การออกแบบและสร้างตัวควบคุมเชิงเลข สำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง

โดย

สมบูรณ์ จงชัยกิจ
กฤษดา วิศวธีรานนท์

โครงการวิจัย เลขที่ 105 - IR - 2530
ทุนส่งเสริมการวิจัยวิศวกรรมศาสตร์

สถาบันวิจัยและพัฒนาของคณะวิศวกรรมศาสตร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
กรุงเทพ ฯ

สิงหาคม 2532

สถาบันวิจัยและพัฒนาของคณะวิศวกรรมศาสตร์ไม่รับผิดชอบต่อผลเสียใดๆ อันอาจเกิดจากการนำความคิดเห็นในเอกสารฉบับนี้ไปใช้ ความคิดเห็นที่ปรากฏในเอกสารเป็นความคิดเห็นของผู้เขียนซึ่งไม่จำเป็นต้องเป็นความคิดเห็นของสถาบันฯ

การออกแบบและสร้างตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง

(Design and Construction of digital Controller for Continuous Industrial Process)

โดย

อาจารย์ ดร.สมบูรณ์ จงชัยกิจ

วุฒิ วศ.บ.(จุฬาฯ) วศ.ม.(จุฬาฯ) D.E.S.(E.S.E.)

Dr.-Ing. (Paris XI)

รองศาสตราจารย์ กฤษดา วิศวธีรานนท์

วุฒิ B.Eng.(Kyoto) M.Eng.(Kyoto)

โครงการวิจัย เลขที่ 105-IR-2530

ทุนส่งเสริมการวิจัยวิศวกรรมศาสตร์

สถาบันวิจัยและพัฒนาของคณะวิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

กรุงเทพฯ

สิงหาคม 2532

บทคัดย่อ

การวิจัยครั้งนี้เป็นการออกแบบและสร้างตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง ซึ่งอาจนำไปพัฒนาในเชิงการค้าได้ในอนาคต

ตัวควบคุมเชิงเลขที่ออกแบบเป็นชนิดที่ผู้ใช้สามารถโปรแกรมโครงสร้างการควบคุมได้เอง โดยมีรายละเอียดดังนี้คือ ซีพียูที่ใช้เป็นไมโครโปรเซสเซอร์ Intel 8088 ขนาด 16 บิท ทำงานที่ความถี่ 5 เมกกะเฮร์ท โดยมีสัญญาณขาเข้าประกอบด้วยสัญญาณอนาลอค 1-5 V_{dc} จำนวน 5 สัญญาณ และสัญญาณดิจิตัลจำนวน 3 สัญญาณ สัญญาณออกประกอบด้วยสัญญาณอนาลอค 4-20 mA_{dc} 1 สัญญาณ 1-5 V_{dc} 2 สัญญาณ และสัญญาณดิจิตัลจำนวน 3 สัญญาณ มีฟังค์ชันในการควบคุมจำพวกคณิตศาสตร์ ตรรก พื้นฐาน และ PID ซึ่งสามารถนำมาโปรแกรมกำหนดรูปแบบการควบคุมให้เป็นแบบที่ซับซ้อนจำพวก Cascade Feedback-feedforward Ratioฯลฯได้ โปรแกรมกำหนดรูปแบบการควบคุมใช้ภาษาคล้ายภาษาแอสแซมบลี้

ตัวควบคุมเชิงเลขที่สร้างขึ้นได้นำไปทดสอบกับโปรเซสจริงในห้องปฏิบัติการ โดยมีรูปแบบการควบคุม 2 แบบ คือ การควบคุมป้อนกลับแบบลูปเดียวและการควบคุมแบบ Cascade เวลาในการทำงานแต่ละรอบมีค่าเท่ากับ 8 และ 24 ms. ตามลำดับ

ตัวควบคุมดังกล่าวสามารถนำไปดัดแปลงเป็นตัวการควบคุมที่ซับซ้อนได้ถึง 8 ลูป

การวิจัยที่จะทำต่อไปได้แก่การออกแบบวงจรพิมพ์ใหม่ให้มีขนาดเล็กลง การทดสอบความทนทานในโรงงานเพื่อหาจุดอ่อนของวงจรสำหรับการแก้ไข การเขียนโปรแกรมส่วนติดต่อกับไมโครคอมพิวเตอร์ การเขียนโปรแกรม Mneumonic Compiler และการพัฒนาไมโครคอมพิวเตอร์เป็นเครื่องปรับตั้งค่าของ PID (PID Controller Tuning)

Abstract

This research is the design and construction of digital controller for continuous industrial process which can be developed to be a commercial product in the future.

The developed digital controller is the one that user can program the control configuration. The 16-bit Intel 8088 general purpose microprocessos runs at 5 MHz is setlected as the CPU. This digital Controller can handle 5 analog inputs (1-5 V_{dc}), 3 digital inputs, 3 analog outputs (on 4-20 mA_{dc} and the others 1-5 V_{dc}) and 3 digital outputs. Control functions are arithmetic, logical, basic and PID. With these functions, the complex controls such as cascade, feedback-feedforward, ratio etc. can be implemented. Assembly-like language was chosen for control configuration programming.

Single-loop feedback control and cascade control are tested with the process plant model in the laboratory. The results have proved to be satisfactory with run time of 8 ms./cycle and 24 ms./cycle, respectively.

This controller can, with some modifications, controls 8 complex controls.

The future research will concern: re-design the printed circuit boards to reduce the controller size, endurance tests in industrial environment to find its weak points and to improve the design, develop the microcomputer interface, mneumonic compiler and using microcomputer as PID controller tuner.

กิตติกรรมประกาศ

ในการวิจัยตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่องนี้ คณะผู้วิจัยขอขอบคุณ นายอำนาย แสงวิโรจนพัฒน์ ซึ่งเป็นผู้ช่วยวิจัยในด้านออกแบบ สร้างและพัฒนาตัวควบคุม นายธงชัย ตากวิริยะนันท์ ผู้ซึ่งช่วยในด้านการเชื่อมโยงไมโครคอมพิวเตอร์เข้ากับตัวควบคุม

ขอขอบคุณ คุณธนวดี นิมมานุทย์ ที่ช่วยพิมพ์ต้นฉบับรายงานการวิจัยนี้ได้อย่างสมบูรณ์และรวดเร็ว

สุดท้ายนี้ คณะผู้วิจัยขอขอบคุณสถาบันวิจัยและพัฒนา คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยที่ได้สนับสนุนการวิจัยนี้ด้วยเงินทุนคณะวิศวกรรมศาสตร์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมา

ปัจจุบันโรงงานอุตสาหกรรมในประเทศไทยไม่ว่าจะเป็นโรงงานขนาดเล็ก ขนาดกลาง หรือขนาดใหญ่ ได้มีการนำเอาระบบควบคุมแบบอัตโนมัติมาใช้ควบคุมขบวนการผลิตอย่างกว้างขวาง ระบบควบคุมแบบอัตโนมัติมีมูลค่าค่อนข้างสูงประมาณ 10-20% ของมูลค่าโรงงาน คิดเป็นมูลค่ารวมไม่ต่ำกว่าปีละ 1000 ล้านบาท อุปกรณ์ต่าง ๆ ที่ประกอบกันขึ้นเป็นระบบควบคุมอัตโนมัติมักจะสั่งซื้อจากต่างประเทศทั้งหมดหรือเกือบทั้งหมด ยังผลทำให้การขาดดุลชำระเงินตราต่างประเทศของประเทศไทยเพิ่มขึ้น แม้ว่ามูลค่าการส่งสินค้าออกจะเพิ่มขึ้นอย่างมากก็ตาม เพื่อที่จะลดการขาดดุลการชำระเงินตราต่างประเทศลง จำเป็นอย่างยิ่งที่ประเทศไทยจะต้องส่งเสริมให้มีการพัฒนาอุปกรณ์ต่าง ๆ ดังกล่าวขึ้นใช้เองภายในประเทศ คณะผู้วิจัยซึ่งเป็นผู้ใกล้ชิดเกี่ยวข้องกับระบบควบคุมอัตโนมัติทางอุตสาหกรรม ทั้งในแง่เป็นวิทยากรอบรมช่างเทคนิคของโรงงานต่าง ๆ และในแง่ที่เป็นอาจารย์ในมหาวิทยาลัย จึงมีความคิดที่จะลองพัฒนาอุปกรณ์ต่าง ๆ ดังกล่าวขึ้น

ตัวควบคุม (Controller) เป็นอุปกรณ์ที่สำคัญมากในระบบควบคุมอัตโนมัติ โดยหลักการแล้ว ตัวควบคุมเป็นเครื่องคำนวณซึ่งผู้ใช้สามารถปรับเปลี่ยนกฎเกณฑ์การคำนวณได้

ในยุคต้น ๆ [1] ตัวควบคุมเป็นแบบระบบลม (Pneumatic Controller) ต่อมาได้มีการนำเอาตัวควบคุมแบบอนาลอค (Analog Controller) มาทดแทน ตัวควบคุมดังกล่าวทั้งสองแบบมีข้อจำกัดสำคัญอยู่ที่ขีดความสามารถในการคำนวณ กล่าวคือตัวควบคุมดังกล่าวเป็นเพียงแค่ตัวควบคุมแบบ PID (Proportional Integral Derivative) เท่านั้น ในระบบควบคุมอัตโนมัติทางอุตสาหกรรมผู้ใช้มักจำเป็นต้องทำการคำนวณอย่างอื่นประกอบ เช่น กำลังสอง การปรับสัญญาณให้ความสัมพันธ์เป็นแบบเชิงเส้นตรง (Linearization) เป็นต้น ในระบบควบคุมอัตโนมัติที่ใช้ตัวควบคุมทั้งสองแบบดังกล่าวผู้ใช้จำเป็นต้องเพิ่มอุปกรณ์เพื่อทำการคำนวณประกอบเข้าไปอีก

ตัวควบคุมแบบเชิงเลข (Digital Controller) มีขีดความสามารถในการคำนวณสูง กล่าวคือ ตัวควบคุมแบบเชิงเลขตัวหนึ่งจะทำการคำนวณหลายอย่างได้ทำให้ผู้ซื้อไม่ต้องเพิ่มอุปกรณ์คำนวณประกอบอื่น ๆ อีก นอกจากนั้นยังอาจจะสามารถควบคุมระบบอัตโนมัติได้ทีละหลายๆ ระบบ ตัวควบเชิงเลขจึงมีราคาถูกกว่า

การนำเอาตัวควบคุมไปใช้งาน อาจจะเป็นแบบตัวควบคุมตัวเดียวแยกเป็นอิสระ (Stand Alone) ซึ่งเหมาะกับอุตสาหกรรมขนาดเล็ก [2,3,4] หรืออาจจะเป็นระบบลำดับชั้น (Hierarchical Control) ซึ่งเหมาะกับอุตสาหกรรมขนาดกลางและขนาดใหญ่ [1]

การวิจัยจะมุ่งเน้นไปที่ตัวควบคุมเชิงเลขแบบเดี่ยวอิสระ (Stand Alone) สำหรับอุตสาหกรรมขนาดเล็ก

1.2 วัตถุประสงค์

เพื่อหาโครงสร้างของตัวควบคุมเชิงเลขที่เหมาะสมทั้งทางด้าน Software และ Hardware สำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง

1.3 เป้าหมาย

- ออกแบบและสร้างตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง ซึ่งอาจนำไปพัฒนาเชิงการค้าในอนาคตได้

- ทดสอบการทำงานของตัวควบคุมเชิงเลขกับโปรเซสจริงในห้องปฏิบัติการ เปรียบเทียบกับตัวควบคุมเชิงเลขขนาดเดียวกับที่มีขายในท้องตลาด

1.4 ขั้นตอนการวิจัย

ศึกษารวบรวมข้อมูลและสร้างอุปกรณ์ทดลอง (Hardware)

1. ศึกษาและรวบรวมข้อมูลเกี่ยวกับตัวควบคุมเชิงเลขขนาดเล็กที่มีขายในท้องตลาด
2. ศึกษาและรวบรวมข้อมูลเกี่ยวกับความต้องการของโรงงาน
3. จัดซื้อและประกอบชุดไมโครโปรเซสเซอร์ แบบแผ่นเดียวและแผ่นวงจร A/D และ D/A

พัฒนาโปรแกรม (Software)

4. เขียน ทดลอง ตรวจสอบและแก้ไข (Debug) โปรแกรมซึ่งทำหน้าที่คำนวณตามฟังก์ชั่นต่าง ๆ ในระบบควบคุม เช่นฟังก์ชั่นของการควบคุมแบบ (PID) ฟังก์ชั่นการถอดรากกำลังสอง ฯลฯ

5. ทดลองนำไปควบคุมแบบจำลองโรงงาน (Plant Model) ซึ่งสร้างเสร็จแล้ว เพื่อนำผลสรุปไปปรับปรุงโปรแกรมจนเป็นที่พอใจ

6. วิเคราะห์และสรุปผลการทดลอง (และแก้ไขปรับปรุง ถ้ามีปัญหา)

งานเอกสาร

7. เขียนรายงานการวิจัย

ตารางเวลาของแต่ละขั้นตอนในการดำเนินการวิจัยแสดงในตารางที่ 1.1

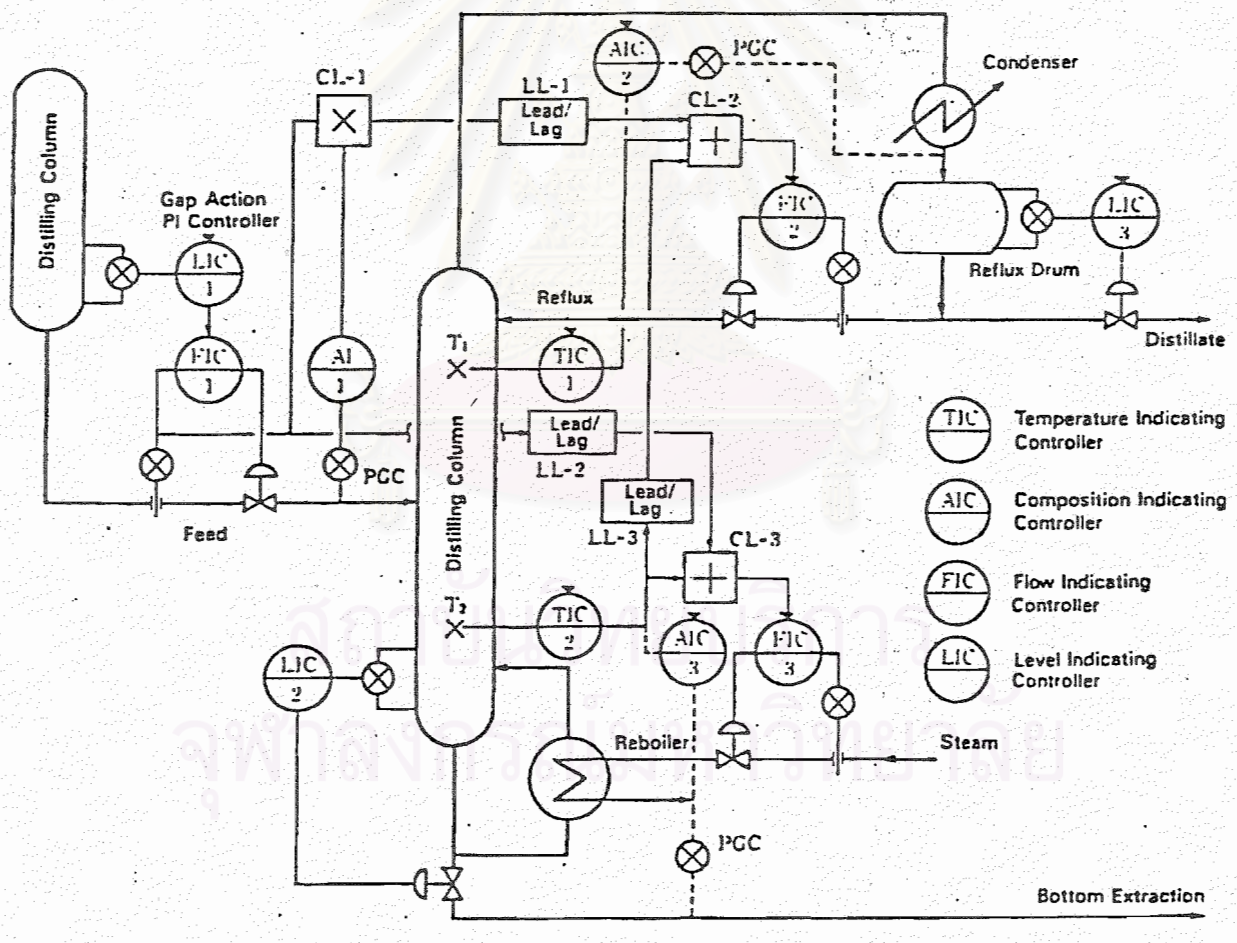
ตารางที่ 1.1 ตารางเวลาแสดงช่วงเวลาของแต่ละขั้นตอนการวิจัย

ขั้นตอนที่	ปี พ.ศ. 2530 - 3531											
	ตค.	พย.	ธค.	มค.	กพ.	มีค.	เมย.	พค.	มิย.	กค.	สค.	กย.
1	* *	* *										
2		*	* *									
3	* *	* *										
4			*	* *	* *	* *						
5						*	* *	* *				
6							*	* *	* *	* *		
7										*	* *	* * *

บทที่ 2

การควบคุมในงานอุตสาหกรรม

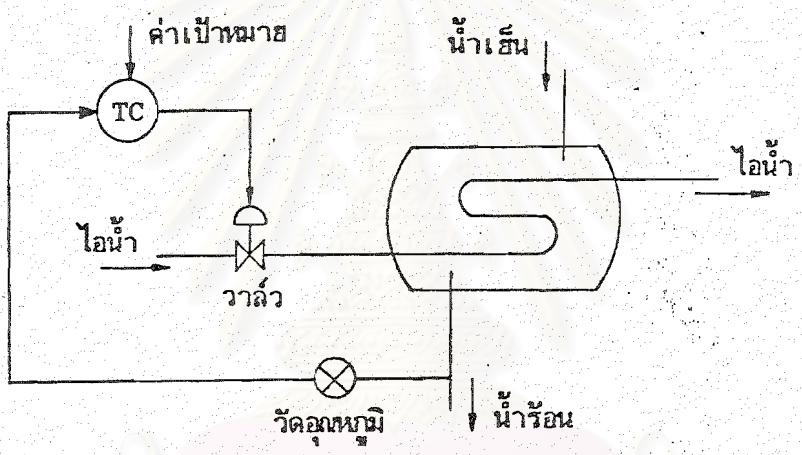
การควบคุมในงานอุตสาหกรรมมีมากมายหลายแบบ ตั้งแต่แบบง่ายๆไปจนถึงแบบซับซ้อนขึ้นกับงานแต่ละชนิด โครงสร้างของระบบควบคุมไม่ว่าจะซับซ้อนเพียงใด โดยส่วนใหญ่เราสามารถแยกแยะออกเป็นระบบควบคุมย่อย ๆ หลาย ๆ ระบบ โดยที่ระบบควบคุมย่อยจะประกอบด้วย การควบคุม PID หนึ่งหรือสองลูปเท่านั้น (ดูรูปที่ 2.1) ในบทนี้จะกล่าวถึงรูปแบบต่าง ๆ ของระบบควบคุมย่อยที่พบเห็นในงานควบคุมทางอุตสาหกรรมทั่วๆไป



รูปที่ 2.1 การควบคุมในงานอุตสาหกรรม (Distilling Column Advanced Control

2.1 องค์ประกอบและนิยามที่สำคัญของการควบคุม

ก่อนที่จะกล่าวถึงรูปแบบการควบคุม (Control Configuration) ชนิดต่างๆ จำเป็นจะต้องทราบถึงองค์ประกอบ และนิยามของเทอมบางเทอมของการควบคุม โดยจะเริ่มจากโปรเซสที่มีตัวแปรที่ถูกควบคุมเพียงตัวเดียวก่อน เช่น โปรเซสของ Heat-exchanger (ดูรูปที่ 2.2) ระบบต้องการควบคุมอุณหภูมิของน้ำร้อน ดังนั้นเครื่องควบคุมจะต้องสามารถควบคุมตัวแปรที่เป็นผลกระทบต่ออุณหภูมิของน้ำ เพื่อให้ได้ระดับอุณหภูมิที่ต้องการ การควบคุมในตัวอย่างนี้เป็นการควบคุมแบบง่ายๆ



รูปที่ 2.2 การควบคุม Heat Exchanger ด้วย Simple Control

จากโปรเซสในรูปที่ 2.2 สามารถเขียนบล็อคไดอะแกรมองค์ประกอบของการควบคุมได้ดังรูปที่ 2.3 ซึ่งนิยามเทอมต่างๆของการควบคุมได้ดังนี้ [5]

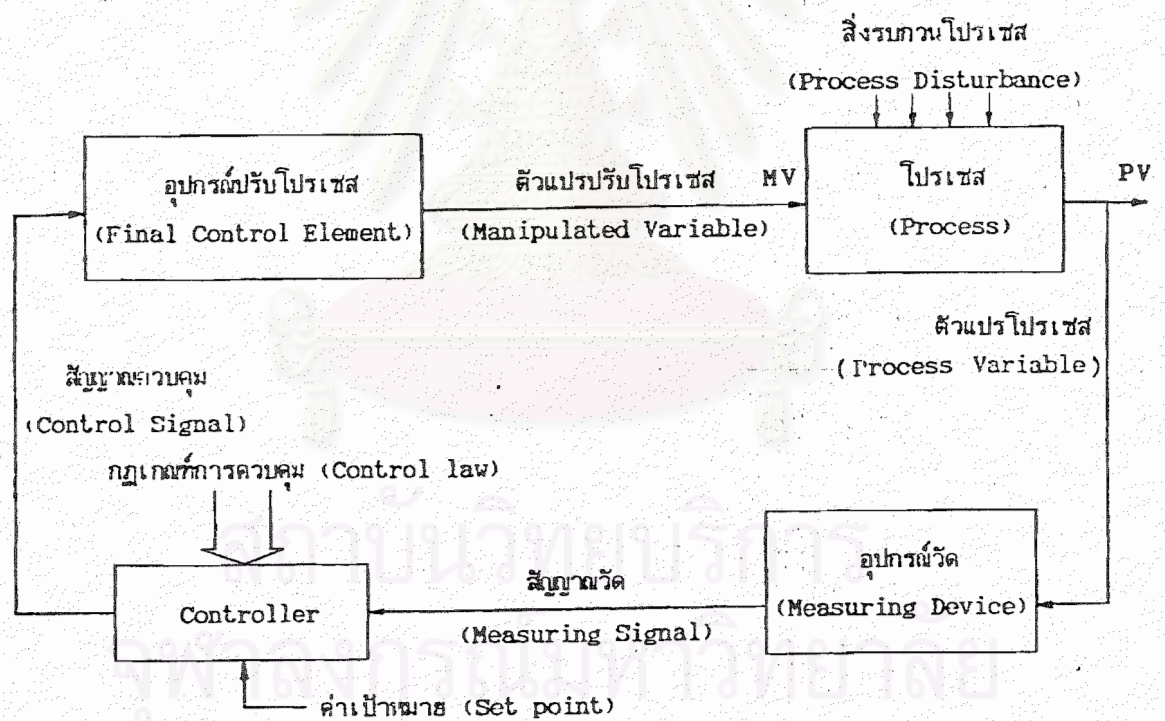
(1) **โปรเซส (Process)** หมายถึง ขบวนการทางฟิสิกส์ที่ต้องการให้มีสภาวะตามต้องการ เมื่อสภาวะการทำงานเปลี่ยนแปลงไป ในรูปที่ 2.2 หมายถึงการทำให้น้ำเย็นกลายเป็นน้ำร้อนโดยการรับพลังงานความร้อนจากไอน้ำ

(2) **ตัวแปรโปรเซส (Process Variable)** คือ ตัวแปรทางฟิสิกส์ ซึ่งแสดงสภาวะของโปรเซส ในรูปที่ 2.2 หมายถึง อุณหภูมิของน้ำร้อน

