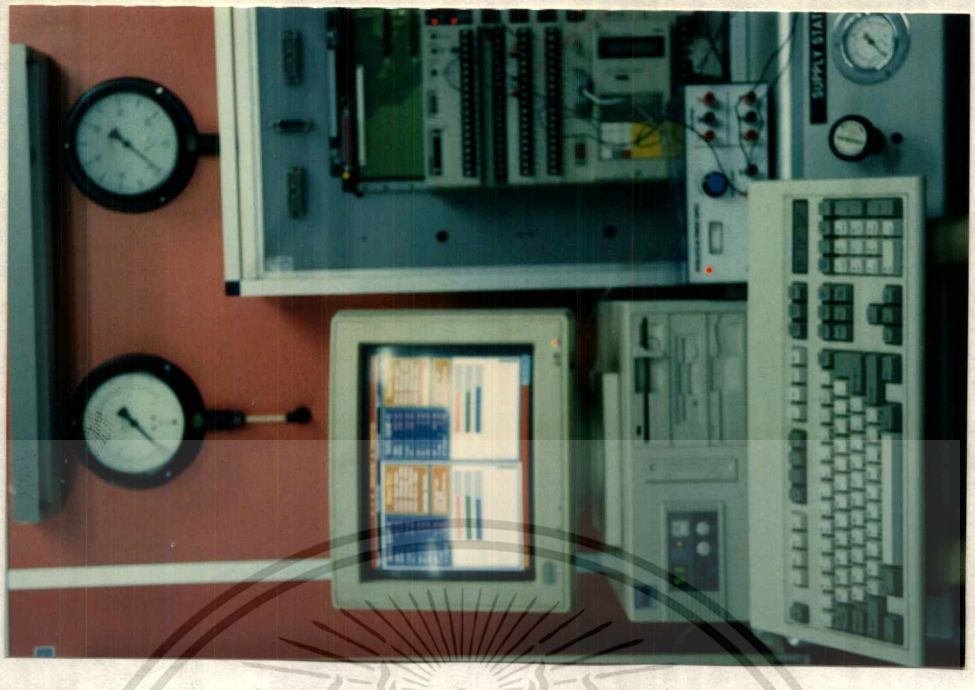
รูปที่ 28 เมื่อระบบทั้ง 2 เข้าสู่สภาวะเสถียรแล้ว ทำการทดลองให้สิ่งรบกวนกับระบบ (disturbance) โดยลดกำลังและเพิ่มกำลังของปั๊มน้ำเล็กน้อยจะมีผลการตอบสนองของการควบคุม ตามรูป



รูปที่ 29 เป็นชุดอุปกรณ์ที่ใช้ทำโครงงาน และชุดทดลองการควบคุมระดับน้ำ ขณะทำการทดลอง

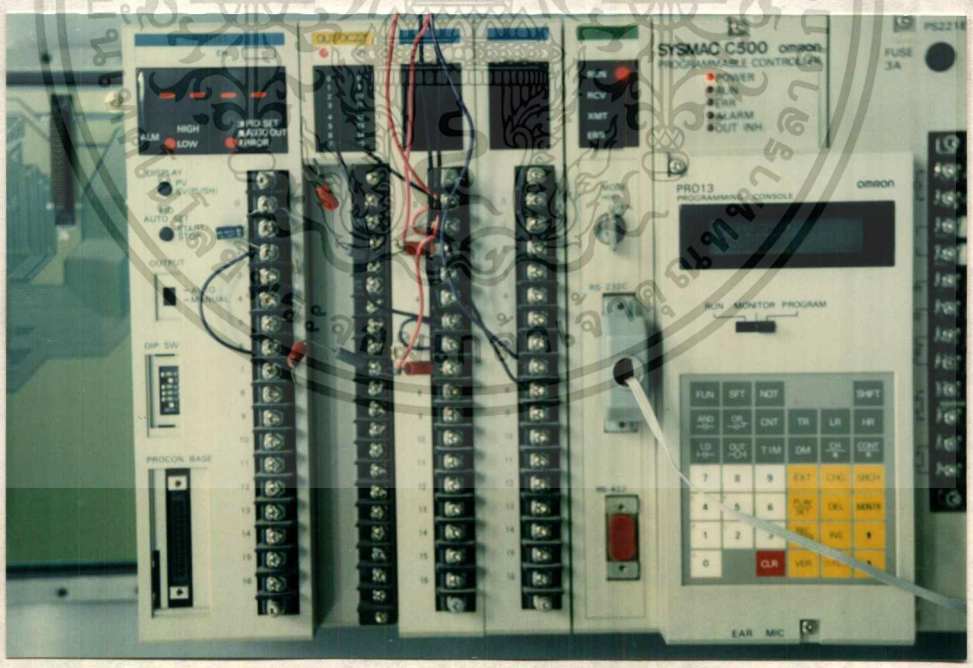


รูปที่ 30 ชุดทดลองการควบคุมระดับน้ำชนิด Second Order ที่ใช้การตรวจวัดโดย Flow Meter และ Level Transmitter (LT) ที่อยู่ภายในห้อง Process ของภาควิชาวิศวกรรมการวัดคุม



รูปที่ 31 ชุดอุปกรณ์ที่ใช้ทำโครงงาน

- CPU 386 SX-20
- จอภาพ SVGA
- Keyboard
- ชุดอุปกรณ์ I/O Interface



รูปที่ 32 ชุดอุปกรณ์ I/O Interface

- อุปกรณ์แปลงสัญญาณอนาลอกเป็นดิจิตอล AD001 (Slot ที่ 3 จากซ้าย)
- อุปกรณ์แปลงสัญญาณดิจิตอลเป็นอนาลอก DA001 (Slot ที่ 4 จากซ้าย)
- พอร์ตอนุกรม LK008 (slot ที่ 5 จากซ้าย)
- CPU ของ PLC รุ่น SYSMAC C500

Data setting range for Controller

Scaling Data.

Maximum data <MAX> = 9,999.9 %
Minimum data <MIN> = -999.9 %

Upper and Lower limit values.

Lower limit <Low> = Minimum data <MIN> %
<= Set-point <Sp> - 0.1 %
Upper limit <Hi> = Maximum data <MAX> %
>= Set-point <Sp> + 0.1 %

PID Constants.

Proportional Gain <Kp>
Minimum = 0.1
Maximum = 999.9
Integal Gain <Ki>
Minimum = 0 $second^{-1}$
Maximum = 999.9 $second^{-1}$
Derivative Gain <Kd>
Minimum = 0 second
Maximum = 999.9 second

Set-point <Sp>

Minimum = Low value <Hi> - 0.1 %
Maximum = High value <Low> + 0.1 %

Sampling Time <Time>

Minimum = 0.1 second
Maximum = 60 second

Process Types

Process Types <Process> = Level process
Flow process

Data setting range for simulation.

Process Orders

Process Orders <Order> = First Order
Second Order

Process Constants

Process Gain <Ks>
Minimum = 99.9
Maximum = -99.9
Time Constant 1 <τ1>
Minimum = 0.1 second
Maximum = 99.9 second
Time Constant 2 <τ2>
Minimum = 0.1 second
Maximum = 99.9 second

ในการเพิ่ม หรือ ลดค่าพารามิเตอร์ต่าง ๆ เราสามารถเพิ่มหรือลดค่าได้ครั้งละ 0.1 แต่ในส่วนของการเพิ่มหรือลดค่า **Manipulated Variable < Mv >** สามารถเพิ่มหรือลดค่าได้ครั้งละ 0.5

ตัวอย่างผลการทดลอง

การทดลองเพื่อที่จะหาค่าพารามิเตอร์ที่เหมาะสมต่าง ๆ สำหรับโปรแกรม โดยใช้โปรแกรมนี้ร่วมกับ เครื่องคอมพิวเตอร์ 386 Dx-40 ต่อผ่านอุปกรณ์ A/D , D/A สำเร็จรูป และบันทึกผลของค่า Pv (Process Variable) และ Mv (Manipulated Variable) โดย อุปกรณ์บันทึกผล (Recorder) ที่มีอัตราการป้อนกระดาษเป็น 1440 มม. ต่อ ชั่วโมง แสดงไว้ด้วยกระดาษกราฟ

หมายเหตุ ในการทดลองบันทึกค่าด้วย Recorder นี้ อุปกรณ์มีค่า Error เท่าที่วัดได้ประมาณ **1** % ตลอดการทดลอง เช่น ที่ค่า Pv. = 50 ค่าที่อ่านได้ประมาณ 5**1** เป็นต้น

การทดลองที่ 1 Process ที่ใช้ทดลองเป็น การ Control Level 2 ถัง แบบ 1 Loop ชนิด Reverse Action เราสามารถหาค่าพารามิเตอร์ที่เหมาะสมได้ดังนี้