(3) **ตัวแปรปรับโปรเซส (Manipulated Variable)** คือ ตัวแปรที่ใช้เปลี่ยนแปลงค่าตัวแปรโปรเซส ได้แก่ อัตราไหลของไอน้ำ

นอกจากนี้ค่าตัวแปรโปรเซสจะเปลี่ยนแปลงได้อีก ถ้าเกิด "สิ่งรบกวนโปรเซส" (Process Disturbance) ซึ่งแบ่งได้เป็น 2 ประเภท ดังนี้

- **Supply Disturbance** หมายถึง การเปลี่ยนแปลงของพลังงาน (วัตถุ) ขาเข้า จากรูปที่ 1 หมายถึง การเปลี่ยนแปลงของคุณภาพ อุณหภูมิ ความดันของไอน้ำร้อนขาเข้า
- **Demand Disturbance** หมายถึง การเปลี่ยนแปลงพลังงาน (วัตถุ) ขาออกจากรูปที่ 1 หมายถึง การเปลี่ยนแปลงของอุณหภูมิ อัตราการไหลของน้ำเย็น รวมทั้งการเปลี่ยนแปลงของการสูญเสียพลังงานความร้อนที่ท่อและถัง



รูปที่ 2.3 บล็อคไดอะแกรมแสดงองค์ประกอบของการควบคุม

(4) **อุปกรณ์วัด (Measuring Device)** เป็นอุปกรณ์ที่ให้สัญญาณขาออกซึ่งมีขนาดสัมพันธ์กับขนาดของตัวแปรโปรเซส (Process Variable)

(5) **เครื่องควบคุม (Controller)** ทำหน้าที่ออกคำสั่ง หรือกำเนิดสัญญาณควบคุม (Control Signal) ตามกฎเกณฑ์การควบคุม (Control law) ที่ถูกกำหนดไว้ล่วงหน้า คำสั่ง หรือสัญญาณควบคุมนี้อาจจะเป็นฟังก์ชันกับเวลา หรือ กับสัญญาณขาเข้าที่ได้รับจากอุปกรณ์วัด

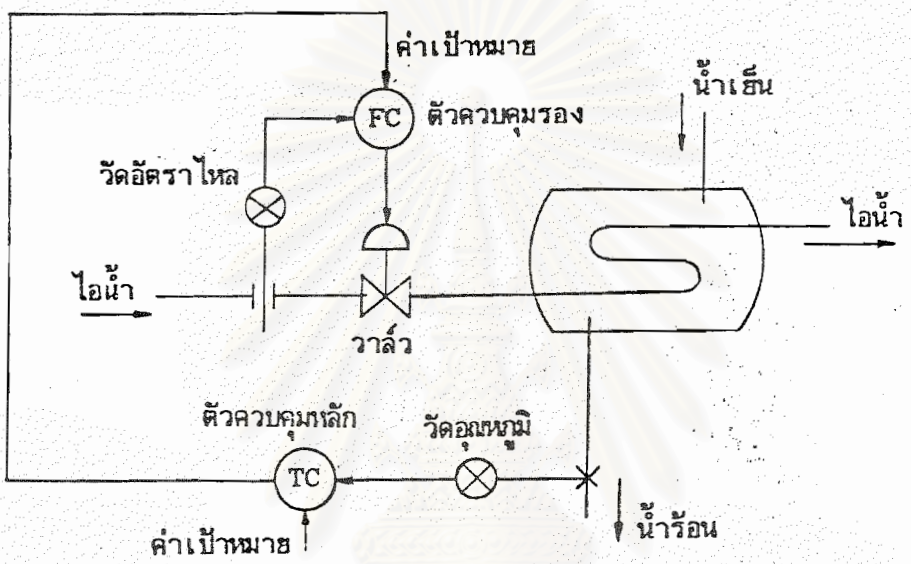
(6) **อุปกรณ์ปรับโปรเซส (Final Control Element)** อุปกรณ์ทำหน้าที่ปรับสภาวะของโปรเซสด้วยการเปลี่ยนแปลงค่าตัวแปรปรับโปรเซส (Manipulated Variable) ตามคำสั่ง หรือสัญญาณควบคุมที่ได้รับจาก Controller ในรูปที่ 2.2 อุปกรณ์ปรับโปรเซส คือ วาล์ว

2.2 รูปแบบการควบคุมในอุตสาหกรรม

เครื่องควบคุมอัตโนมัติที่ใช้ในการควบคุมมีหน้าที่หลัก คือ การคำนวณหาค่าตัวแปรปรับโปรเซส (Manipulated Variable) ที่เหมาะสม เพื่อทำให้ค่าตัวแปรโปรเซสมีค่าเท่ากับค่าเป้าหมาย ซึ่งลักษณะของค่าเป้าหมาย (Set point) ในการควบคุมสามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆ คือ แบบ Servo ปกติจะมีค่าเป้าหมายเปลี่ยนแปลงตลอดเวลา เครื่องควบคุมจะต้องควบคุมให้ตัวแปรโปรเซสเปลี่ยนแปลงตามให้ทัน และแบบ Regulator ซึ่งจะมีค่าเป้าหมายคงที่เกือบตลอดเวลาหรือนานๆเปลี่ยนที เครื่องควบคุมจะต้องควบคุมให้ PV มีค่าเปลี่ยนไปตาม SV แม้ว่าจะมีสัญญาณรบกวน (Disturbance) เข้ามารบกวนระบบทำให้ PV เปลี่ยนไปก็ตาม สัญญาณรบกวนมีทั้งชนิดที่ไม่สามารถรู้ล่วงหน้าหรือคาดเดาได้ และชนิดที่สามารถทำนายได้ ซึ่งมักเกิดจากการเปลี่ยนแปลงของโปรเซสอื่นๆที่เกี่ยวข้อง ซึ่งรูปแบบการควบคุมที่ถูกใช้เพื่อให้ได้ตามจุดประสงค์ที่กล่าวมานั้น มีวิธีการควบคุมได้หลายแบบดังนี้ [5]

2.2.1 การควบคุมแบบง่ายๆ (Simple Control) ในที่นี้หมายถึงระบบที่ประกอบด้วยตัวควบคุมตัวเดียว ควบคุมโปรเซสเพียงโปรเซสเดียว สัญญาณวัด สัญญาณควบคุม ตัวแปรปรับโปรเซส และตัวแปรโปรเซสเพียงอย่างละตัวเท่านั้น (ดูรูปที่ 2.2) ลักษณะนี้จะเป็นการควบคุมแบบ ลูปเดียว (Single loop Control) หลักการควบคุมจะเป็นแบบป้อนกลับ (Feedback Control) ซึ่งประกอบด้วย ตัวควบคุมแบบ PID หนึ่งตัว ควบคุมโปรเซสแบบ Single-Input Single-Output (SISO) ระบบแบบนี้มีใช้กันมาก เนื่องจากสามารถใช้งานกับโปรเซสทั่วๆไปได้ โดยไม่ต้องรู้ค่าคงตัวของโปรเซสต่างๆ ข้อเสียที่สำคัญ คือ สัญญาณรบกวนจะต้องทำให้โปรเซสเปลี่ยนไปจน PV เปลี่ยนแปลงไปก่อน ตัวควบคุมจึงจะเริ่มทำการแก้ไขและลูปที่ควบคุมจะต้องไม่ถูกรบกวนด้วยลูปอื่นๆมากจนเกินไป

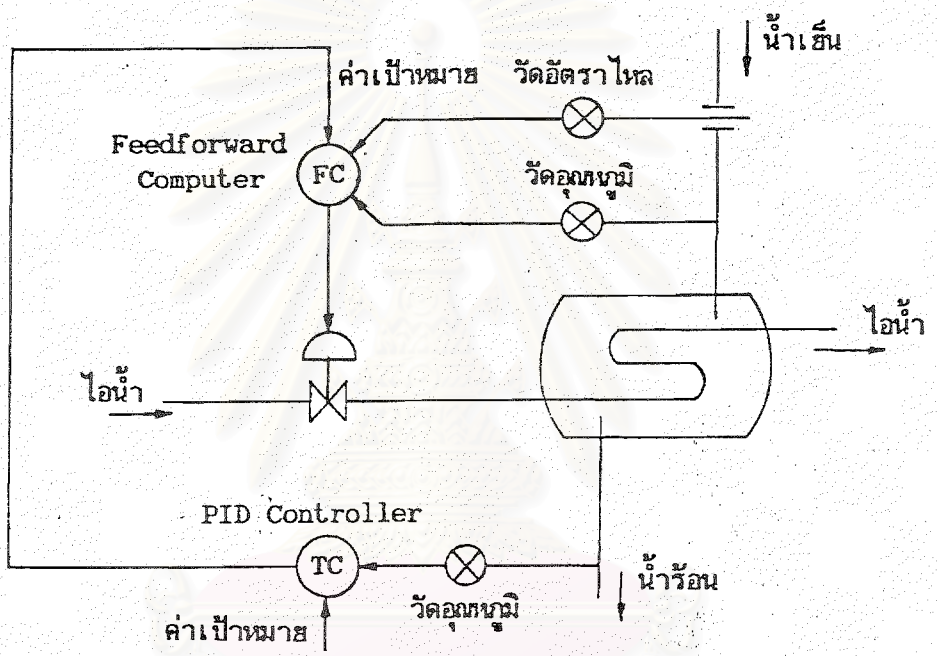
2.2.2 Cascade Control การควบคุมแบบนี้จะใช้เมื่อต้องการผลการควบคุมที่ดีกว่าการควบคุมแบบง่ายๆ (แบบลูปเดียว) ในการควบคุมแบบง่ายๆความสัมพันธ์ระหว่างสัญญาณขาออกกับสัญญาณขาเข้าของอุปกรณ์ปรับโปรเซส อาจเปลี่ยนแปลงได้เมื่อเกิดสิ่งรบกวน ตัวอย่างแสดงในรูปที่ 2.4 อัตราไหลของไอน้ำขาออกของวาล์วอาจจะเปลี่ยนแปลงได้ แม้ว่าสัญญาณควบคุมจะคงที่ ถ้าความดันของไอน้ำเข้าเปลี่ยนแปลง



รูปที่ 2.4 การควบคุม Heat Exchanger ด้วย Cascade Control

การควบคุมแบบ Cascade Control ประกอบด้วยลูปการควบคุมสองลูป ลูปการควบคุมหลัก (Primary หรือ Master Control Loop) ทำหน้าที่เหมือนกับการควบคุมแบบง่ายๆ ส่วนลูปการควบคุมรอง (Secondary หรือ Slave Control Loop) จะเป็นส่วนที่เพิ่มเติมขึ้นเพื่อการปรับปรุงการทำงานของอุปกรณ์ปรับโปรเซส ให้ความสัมพันธ์ระหว่างสัญญาณขาออกกับสัญญาณขาเข้าคงที่ แม้ว่าจะเกิดสิ่งรบกวนระบบ ตัวควบคุมที่ใช้ในลูปทั้งสองจะเป็นแบบ PID (PID Controller)

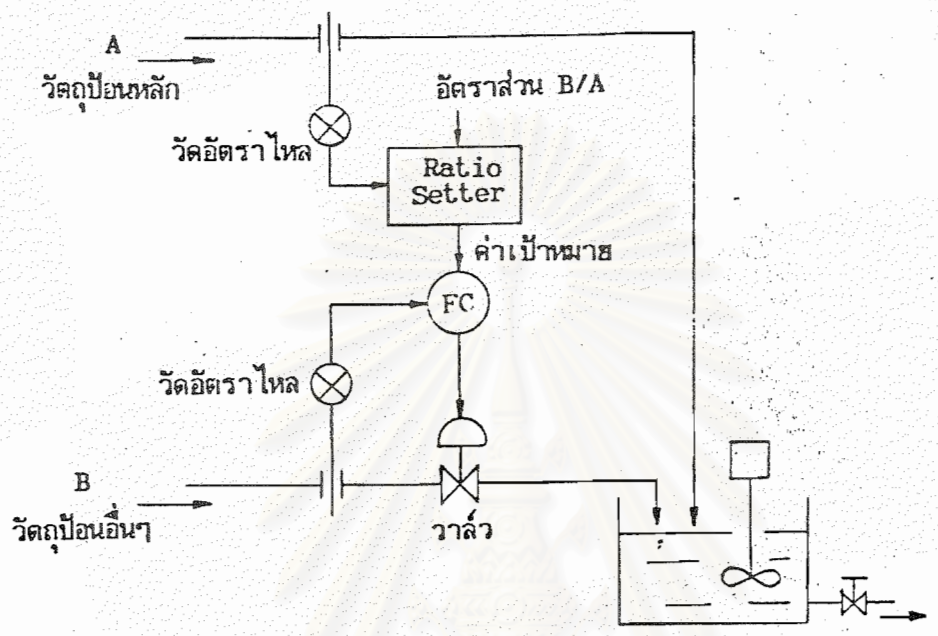
2.2.3 Feedforward-Feedback Control ในการออกแบบระบบควบคุมแบบ Feedforward Control จำเป็นต้องทราบความสัมพันธ์ระหว่างตัวแปรโปรเซสกับสัญญาณควบคุม ค่าเป้าหมาย และสิ่งรบกวน อย่างชัดเจน เพื่อใช้ในการคำนวณหาผลกระทบต่อระบบได้อย่างถูกต้อง ทำให้ระบบ Feedforward-Feedback Control สามารถควบคุมโปรเซสได้โดยไม่ต้องมีระบบป้อนกลับ แต่ในความเป็นจริงแล้วมักจะทราบความสัมพันธ์เพียงคร่าวๆเท่านั้น ในทางปฏิบัติจึงต้องเพิ่ม Feedback Control (ดังรูปที่ 2.5) เพื่อทำหน้าที่หาค่าเป้าหมายที่เหมาะสมสำหรับ Feedforward Control ตัวควบคุมที่ใช้จะเป็นแบบ PID Controller และ Feedforward Computer



รูปที่ 2.5 การควบคุม Heat Exchanger ด้วย Feedforward-Feedback Control

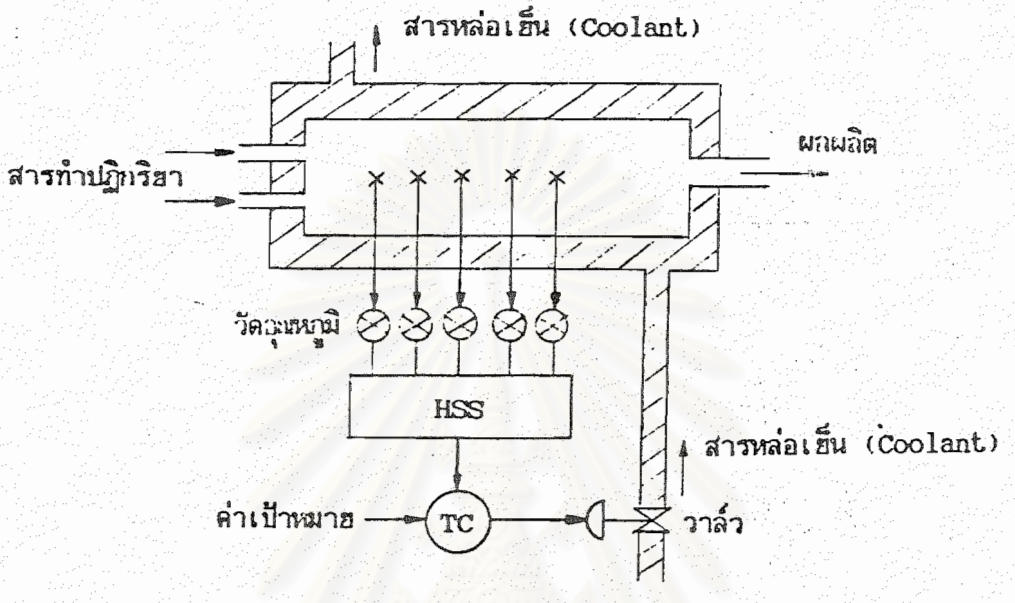
2.2.4 Ratio Control การควบคุมแบบนี้ถูกใช้เมื่อต้องการควบคุม อัตราส่วนของปริมาณวัตถุป้อนหลายๆชนิด ให้มีค่าคงที่ตลอดเวลา เช่น การควบคุมอัตราส่วนระหว่างอากาศกับเชื้อเพลิงในเตาเผา การผสมอาหารสัตว์ เป็นต้น

ในทางปฏิบัติมักจะกำหนดวัตถุป้อนชนิดหนึ่งเป็นวัตถุป้อนหลัก อัตราไหลของวัตถุป้อนหลักอาจจะไม่ถูกควบคุม หรือ อาจจะถูกควบคุมก็ได้แล้วแต่ชนิดของงาน ส่วนอัตราการไหลของวัตถุป้อนอื่นๆ ที่เหลือทุกตัวจะถูกควบคุมให้มีอัตราไหลเป็นอัตราส่วนที่แน่นอนกับอัตราไหลของวัตถุป้อนหลักตลอดเวลา ตัวอย่างการควบคุมแบบ Ratio Control ที่ตัวป้อนหลักไม่ถูกควบคุมแสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 การควบคุมแบบ Ratio Control

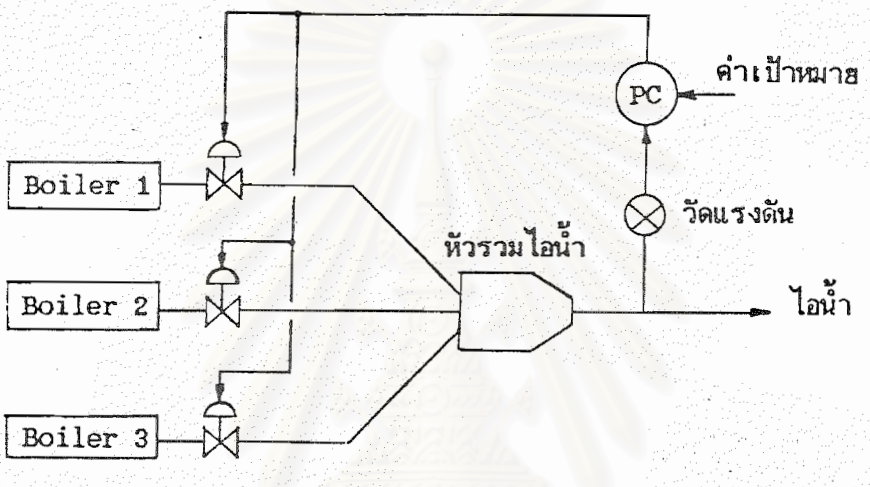
2.2.5 Selective Control เป็นการควบคุมเพื่อจำกัดไม่ให้ตัวแปรโปรเซสมีค่าเกินกว่า หรือต่ำกว่าค่าที่กำหนด เพื่อความปลอดภัยของอุปกรณ์ ลักษณะเฉพาะตัวของการควบคุมแบบนี้ คือ การใช้สวิทช์ HSS (High Selector Switch) เลือกค่าสูงสุด หรือสวิทช์ LSS (Low Selector Switch) เลือกค่าต่ำสุด ตัวอย่างการควบคุมแบบ Selective ในอุตสาหกรรม คือ Auctioneering Control มีลักษณะการทำงานเริ่มจากการเลือกระหว่างสัญญาณวัดชนิดเดียวกันหลายๆค่า ว่าสัญญาณใดมีค่าสูงสุด แล้วนำเอาสัญญาณวัดค่านั้นส่งไปยังตัวควบคุม (ซึ่งมีตัวเดียว) เพื่อควบคุมโปรเซสต่อไป ตัวอย่างงานได้แก่ การป้องกัน Hot Spot ในเตาเผา หรือในหม้อปฏิกิริยาแบบ Catalytic (รูปที่ 2.7) ที่เกิดความร้อนสูงๆ เป็นต้น



รูปที่ 2.7 การควบคุม Auctioneering Control ในหม้อปฏิกริยา

2.2.6 Split-Range Control การควบคุมแบบนี้จะใช้เมื่อจำเป็นต้องใช้อุปกรณ์ปรับโปรเซส (Final Control Element) หลายตัวทำหน้าที่ประสานกันในการปรับโปรเซสเพียงโปรเซสเดียว

ลักษณะพิเศษของการควบคุมแบบนี้ คือ ในระบบควบคุมจะมีอุปกรณ์วัดตัวเดียว และอุปกรณ์ปรับโปรเซสหลายตัว ส่วนตัวควบคุมอาจมีตัวเดียวหรือหลายตัวก็ได้ ตัวอย่างที่ของงานคือการควบคุมแรงดันก๊าซในถัง การควบคุมแรงดันของไอน้ำจากแหล่งจ่าย (Boiler) หลายๆ แหล่ง (รูปที่ 2.8) เป็นต้น



รูปที่ 2.8 การควบคุมแรงดันไอน้ำจากแหล่งจ่ายหลายแห่ง

จากรูปแบบการควบคุมต่างๆ ที่ได้กล่าวมาทั้งหมด จะเห็นได้ว่า การควบคุมที่มีความซับซ้อนจำเป็นต้องมีการคำนวณและฟังก์ชันต่างๆ ที่ช่วยในการควบคุม ดังนั้นการนำไมโครโปรเซสเซอร์ ซึ่งมีความสามารถในการคำนวณมาใช้ในเครื่องควบคุม ทำให้สามารถโปรแกรมกำหนดรูปแบบการควบคุมใช้กับงานควบคุมที่ซับซ้อนได้ โดยไม่จำเป็นต้องมีอุปกรณ์ช่วยมากมายนัก สำหรับรายละเอียดของฟังก์ชันต่างๆที่จำเป็นในการสร้างเครื่องควบคุมที่โปรแกรมได้ จะกล่าวถึงในบทที่ 5

บทที่ 3

การออกแบบเครื่องควบคุมเชิงเลข

จากการศึกษาตัวควบคุมเชิงเลขแบบเดี่ยวอิสระ (Stand Alone) ที่มีขายในท้องตลาดพบว่าตัวควบคุมดังกล่าวยังแบ่งได้เป็นสองประเภทย่อยคือ ตัวควบคุมเชิงเลขแบบที่ผู้ใช้ไม่สามารถและสามารถเปลี่ยนแปลงโครงสร้างการควบคุม (Control Configuration) ได้เอง

ตัวควบคุมแบบแรกมีขีดความสามารถค่อนข้างจำกัดกล่าวคือ สามารถควบคุมได้เพียง loop เดียว มีสัญญาณขาเข้าเพียงสองสัญญาณ (PV:สัญญาณจากอุปกรณ์วัด และ SV:สัญญาณค่าเป้าหมายจากภายนอก) มีสัญญาณขาออกสามสัญญาณแบ่งออกเป็นสัญญาณอนาลอค 4-20 mA_{dc} หนึ่งสัญญาณ และสัญญาณดิจิตัล (High & Low Alaums) สองสัญญาณ ฟังก์ชั่นในการควบคุมตายตัวเพียงหนึ่งหรือสองฟังก์ชัน ซึ่งมักจะได้แก่ฟังก์ชัน PID และฟังก์ชัน Linearization

ตัวควบคุมแบบหลัง [2,3,4] มีขีดความสามารถสูงกว่ากล่าวคือ ควบคุมได้สอง loop เป็นอย่างน้อยโดยสามารถนำไปควบคุมระบบควบคุมแบบ Cascade Ratio Feedback-feedforward ฯลฯได้ โดยผู้ใช้สามารถโปรแกรมรูปแบบของการควบคุม (Control Configuration) ได้เอง ตัวควบคุมแบบนี้จะมีสัญญาณขาเข้าอนาลอค 4-5 สัญญาณ สัญญาณขาเข้าดิจิตัล 2-5 สัญญาณ สัญญาณขาออกอนาลอค 2-4 สัญญาณ สัญญาณขาออกดิจิตัล 2-4 สัญญาณ ฟังก์ชั่นในการควบคุมมีมากมายเช่น PID Linearization Square-root Extraction Lead-lag Compensation บวก ลบ คูณ หาร เป็นต้น

การวิจัยจะทำการออกแบบและสร้างตัวควบคุมแบบหลัง ซึ่งเป็นแบบที่ผู้ใช้สามารถโปรแกรมรูปแบบของการควบคุมได้เอง เนื่องจากความรู้ในการวิจัยครั้งนี้จะสามารถนำไปพัฒนาตัวควบคุมเชิงเลขแบบแรกในอนาคตได้โดยไม่ยาก โดยคณะวิจัยจะเรียกตัวควบคุมแบบนี้ว่า "ตัวควบคุมเชิงเลขชนิดโปรแกรมได้ (Programmable Digital Controller)"

3.1 ข้อกำหนดเป้าหมาย (Target Specification)

จากประสบการณ์ของคณะวิจัยในงานควบคุมทางอุตสาหกรรม และผลจากการศึกษาข้อมูลของตัวควบคุมเชิงเลขชนิดโปรแกรมได้ในท้องตลาด ตัวควบคุมเชิงเลขชนิดโปรแกรมได้ที่

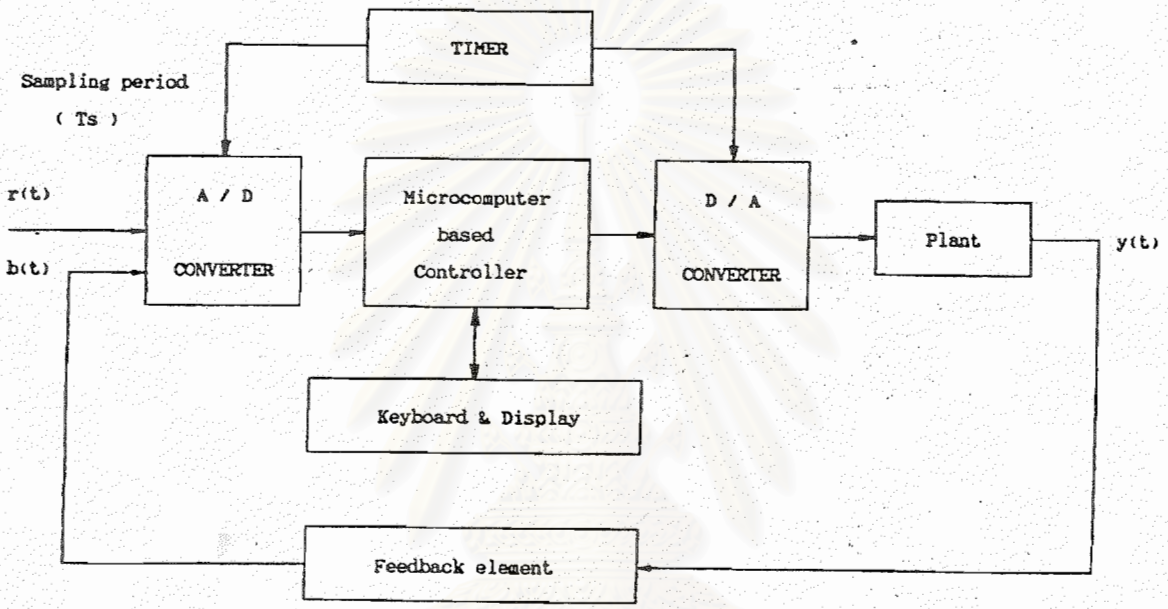
จะทำการใช้งานจะต้องมีขีดความสามารถและประสิทธิภาพที่เทียบเท่าหรือดีกว่า ตัวควบคุมเชิงเลขที่มีขายในท้องตลาดปัจจุบัน โดยมีรายละเอียดและข้อกำหนดเป้าหมายดังต่อไปนี้

- ชนิดความสามารถในการควบคุม :
  Simple PID Control Cascade Control Feedforward-feedback Control Ratio Control Selective Control Split-range Control
- อื่นๆที่เป็นประโยชน์ต่อการควบคุม :
  การ ลบ คูณ หาร Linearization Square-root Extraction Lead-lag Compensation High & Low Limits.
- Self-diagnostic :
  สามารถตรวจความผิดพลาดของการทำงานของ CPU A/D & D/A Input Overrange Arithmatic Operation Overrange Battery Back-up Low
- จำนวนของ Input & Output :
  Analog Inputs : 5 (1-5 V_{dc})
  Digital Inputs : 3
  Analog Outputs : 3 (4-20 mA_{dc} และ 1-5 V_{dc})
  Digital Outputs : 4
- แสดงผลและเปลี่ยนแปลงค่าพารามิเตอร์ที่ใช้ในการควบคุมเช่น SV PB Ti Td เป็นต้นได้
- ความเร็วของการทำงานในแต่ละรอบ (run time/cycle) ต้องน้อยกว่า 0.2 วินาที

3.2 การออกแบบฮาร์ดแวร์

เนื่องจากในงานควบคุมทางอุตสาหกรรม เครื่องควบคุมเชิงเลขจะรับและส่งสัญญาณกับอุปกรณ์อื่นๆเป็นสัญญาณอนาลอก (Analog Signal) แต่เครื่องควบคุมเชิงเลขซึ่งเป็นไมโครคอมพิวเตอร์ทำงานโดยใช้สัญญาณเชิงเลข ดังนั้นจึงต้องเปลี่ยนสัญญาณจากสัญญาณอนาลอกไปเป็นสัญญาณเชิงเลข (Digital Signal) ในขณะที่รับข้อมูลเข้าสู่เครื่อง และแปลงสัญญาณแบบ

เชิงเลขไปเป็นสัญญาณอนาลอคในขณะที่ส่งสัญญาณการควบคุมออกจากเครื่อง การแปลงค่าสัญญาณเหล่านี้สามารถทำได้โดยใช้ไอซี ADC และ DAC ซึ่งจะอยู่ในส่วนฮาร์ดแวร์ที่ใช้ติดต่อ (Interface) ระหว่างเครื่องควบคุมกับสัญญาณภายนอก นอกจากนี้เครื่องควบคุมเชิงเลขจำเป็นต้องมีส่วนที่ใช้ติดต่อกับแป้นกดและส่วนแสดงผล เพื่อช่วยในการแสดงผลและเปลี่ยนแปลงค่าพารามิเตอร์ต่างๆในระบบ การแสดงผลมีหลายชนิด เช่น แบบ Bar Graph, ตัวอักษรและตัวเลข รูปที่ 3.1 แสดงโครงสร้างของเครื่องควบคุมเชิงเลขอย่างคร่าวๆ

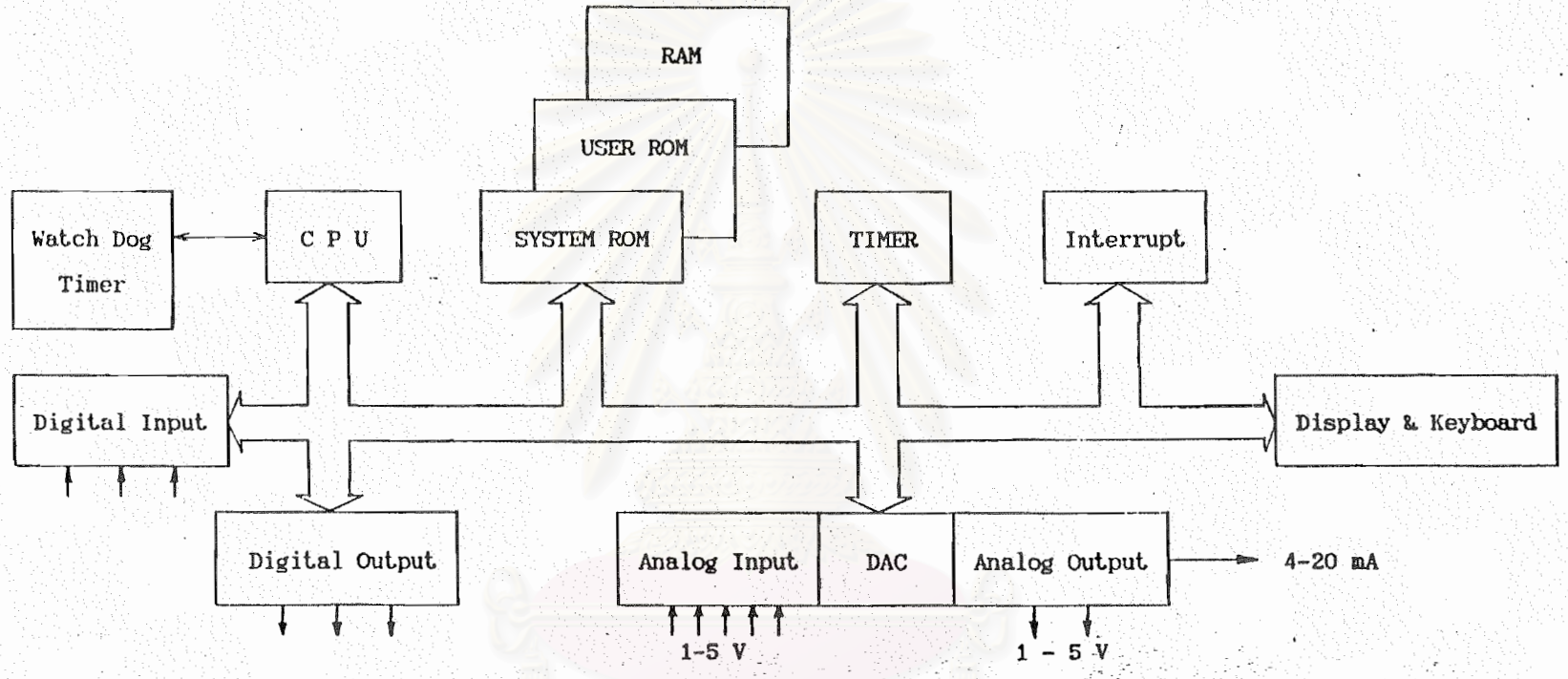


รูปที่ 3.1 โครงสร้างโดยทั่วๆไปของเครื่องควบคุมเชิงเลข

3.2.1 ฮาร์ดแวร์ที่จำเป็นในการสร้างเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

รูปที่ 3.2 แสดงโครงสร้างของฮาร์ดแวร์ที่จำเป็น ซึ่งประกอบด้วยส่วนต่าง ๆ ดังนี้

3.2.1.1 ส่วนประมวลผลกลาง (CPU) เนื่องจากซอฟแวร์ในเครื่องควบคุมจำเป็นต้องมีการคำนวณที่ซับซ้อน ผู้วิจัยเลือกใช้ไอซี Intel 8088 ขนาด 16 บิท ความเร็ว 5 MHz ซึ่งมีคำสั่งที่เหมาะสมกับการคำนวณที่ซับซ้อน และเป็นซีพียูที่ใช้ในเครื่องไมโครคอมพิว



รูปที่ 3.2 โครงสร้างฮาร์ดแวร์เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

เตอร์ ไอบีเอ็ม ซึ่งนิยมใช้กันทั่วไป และมีโปรแกรมซอฟแวร์สำเร็จรูปชื่อ DEBUG ช่วยในการทดสอบอัลกอริทึมของโปรแกรมโดยไม่เกี่ยวข้องกับฮาร์ดแวร์ ทำให้มีความสะดวกในการวิจัย

3.2.1.2 **ส่วนหน่วยความจำ** แบ่งเป็น 2 ชนิด คือ

(1) ROM แบ่งออกเป็น 2 ส่วน คือ

- SYSTEM ROM สำหรับเก็บโปรแกรมจัดการระบบ
- USER EPROM สำหรับเก็บโปรแกรมกำหนดรูปแบบการควบคุม

(2) RAM ใช้เก็บค่าสถานะของการทำงาน, พารามิเตอร์ต่างๆ ที่ใช้ในการควบคุมและรวมทั้งใช้เป็นพื้นที่ใช้งานของโปรแกรม (Working Area)

3.2.1.3 **ส่วน TIMER** สำหรับการนับและงานที่เกี่ยวข้องกับเวลา เช่น ใช้กำหนดเวลาให้กับโปรแกรมฟังก์ชัน TIMER ของโปรแกรมจัดการระบบ และช่วยกำหนดเวลาสำหรับการสุ่ม (Sampling Time) เพื่อรับสัญญาณแบบต่อเนื่องจากภายนอก

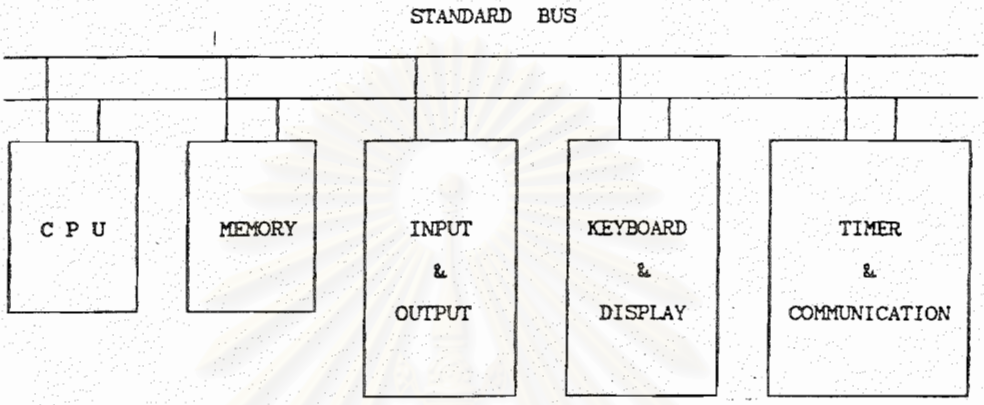
3.2.1.4 **ส่วน Interrupt** สำหรับจัดการเกี่ยวกับการรับสัญญาณอินเตอร์รัพจากฮาร์ดแวร์ ทำงานร่วมกับส่วน TIMER ในการนับเวลา

3.2.1.5 **ส่วนแสดงผลและรับข้อมูลจากแป้นพิมพ์** (Display & Keyboard Interface) สำหรับจัดการเกี่ยวกับการรับข้อมูลเพื่อเปลี่ยนแปลงค่าพารามิเตอร์หรือแสดงผลค่าสถานะของการควบคุม

3.2.1.6 **ส่วนอินพุทและเอาท์พุท** (Input & Output Interface) สำหรับรับและส่งสัญญาณกับโปรเซสภายนอก และระหว่างการรับส่งจะต้องมีการเปลี่ยนแปลงค่าสัญญาณระหว่างสัญญาณแบบต่อเนื่องและสัญญาณแบบไม่ต่อเนื่อง ดังนั้นฮาร์ดแวร์ในส่วนนี้จำเป็นต้องมี DAC (Digital to Analog Converter) และ ADC (Analog to Digital Converter)

3.2.2 โครงสร้างและองค์ประกอบของเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

เมื่อพิจารณาฮาร์ดแวร์ที่จำเป็นในการสร้างเครื่องควบคุมเชิงเลข จากหัวข้อ 3.2.1 สามารถออกแบบและแบ่งฮาร์ดแวร์ที่ใช้ในงานวิจัยออกเป็นหลายบอร์ด ดังรูปที่ 3.3 ในแต่ละบอร์ดมีหน้าที่ดังนี้



รูปที่ 3.3 องค์ประกอบของเครื่องควบคุมเชิงเลข

- CPU Board เป็นส่วนประมวลผลกลางมี CPU Intel 8088, Clock Generator 8284, Programmable Interrupt Control 8259 ทำหน้าที่ควบคุมระบบทั้งหมด และ จัดการเกี่ยวกับการอินเตอร์รัน
- Memory บอร์ดที่ใช้เก็บโปรแกรมจัดการระบบ และเก็บสถานะการทำงานของระบบ ประกอบด้วย ROM 2764 RAM 6264
- TIMER & Communication บอร์ดที่ทำงานเกี่ยวกับเวลาของระบบ รวมทั้งการสื่อสารแบบอนุกรมเพื่อใช้ในการติดต่อกับเครื่องไมโครคอมพิวเตอร์
- Input/Output บอร์ดสำหรับจัดการรับส่งข้อมูลและแปลงค่าสัญญาณระหว่างค่าภายในเครื่องควบคุม กับ ภายนอก ซึ่งมี ADC ช่วยในการแปลงค่าสัญญาณ
- Keyboard & Display บอร์ดสำหรับการแสดงผลและเปลี่ยนแปลงค่าที่จำเป็นในการควบคุม
- EUROCON BUS เป็นบัสมาตรฐานสำหรับเชื่อมโยงบอร์ดต่างๆเข้าด้วยกัน

3.3 แนวความคิดของโปรแกรมจัดการระบบ

แนวความคิดในการเขียนซอฟแวร์ที่ควรจะนำมาพิจารณามีดังนี้

3.3.1 เวลาการสุ่มข้อมูล (Sampling Time) การอ่านข้อมูลของสัญญาณต่อเนื่องภายนอกระบบให้ถูกต้อง จำเป็นต้องมีเวลาการอ่านสุ่มข้อมูลที่เหมาะสม เพราะจะทำให้เกิดปัญหาเรื่องความผิดพลาดของข้อมูลได้ ถ้าเวลาการอ่านสุ่มข้อมูลช้าเกินไป

ค่าเวลาในการสุ่มข้อมูลที่เหมาะสมกับโปรเซสที่มีตัวแปรต่างชนิดกัน จะมีค่าไม่เท่ากัน ดังตารางที่ 3.1 [6]

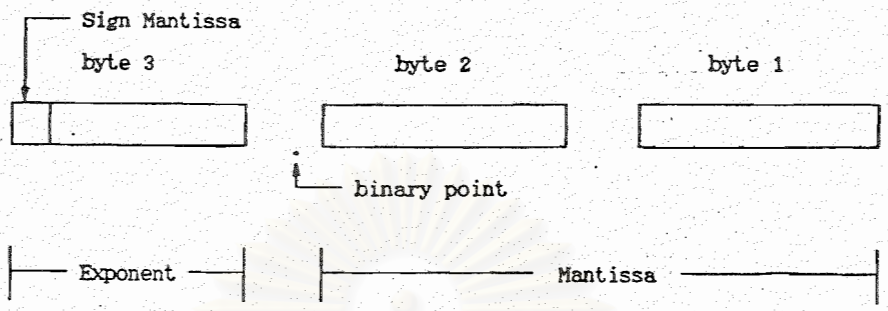
ตารางที่ 3.1

ชนิดของตัวแปร	เวลาการสุ่ม (วินาที)
การไหล	1 - 3
ระดับ	5 - 10
ความดัน	1 - 5
อุณหภูมิ	10 - 20

สำหรับเครื่องควบคุมเชิงเลขในทางการค้าโดยทั่วไป จะใช้เวลาการสุ่มข้อมูลประมาณ 0.2 วินาที

3.3.2 ปัญหาเรื่องชนิดและขนาดของตัวเลข เครื่องควบคุมเชิงเลขมีการคำนวณที่ซับซ้อนและต้องการความถูกต้องสูง จึงจำเป็นต้องมีการคำนวณโดยใช้ตัวเลขแบบมีทศนิยม (floating point number) และเมื่อคำนึงถึงการทำงานของโปรแกรมจัดการระบบที่ทำงานแบบ Real Time Programming ต้องการใช้เวลาในการคำนวณของโปรแกรมหลักและโปรแกรมย่อยต่างๆน้อยที่สุดเพื่อทำให้เวลาในการสุ่มสั้น รวมทั้งการช่วยให้ขนาดหน่วยความจำที่เก็บค่าตัวแปรมีขนาดลดลง จึงจำเป็นต้องมีการเลือกชนิดและขนาดของตัวเลขให้มีค่าเหมาะสม

องค์ประกอบของหน่วยความจำที่ใช้เก็บค่าตัวเลขแบบที่มีทศนิยม (ดูรูปที่ 3.4) ค่าความถูกต้องของตัวเลขที่มีจุดทศนิยมด้วยโครงสร้างนี้ สามารถคำนวณได้จากสมการที่ (3.1)[7]



รูปที่ 3.4 โครงสร้างหน่วยความจำที่ใช้เก็บค่าตัวเลขทศนิยม

$$(m-2)\log 2 \leq d \leq (m-1)\log 2 \quad \ldots\ldots \quad (3.1)$$

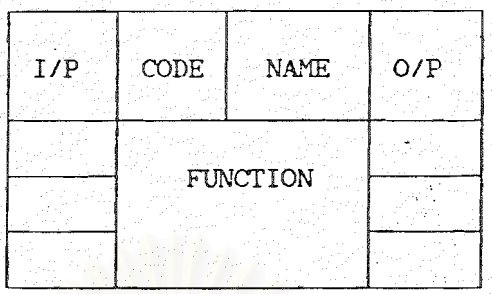
m - จำนวนของบิทในส่วนของ Mantissa

d - จำนวนตัวเลขฐานสิบหลังทศนิยมที่ถูกต้อง

จากโครงสร้างที่ใช้ดังรูป 3.4 มีค่า m = 24, d = 5 ดังนั้นโครงสร้างที่ใช้จะได้ค่าตัวเลขที่ถูกต้องหลังทศนิยมจำนวน 5 ตัว เหมาะสมกับ DAC ขนาด 12 บิท ซึ่งเมื่อใช้แทนค่าจาก 0 ถึง 1 DAC จะให้ความละเอียดของแต่ละบิทเท่ากับ 0.000244

3.3.3 ลักษณะของโปรแกรมกำหนดรูปแบบการควบคุม เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้โดยทั่วไปจะมีส่วนฮาร์ดแวร์ เพื่อใช้ในการเขียนโปรแกรมกำหนดรูปแบบการควบคุม วิธีการโปรแกรมส่วนใหญ่แบ่งได้เป็น 2 วิธี ดังนี้

1. **การโปรแกรมแบบเวเฟอร์** (COMPUTATION WAFER) [3] มีลักษณะของคำสั่งเป็นเวเฟอร์ โครงสร้างแสดงดังรูปที่ 3.4 สำหรับวิธีการโปรแกรมทำได้โดยการนำเวเฟอร์ของคำสั่งต่างๆ มาเรียงต่อกัน



รูปที่ 3.4 โครงสร้างคำสั่งแบบเวเฟอร์

ส่วนต่างๆ ภายในเวเฟอร์ มีความหมายดังนี้

NAME : ชื่อย่อของเวเฟอร์ เช่น RFSUMM หมายถึง บวก

FUNCTION : สัญญลักษณ์/ตัวอักษรของเวเฟอร์ เช่น +,-

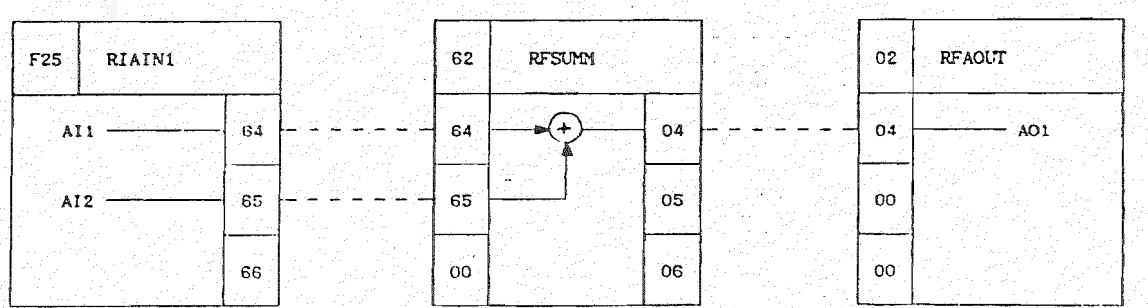
CODE : รหัสแทนคำสั่ง เช่น 62 แทนคำสั่งบวก

O/P : เลขฐานสิบหกของสัญญาณออก กำหนดโดยตัวเวเฟอร์เอง หรือจากตำแหน่งของเวเฟอร์ที่ตั้งอยู่ในทางเดินของสัญญาณ

I/P : เลขฐานสิบหกกำหนดที่มาของสัญญาณ

วิธีการโปรแกรม แสดงได้ดังตัวอย่างการเขียนโปรแกรมบวกสัญญาณ ดังต่อไปนี้

- นำวงจรบวกสัญญาณมาเปรียบเทียบหาเวเฟอร์ที่เหมาะสม มาต่อกันดังนี้



- ป้อนโปรแกรมลงบนเครื่องควบคุม โดยการป้อนค่าตัวเลขฐานสิบหกที่อยู่ในเวเฟอร์ต่างๆ ลงบนเครื่องควบคุมตามตำแหน่งที่กำหนด เช่น เริ่มต้นที่ตำแหน่ง 4640 ลักษณะการป้อนจะเป็นดังนี้