<MAX> = 100.00 % <Kp> = 3.50 <Sp> = 50.00 %
<MIN> = 0.00 % <Ki> = 0.1 sec^{-1} <Time> = 1.00 sec
<Low> = 10.00 % <Kd> = 7.00 sec
<Hi> = 90.00 %

ซึ่งค่าคงที่ต่าง ๆ เหล่านี้เป็นค่าตัวอย่างที่เหมาะสมกับชนิดของกระบวนการ และ ฮาร์ดแวร์ ซึ่งจากกราฟความสัมพันธ์จะเป็นดัง **กราฟรูปที่ 1** ในช่วงที่ (1)

ช่วงที่ (2) เป็นการทดลองเปลี่ยน Set-Point จาก 50 % เป็น 60 % จะเห็นว่า ค่า Mv. มีการปรับตัวเพิ่มขึ้นเล็กน้อย แล้วก็พยายามควบคุมค่า Pv. ให้ปรับตัวตามจนเข้าสู่ Set-Point ใหม่

ช่วงที่ (3) เปลี่ยน Set-Point กลับจาก 60 % เป็น 40 %

กราฟรูปที่ 2 ช่วงที่ (4) เราหาความสัมพันธ์ของค่า Pv. และ Mv. โดยมีการรบกวนของการเปลี่ยนแปลงค่า กำลังของ Pump ที่สูบน้ำ โดยจากค่าที่ F50 เป็น F60 ค่า Mv. จะมีการเปลี่ยนแปลงดังกราฟ

ช่วงที่ (5) เปลี่ยนค่ากำลังของ Pump จาก F60 เป็น F40

กราฟรูปที่ 3 แสดงผลของการเปลี่ยนค่า Gain ต่าง ๆ

ช่วงที่ (6) เป็นการเปลี่ยนค่า Kp โดย จากเดิม 3.50 เพิ่มขึ้นเป็น 6.50

ช่วงที่ (7) เป็นการเปลี่ยนค่า Kp จากที่ 6.50 กลับเป็น 0.50

เราพบว่าการเปลี่ยนค่า Kp ในกรณีนี้ไม่ค่อยมีผลมากนัก เพราะค่าของ Pv. เริ่มเข้าสู่ Set-Point แล้ว จึงไม่แสดงการเปลี่ยนแปลงเท่าไรนัก

ช่วงที่ (8) เป็นการเปลี่ยนค่า Ki โดย จากเดิม 0.10 sec^{-1} เพิ่มขึ้นเป็น 5.00 sec^{-1} ในกรณีนี้ ค่า Mv. มีการเปลี่ยนแปลงเป็นแบบ On-Off และค่า Pv. มีแนวโน้มที่จะเข้าสู่สภาวะไม่เสถียร (Underdamped)

ช่วงที่ (9) เป็นการเปลี่ยนค่า Ki จากที่ 5.00 sec^{-1} กลับเป็น 0.0 sec^{-1} จากผลของค่า Mv. ทำให้ ค่า Pv. ที่เกิดขึ้นไม่เข้าสู่สภาวะ Set-Point ที่ต้องการ และเกิดสภาวะค้าง ที่ค่าประมาณ 52%

กราฟรูปที่ 4 เป็นการแสดงผลของการเปลี่ยนแปลงค่า Kd

ช่วงที่ (10) เป็นการเปลี่ยนค่า Kd โดย จากเดิม 7.00 sec เพิ่มขึ้นเป็น 30.00 sec จากผลของการเพิ่มค่า ช่วงของการเปลี่ยนค่า Mv. มีช่วงการเปลี่ยนแปลงมาก และค่าของ Pv. มีการแกว่งมาก

ช่วงที่ (11) เป็นการเปลี่ยนค่า Kd จากที่ 30.00 sec กลับเป็น 0.00 sec ค่า Mv. มีระยะการแกว่งที่ลดลง รวมทั้งค่า Pv. ด้วย แต่ก็มีแนวโน้มที่จะไม่เสถียร

ช่วงที่ (12) เป็นการเปลี่ยนค่าทุกค่าให้กลับเป็นค่าที่เหมาะสมค่าเดิมก่อนการเปลี่ยนแปลง จะเห็นว่าระบบเริ่มเข้าสู่สภาวะเสถียรอีกครั้ง

การทดลองที่ 2 Process ที่ใช้ทดลองเป็น การ Control Flow แบบ 1 Loop ชนิด Reverse Action เราสามารถหาค่าพารามิเตอร์ที่เหมาะสมได้ดังนี้

<MAX> = 100.00 % <Kp> = 1.50 <Sp> = 50.00 %

<MIN> = 0.00 % <Ki> = 0.3 sec^{-1} <Time> = 1.00 sec

<Low> = 10.00 % <Kd> = 0.60 sec

<Hi> = 90.00 %

หมายเหตุ ในการทดลองนี้ ค่าของ Set-Point และ Pv. ของ Process แบบ Flow การคำนวณจำเป็นต้องมีการยกกำลังสองแล้วหารด้วยค่า 100 ตามสูตร

(ค่าที่ต้องการ)^2/100 เช่น ต้องการค่า Set-Point ที่ 50 ค่าที่ปรากฏในกราฟจะเป็น (50*50)/100 =25 เป็นต้น (แสดงด้วยค่าในวงเล็บ)

ซึ่งค่าคงที่ต่าง ๆ เหล่านี้เป็นค่าตัวอย่างที่เหมาะสมกับชนิดของกระบวนการ และ ฮาร์ดแวร์ ซึ่งจากกราฟความสัมพันธ์จะเป็นดัง **กราฟรูปที่ 5** ในช่วงที่ (1)

ช่วงที่ (2) เป็นการทดลองเปลี่ยน Set-Point จาก 50 (25)% เป็น 60 (36)% จะเห็นว่า ค่า Mv. มีการปรับตัวเพิ่มขึ้นเล็กน้อย แล้วก็พยายามควบคุมค่า Pv. ให้ปรับตัวตามจนเข้าสู่ Set-Point ใหม่ โดยมีการรบกวนของสภาวะภายนอกตลอดเวลา

กราฟรูปที่6 ช่วงที่ (3) เปลี่ยน Set-Point กลับจาก 60 (36) % เป็น 40 (16) %

ช่วงที่ (4) เราหาความสัมพันธ์ของค่า Pv. และ Mv. โดยมีการรบกวนของการเปลี่ยนแปลงค่ากำลังของ Pump ที่สูบน้ำ โดยจากค่าที่ F50 เป็น F60 ค่า Mv. จะมีการเปลี่ยนแปลงดังกราฟ

ช่วงที่ (5) เปลี่ยนค่ากำลังของ Pump จาก F60 เป็น F40 ค่า Mv. จะมีการเพิ่มมากขึ้น

กราฟรูปที่ 7 แสดงผลของการเปลี่ยนค่า Gain ต่าง ๆ

ช่วงที่ (6) เป็นการเปลี่ยนค่า Kp โดย จากเดิม 1.50 เพิ่มขึ้นเป็น 15.00 ค่า Mv. จะมีลักษณะ On-Off และ ช่วงของค่า Pv. มีการแกว่งในช่วงแคบ ๆ

ช่วงที่ (7) เป็นการเปลี่ยนค่า Kp จากที่ 15.00 กลับเป็น 0.10

เราพบว่าการเปลี่ยนค่า Kp ในกรณีนี้ไม่ค่อยมีผลมากนัก เมื่อเทียบกับช่วงที่ (6) ค่าของ Pv. เริ่มเข้าสู่ Set-Point แล้ว และมีการแกว่งน้อย

กราฟรูปที่ 8 ช่วงที่ (8) เป็นการเปลี่ยนค่า Ki โดย จากเดิม 0.30 sec^{-1} เพิ่มขึ้นเป็น 10.0 sec^{-1}ในกรณีนี้ ค่า Mv. มีการเปลี่ยนแปลงเป็นแบบ On-Off และค่า Pv. มีการแกว่งในช่วงที่มากขึ้น

ช่วงที่ (9) เป็นการเปลี่ยนค่า Ki จากที่ 10.00 sec^{-1} กลับเป็น 0.0 sec^{-1} จากผลของค่า Mv. ทำให้ ค่า Pv. ที่เกิดขึ้นไม่เข้าสู่สภาวะ Set-Point ที่ต้องการ และเกิดสภาวะค้าง ที่ค่าประมาณ 28.5(8.50)%

ช่วงที่ (10) เป็นการเปลี่ยนค่า Kd โดย จากเดิม 0.60 sec เพิ่มขึ้นเป็น 10.00 sec จากผลของการเพิ่มค่า ช่วงของการเปลี่ยนค่า Mv. มีช่วงการเปลี่ยนแปลงมาก และค่าของ Pv. มีการแกว่งมาก

ช่วงที่ (11) เป็นการเปลี่ยนค่า Kd จากที่ 10.00 sec กลับเป็น 0.00 sec ค่า Mv. มีระยะการแกว่งที่ลดลงในช่วงแคบ ๆ รวมทั้งค่า Pv. ด้วย