ตำแหน่ง	เลขฐานสิบหก
4640	62
4641	64
4642	65
4643	00
4644	02
4645	04
4646	00
4647	00

2. **การเขียนโปรแกรมแบบแอสแซมบลี้** [2] วิธีนี้คำสั่งที่ใช้ในการกำหนดรูปแบบมีลักษณะคล้ายภาษาแอสแซมบลี้ ซึ่งการป้อนโปรแกรมลงบนเครื่องควบคุมจะป้อนคำสั่งเป็นแบบ mnemonic ตัวอย่างการเขียนโปรแกรมควบคุมแบบ PID

```
1. LD   IN1     : อ่านค่าสัญญาณเข้าช่องที่ 1
2. BPID         : โปรแกรม PID
3. ST   OUT1    : ส่งสัญญาณออกช่องที่ 1
4. END          : จบโปรแกรม
```

เมื่อพิจารณาจากวิธีการเขียนโปรแกรมทั้งสองวิธีแล้ว จะเห็นว่าการเขียนโปรแกรมในแบบเวเฟอร์นั้นมีข้อเสียที่ไม่สะดวกในการใช้งาน คือ การป้อนโปรแกรมจะใช้วิธีการป้อนโปรแกรมเป็นเลขฐานสิบหก ซึ่งทำให้ยากในการอ่านและตรวจสอบโปรแกรมที่อยู่บนเครื่องควบคุม จำเป็นต้องมีรูปภาพการต่อของเวเฟอร์เพื่อช่วยในการอ่านและตรวจสอบ แต่วิธีเขียนแบบแอสแซมบลี้จะป้อนโปรแกรมแบบ mnemonic ซึ่งทำให้ง่ายต่อการอ่านและเข้าใจโปรแกรม

สำหรับการป้อนโปรแกรมของตัวควบคุมเชิงเลขชนิดโปรแกรมได้ โดยทั่วไปนั้นตัวควบคุมจะมีฮาร์ดแวร์ที่ช่วยในการป้อนโปรแกรม (Programming Unit) ติดมาด้วย ซึ่งฮาร์ดแวร์ส่วนนี้จะใช้ป้อนโปรแกรมและแสดงผลทีละคำสั่ง ทำให้ไม่สะดวกในการอ่านโปรแกรมทั้งหมด ดังนั้นในการวิจัยนี้จึงมีแนวความคิดที่จะให้มีการป้อนโปรแกรมบนไมโครคอมพิวเตอร์ ซึ่งทำให้สา

มารถแสดงผลโปรแกรมได้ทีละหลายบรรทัด และยังช่วยลดจำนวนฮาร์ดแวร์ที่ใช้ในการป้อนโปรแกรม การที่เลือกใช้ไมโครคอมพิวเตอร์ เพราะในปัจจุบันไมโครคอมพิวเตอร์นั้นจะมีใช้กันทั่วไป และในเครื่องควบคุมเชิงเลขโดยทั่วไปแล้ว จะมีวงจรทางการสื่อสารกับไมโครคอมพิวเตอร์อยู่แล้ว ช่วยให้สามารถส่งโปรแกรมจากคอมพิวเตอร์ไปยังเครื่องควบคุมเพื่อทดสอบโปรแกรมได้ เมื่อทดลองจนแน่ใจแล้ว จึงจะทำการเขียนโปรแกรมลงบน USER ROM เพื่อใช้เป็นโปรแกรมกำหนดรูปแบบการควบคุมจริง

สำหรับเครื่องควบคุมเชิงเลขที่ทำการวิจัยจะเลือกการเขียนโปรแกรมแบบแอสแซมบลี้ และ สำหรับการป้อนโปรแกรมจะป้อนโปรแกรมบนเครื่องไมโครคอมพิวเตอร์

บทที่ 4

ฮาร์ดแวร์ของเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

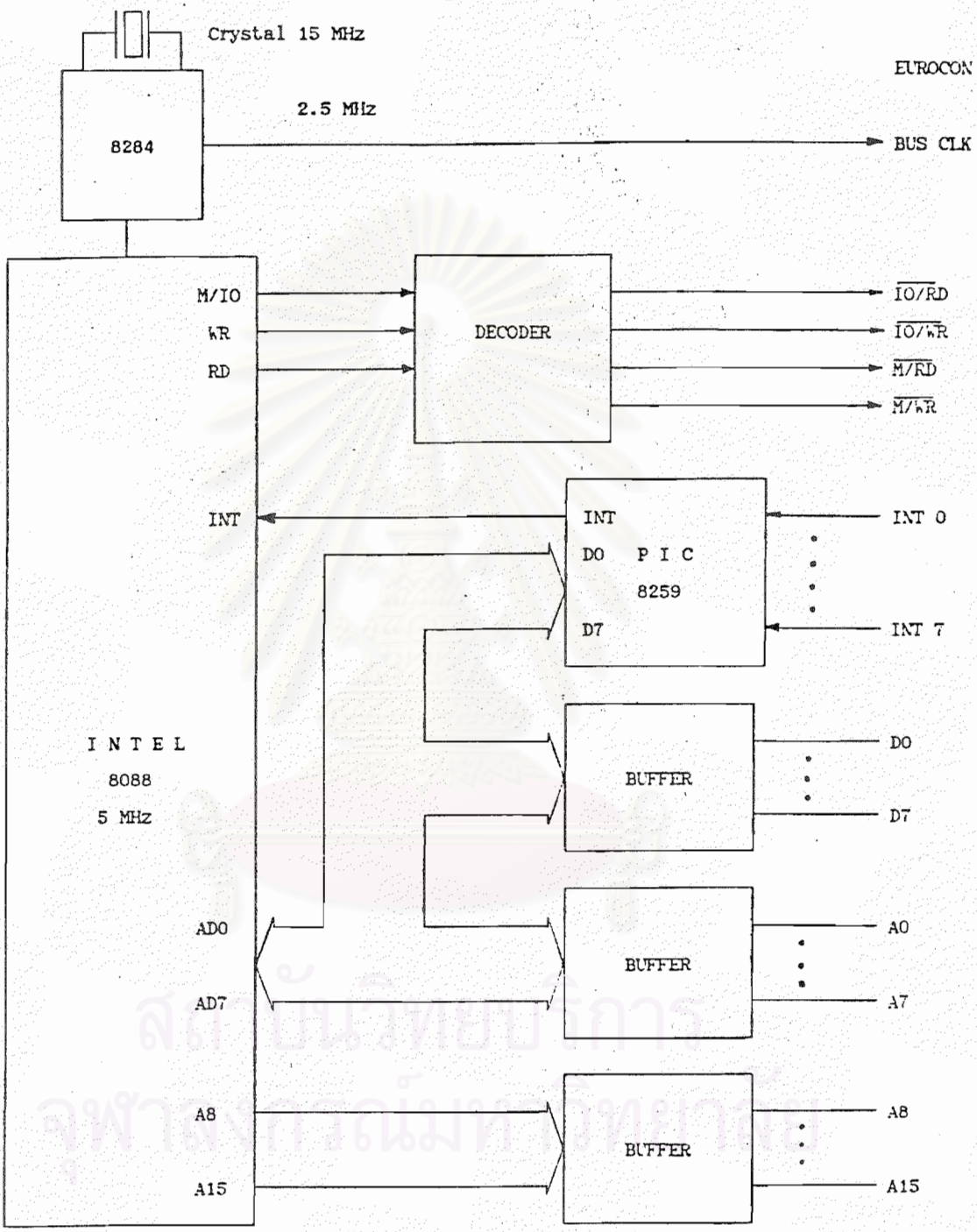
ฮาร์ดแวร์ของเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ สามารถแบ่งเป็นส่วนใหญ่ๆ ได้ 4 ส่วนคือ ส่วนประมวลผลกลาง, ส่วนหน่วยความจำ, ส่วนตั้งเวลาและการสื่อสารแบบอนุกรม, ส่วนอินพุทและเอาท์พุท ในแต่ละส่วนมีลักษณะเป็นแผ่นวงจรพิมพ์ขนาด 100x160 มิลลิเมตร ยกเว้นส่วนแสดงผลและแป้นพิมพ์ และจะเชื่อมโยงกันผ่านทางบัส EUROCON* ขนาด 64 ขา รายละเอียดในแต่ละส่วนมีดังนี้

4.1 ส่วนประมวลผลกลาง

ส่วนประมวลผลกลางประกอบด้วย (ดูรูปที่ 4.1) CPU Intel 8088 ทำงานด้วยความเร็ว 5 เมกกะเฮิร์ท, Clock Generator 8284 ทำหน้าที่สร้างสัญญาณนาฬิกาให้กับระบบ โดยรับสัญญาณเข้าจากคริสตอลความถี่ 15 เมกกะเฮิร์ท และสร้างสัญญาณนาฬิกาความถี่ 5 เมกกะเฮิร์ท สำหรับจ่ายให้กับซีพียู และความถี่ 2.5 เมกกะเฮิร์ท สำหรับวงจรส่วนต่างๆ ที่ต้องการใช้สัญญาณนาฬิกา เช่น ส่วนตั้งเวลา เป็นต้น, Programmable Interrupt Control 8259 จัดการเกี่ยวกับอินเตอร์รัพจากวงจรภายนอกได้จำนวน 8 จุด (INT 0-7) โดยจะส่ง Interrupt vector ออกมาที่บัสข้อมูลเพื่อให้ซีพียูนำค่าไปคำนวณหาตำแหน่งของหน่วยความจำที่เก็บตำแหน่งของโปรแกรมบริการอินเตอร์รัพ สำหรับวิธีการจัดลำดับการบริการของสัญญาณอินเตอร์รัพทั้ง 8 จุด ขึ้นกับการโปรแกรมกำหนดการทำงานของ 8259 (รายละเอียดจะกล่าวในบทที่ 5) สำหรับสัญญาณการควบคุมที่ใช้ในการอ่านเขียนหน่วยความจำและอุปกรณ์อินพุทเอาท์พุท จะมีตัวถอดรหัส 74LS138 ช่วยในการถอดรหัส โดยนำสัญญาณจากซีพียู คือ IO/M, RD, WR มาถอดรหัสเพื่อแยกสัญญาณที่ใช้ในการควบคุมการอ่านเขียนหน่วยความจำ และอุปกรณ์อินพุทเอาท์พุท (IO/RD, IO/WR, M/RD, M/WR)

---

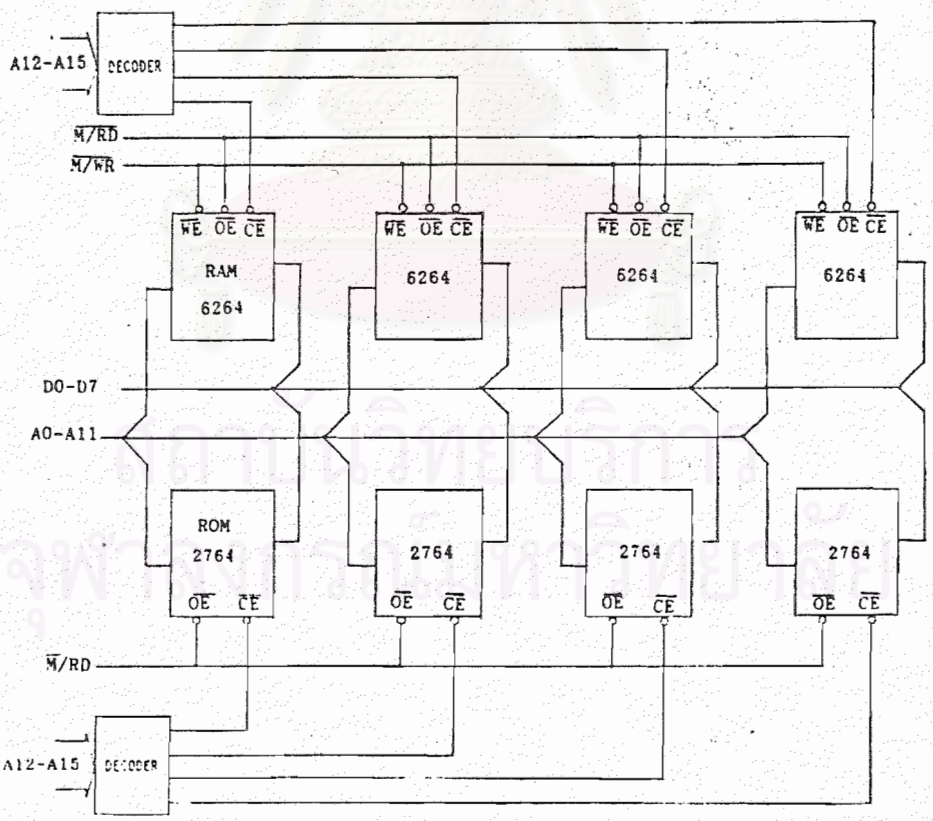
* รายละเอียดดูใน ภาคผนวก ก.



รูปที่ 4.1 บล็อคไดอะแกรมของวงจรส่วนประมวลผลกลาง

4.2 ส่วนหน่วยความจำ

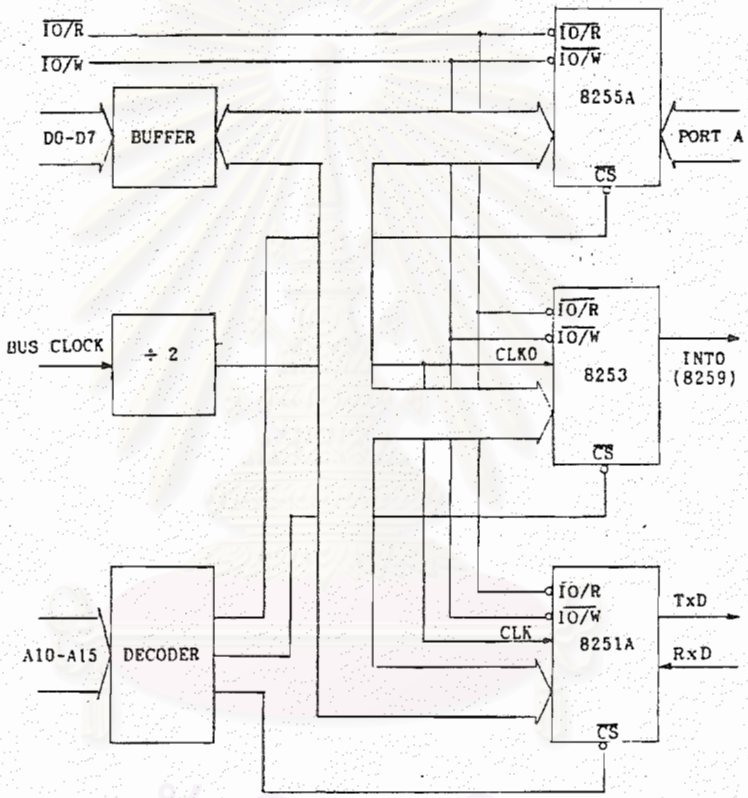
ส่วนหน่วยความจำถูกออกแบบให้มีความจุสูงสุดถึง 64 กิโลไบท์ โดยแบ่งเป็น 2 ส่วน คือ หน่วยความจำประเภท RAM ใช้ 6264 ที่ตำแหน่ง 0000H-7999H และหน่วยความจำประเภท ROM ใช้ 2764 ที่ตำแหน่ง 8000H-FFFFH สาเหตุของการจัดตำแหน่งของ ROM ที่ตำแหน่งสูงและ RAM ที่ตำแหน่งต่ำ เพราะลักษณะการทำงานของ ซีพียู Intel 8088 เมื่อรับสัญญาณ RESET จะอ่านคำสั่งที่ตำแหน่ง FFF0H เพื่อทำงานโดยหน่วยความจำจากตำแหน่ง FFF0H ถึง FFFFH จะเก็บคำสั่งการกำหนดค่าเริ่มต้นและคำสั่งกระโดดไปยังจุดเริ่มต้นของโปรแกรมจัดการระบบ สำหรับหน่วยความจำ RAM ใน 1 กิโลไบท์แรก จะถูกจองไว้สำหรับเก็บค่าตัวชี้ตำแหน่งของโปรแกรมบริการอินเตอร์รัพท์ชนิดต่างๆ โดยการอินเตอร์รันมี 256 ชนิด แต่ละค่าตัวชี้ใช้หน่วยความจำขนาด 4 ไบท์ โดยเก็บค่าของ Segment และ Offset ที่ใช้ชี้ตำแหน่งโปรแกรม สำหรับการถอดรหัสเพื่อเลือกตำแหน่งของหน่วยความจำ จะใช้ 74LS138 บล็อคไดอะแกรมของวงจรส่วนหน่วยความจำแสดงดังรูปที่ 4.2



รูปที่ 4.2 บล็อคไดอะแกรมของวงจรส่วนหน่วยความจำ

4.3 ส่วนตั้งเวลาและการสื่อสาร

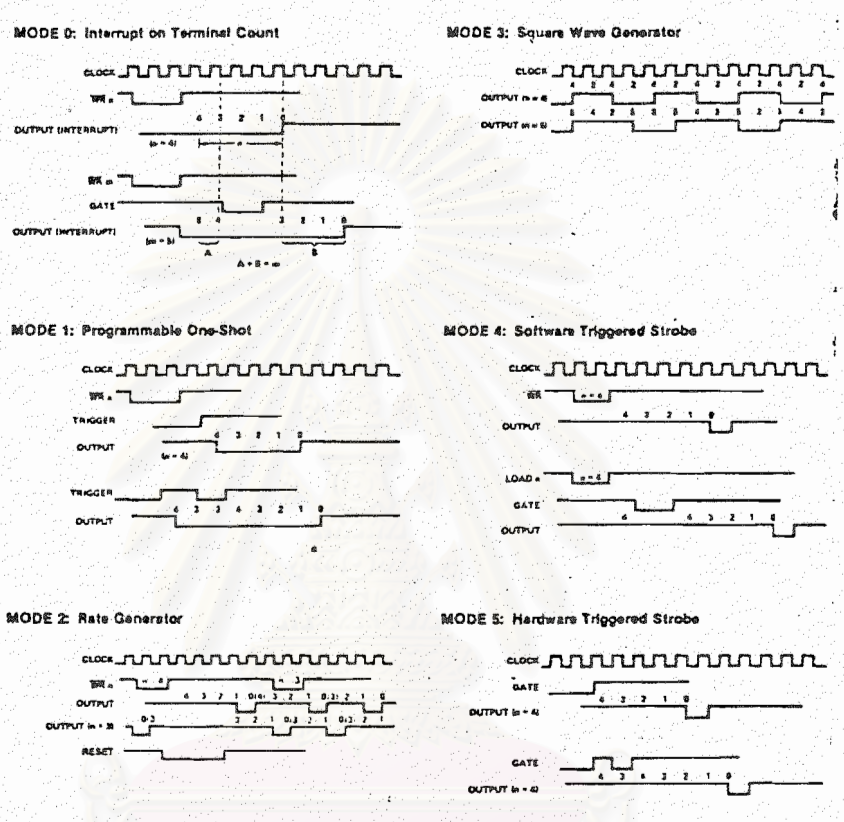
ส่วนตั้งเวลาและการสื่อสารแบบอนุกรม ประกอบด้วยไอซีหลัก 3 ตัว คือ 8253 (Programmable Interval Timer), 8255A (Programmable Peripheral Interface), 8251A USART บล็อคไดอะแกรมแสดงดังรูปที่ 4.3 รายละเอียดการทำงานของแต่ละไอซีมีดังนี้



รูปที่ 4.3 บล็อคไดอะแกรมส่วนตั้งเวลาและการสื่อสาร

(1) 8253 โครงสร้างภายในประกอบด้วยตัวนับขนาด 16 บิท ที่โปรแกรมการทำงานได้ 3 ตัว ใช้กับสัญญาณอินพุทที่ความถี่ไม่เกิน 2.6 MHz (ขนาดความถี่ไม่เกิน 2 MHz สำหรับ 8253-5) การทำงานของ 8253 มีได้หลาย mode แสดงได้ ดังรูปที่ 4.4 นำมาใช้งานร่วมกับ PIC 8259 ในการตั้งเวลาการสุ่มข้อมูล โดยใช้ตัวนับตัวที่ 0 ของ 8253 ทำงานใน mode 3 (สร้างสัญญาณจัตุรัส) รับสัญญาณความถี่ 1.25 MHz ซึ่งมีค่าเป็นครึ่งหนึ่งของความถี่ของ bus clock บนบัส EUROCON เพื่อสร้างสัญญาณนาฬิกาซึ่งมีค่าคาบเวลาเท่ากับ 50 ms.