กราฟรูปที่ 9 ช่วงที่ (12) เป็นการเปลี่ยนค่าทุกค่าให้กลับเป็นค่าที่เหมาะสมค่าเดิมก่อนการเปลี่ยนแปลง จะเห็นว่าระบบเริ่มเข้าสู่สภาวะเสถียรอีกครั้ง แต่ก็มีการแกว่งในช่วงแคบ ๆ จากผลการรบกวนของระบบ

การทดลองที่ 3 Process ที่ใช้ทดลองเป็น การ Control แบบ 2 Loop Casscade ชนิด Reverse Action เราสามารถหาค่าพารามิเตอร์ที่เหมาะสมได้ดังนี้

Loop ที่ 1 Process Level : Primary Controller [Master]

<MAX> = 100.00 % <Kp> = 4.20 <Sp> = 50.00 %

<MIN> = 0% <Ki> = 0.10 sec^{-1} <Time> = 1.00 sec

<Low> = 10.00 % <Kd> = 7.50 sec

<Hi> = 90.00 %

Loop ที่ 2 Process Flow : Secondary Controller [Slave]

<MAX> = 100.00 % <Kp> = 1.90 <Sp> = เปลี่ยนตามค่า Mv. Loop 1

<MIN> = 0.00 % <Ki> = 0.30 sec^{-1} <Time> = 1.00 sec

<Low> = 10.00 % <Kd> = 0.60 sec

<Hi> = 90.00 %

ซึ่งค่าคงที่ต่าง ๆ เหล่านี้เป็นค่าตัวอย่างที่เหมาะสมกับชนิดของกระบวนการ และ ฮาร์ดแวร์ ซึ่งจากกราฟความสัมพันธ์จะแสดงเฉพาะค่าของ Pv. Loop 1 และ Mv. Loop 2 เป็นดัง **กราฟรูปที่ 10** ในช่วงที่ (1) จะเป็นการเปลี่ยนตามค่าที่ตั้งไว้

ช่วงที่ (2) เป็นการทดลองเปลี่ยน Set-Point จาก 50.00% เป็น 60.00% จะเห็นว่า ค่า Mv. มีการปรับตัวเพิ่มขึ้น แล้วก็พยายามควบคุมค่า Pv. ให้ปรับตัวตามจนเข้าสู่ Set-Point ใหม่

ช่วงที่ (3) เปลี่ยน Set-Point กลับจาก 60.00% เป็น 40.00%

ช่วงที่ (4) เราหาความสัมพันธ์ของค่า Pv. และ Mv. โดยมีการรบกวนของการเปลี่ยนแปลงค่ากำลังของ Pump ที่สูบน้ำ โดยจากค่าที่ F50 เป็น F60 ค่า Mv. จะมีการเปลี่ยนแปลงดังกราฟ

ช่วงที่ (5) เปลี่ยนค่ากำลังของ Pump จาก F60 เป็น F40

กราฟรูปที่ 11 แสดงผลของการเปลี่ยนค่า Gain ต่าง ๆ

ช่วงที่ (6) เป็นการเปลี่ยนค่า Kp โดย จากเดิม 4.20 เพิ่มขึ้นเป็น 15.00 การเปลี่ยนแปลงในช่วงนี้ค่า Mv. จะมีช่วงการเปลี่ยนที่ค่อนข้างกว้างประมาณ 20-100

ช่วงที่ (7) เป็นการเปลี่ยนค่า Kp จากที่ 15.00 กลับเป็น 0.50 ช่วงของ Mv. มีการเปลี่ยนแปลงที่แคบลงส่วนค่า Pv. ก็มีการแกว่งในช่วงแรกแล้วค่อย ๆ ปรับเข้าสู่ภาวะเสถียร

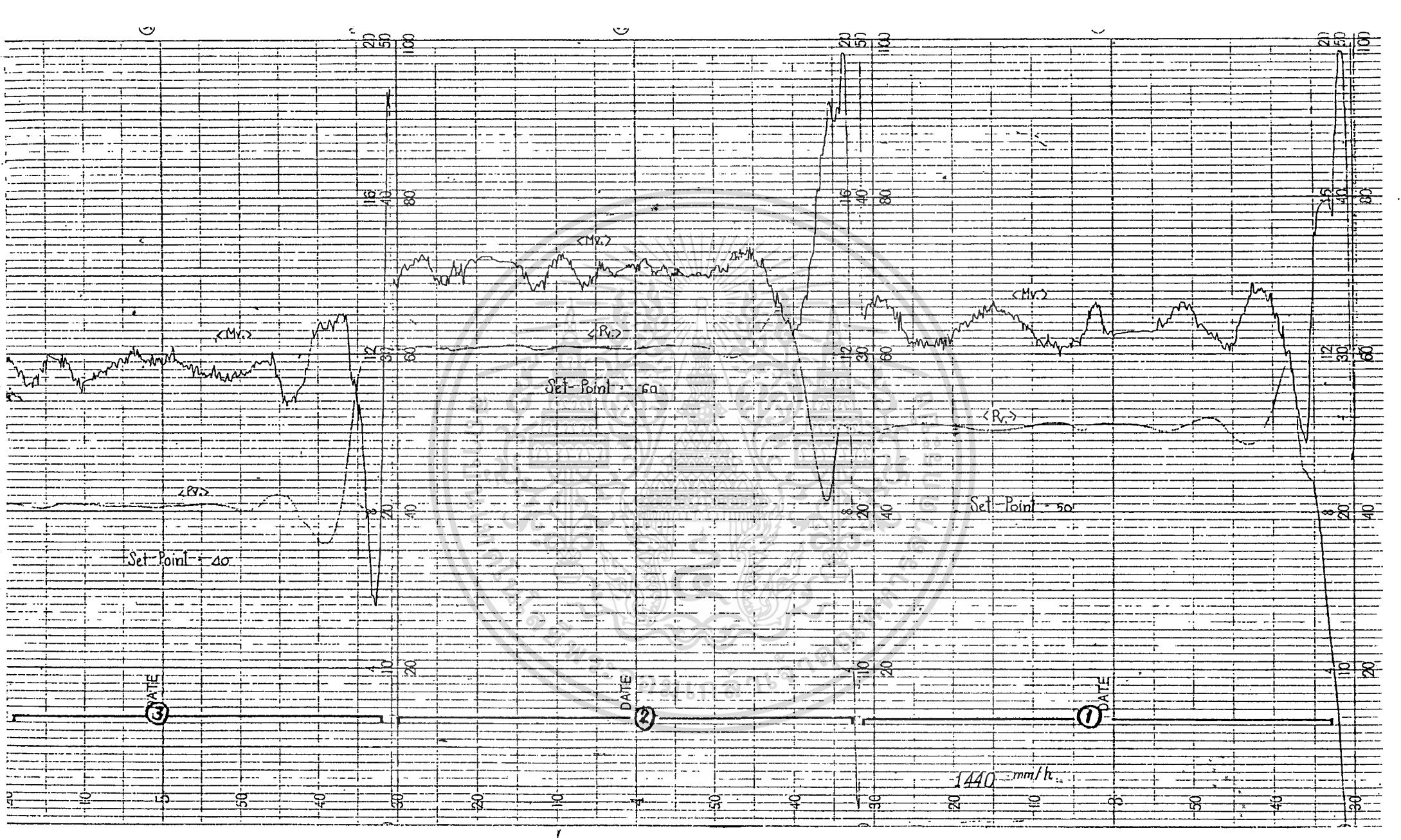
กราฟรูปที่ 12 ช่วงที่ (8) เป็นการเปลี่ยนค่า Ki โดย จากเดิม 0.10 sec^{-1} เพิ่มขึ้นเป็น 10.00 sec^{-1} ในกรณีนี้ ค่า Mv. มีการเปลี่ยนแปลงเป็นแบบ On-Off จาก 0%-100% และค่า Pv. มีแนวโน้มที่จะเข้าสู่สภาวะไม่เสถียร (Underdamped)

ช่วงที่ (9) เป็นการเปลี่ยนค่า Ki จากที่ 10.00 sec^{-1} กลับเป็น 0 sec^{-1} จากผลของค่า Mv. ทำให้ ค่า Pv. ที่เกิดขึ้นเกิดสภาวะค้าง ที่ค่าประมาณ 4

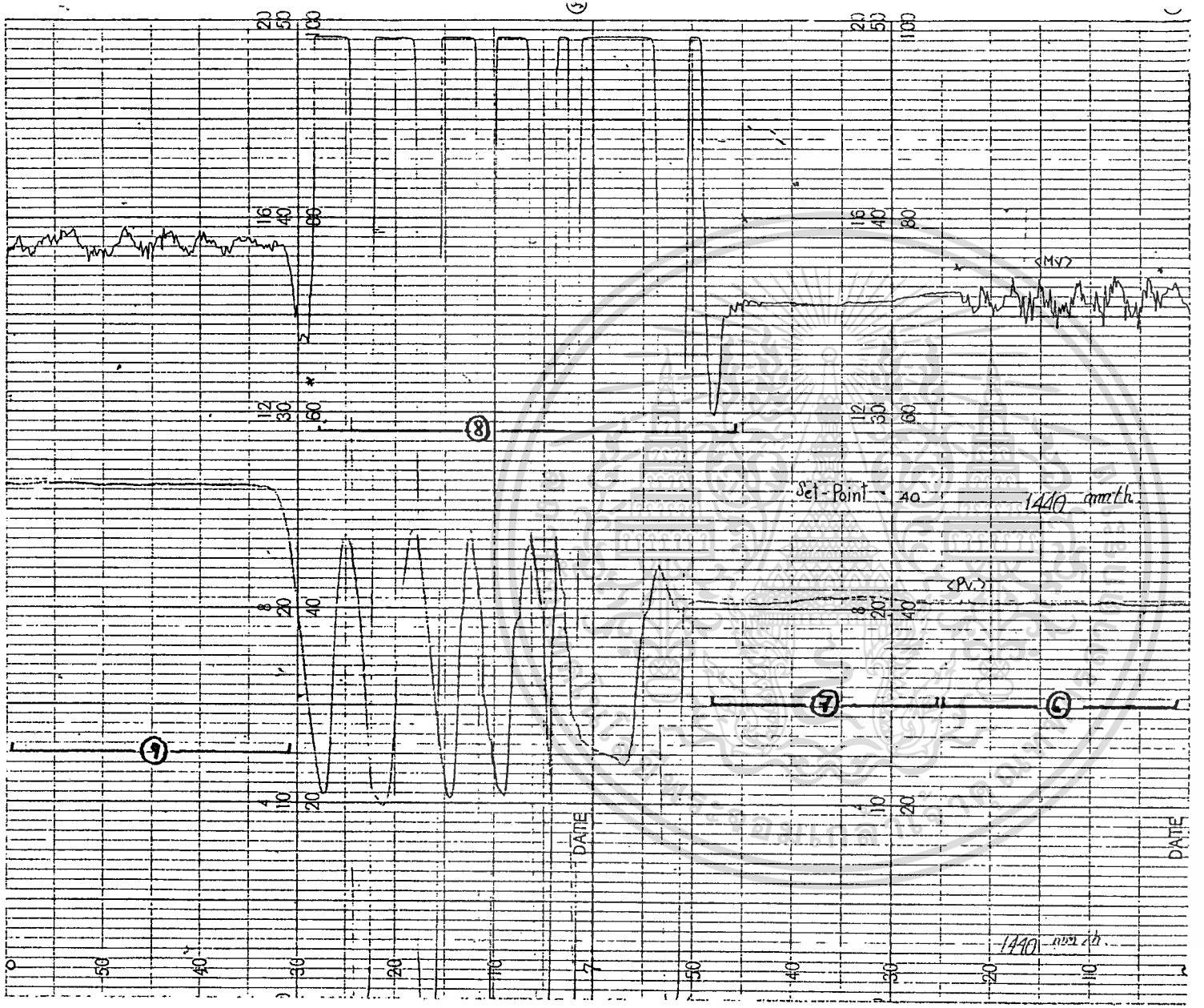
ช่วงที่ (10) เป็นการเปลี่ยนค่า Kd โดย จากเดิม 7.50 sec เพิ่มขึ้นเป็น 20.00 sec จากผลของการเพิ่มค่า ช่วงของการเปลี่ยนค่า Mv. มีช่วงการเปลี่ยนแปลงพอประมาณ

ช่วงที่ (11) เป็นการเปลี่ยนค่า Kd จากที่ 20.00 sec กลับเป็น 0.0 sec ค่า Mv. มีระยะการแกว่งที่มาก รวมทั้งค่า Pv. ด้วย และมีแนวโน้มที่จะไม่เสถียร

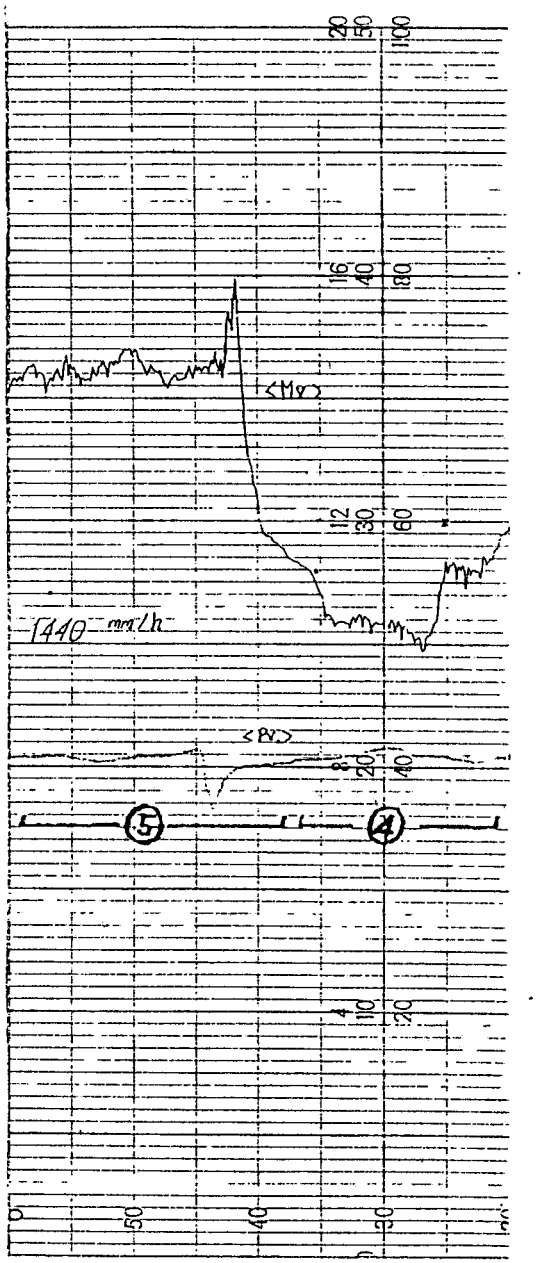
กราฟรูปที่ 13 ช่วงที่ (12) เป็นการเปลี่ยนค่าทุกค่าให้กลับเป็นค่าที่เหมาะสมค่าเดิมก่อนการเปลี่ยนแปลง จะเห็นว่าระบบเริ่มเข้าสู่สภาวะเสถียรอีกครั้ง



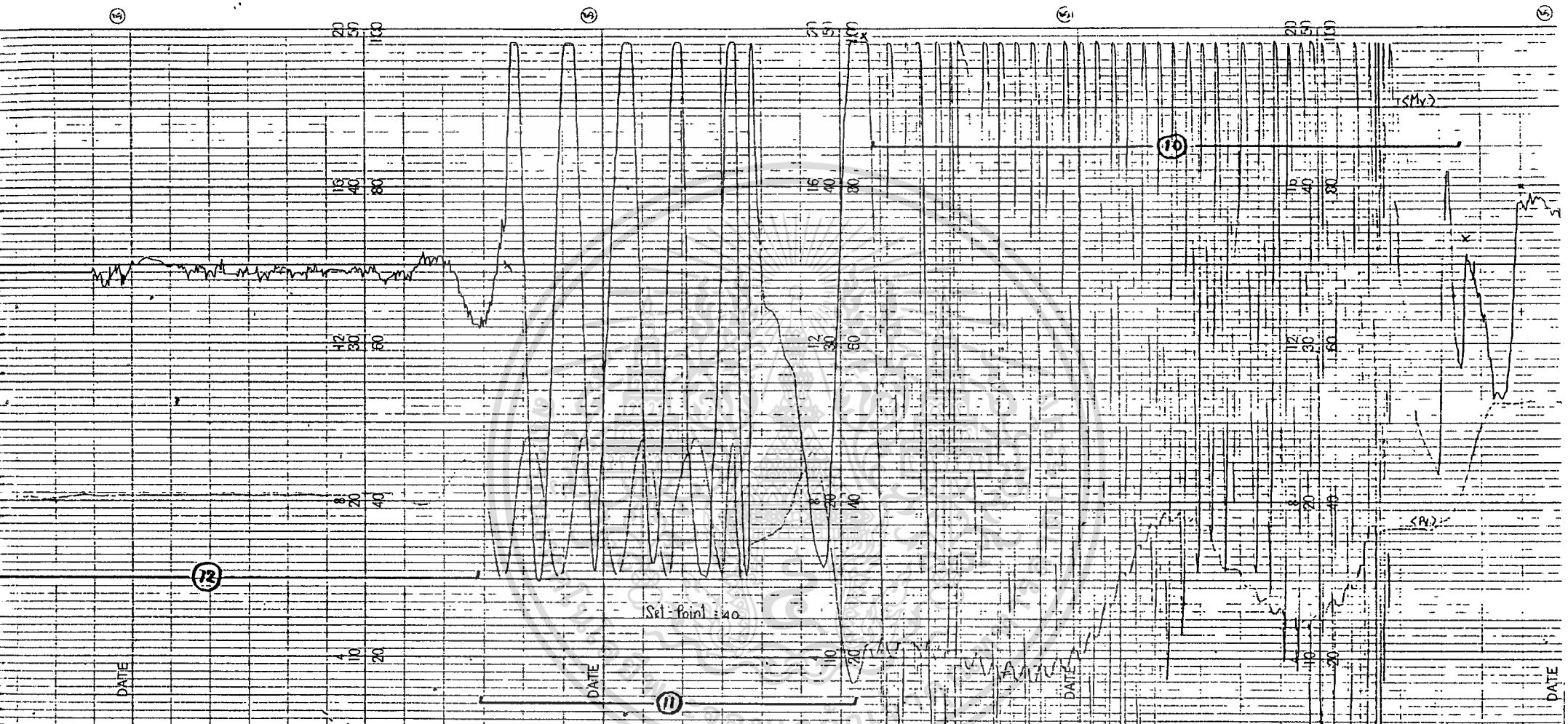
กราฟรูปที่ 1 การเปลี่ยนค่า Set-Point การทดลองที่ 1