ส่งไปอินเตอร์รัพท์ซีพียู นอกจากนั้นสัญญาณนี้จะใช้กับฟังก์ชัน TIMER สำหรับตัวตัวนับอีก 2 ตัว ถูกจองไว้สำหรับวงจรส่วนต่างๆ ของระบบ



รูปที่ 4.4 โหมดการทำงานแบบต่างๆของ 8253

(2) 8255A โครงสร้างภายในประกอบด้วยพอร์ทแบบขนาน (Parallel Port) 3 ชุด ที่สามารถโปรแกรมเลือกแต่ละพอร์ทให้เป็นอินพุทหรือเอาท์พุทได้ ในวงจรนี้ใช้พอร์ท A เพียงพอร์ทเดียวทำหน้าที่เป็นอินพุทเก็บสถานะของการทำงาน เช่น ค่าการเปรียบเทียบสัญญาณจาก DAC กับสัญญาณอินพุทแบบต่อเนื่อง ใช้ร่วมกับโปรแกรมเพื่อหาค่าเชิงเลขที่ถูกต้องของสัญญาณอิเนุท, เก็บสถานะสวิทช์ที่ให้เลือกรูปแบบในการคำนวณของโปรแกรม PID เป็นแบบ Direct หรือ Inverse, เก็บสถานะการเลือกอ่านโปรแกรมกำหนดรูปแบบบน ROM หรือ RAM เป็นต้น ความหมายของแต่ละบิทในพอร์ท A ของ 8255A แสดงดังตารางที่ 4.1

ตารางที่ 4.1

บิทที่	ความหมาย
0	ผลการเปรียบเทียบสัญญาณระหว่าง DAC กับ สัญญาณเข้า
1	กำหนด PID ลูปแรก แบบ DIRECT หรือ REVERSE
2	กำหนด PID ลูปสอง แบบ DIRECT หรือ REVERSE
3	กำหนดอ่านโปรแกรมบน ROM หรือ RAM
4-7	ไม่ถูกใช้งาน

(3) 8251A ใช้สื่อสารกับไมโครคอมพิวเตอร์ ส่วนนี้ไม่อยู่ในขอบเขตของวิทยานิพนธ์ แต่ใส่ไว้เพื่อขยายความสามารถของตัวควบคุมเชิงเลขในอนาคต

ตำแหน่งและหน้าที่ของพอร์ทบนบอร์ด แสดงได้ดังตารางที่ 4.2 ดังนี้

ตารางที่ 4.2

ตำแหน่งพอร์ท	หน้าที่พอร์ท
C000	อ่านเขียนข้อมูลบน 8251A
C001	อ่านสถานะหรือเขียนคำสั่งควบคุมบน 8251A
C800	อ่านเขียนข้อมูลบน TIMER 0 ของ 8253
C801	อ่านเขียนข้อมูลบน TIMER 1 ของ 8253
C802	อ่านเขียนข้อมูลบน TIMER 2 ของ 8253
C803	อ่านสถานะหรือเขียนคำสั่งควบคุมบน 8253
CC00	อ่านหรือเขียนข้อมูลที่พอร์ท A ของ 8255A
CC01	อ่านหรือเขียนข้อมูลที่พอร์ท B ของ 8255A
CC02	อ่านหรือเขียนข้อมูลที่พอร์ท C ของ 8255A
CC03	เขียนคำสั่งควบคุมการทำงานของ 8255A

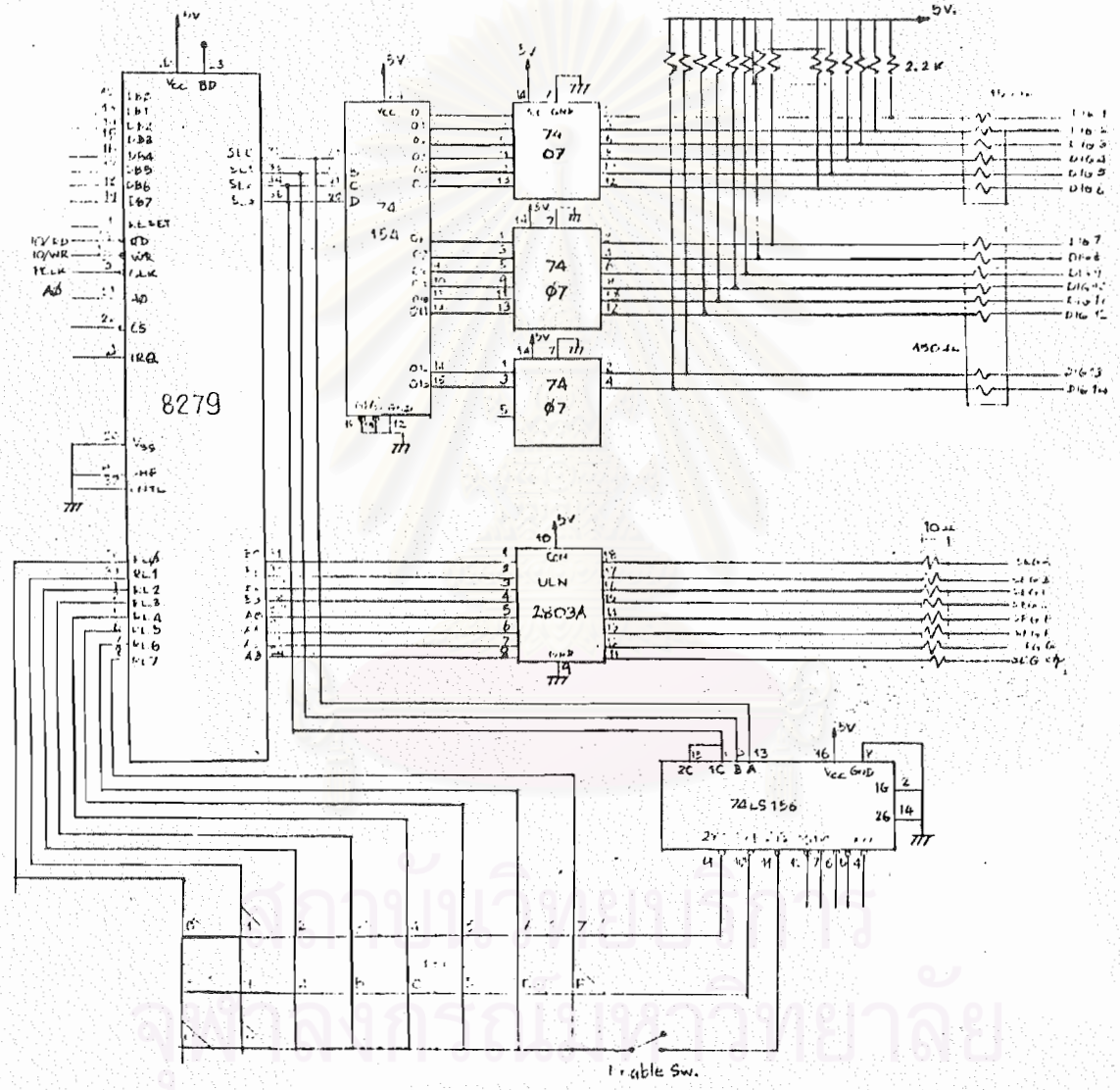
4.4 ส่วนแสดงผลและแป้นพิมพ์ (Display & Keyboard Interface)

ส่วนแสดงผลและแป้นพิมพ์ออกแบบไว้เพื่อการแสดงผล และการเปลี่ยนแปลงค่าต่างๆ โดยจำแนกการทำงานเป็น 2 ส่วนแบ่งเป็น 2 ส่วน คือ ส่วนแรกจัดการเกี่ยวกับ ค่าตัวแปรโปรเซส ตัวแปรควบคุม ค่าเป้าหมาย และโหมดการทำงานของระบบ โดยแสดงค่าทั้งแบบ Bar Graph และแบบ LED 7 ส่วน ส่วนที่สองจัดการเกี่ยวกับ ค่าพารามิเตอร์ต่างๆ ที่ใช้ในการควบคุม เช่น ค่า Proportional Band, Integral Time, Derivative Time ในส่วนนี้แสดงผลโดยใช้ไอซีที่มีส่วนแสดงผลเป็นแบบ 16 Segments

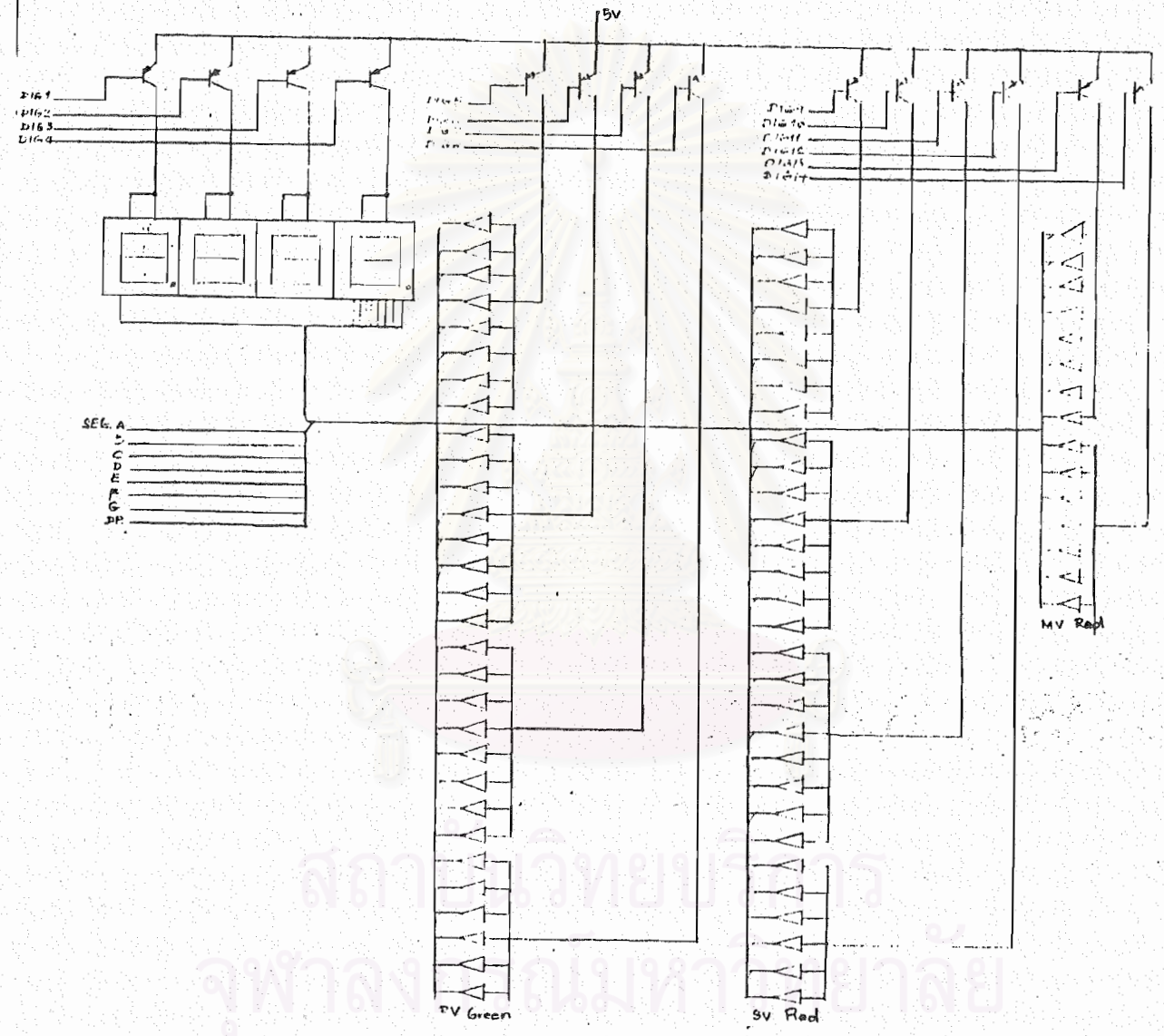
วงจรในรูปที่ 4.5,4.6 และ 4.7 เป็นวงจรของส่วนแสดงผลและแป้นพิมพ์ ใช้ไอซี 8279 Programmable Keyboard/Display Interface ทำหน้าที่สแกนอ่านค่าคีย์บอร์ดจำนวน 14 ตัว และสแกนส่วนแสดงผล 16 ตัว แบ่งออกเป็น 2 ส่วน คือ ส่วนแสดงผลด้วย LED 7 ส่วน สำหรับ 4 ตัวแรก เพื่อแสดงค่าตัวแปรของระบบเป็นตัวเลข สำหรับอีกส่วนแสดงผลเป็น Bar Graph โดยใช้ตัวที่ 5-8 สำหรับแสดงผลค่าตัวแปรโปรเซส, ตัวที่ 9-12 แสดงค่าเป้าหมาย และตัวที่ 13-14 แสดงค่าตัวแปรควบคุมระบบ

การทำงานของส่วนสแกนค่าแป้นพิมพ์ (ดูรูปที่ 4.5) เริ่มจาก 8279 ส่งสัญญาณจาก SL0-SL3 ผ่านตัวถอดรหัส 2 ออก 4 ไอซีเบอร์ 74LS156 เพื่อเลือกแถวของแป้นพิมพ์ที่ต้องการสแกนอ่าน ตัวแป้นพิมพ์ต่ออยู่ระหว่างขา SL0-SL3 กับ RL0-RL7 ของ 8279 เมื่อแป้นพิมพ์ถูกกด 8279 จะอ่านค่าลอจิคของ RL0-RL7 ถ้ามีค่า "0" 8279 จะคอยเพื่อ debounce 10.3 ms. ถ้ายังคงมีค่า "0" 8279 จะทำการแปลงรหัสเขียนลงบน RAM ส่วนคีย์บอร์ด

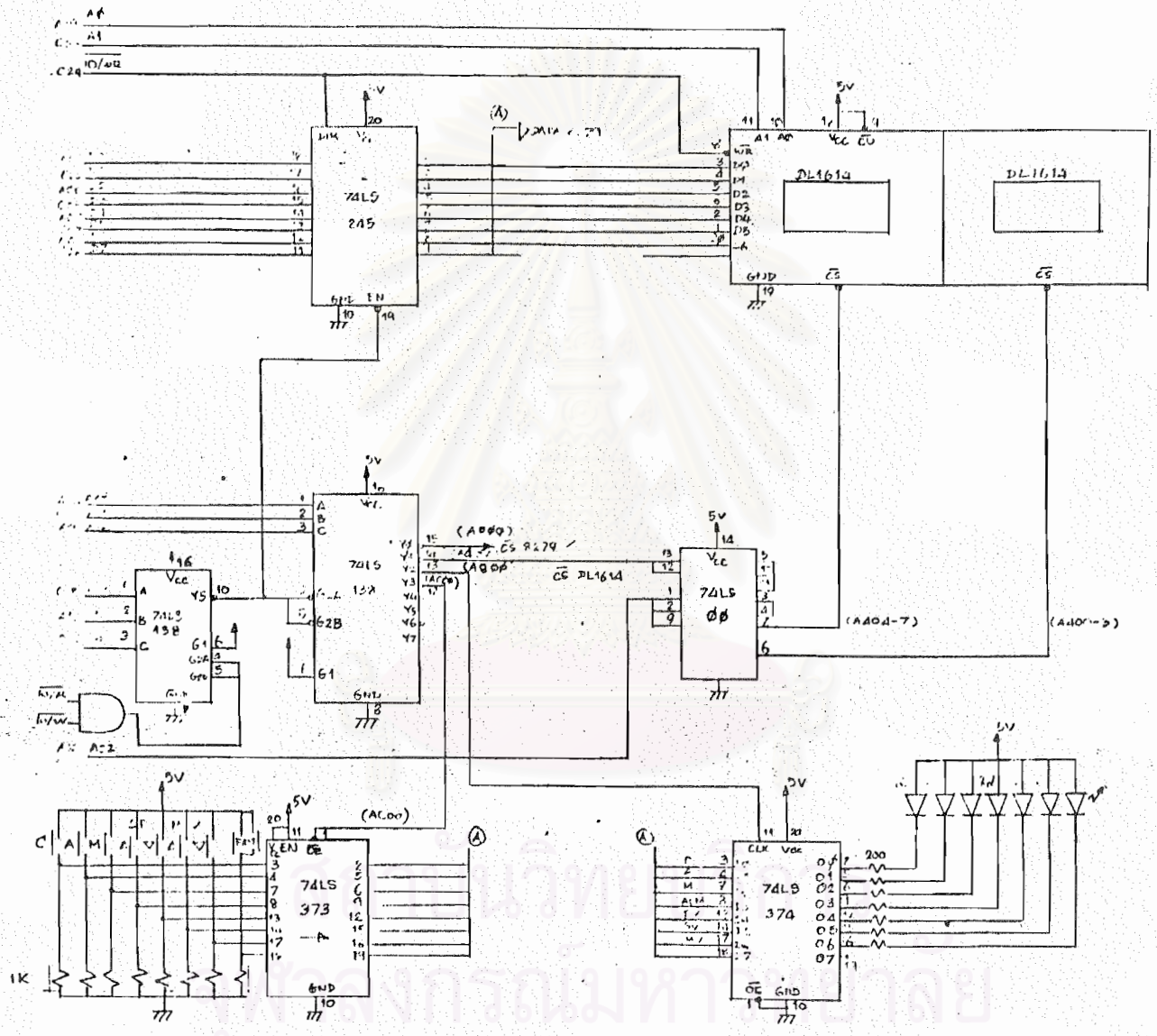
การทำงานของส่วนแสดงผลมีรายละเอียดสำคัญดังต่อไปนี้ จากรูปที่ 4.6 ส่วนแสดงผลทุกตัวเป็นแบบอาโนดร่วม (common-anode) โดยแต่ละตัวมีทรานซิสเตอร์สวิทช์ PNP ต่ออยู่ระหว่างขั้วอาโนดกับ 5 โวลท์ และในส่วนของ LED และ Bar Graph ในแต่ละ Segment จะต่อกับ บัส Segment เดียวกัน เพราะการแสดงผลของ 8279 เป็นแบบมัลติเพล็ก ซึ่งในการแสดงผลแต่ละครั้งจะมีเพียงตัวเดียวที่ถูกเลือกให้แสดงผล โดยรับสัญญาณลอจิก "0" เพื่อใช้ขับทรานซิสเตอร์เลือกตัวแสดงผล มาจากตัวถอดรหัส 4 ออก 16 ไอซีเบอร์ 74154 อินพุทของ 74154 มาจากขา SL0-SL3 ของ 8279 โดย 8279 จะให้สัญญาณที่ขา SL0-SL3 วนเป็นรอบจาก 0000 ถึง 1111 โดยเพิ่มค่าทีละหนึ่ง ทำให้ 74154 สร้างสัญญาณเลือกตัวแสดงผลทีละตัวจนครบ บัสของ Segment ส่วนแสดงผล (ดูรูปที่ 4.6) ต่อกับขา A3-A0 และ B3-B0 ของ 8279 การทำงานจะเริ่มจาก 8279 ส่งข้อไปเลือก Segment ที่ต้องการให้สว่างบนตัว



รูปที่ 4.5 วงจรส่วนแสดงผลและแป้นพิมพ์



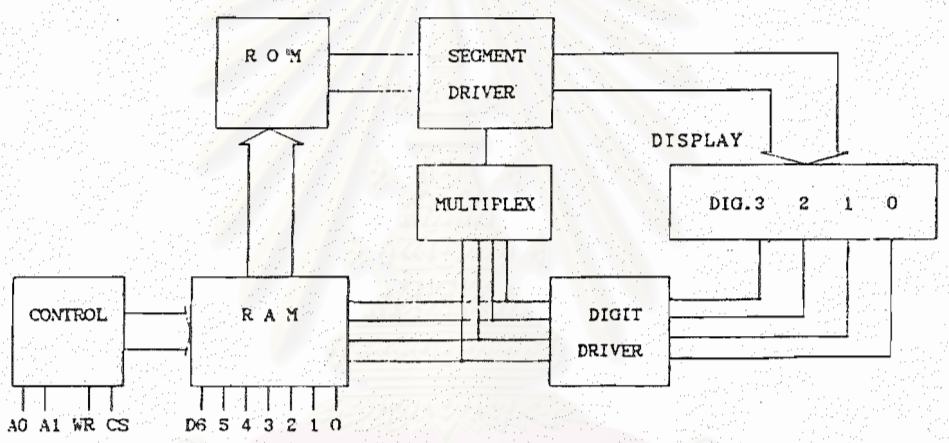
รูปที่ 4.6 วงจรส่วนแสดงผลและแป้นพิมพ์ (ต่อ)



รูปที่ 4.7 วงจรส่วนแสดงผลและแป้นพิมพ์ (ต่อ)

แสดงผลที่ถูกเลือก โดยข้อมูลถูกส่งผ่านทางบัฟเฟอร์ ULN2803A ข้อมูลที่ส่งนี้ถูกนำมาจากหน่วยความจำ RAM ส่วนแสดงผลจำนวน 16 ไบท์ ภายใน 8279

สำหรับส่วนแสดงผล 16 Segments ใช้ไอซีเบอร์ DL1416 (4 Digit LED Intelligent Displays) โครงสร้างภายในแสดงได้ดังรูปที่ 4.8 ทำงานโดยรับสัญญาณควบคุมจาก A0,A1,IO/WR,CS และรับข้อมูลจากบัสข้อมูล เพื่อเขียนข้อมูลลงบน RAM ภายใน DL1416 จากนั้น DL1416 จะนำค่าใน RAM ไปถอดรหัสเพื่อแสดงผลบน LED 16 segments โดยการถอดรหัสเพื่อหาตัวอักษร จะใช้การถอดรหัสแบบ ASCII 7 บิท



รูปที่ 4.8 โครงสร้างภายใน DL1416

วงจรส่วนแสดงผลสถานะการทำงานของระบบ (ดูรูปที่ 4.7) เช่น แสดงโหมดของการควบคุม (AUTO, MANUAL, CASCADE) ใช้ LED ต่อผ่าน Latch 74LS374 สำหรับวงจรแป้นพิมพ์ที่เหลือคือ แป้นพิมพ์กดเลือกโหมดการควบคุม, เปลี่ยนแปลงค่าเป้าหมาย และค่าสัญญาณควบคุมโปรเซส ใช้สวิทช์ต่อผ่าน Latch 74LS373

สำหรับการถอดรหัสหาค่าตำแหน่งพอร์ทต่างๆ บนเบอร์ดนี้ (ดูรูปที่ 4.7) ใช้ตัวถอดรหัส 74LS138 โดยตำแหน่งและหน้าที่ของพอร์ท แสดงได้ดังตารางที่ 4.3

ตารางที่ 4.3

ตำแหน่งพอร์ท	หน้าที่พอร์ท
A000	อ่าน/เขียน 8279 RAM แสดงผล หรือ อ่าน 8279 FIFO
A001	อ่านสถานะของ 8279 หรือ เขียนคำสั่งบน 8279
A400-A407	เขียนข้อมูลบน DL1416 เพื่อแสดงผลจากตำแหน่ง 0-7
A800	เขียนข้อมูลบน 74LS374 เพื่อขับ LED
AC00	อ่านข้อมูลจาก keyboard ผ่าน 74LS373

ฮาร์ดแวร์ของวงจรแสดงผลและแป้นพิมพ์ ประกอบด้วยบอร์ดวงจร 3 บอร์ด คือ

(1) บอร์ดขนาด 100x160 mm. ใช้เชื่อมต่อกับบัส EUROCON เพื่อดึงสัญญาณที่จำเป็นบางส่วนมาใช้กับวงจรแสดงผลและแป้นพิมพ์

(2) บอร์ดแสดงผลด้านข้าง ส่วนแสดงผลเป็นแบบ LED 16 segments

(3) บอร์ดแสดงผลด้านหน้า ส่วนแสดงผลเป็นแบบ LED 7 segments และ bar graph

การเชื่อมต่อสัญญาณระหว่างบอร์ดที่ 1 และ 2 ใช้ connector ขนาด 24 ขา รายละเอียดแสดงได้ ดังรูปที่ 4.9

1	2	3	4	5	6	7	8	9	10	11	12
5 V.	BCLK	IO/R	A0	A2	A11	A13	A15	D1	D3	D5	D7
GND	RES	IO/W	A1	A3	A12	A14	D0	D2	D4	D6	RES
24	23	22	21	20	19	18	17	16	15	14	13

รูปที่ 4.9 รายละเอียดของ Connector ขนาด 24 ขา

BCLK : สัญญาณนาฬิกาความถี่ 2.5 MHz จากบัส EUROCON

IO/R, IO/W : สัญญาณควบคุมการอ่านเขียนอินพุทเอาท์พุท

A0-A15 : Address Bus

D0-D7 : Data Bus

RES : ไม่ได้ใช้งาน

การเชื่อมต่อสัญญาณระหว่างบอร์ดที่ 2 และ 3 ใช้ connector ขนาด 36 ขา รายละเอียดแสดงได้ ดังรูปที่ 4.10

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5 V.	Sel1	RL2	D0	D2	D4	D6	SgA	SgC	SgE	SgG	Dg1	Dg3	Dg5	Dg7	Dg9	Dg11	Dg13
GND	Sel2	RL4	D1	D3	D5	D7	SgB	SgD	SgF	DP	Dg2	Dg4	Dg6	Dg8	Dg10	Dg12	Dg14
36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19