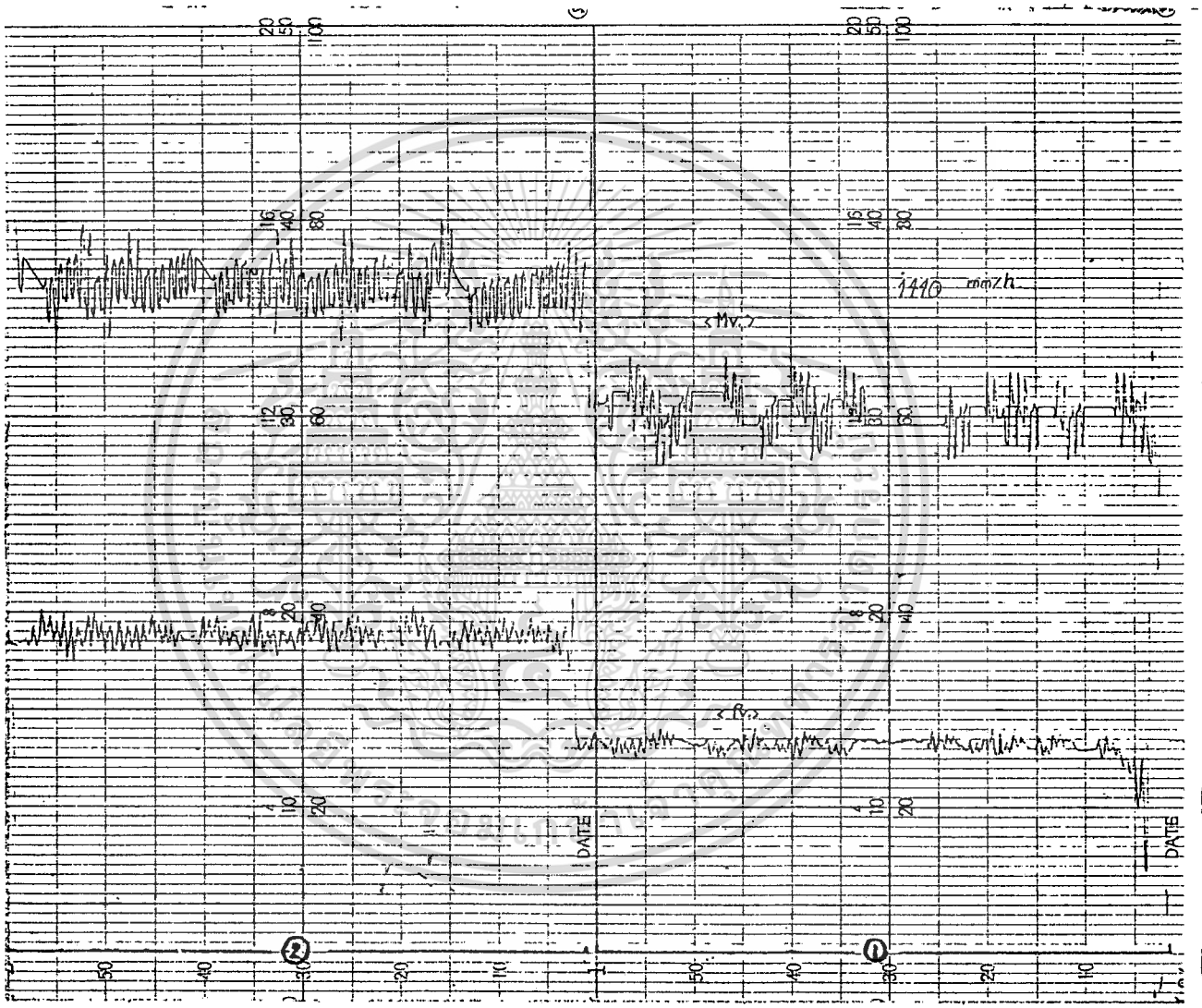
กราฟรูปที่ 3 การเปลี่ยนค่า Gain ต่าง ๆ (Kp , Ki)



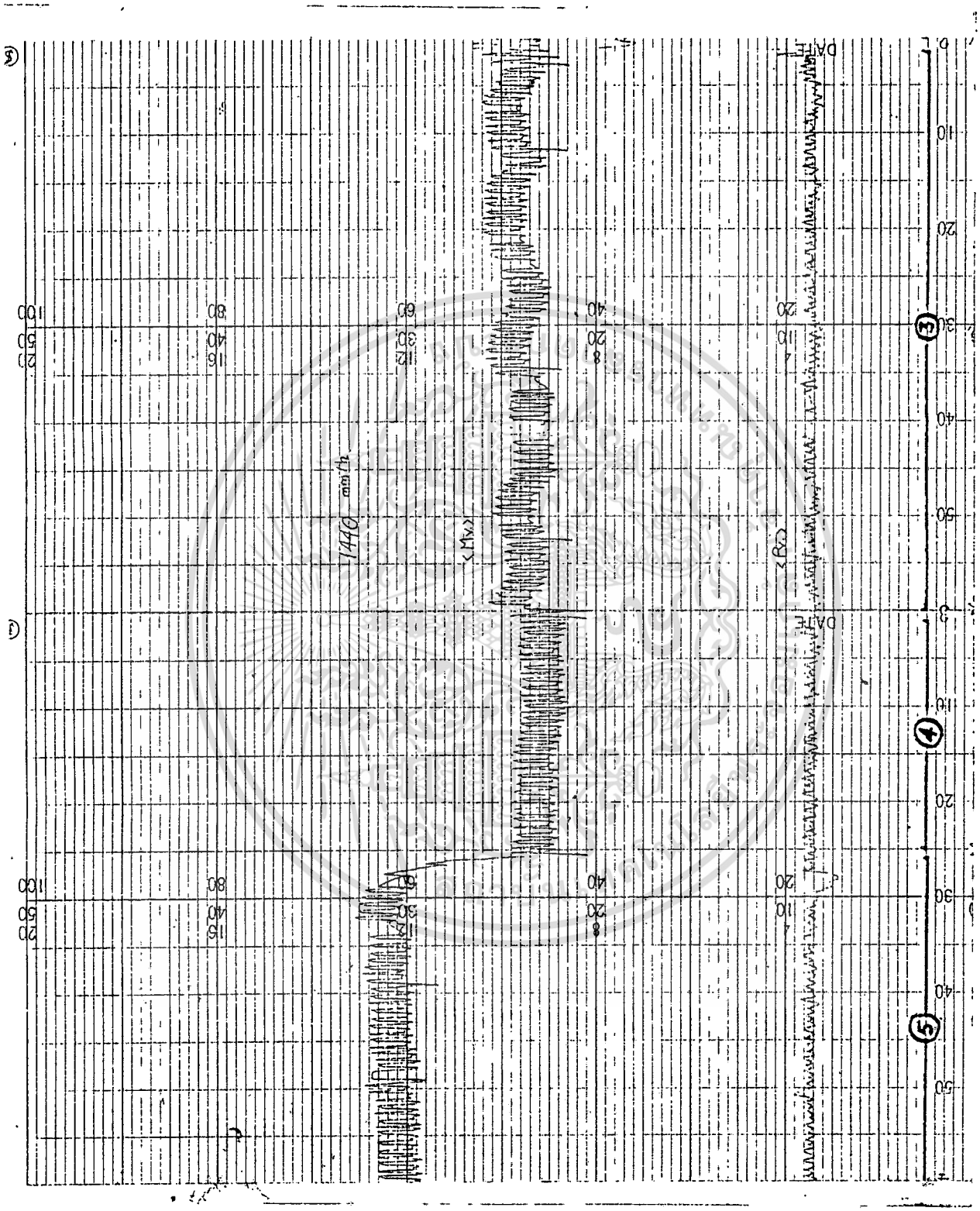
กราฟรูปที่ 2 ผลกระทบภายนอก



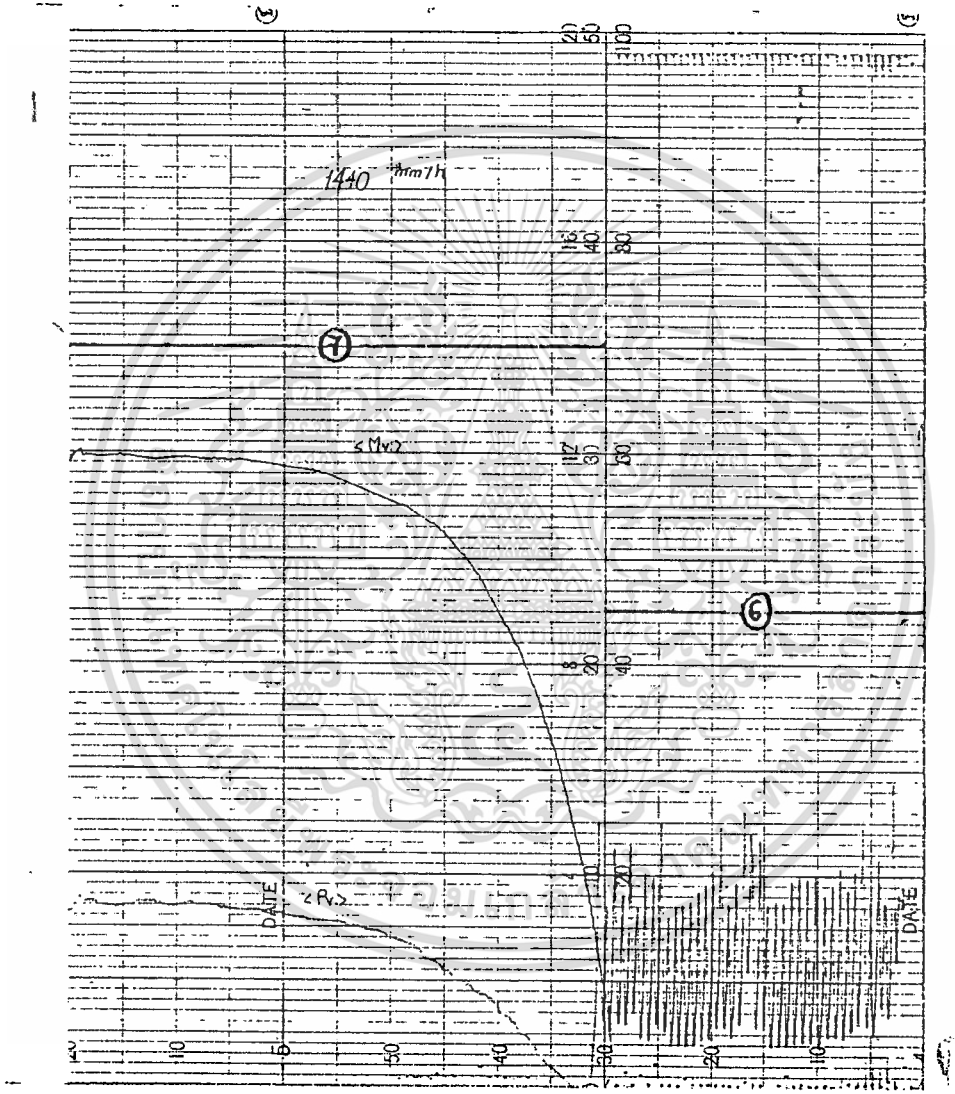
กราฟรูปที่ 4 การเปลี่ยนค่า Gain ต่าง ๆ (Kd) และ การเข้าสู่สภาวะเสถียร



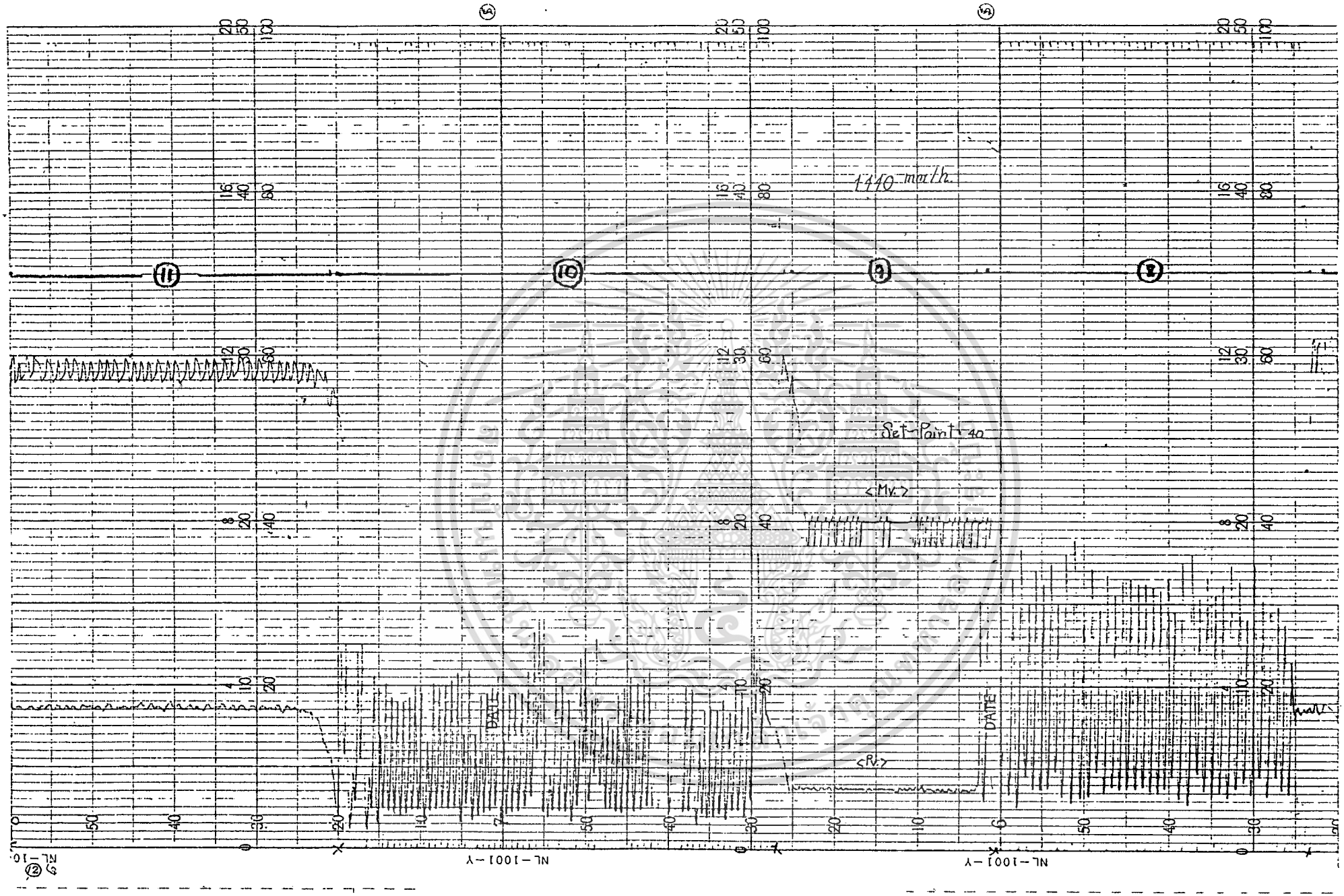
กราฟรูปที่ 5 การเปลี่ยนค่า Set-Point การทดลองที่ 2