รูปที่ 4.10 รายละเอียดของ Connector ขนาด 36 ขา

Sel 1 : สัญญาณเลือกเอาท์พุทที่ 74LS374

Sel 2 : สัญญาณเลือกอินพุทของ 74LS373

RL2,4 : สัญญาณจากขา RL2,4 ของ 8279

D0-D7 : Data Bus

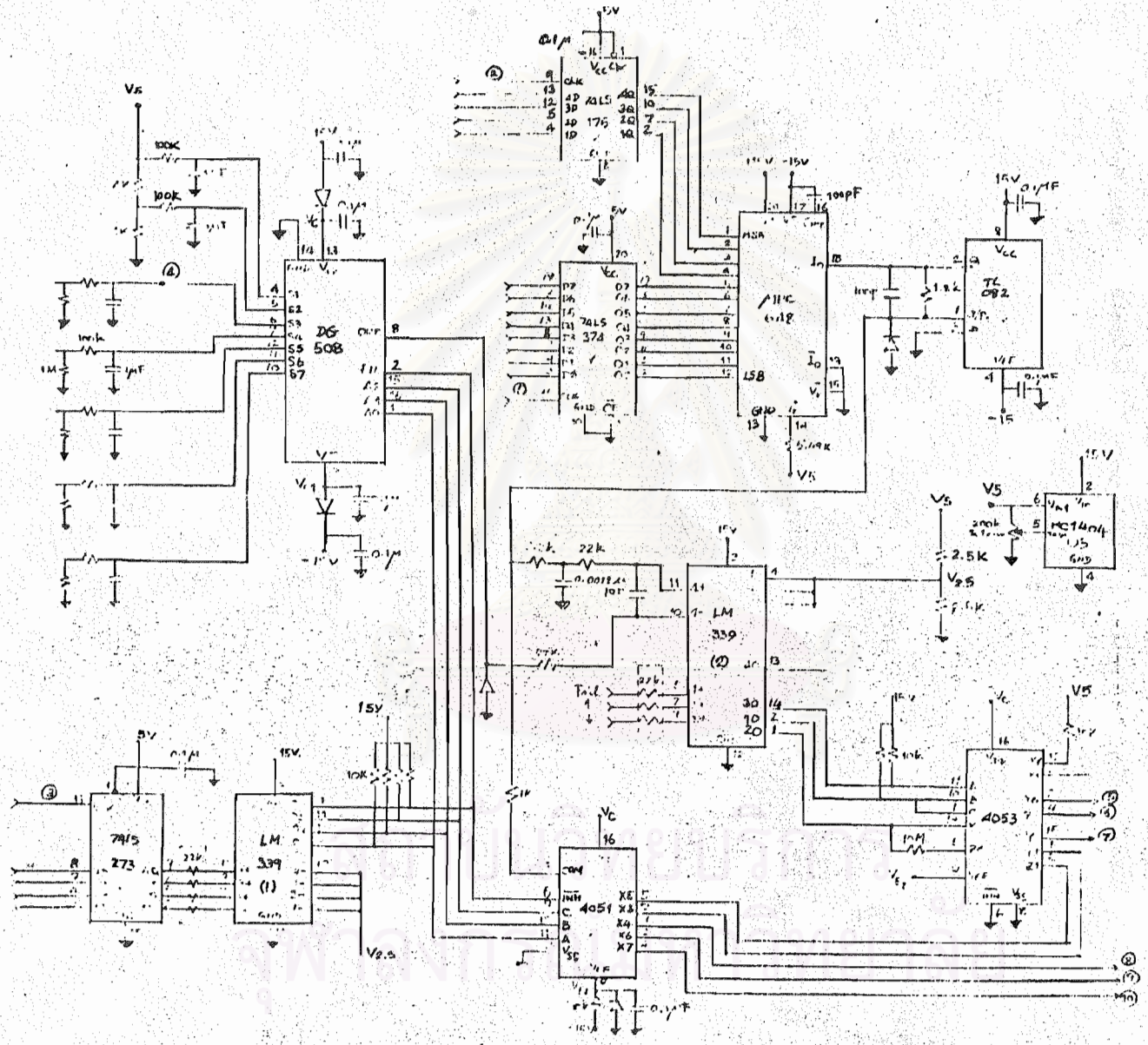
Sg A-F : Segment ของ LED 7 ส่วน

Dg 1-14 : ตำแหน่งของส่วนแสดงผลของ 8279

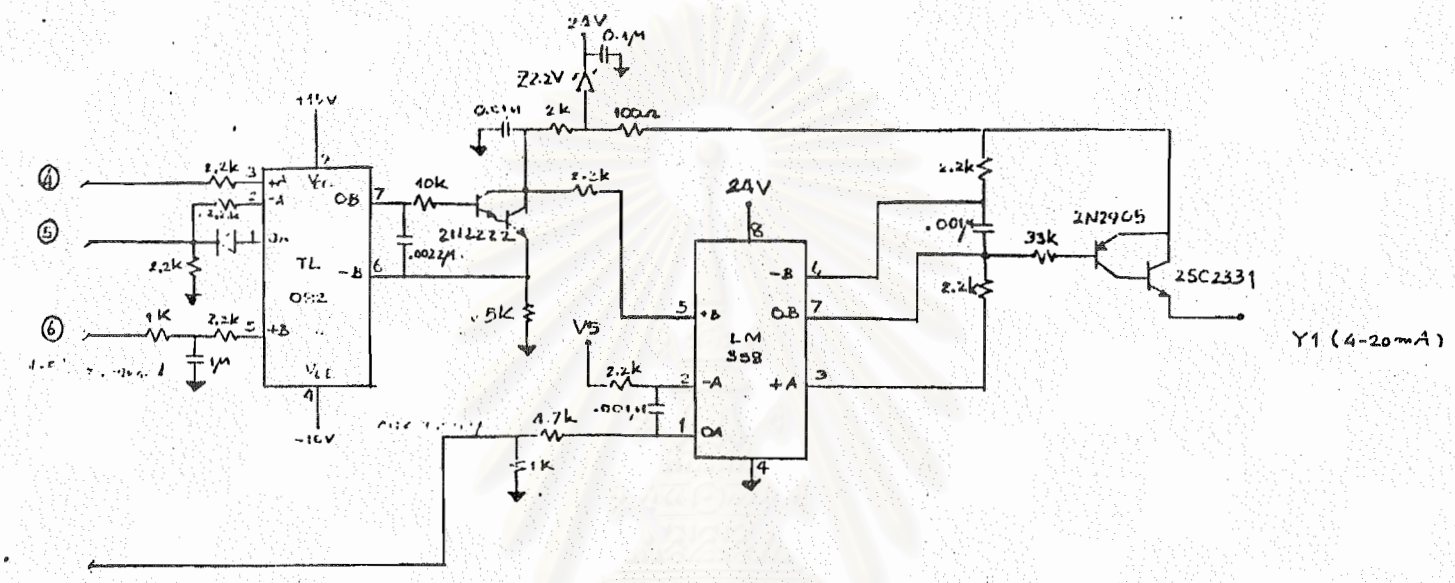
4.5 ส่วนอินพุทและเอาท์พุท

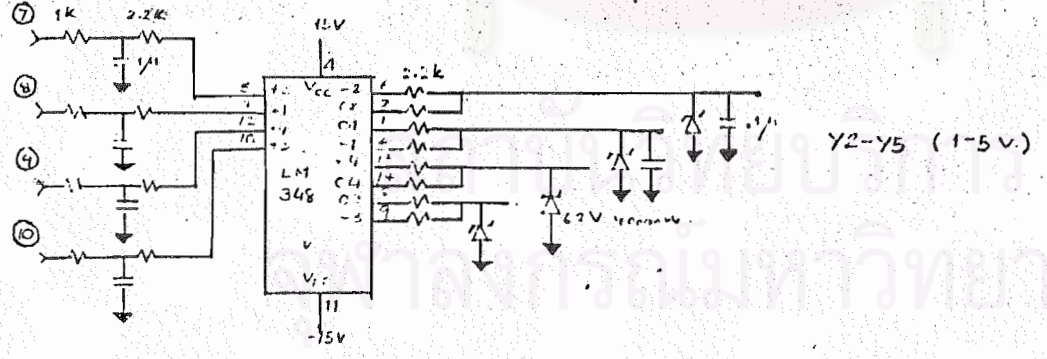
ส่วนอินพุทและเอาท์พุท วงจรแบ่งออกเป็น 2 ส่วน คือ ส่วนติดต่อกับสัญญาณอนาลอค และสัญญาณดิจิตอล

(1) ส่วนสัญญาณอนาลอค ใช้ติดต่อกับสัญญาณอนาลอคภายนอก ซึ่งสัญญาณขาเข้าเป็นสัญญาณแรงดันมาตรฐาน 1-5 V_{dc} สำหรับสัญญาณขาออกเป็นทั้งสัญญาณแรงดัน 1-5 V_{dc} และสัญญาณกระแสมาตรฐาน 4-20 mA_{dc} (ดูรูปที่ 4.11) วงจรประกอบด้วย DG508 8-Channel Analog Multiplexer ทำหน้าที่เลือกสัญญาณเข้าจากภายนอกทั้งหมด 5 ช่องสัญญาณ ส่วนทางด้านเอาท์พุทใช้ไอซี 4051 8-Channel Analog Switch เลือกช่องสัญญาณออก โดยการเลือกช่องสัญญาณของทั้ง DG508 และ 4051 ทำได้โดยการเขียนข้อมูล 4 บิท ผ่าน 74LS273 แบ่งเป็น 3 บิท เพื่อเลือกช่องสัญญาณ และอีก 1 บิท เพื่อเลือก Analog Switch (DG508 และ 4051 จะทำงานไม่พร้อมกัน เนื่องจากสัญญาณ Enable ของ DG508 มีลอจิค "1" ส่วน 4051 มีลอจิค "0") ตารางที่ 4.4 แสดงข้อมูลที่เขียนผ่าน 74LS273 (พอร์ทที่ B400) เพื่อเลือกช่องสัญญาณอินพุทเอาท์พุท

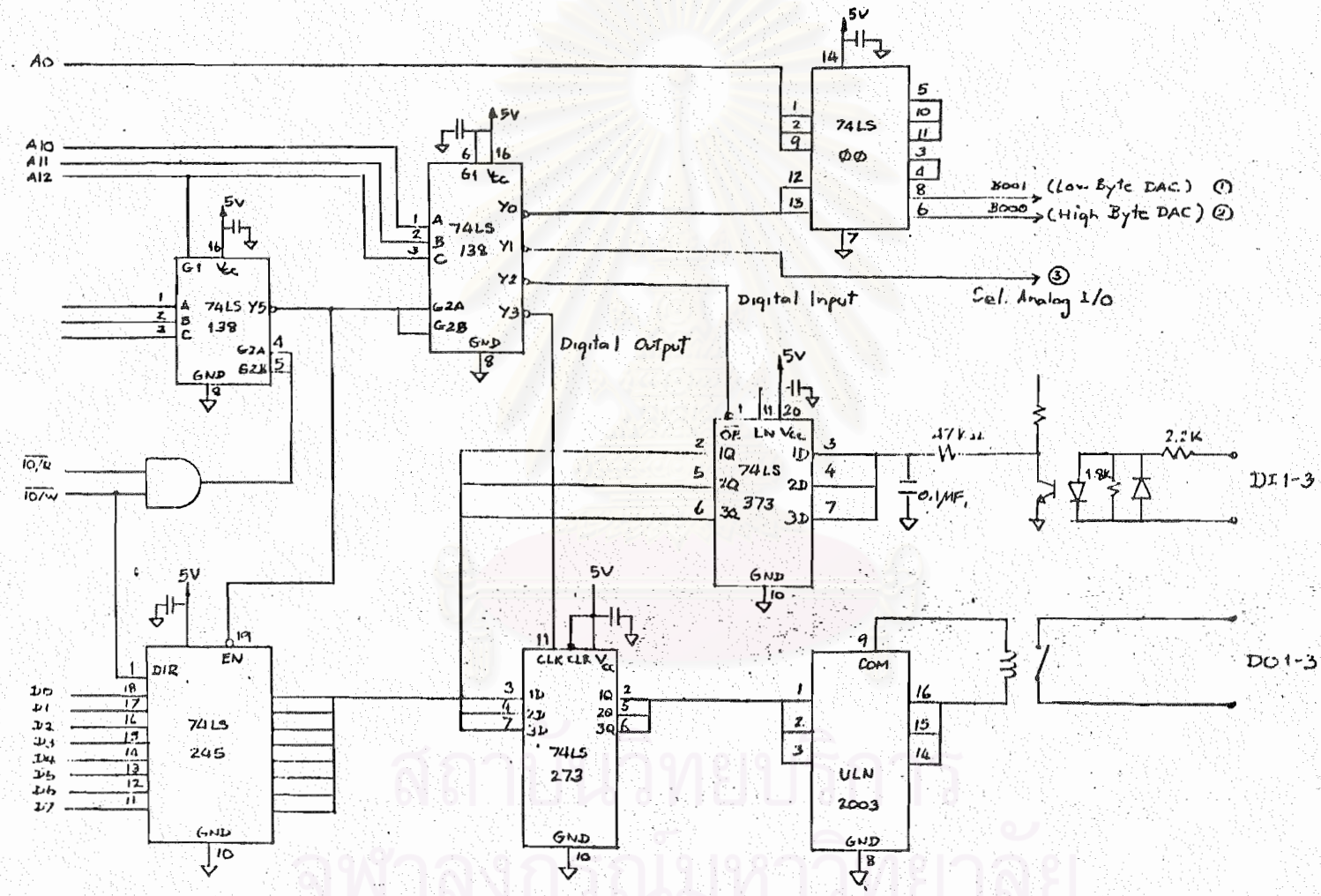


รูปที่ 4.11 วงจรส่วนอินพุทเอาท์พุท





รูปที่ 4.11 วงจรส่วนอินพุทเอาท์พุท (ต่อ)

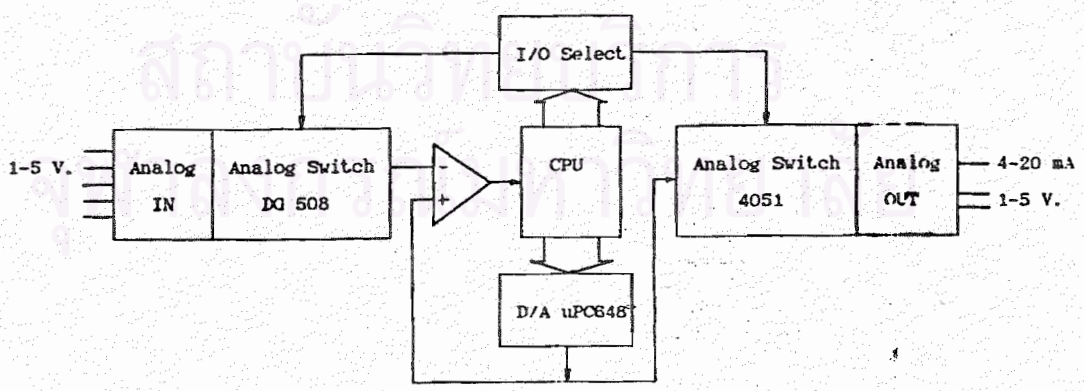


รูปที่ 4.11 วงจรส่วนอินพุทเอาท์พุท (ต่อ)

ตารางที่ 4.4

อินพุท	ข้อมูลพอร์ท B400	เอาท์พุท	ข้อมูลพอร์ท B400
1	0A	1	05
2	0B	2	09
3	06	3	04
4	07	4	0C
5	0E		

การแปลงค่าไปกลับระหว่างสัญญาณอนาลอค กับสัญญาณดิจิตอล (ดูรูปที่ 4.12) ใช้ไอซี uPC648 12 bit DAC การทำงานในกรณีที่เปลี่ยนค่าสัญญาณอนาลอคเป็นดิจิตอล วงจรจะทำงานแบบ Sucessive Approximation ADC คือวงจรจะทำงานร่วมกับโปรแกรม โดยโปรแกรมส่งค่าดิจิตอลมาที่ 74LS374 สำหรับ low byte และที่ 74LS175 สำหรับ high byte ผ่านเข้า DAC ได้สัญญาณอนาลอคมาเปรียบเทียบกับสัญญาณอินพุทจาก DG508 เพื่อหาค่าดิจิตอลที่เหมาะสม สำหรับการทำงานในกรณีที่เปลี่ยนค่าสัญญาณดิจิตอลเป็นอนาลอค วงจรสามารถส่งค่าดิจิตอลผ่าน DAC ออกทาง 4051 ได้ทันที



รูปที่ 4.12 บล็อคไดอะแกรมแสดงการแปลงสัญญาณอนาลอค/ดิจิตอลและดิจิตอล/อนาลอค

การถอดรหัสหาค่าตำแหน่งของพอร์ทต่างๆบนอินพุทเอาท์พุทบอร์ด ใช้ตัวถอดรหัส 74LS138 ตำแหน่งและหน้าที่พอร์ทแสดงได้ดังตารางที่ 4.5

ตารางที่ 4.5

ตำแหน่งพอร์ท	หน้าที่พอร์ท
B000	เขียนข้อมูล D0-D7 ให้ DAC
B001	เขียนข้อมูล D8-D11 ให้ DAC
B400	เขียนข้อมูลเลือกช่องสัญญาณอนาลอคเข้าหรือออก
B800	อ่านข้อมูลสัญญาณเข้าแบบดิจิตอล
BC00	เขียนข้อมูลรีบรีเลย์

(2) ส่วนสัญญาณดิจิตอล (ดูรูปที่ 4.11) แบ่งออกเป็น 2 ส่วน

- อินพุท ใช้อินเตอร์เฟสกับสวิทช์ ปุ่มกด และ sensor ของอุปกรณ์ภายนอก โดยรับสัญญาณผ่าน Opto-couple เบอร์ TLP-521-4

- เอาท์พุท ใช้อินเตอร์เฟสกับอุปกรณ์ภายนอกที่ต้องการควบคุม วงจรประกอบด้วย รีเลย์ทำงานที่ 5 V_{dc} หน้าสัมผัสทนกระแสไฟฟ้าได้ 0.5 A โดยมีบัฟเฟอร์ ULN2003 เป็นตัวขับรีเลย์

บทที่ 5

ซอฟแวร์เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

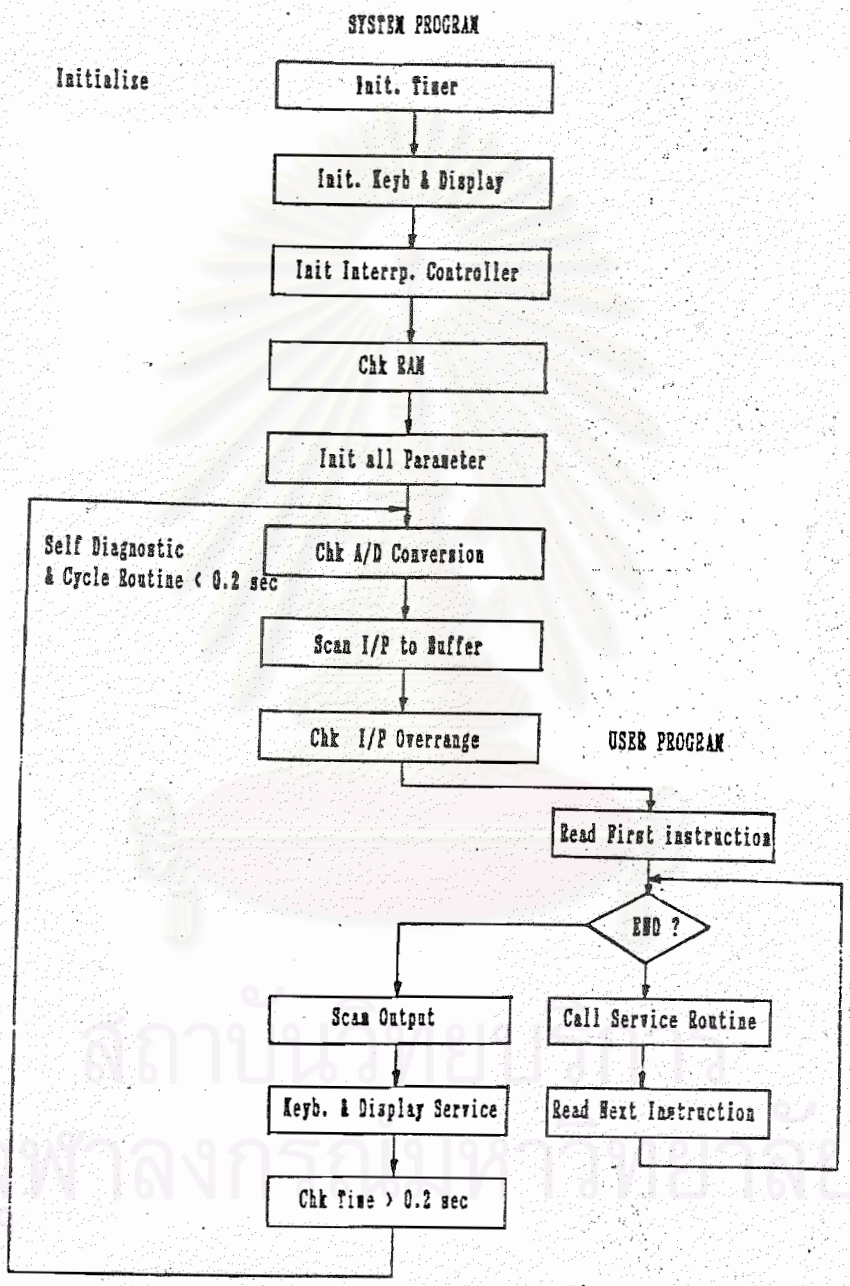
เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ สามารถทำการควบคุมได้หลายรูปแบบ ซึ่งการกำหนดรูปแบบการควบคุมขึ้นกับโปรแกรมที่เขียนโดยผู้ใช้งาน ซึ่งการพิจารณาจากแนวความคิดการออกแบบใน บทที่ 4 ได้ลักษณะของโปรแกรมที่เลือกใช้ในการเขียนเพื่อกำหนดรูปแบบการควบคุม คือ โปรแกรมที่มีลักษณะภาษาแบบ แอสแซมบลี้ สำหรับภาษาที่ใช้ในการพัฒนาซอฟแวร์บนเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ ในการวิจัยนี้ ใช้ภาษา PL/M-86 ซึ่งเป็นภาษาชั้นสูง (High level language) ทำให้สะดวกในการเขียนโปรแกรมโครงสร้าง (Structure Programming) และทำการตรวจสอบและแก้ไขโปรแกรมได้ง่าย แต่ในบางโมดูลของโปรแกรมจำเป็นต้องใช้ภาษาแอสแซมบลี้ ช่วยในการเขียน เพราะในภาษา PL/M-86 ไม่มีคำสั่งที่เข้าถึงระดับรีจีสเตอร์ของซีพียู Intel 8088 เช่น คำสั่งจัดการเกี่ยวกับ Stack (PUSH, POP)

5.1 โปรแกรมควบคุมระบบ

เมื่อพิจารณาจากการทำงานของเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ สามารถเขียนโปรแกรมควบคุมระบบ ซึ่งมีโฟว์ชาร์ทแสดงการทำงานได้ ดังรูปที่ 5.1 โดยแบ่งโปรแกรมตามประเภทการใช้งานเป็น 2 ประเภท คือ

5.1.1 โปรแกรมหลักของระบบ (System Programming) แบ่งเป็น 5 โปรแกรม

(1) โปรแกรมกำหนดการทำงานของฮาร์ดแวร์ (Initialize Hardwares) เพื่อควบคุมรูปแบบการทำงานของไอซี เนื่องจากไอซีบางตัวในเครื่องควบคุมเป็นไอซีประเภทที่มีรูปแบบการทำงานได้หลายรูปแบบขึ้นกับโปรแกรม เช่น 8255A, 8259A, 8253, 8279



รูปที่ 5.1 โฟว์ชาร์ทการทำงานของเครื่องควบคุมเชิงเลข

(2) โปรแกรมกำหนดค่าเริ่มต้นของพารามิเตอร์ เนื่องจากเครื่องควบคุมเชิงเลขมีการทำงานได้หลายโหมด คือ Auto, Manual, Cascade ดังนั้นจึงจำเป็นต้องมีการกำหนดค่าเริ่มต้นของพารามิเตอร์ เพื่อใช้ในการกำหนดโหมดการทำงานเริ่มต้น และ ค่าเริ่มต้นของพารามิเตอร์ที่ใช้ในการคำนวณและการควบคุมที่ถูกแสดงผลและเปลี่ยนแปลงค่าขณะทำงาน

(3) โปรแกรมตรวจสอบความผิดพลาดของระบบ ประกอบด้วยโปรแกรมตรวจสอบส่วนต่างๆ ดังนี้

- การแปลงค่าจากสัญญาณอนาลอคเป็นสัญญาณดิจิตอลของ DAC
- ความผิดพลาดของ RAM
- เวลาใช้ในการคำนวณแต่ละรอบเกิน 0.2 วินาที
- อินพุทเกินพิสัย (Overrange)

(4) โปรแกรมสแกนอินพุทและเอาท์พุท เป็นโปรแกรมที่ใช้อ่านหรือส่งสัญญาณกับอุปกรณ์ภายนอก ทั้งสัญญาณอนาลอคและดิจิตอล

(5) โปรแกรมรับค่าจากแป้นพิมพ์และแสดงผล ทำหน้าที่จัดการแสดงผลหรือเปลี่ยนค่าพารามิเตอร์ที่ใช้ในการคำนวณของระบบ โดยส่วนแสดงผลแบ่งออกเป็น 2 ส่วน คือ

- ส่วนแสดงผลด้านหน้า (Front Panel) แสดงผลตัวแปรที่ใช้ในการควบคุม คือ ตัวแปรโปรเซส, ตัวแปรควบคุม, ค่าเป้าหมาย แสดงผลเป็นตัวเลข และ Bar Graph
- ส่วนแสดงผลด้านข้าง (Side Panel) แสดงผลค่าพารามิเตอร์ที่ใช้ในการคำนวณ เช่น ค่า Proportional Band, Integral Time เป็นต้น แสดงผลแบบตัวอักษรและตัวเลข