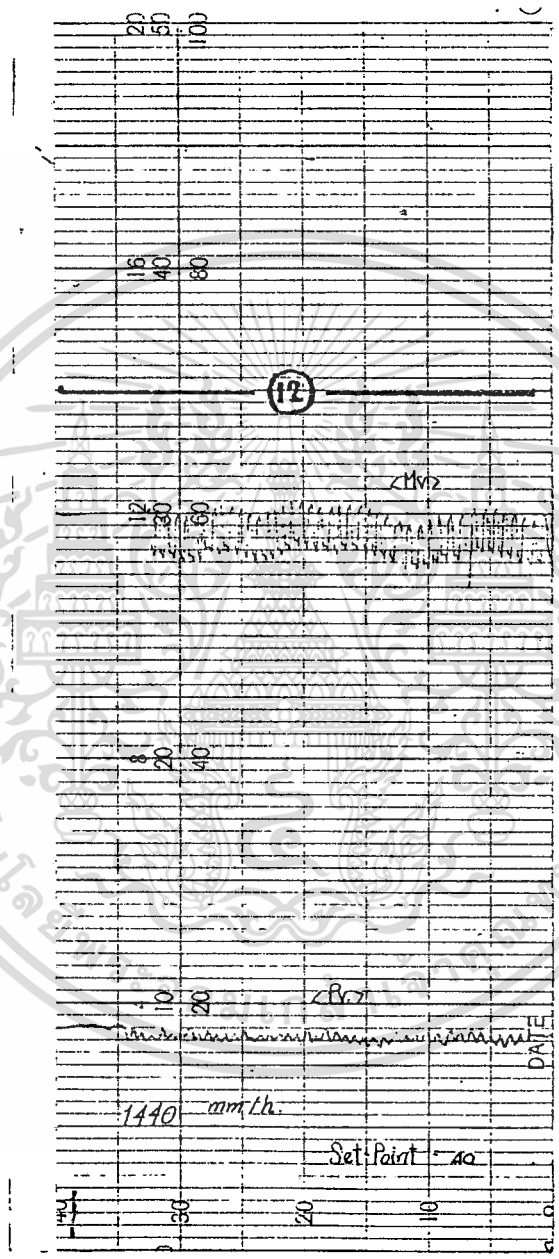
กราฟรูปที่ 6 การเปลี่ยนค่า Set-Point และผลกระทบภายนอก



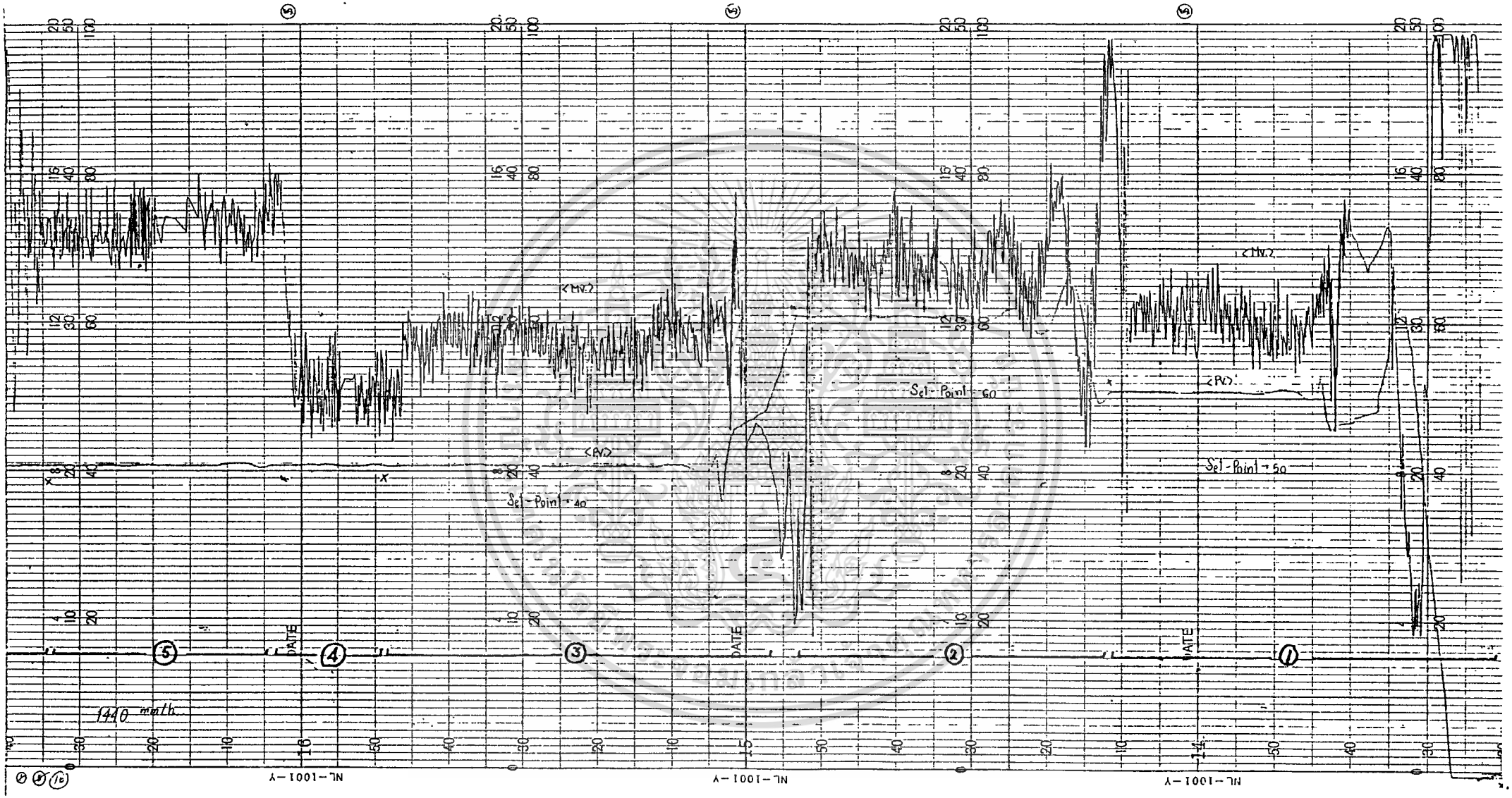
กราฟรูปที่ 7 การเปลี่ยนค่า Gain ต่าง ๆ (Kp)



กราฟรูปที่ 8 การเปลี่ยนค่า Gain ต่าง ๆ (Ki , Kd)

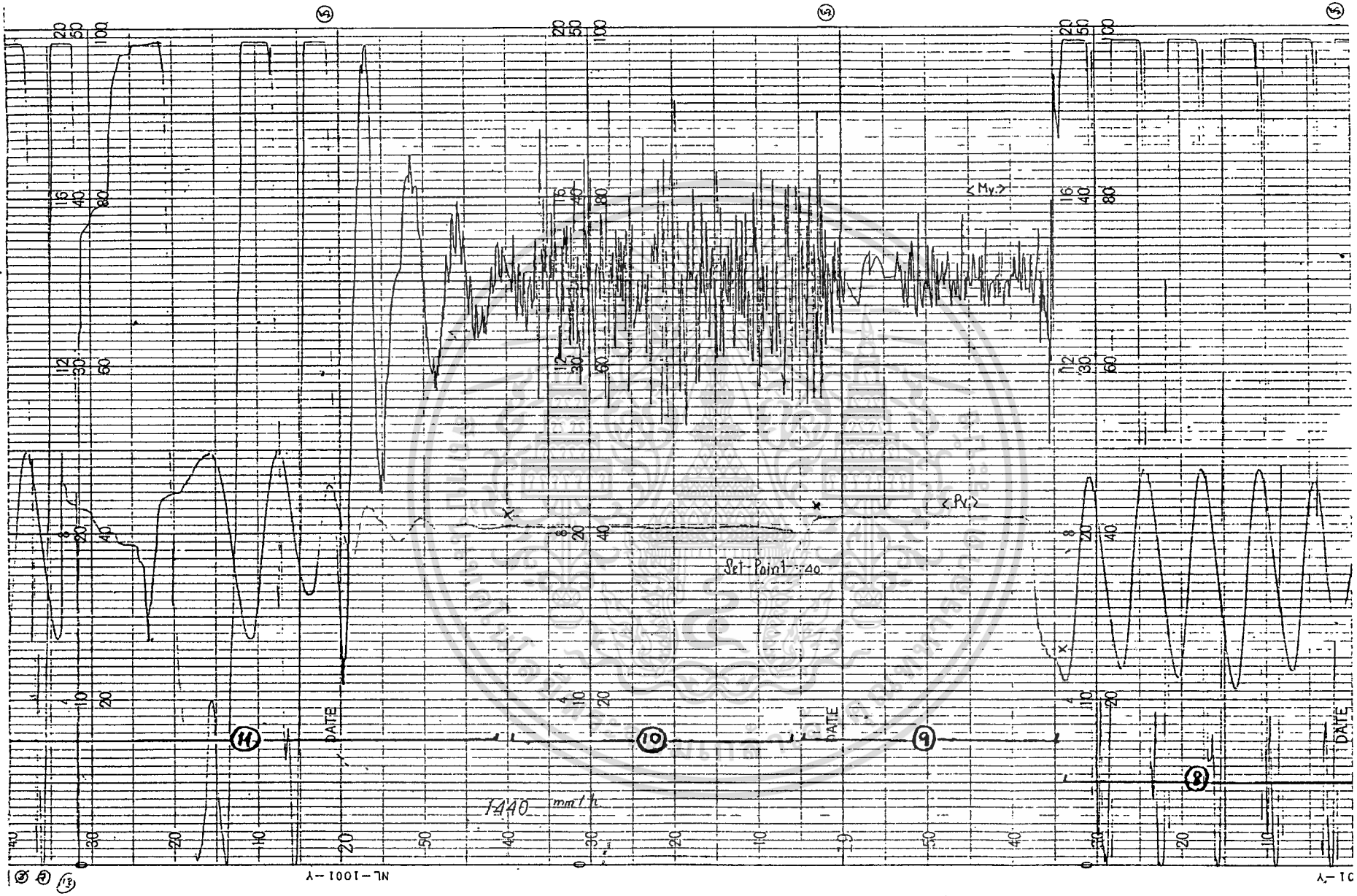


กราฟรูปที่ 9 การเข้าสู่สภาวะเสถียร

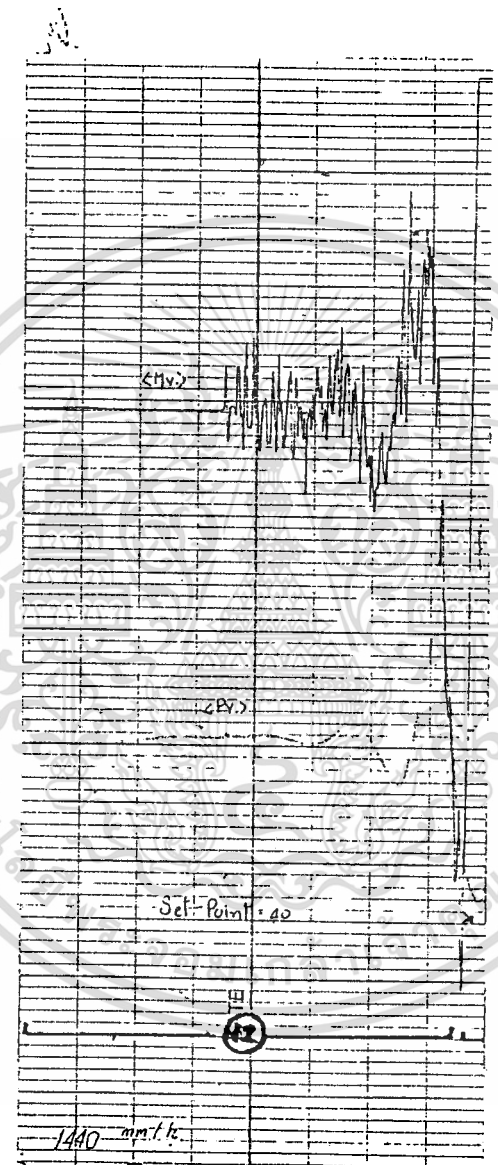


กราฟรูปที่ 10 การเปลี่ยนค่า Set-Point การทดลองที่ 3 และผลกระทบภายนอก

กราฟรูปที่ 11 การเปลี่ยนค่า Gain ต่าง ๆ (Kp)



กราฟรูปที่ 12 การเปลี่ยนค่า Gain ต่าง ๆ (Ki , Kd)



กราฟรูปที่ 13 การเข้าสู่สภาวะเสถียรอีกครั้ง

กราฟรูปที่ 14 แสดงการตรวจสอบค่า Error ที่เกิดขึ้นจากอุปกรณ์ Recorder ด้วยการใส่ค่า Input ทีละ 10 %

ภาค 5

ปัญหาที่เกิดขึ้น และ ข้อเสนอแนะ.

จากประสบการณ์ในการทำโครงงานชิ้นนี้ ปัญหาส่วนใหญ่มักเกิดขึ้นจากเทคนิคในการเขียนโปรแกรมด้วยภาษาซี ซึ่งคณะผู้ทำโครงงานนี้ด้อยประสบการณ์ในการเขียนโปรแกรม ทำให้ในบางครั้งไม่อาจบรรลุตามจุดประสงค์ที่ต้องการได้ อาทิเช่น ในการสร้างฐานเวลาเพื่อเป็นจุดอ้างอิงการทำงานของโปรแกรม เมื่อเปรียบเทียบกับ **real time** ของ **personal computer** ยังมีความคลาดเคลื่อนอยู่ มีผลให้ **sampling period** ของโปรแกรมมีความคลาดเคลื่อนตามไปด้วย แนวทางแก้ไขในข้อบกพร่องข้อนี้คือการศึกษาถึงการ **interupt** และนำ **clock** ของ **computer** มาใช้ให้เป็นประโยชน์

การศึกษาเกี่ยวกับการติดต่อกับ I/O มีความสำคัญอย่างยิ่งเพราะการทำงานด้านนี้มักจะมีข้อผิดพลาดเกิดขึ้นเสมอๆ ถ้าหากผู้ที่ต้องการนำโครงงานนี้ไปปรับปรุงหรือใช้งานจริงควรจะศึกษาทางด้านนี้ให้เชี่ยวชาญอันจะช่วยให้โปรแกรมนี้สมบูรณ์ยิ่งขึ้น

การแสดงผลหน้าจอที่มีทั้ง **trend graph** และ **bar graph** ก็เป็นข้อดีอีกอย่างหนึ่งที่ทำให้ผู้ใช้โปรแกรมนี้มีความสะดวก เข้าใจง่ายกว่าการใช้งาน **controller** ทั่วๆ ไป ซึ่งประโยชน์ในข้อนี้สามารถนำมาประยุกต์ใช้เป็นคู่มือในการเรียนการสอนเกี่ยวกับวิชาที่ว่าด้วย การควบคุมกระบวนการและทฤษฎีการควบคุมกระบวนการ ให้สามารถเข้าใจได้ง่ายยิ่งขึ้น แต่ในส่วนของการแสดงผลหน้าจอค่อนข้างใช้เวลานาน ทำให้ไม่สามารถควบคุมกระบวนการที่ต้องการ **sampling period** ต่ำกว่าประมาณ 1000 มิลลิวินาทีได้ แต่โดยทั่วไปแล้ว กระบวนการส่วนใหญ่ก็มักจะไม่จำเป็นต้องมีค่า **sampling period** น้อยๆ

เนื่องจากการใช้โปรแกรมในการทำงาน และแสดงผลในโหมดการทำงานแบบsimulation ดูได้จากกราฟการแสดงผลของโปรแกรม ทำให้นักศึกษาสามารถเห็นแนวโน้มของการควบคุมเพื่อให้ได้ตามความต้องการ ซึ่งจะพบว่าค่าตัวแปรต่างๆ ในกระบวนการมีผลอย่างไรต่อการควบคุม แต่คณะผู้จัดทำได้จำลองกระบวนการไว้เพียง **first-order** และ **second-order** 2 ประเภทเท่านั้น ซึ่งในอนาคตอาจจะมีการเพิ่มเติมในส่วนของการจำลองกระบวนการให้มีความซับซ้อนมากยิ่งขึ้นเพื่อรองรับกับกระบวนการต่างๆ ได้มากที่สุดเท่าที่จะเป็นไปได้

หนังสืออ้างอิง

ร.ศ. มัณฑนา ปราการสมุทร, **การเขียนชุดคำสั่งภาษา C**. บริษัท ดวงกลมสมัย จำกัด, มิถุนายน 2534.

มนตรี พจนารถลาวัลย์, **การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี**, บริษัท ซีเอ็ดยูเคชั่น จำกัด

สมพัฒน์ รุ่งตะวันเรืองศรี, **เรียนรู้คอมพิวเตอร์กราฟิกส์ 2 มิติ ด้วยภาษา C**. บริษัท ซีเอ็ดยูเคชั่น จำกัด

ศิวัฒน์ ศิวะบวร, พรชัย จักรธำรงค์, จิรศักดิ์ ชัยวิริยะกุล, **การประยุกต์ใช้งานภาษาซี**, บริษัท ซีเอ็ดยูเคชั่น จำกัด

กิตติ ตีรเศรษฐ, **การวิเคราะห์ระบบควบคุมเชิงเส้นเล่ม 1** ,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, เมษายน 2535

กิตติ ตีรเศรษฐ, **พื้นฐานวิศวกรรมระบบควบคุม**, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, มิถุนายน 2535

สุเธียร เกียรติสุนทร, **พื้นฐานวิศวกรรมระบบควบคุมในกระบวนการอุตสาหกรรม**, สมาคมส่งเสริมเทคโนโลยีไทย-ญี่ปุ่น, พฤษภาคม 2536

วีรยุทธ์ เลิศนที, อรชร อินคานุวัฒน์, **คู่มือการใช้โปรแกรม Page Maker 4 บน Windows** , บริษัท ซีเอ็ดยูเคชั่น จำกัด

Carlos A. Smith, Armando B. Corripio, **Principle and Practice of Automatic Process Control**, John Wiley&Sons.

Curtis D. Johnson, **Microprocessor-Based Process Control,** Prentice-Hall, INC.

Peter Hamann, Steve Willings; **Mini H-Type Programmable Controller Operation Manual**, Omron, 1990.

Omron Tateisi Electronics Co., **Omron User's Manual PID Unit**, Omron, 1990.