(6) โปรแกรมบริการอินเตอร์รัพท์ ใช้ในการนับเพื่อจับเวลาสำหรับการสุ่มอ่านข้อมูลจากสัญญาณภายนอก

5.1.2 โปรแกรมย่อยบริการผู้ใช้ (Service User Program)

โปรแกรมย่อยบริการผู้ใช้ เป็นโปรแกรมคำสั่งใช้ในการคำนวณและการควบคุมที่สร้างขึ้นและเก็บภายในเครื่องควบคุม เพื่อให้ผู้ใช้นำคำสั่งเหล่านี้มาเขียนโปรแกรมกำหนดรูปแบบการควบคุม โปรแกรมย่อยบริการผู้ใช้แบ่งออกเป็น 4 ประเภท ตามลักษณะงาน ดังนี้

(1) โปรแกรมการคำนวณทางคณิตศาสตร์
- บวก, ลบ, คูณ, หาร
- ถอดรากที่สอง (Square Root Extraction)
- ค่าสัมบูรณ์ของเลข (Absolute Value)

(2) โปรแกรมย่อยทางตรรก (Logical Function)
- AND, OR, NOT
- โปรแกรมการกระโดด (Branching)
- การเปรียบเทียบค่า (Comparison)

(3) โปรแกรมฟังก์ชั่นพื้นฐาน
- First Order Lag
- First Order Lead
- Deadtime
- Timer
- High & Low Selector
- Interporation

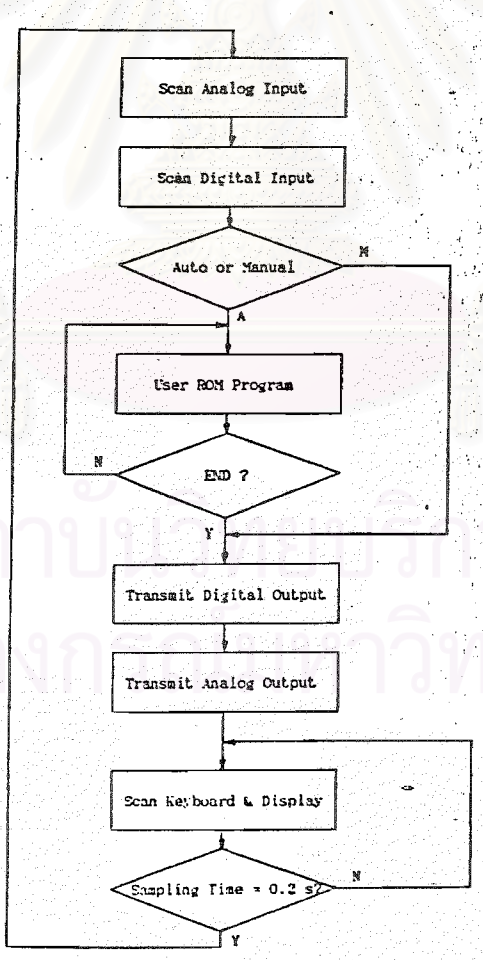
(4) โปรแกรมควบคุมแบบ PID
- Basic PID
- Cascade PID

5.2 การทำงานของโปรแกรมควบคุมระบบ

จากโฟว์ชาร์ทรูปที่ 5.1 แบ่งลักษณะการทำงานของโปรแกรมได้ 2 ลักษณะ คือ

5.2.1 โปรแกรมทำงานแบบครั้งเดียว เป็นโปรแกรมกำหนดการทำงานเริ่มต้นของไอซีประเภทโปรแกรมได้ และกำหนดค่าเริ่มต้นของพารามิเตอร์ ซึ่งโปรแกรมประเภทนี้จะทำงานเพียงครั้งเดียว ภายหลังจาก Power On เท่านั้น

5.2.2 โปรแกรมทำงานแบบวนรอบ (Cycle) เป็นโปรแกรมซึ่งเครื่องควบคุมจะต้องทำงานวนรอบตลอดเวลาเพื่อควบคุมโปรเซส ประกอบด้วย โปรแกรมตรวจสอบความผิดพลาด, สแกนอ่านข้อมูล, โปรแกรมบริการผู้ใช้, สแกนข้อมูลออกเพื่อควบคุม และ แสดงผลข้อมูล การทำงานของโปรแกรมทั้งหมดในแต่ละรอบใช้เวลาไม่เกิน 0.2 วินาที รูปที่ 5.2 แสดงโฟว์ชาร์ทการทำงานแบบวนรอบ

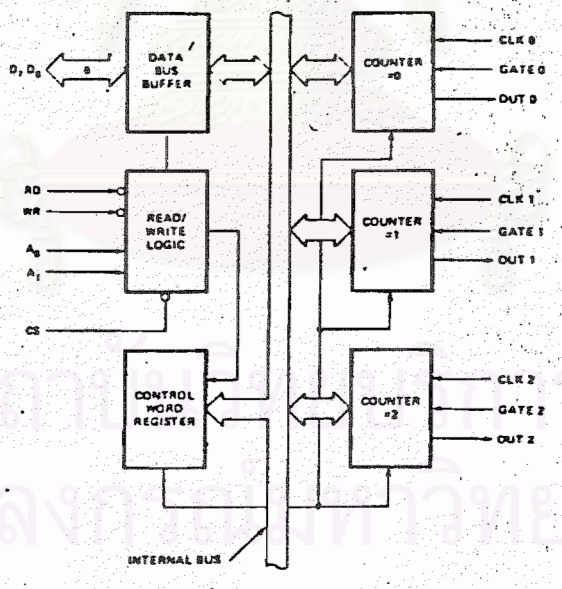


รูปที่ 5.2 โฟว์ชาร์ทการทำงานแบบวนรอบ

5.3 รายละเอียดโปรแกรมหลักของระบบ

5.3.1 โปรแกรมกำหนดการทำงานของฮาร์ดแวร์ ในเครื่องควบคุมมีไอซีประเภทโปรแกรมได้หลายตัว มีรายละเอียดในการโปรแกรมกำหนดการทำงานของไอซีแต่ละตัว ดังนี้

(1) 8253 (Programmable Interval Timer) โครงสร้างภายใน 8253 แสดงได้ดังรูปที่ 5.3 ประกอบด้วยตัวนับขนาด 16 บิท 3 ตัว โดยนับได้ทั้งแบบไบนารีหรือBCD สำหรับการทำงานของ 8253 เลือกได้หลายแบบ (ดังรูปที่ 4.5) ขึ้นกับการโปรแกรมคำสั่งผ่านทางพอร์ทควบคุมของ 8253 (พอร์ทที่ C803) โดยฟอร์แมทของคำสั่งควบคุมและความหมายของแต่ละบิทภายในคำสั่งควบคุมแสดงดังรูปที่ 5.4



รูปที่ 5.3 โครงสร้างภายในของ 8253

Control Word Format

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Definition of Control

SC — Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL — Read/Load:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section)
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

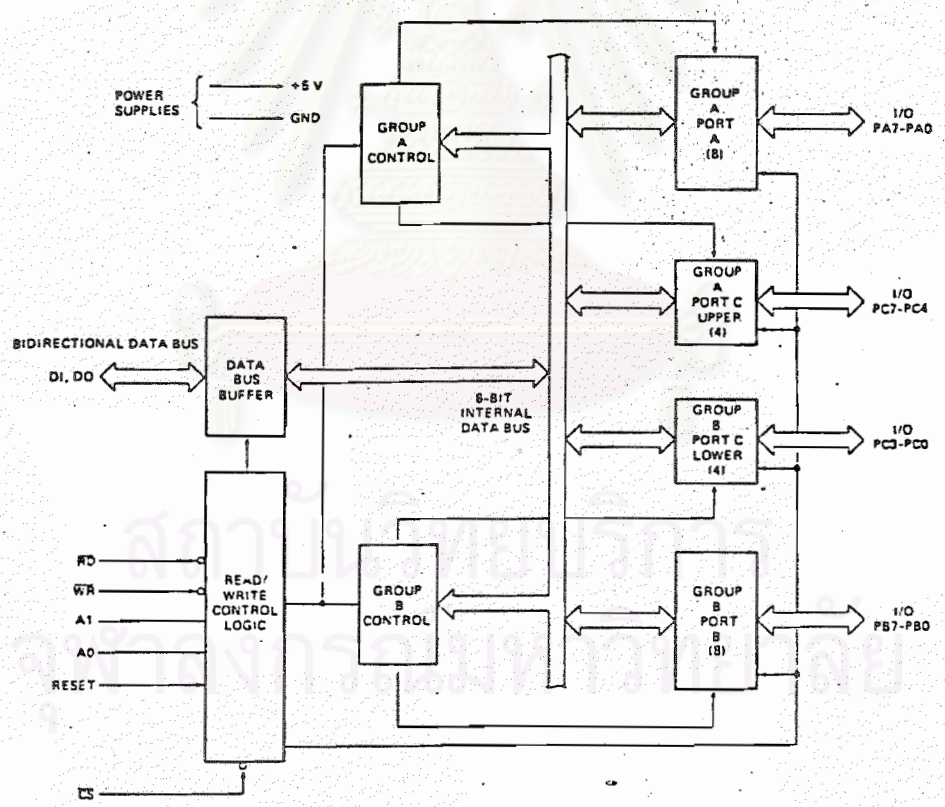
0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

รูปที่ 5.4 ฟอร์แมทคำสั่งควบคุมของ 8253

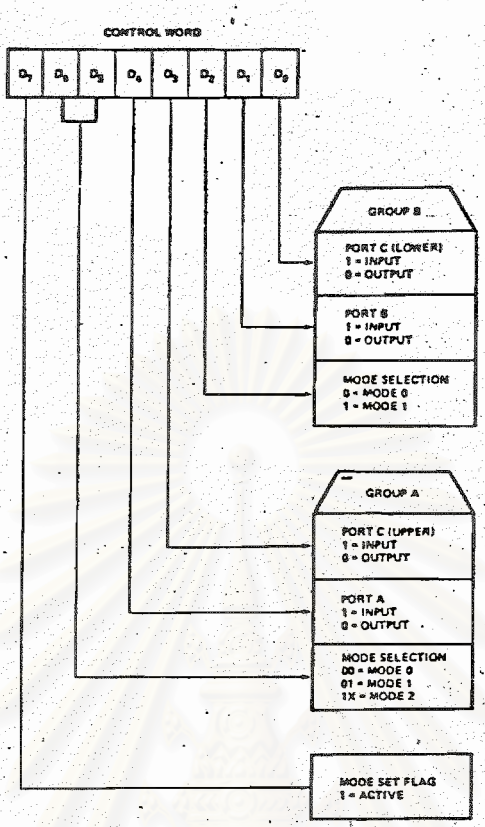
เครื่องควบคุมโปรแกรมให้ 8253 ทำหน้าที่นับเวลา เพื่อจับเวลาการสุ่มข้อมูลจากภายนอก โดย 8253 จะรับสัญญาณเข้ามีความถี่ 1.25 MHz เข้าที่ตัวนับที่ 0 และสร้างสัญญาณจัตุรัสขนาด 50 ms. เมื่ออินเตอร์รัพซีพียู การทำงานถูกกำหนดโดยการส่งข้อมูลจากซีพียู ไปยังพอร์ทเบอร์ 0C803H ของ 8253 3 ครั้ง ตามลำดับ ดังนี้

```
OUTPUT(0C803H) = 036H ;  เลือกตัวนับที่ 0
OUTPUT(0C803H) = 014H ;  กำหนดจำนวนนับ low byte
OUTPUT(0C803H) = 0F0H ;  กำหนดจำนวนนับ high byte
```

(2) 8255A (Programmable Peripheral Interface) โครงสร้างภายใน 8255A แสดงดังรูปที่ 5.5 ประกอบด้วยพอร์ทแบบขนานขนาด 8 บิท 2 พอร์ท กับพอร์ทแบบขนานขนาด 4 บิท 2 พอร์ท โดยการทำงานของแต่ละพอร์ทสามารถโปรแกรมให้ทำงานเป็นได้ทั้ง อินพุทหรือเอาท์พุท ซึ่งในเครื่องควบคุมใช้ พอร์ท A เป็นอินพุท เพื่อตรวจสอบสถานะของการทำงาน เช่น ผลของการเปรียบเทียบสัญญาณจาก DAC กับสัญญาณอินพุทแบบต่อเนื่อง, เก็บสถานะสวิทช์เลือกรูปแบบในการคำนวณโปรแกรม PID แบบ Direct หรือ Reverse เป็นต้น การโปรแกรม 8255A ทำได้โดยเขียนคำสั่งควบคุมที่พอร์ทควบคุมของ 8255A (พอร์ทที่ CC03) ฟอร์แมทของคำสั่งควบคุมแสดงดังรูปที่ 5.6



รูปที่ 5.5 โครงสร้างภายใน 8255A



รูปที่ 5.6 ฟอร์แมทคำสั่งควบคุมของ 8255A

8255A จะถูกโปรแกรมให้พอร์ท A และ B เป็นพอร์ทอินพุทและพอร์ท C เป็นพอร์ทเอาท์พุท การโปรแกรมทำได้โดยเขียนข้อมูล "092H" เข้าที่พอร์ท CC03

(3) 8279 (Programmable Keyboard/Display Interface) โครงสร้างภายใน 8279 ประกอบด้วยส่วนแสดงผลและแป้นพิมพ์ โดยส่วนแสดงผลประกอบด้วย RAM ขนาด 8 บิท 16 ตัว 8279 ใช้ข้อมูลเหล่านี้มาสแกนเพื่อแสดงผล และส่วนแป้นพิมพ์มี RAM ขนาด 8 บิท 8 ตัว เพื่อเก็บรหัสของข้อมูลที่ได้จากการอ่านแป้นพิมพ์ การกำหนดรูปแบบการทำงานส่วนแสดงผลและแป้นพิมพ์ โดยการเขียนคำสั่งควบคุมที่พอร์ทควบคุมของ 8279 (พอร์ทที่ A001) ฟอร์แมทของคำสั่งดังรูปที่ 5.7

Keyboard/Display Mode Set

	MSB						LSB
Code:	0	0	0	D	D	K	K	K

Where DD is the Display Mode and KKK is the Keyboard Mode.

DD

0	0	8 8-bit character display — Left entry
0	1	16 8-bit character display — Left entry*
1	0	8 8-bit character display — Right entry
1	1	16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

KKK

0	0	0	Encoded Scan Keyboard — 2 Key Lockout*
0	0	1	Decoded Scan Keyboard — 2-Key Lockout
0	1	0	Encoded Scan Keyboard — N-Key Rollover
0	1	1	Decoded Scan Keyboard — N-Key Rollover
1	0	0	Encoded Scan Sensor Matrix
1	0	1	Decoded Scan Sensor Matrix
1	1	0	Strobed Input, Encoded Display Scan
1	1	1	Strobed Input, Decoded Display Scan

รูปที่ 5.7 ฟอร์แมทคำสั่งเลือกโหมดการทำงาน 8279

เครื่องควบคุมโปรแกรมให้ 8279 แสดงผล 16 ตัวอักษร และเป็นแบบเริ่มจากซ้าย (left entry) และแป้นพิมพ์ทำงานแบบ Encoded Scan Keyboard - 2 Key Lockout โดยการเขียนข้อมูล "08H" ที่พอร์ท A001 การทำงานของ 8279 จำเป็นต้องมีสัญญาณนาฬิกาที่เหมาะสมเพื่อใช้ในการสแกนแป้นพิมพ์ ความถี่ที่เหมาะสมคือ 100 kHz ทำให้เวลาในการสแกนแป้นพิมพ์แต่ละครั้งเท่ากับ 5 ms. และเวลาในการ debounce 10.3 ms. ในเครื่องควบคุมสัญญาณนาฬิกามีความถี่ 2.5 MHz แต่สามารถเขียนคำสั่งควบคุมไปที่พอร์ท A001 เพื่อหารค่าสัญญาณให้ได้ 100 KHz โดยคำสั่งควบคุมมีฟอร์แมทดังรูปที่ 5.8

Program Clock

Code:	0	0	1	P	P	P	P	P

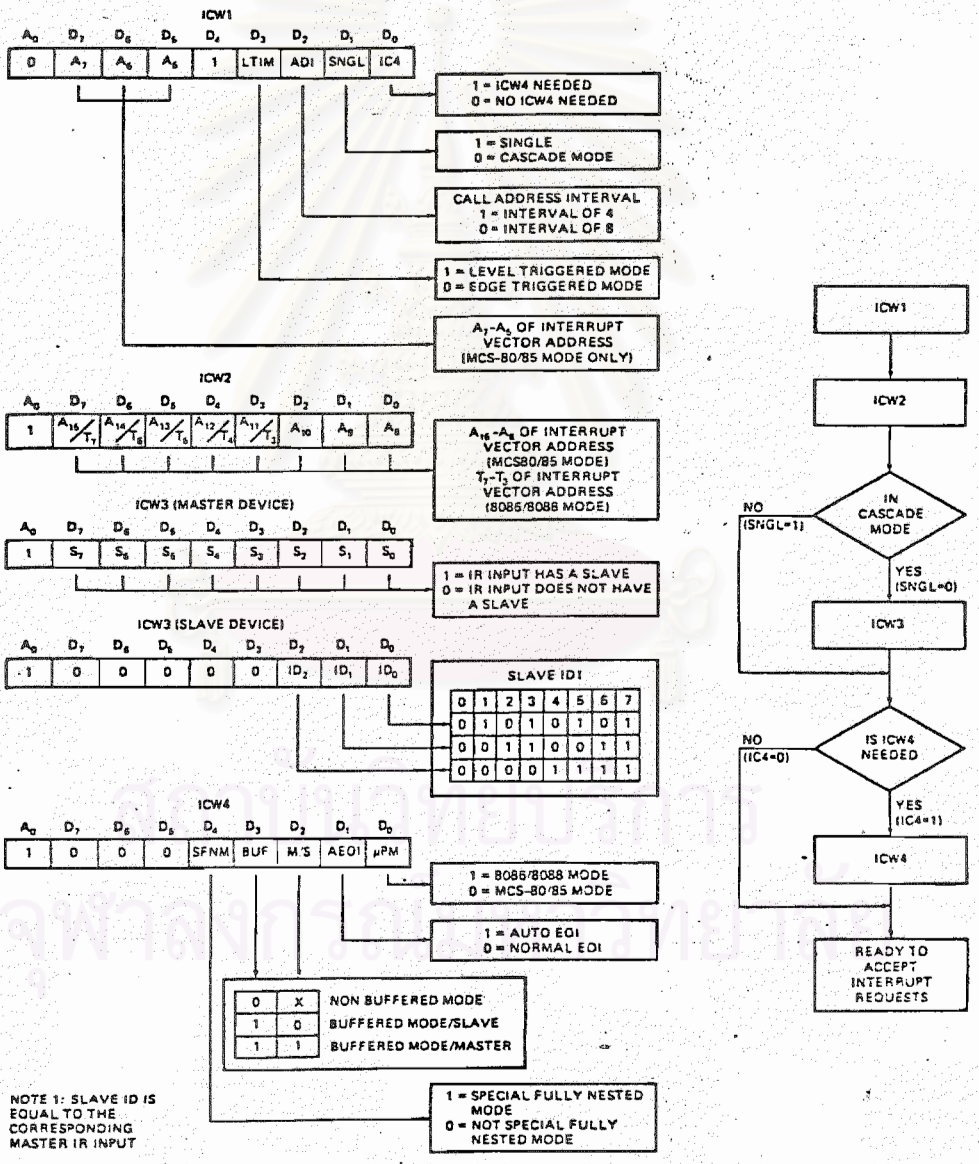
All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits PPPPP determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, PPPPP should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

รูปที่ 5.8 ฟอร์แมทคำสั่งของ Internal Clock ของ 8279

สัญญาณความถี่ 2.5 MHz จะถูกหารด้วยค่า 25 โดยการเขียนข้อมูล "039H" ที่พอร์ท A001 ดังนี้

```
OUTPUT(0A001H) = 039H
OUTPUT(0A001H) = 08H
```

(4) 8259A (Programmable Interrupt Controller) การทำงานของซีพียู Intel 8088 เมื่อได้รับสัญญาณอินเตอร์รัพท์จากภายนอก 8088จะอ่านค่า Interrupt Vector จากอุปกรณ์ภายนอก เพื่อค้นหาตำแหน่งของโปรแกรมบริการอินเตอร์รัพท์ เครื่องควบคุมใช้ 8259A สำหรับส่ง Interrupt Vector โดยรับอินเตอร์รัพท์จากภายนอกได้ 8 สัญญาณ ซึ่งใน 8088 มี Interrupt 256 type ดังนั้นจำเป็นต้องระบุ type สำหรับอินเตอร์รัพท์เรกบน 8259A การโปรแกรม 8259A มีลำดับขั้นตอนและฟอร์แมทของคำสั่งควบคุมแสดงดังรูปที่ 5.9



รูปที่ 5.9 ลำดับขั้นตอนการโปรแกรมและฟอร์แมทคำสั่งควบคุมของ 8259A

การทำงานของ 8259A ในเครื่องควบคุมทำงานในโหมด Single และต้องการใช้ ICW4 โดยเขียนข้อมูล "013H" ที่พอร์ท D400 กรณีนี้ของเครื่องควบคุมไม่ต้องใช้ ICW2 จากนั้นใช้ ICW3 กำหนด Interrupt vector เริ่มต้นของ 8259A เครื่องควบคุมใช้ Interrupt vector แรกคือ 64 โปรแกรมโดยเขียนข้อมูล "040H" ใน ICW3 สำหรับ ICW4 เป็นตัวกำหนดให้ 8259A ถูกใช้งานโดย 8088 โดยเขียนข้อมูล "0DH" ที่ ICW4 ดังนี้

OUTPUT(0D400H) = 013H
OUTPUT(0D402H) = 040H
OUTPUT(0D402H) = 0DH

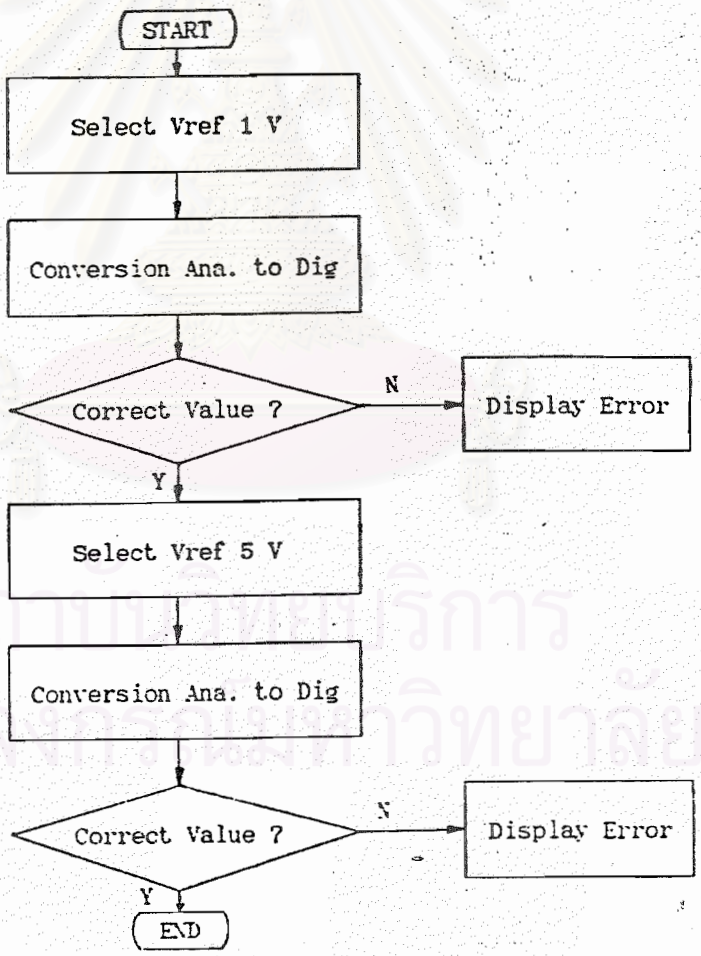
5.3.2 โปรแกรมกำหนดค่าเริ่มต้นของพารามิเตอร์ เครื่องควบคุมโปรแกรมโหมดการทำงานเริ่มต้นเป็นแบบ Manual ในโหมดนี้สัญญาณที่ส่งออกไปควบคุมโปรเซสถูกควบคุมโดยแป้นพิมพ์ สำหรับโปรแกรมกำหนดค่าเริ่มต้นของพารามิเตอร์ในการควบคุม โปรแกรมจะกำหนดค่าเริ่มต้นดังแสดงใน ตารางที่ 5.1 โดยพารามิเตอร์เหล่านี้สามารถแสดงผลและเปลี่ยนแปลงได้โดยแป้นพิมพ์

ตารางที่ 5.1

พารามิเตอร์	จำนวน	ความหมาย	หน่วย	ค่าเริ่มต้น
Pn	1-8	ค่าคงที่ช่วยในการคำนวณ	-	0.0
PB	1,2	ค่า Proportional band	%	900
TI	1,2	ค่า Integral time	sec.	1
TD	1,2	ค่า Derivative time	sec.	0
MH	1	ค่าHigh limitของสัญญาณควบคุม	%	100
ML	1	ค่าLow limitของสัญญาณควบคุม	%	0
PH	1,2	ค่าHigh Alarmของตัวแปรโปรเซส	%	100
PL	1,2	ค่าLow Alarmของตัวแปรโปรเซส	%	0
FX	1-10	Segments Interporation	%	0

5.3.3 โปรแกรมตรวจสอบความผิดพลาดของระบบ

(1) โปรแกรมตรวจสอบ DAC โปรแกรมตรวจสอบการแปลงค่าจากสัญญาณแบบต่อเนื่องเป็นสัญญาณเชิงเลขของ DAC ได้โดยใช้แรงดันอ้างอิง 1 V.และ 5 V. ซึ่งต่ออยู่กับอนาลอคสวิทช์ โดยโปรแกรมจะอ่านค่าจากช่องสัญญาณที่ต่ออยู่กับแรงดันอ้างอิง 1 V. มาแปลงเป็นสัญญาณเชิงเลขโดยวิธี Successive Approximation นำค่าเชิงเลขที่ได้จากการแปลงค่าของ DAC มาเปรียบเทียบกับค่าตัวเลขที่ถูกต้อง (ค่าที่ถูกต้องสำหรับการแปลงค่าที่ถูกต้องคือ 0211H สำหรับ 1 V. และค่า 0BE0H สำหรับ 5 V.) ถ้าถูกต้องจะทำการตรวจสอบในทำนองเดียวกันกับช่องสัญญาณของ 5 โวลท์ โฟว์ชาร์ทแสดงการทำงานของโปรแกรมแสดงดังรูปที่ 5.10

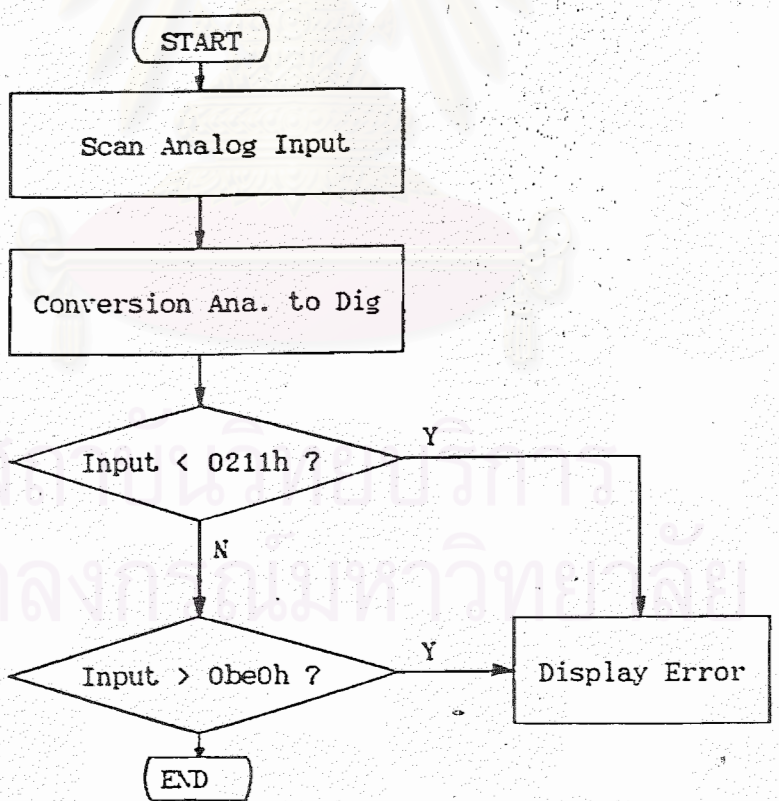


รูปที่ 5.10 โฟว์ชาร์ทของโปรแกรมตรวจสอบ DAC

(2) โปรแกรมตรวจสอบ RAM โปรแกรมตรวจสอบได้โดยเขียนข้อมูลลงบน RAM ที่ต้องการตรวจสอบ และอ่านค่ากลับมาเปรียบเทียบ

(3) โปรแกรมตรวจสอบเวลาในการควบคุม ในการทำงานของโปรแกรมแบบวนรอบ หลังจากทำงานในแต่ละรอบแล้ว โปรแกรมทำการตรวจสอบค่าตัวแปรที่ใช้ในการนับเวลา ซึ่งถูกนับโดยการอินเตอร์รัพท์ของ 8253 โดยแต่ละหน่วยของการนับมีค่าเท่ากับ 50 ms. โปรแกรมจะตรวจสอบว่าค่าตัวแปรนี้มีค่ามากกว่า 4 หรือไม่ ถ้ามากกว่าโปรแกรมจะแสดงผลความผิดพลาด ถ้าน้อยกว่า 4 โปรแกรมจะ clear ค่าตัวแปรที่ใช้นับ

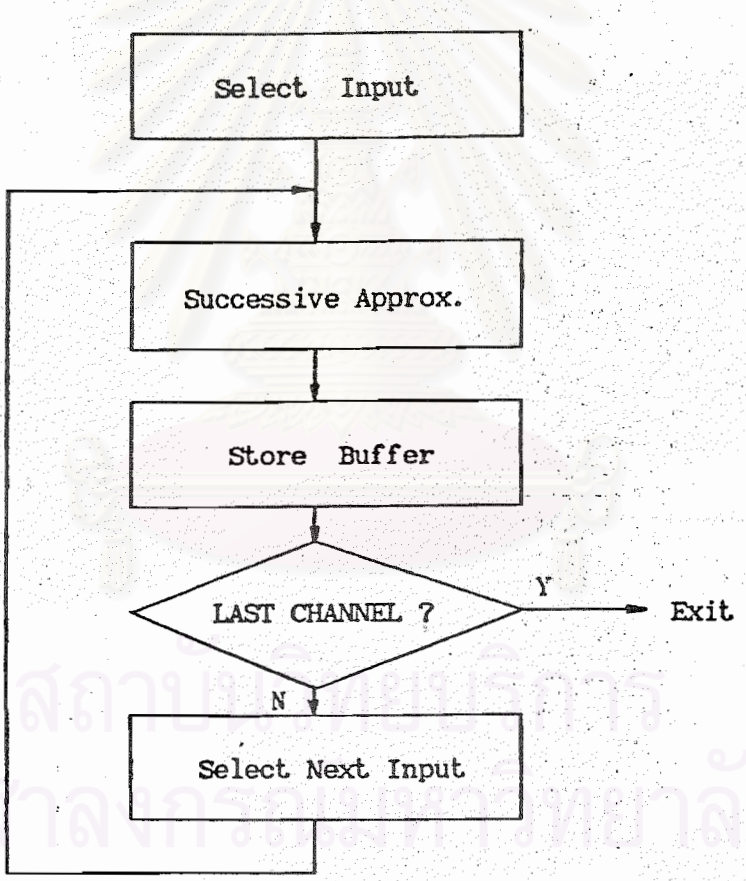
(4) โปรแกรมตรวจสอบอินพุทเกินพิสัย อินพุทของเครื่องควบคุมมีพิสัยอยู่ระหว่างค่า 1-5 V.(ค่าที่ได้จาก DAC คือ 0211H กับ 0BE0H) โปรแกรมจะทำการตรวจสอบค่าตัวแปรที่ได้จากโปรแกรมสแกนอินพุท ว่ามีค่าอยู่ในพิสัยหรือไม่ โฟว์ชาร์ทแสดงการทำงานของโปรแกรมแสดงได้ดังรูปที่ 5.11



รูปที่ 5.11 โฟว์ชาร์ทของโปรแกรมตรวจสอบอินพุทเกินพิสัย

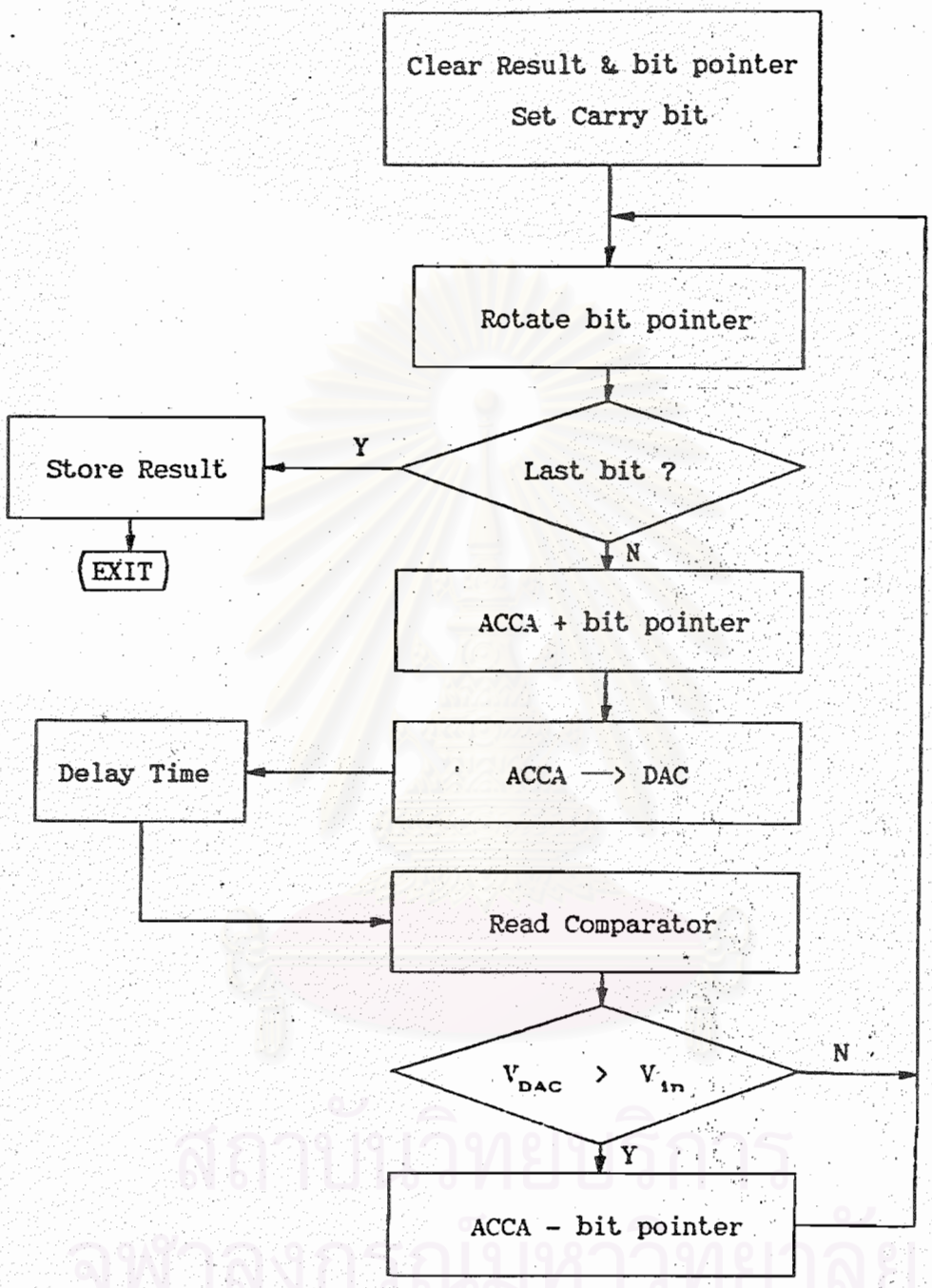
5.3.4 โปรแกรมสแกนอินพุทและเอาท์พุท

(1) **โปรแกรมสแกนอนาลอคอินพุท** เนื่องจากเครื่องควบคุมมีอินพุทและเอาท์พุทหลายช่องสัญญาณ ฮาร์ดแวร์ในส่วนอินเตอร์เฟสนี้จึงมีการออกแบบเพื่อลดจำนวนไอซี ADC และ DAC โดยทุกๆช่องสัญญาณของอินพุทและเอาท์พุทจะใช้ DAC เพียงตัวเดียว กับอนาลอคสวิทช์และตัวเปรียบเทียบ (Comparator) ทำงานร่วมกับโปรแกรมเพื่อแปลงสัญญาณอนาลอคเป็นสัญญาณเชิงเลข โดยอัลกอริทึมในการแปลงใช้วิธี Succesive Approximation และเก็บค่าไว้ที่บัฟเฟอร์สำหรับโปรแกรมอื่นเรียกใช้ โฟว์ชาร์ทแสดงดังรูปที่ 5.12



รูปที่ 5.12 โฟว์ชาร์ทของโปรแกรมสแกนอนาลอคอินพุท

สำหรับอัลกอริทึมการแปลงสัญญาณแบบ Successive Approximation มีโฟว์ชาร์ท แสดงดังรูปที่ 5.13 [8]



รูปที่ 5.13 โฟว์ชาร์ทของ Successive Approximation

(2) โปรแกรมสแกนอนาลอคเอาท์พุท โปรแกรมจะส่งสัญญาณอนาลอคออกที่เอาท์พุท โดยการเลือกช่องสัญญาณของอนาลอคสวิทช์ก่อนส่งข้อมูลผ่าน DAC เพื่อแปลงค่าสัญญาณอนาลอคออกที่ช่องสัญญาณที่เลือกไว้

(3) โปรแกรมอ่านดิจิตอลอินพุท โปรแกรมอ่านสถานะอินพุทผ่านพอร์ทได้ข้อมูล 1 ไบท์ ซึ่งเก็บสถานะของอินพุท 3 จุด และทำการ unpack ออกเป็น 3 ไบท์ โดยแต่ละไบท์เก็บสถานะของแต่ละจุด ดังรูปที่ 5.14

Packed Data	0 0 0 0 0 1 0 1	ข้อมูลที่อ่านจากพอร์ท
Unpacked Data	0 0 0 0 0 0 0 1	สถานะของช่องที่ 1
	0 0 0 0 0 0 0 0	สถานะของช่องที่ 2
	0 0 0 0 0 0 0 1	สถานะของช่องที่ 3

รูปที่ 5.14 การ unpack ดิจิตอลอินพุท

(4) โปรแกรมขับดิจิตอลเอาท์พุท โปรแกรมจะขับดิจิตอลเอาท์พุททั้งหมดโดยเขียนข้อมูลเพียง 1 ไบท์ ออกที่ดิจิตอลเอาท์พุทพอร์ท ซึ่งได้มาจากการ pack ไบท์ข้อมูลของแต่ละเอาท์พุทมารวมกัน ดังรูปที่ 5.15

Unpacked Data	0 0 0 0 0 0 0 1	สถานะของช่องที่ 1
	0 0 0 0 0 0 0 0	สถานะของช่องที่ 2
	0 0 0 0 0 0 0 1	สถานะของช่องที่ 3
Packed Data	0 0 0 0 0 1 0 1	ข้อมูลใช้เขียนออกพอร์ท

รูปที่ 5.15 การ pack ข้อมูลของดิจิตอลเอาท์พุท

5.3.5 โปรแกรมรับค่าจากแป้นกดและแสดงผล โปรแกรมแบ่งเป็น 2 ส่วน คือ

5.3.5.1 โปรแกรมรับค่าแป้นกดและแสดงผลด้านหน้า โปรแกรมส่วนนี้แบ่งเป็นโปรแกรมย่อยได้ 2 โปรแกรม ดังนี้

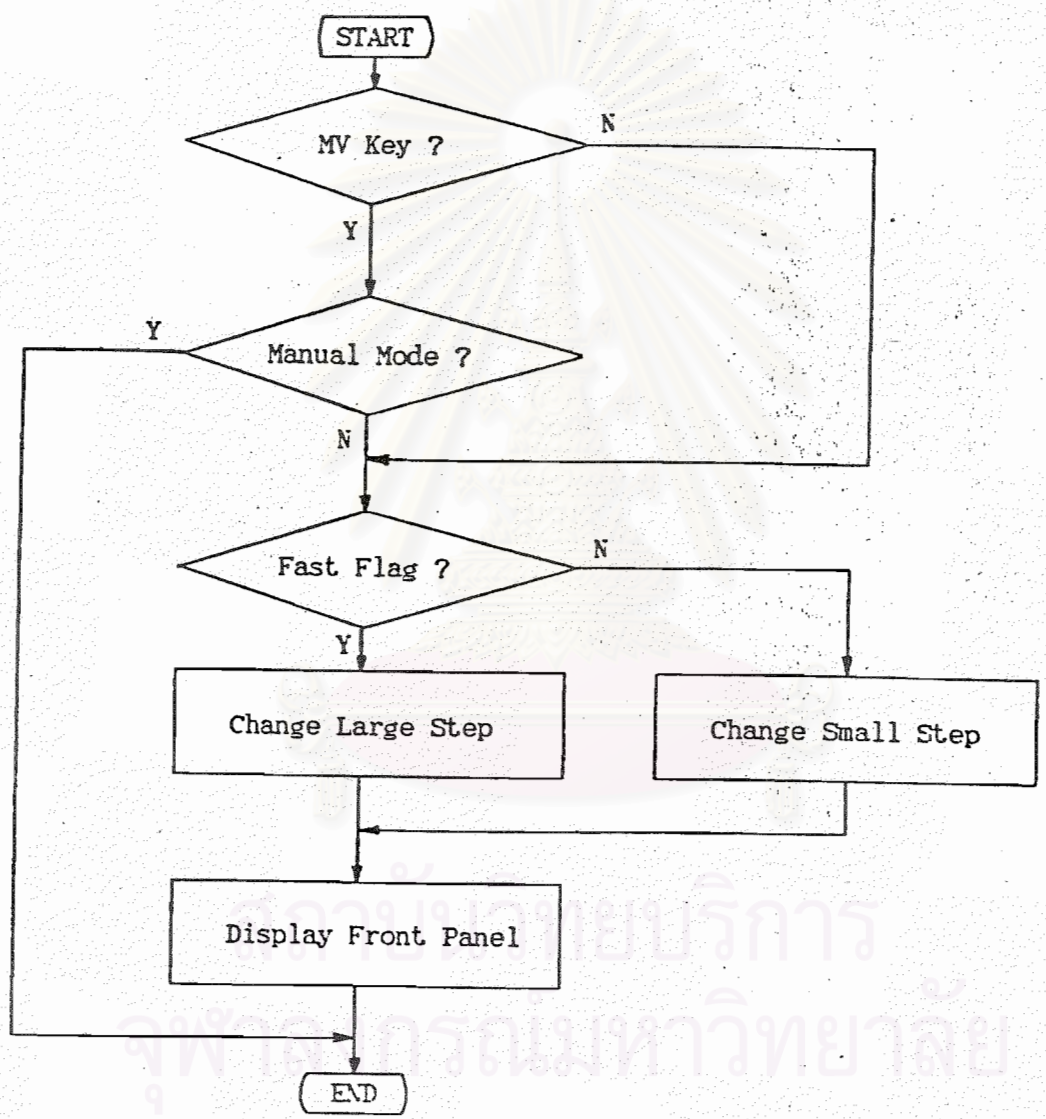
(1) โปรแกรมตรวจสอบแป้นกดด้านหน้า ด้านหน้าของเครื่องควบคุมมี 10 แป้นกด แต่ละตัวมีหน้าที่ดังตารางที่ 5.2

ตารางที่ 5.2

แป้นกด	หน้าที่
A	กำหนดเครื่องควบคุมทำงานในโหมด AUTO
C	กำหนดเครื่องควบคุมทำงานในโหมด CASCADE
M	กำหนดเครื่องควบคุมทำงานในโหมด MANUAL
SV	เพิ่มค่าเป้าหมายของการควบคุม
SV	ลดค่าเป้าหมายของการควบคุม
MV	เพิ่มค่าสัญญาณควบคุม
MV	ลดค่าสัญญาณควบคุม
F	เพิ่มขั้นในการเพิ่มหรือลดค่า SV และ MV
Sel	เลือกค่าที่แสดงผลบน LED 7 ส่วน (PV,SV,MV)

แป้นกด 8 ตัวแรกในตารางที่ 5.2 ต่อกับอินพุทพอร์ทที่ AC00 การทำงานของโปรแกรมตรวจสอบแป้นกด เมื่อมีการกดแป้นกดแต่ละตัวมีดังนี้

- F โปรแกรมจะ set flag เพื่อกำหนดขั้นการเพิ่มหรือลดค่าสำหรับการเปลี่ยนแปลงค่าของ SV และ MV
- M,A,C โปรแกรม set flag กำหนดโหมดการทำงานของเครื่องควบคุม
- SV ,SV ,MV ,MV โปรแกรมทำการเพิ่มหรือลดค่า MV หรือ SV แต่ในกรณีของ MV จะมีการตรวจสอบเพิ่มเติม คือ การเปลี่ยนแปลงค่าจะทำได้ เมื่อเครื่องควบคุมทำงานอยู่ในโหมด MANUAL เท่านั้น โฟว์ชาร์ทแสดงการทำงานแสดงดังรูปที่ 5.16



รูปที่ 5.16 โฟว์ชาร์ทของโปรแกรมการเปลี่ยนแปลงค่า MV,SV

(2) โปรแกรมแสดงผลด้านหน้า ส่วนแสดงผลด้านหน้าประกอบด้วย

- LED 7 ส่วน 4 ตัว เพื่อแสดงค่าตัวแปรโปรเซส (PV), ค่าเป้าหมาย (SV), ค่าตัวแปรควบคุม (MV) ขึ้นอยู่กับแป้นกดพิมพ์ Sel
- LED 7 ตัว เพื่อแสดงสถานะการทำงานของระบบ ได้แก่ โหมดการทำงาน, แสดงค่าที่ถูกแสดงบน LED 7 ส่วนเป็นของตัวแปร PV,SV หรือ MV
- Bar graph แสดงผลของ PV, SV และ MV

ดังนั้นโปรแกรมควบคุมส่วนแสดงผลด้านหน้าแบ่งเป็น 3 ส่วน คือ

(ก) **โปรแกรมควบคุมการแสดงผล LED 7 ส่วน** โปรแกรมต้องทำการแปลงค่า PV,SV หรือ MV ซึ่งมีโครงสร้างการเก็บข้อมูลแบบ floating point ให้อยู่ในรูปของรหัส BCD (Binary Code Decimal) เพื่อทำ table look up หารหัสในการแสดงผลบน LED 7 ส่วน

(ข) **โปรแกรมแสดงผล LED สถานะการทำงาน** โปรแกรมนำสถานะของระบบในหลายไบท์มารวมกัน (pack) เป็น 1 ไบท์ เขียนออกที่พอร์ท A800 เพื่อขับ LED ให้สว่าง

(ค) **โปรแกรมแสดงผล bar graph** โปรแกรมนำค่า MV,SV และ MV มาแสดงผลบน bar graph โดยค่าจะถูกคำนวณเพื่อหาตำแหน่งของ LED ที่ต้องการให้สว่างบน bar graph โปรแกรมจะให้ LED สว่างเพียงจุดเดียวเพื่อประหยัดกระแสในการขับ LED ซึ่ง bar graph ของแต่ละค่าจะเริ่มต้นที่ตำแหน่งของ display RAM บน 8279 ต่างกัน คือ PV ใช้ display RAM ตำแหน่งที่ 5-8, SV ใช้ตำแหน่งที่ 9-12, MV ใช้ตำแหน่งที่ 13-14

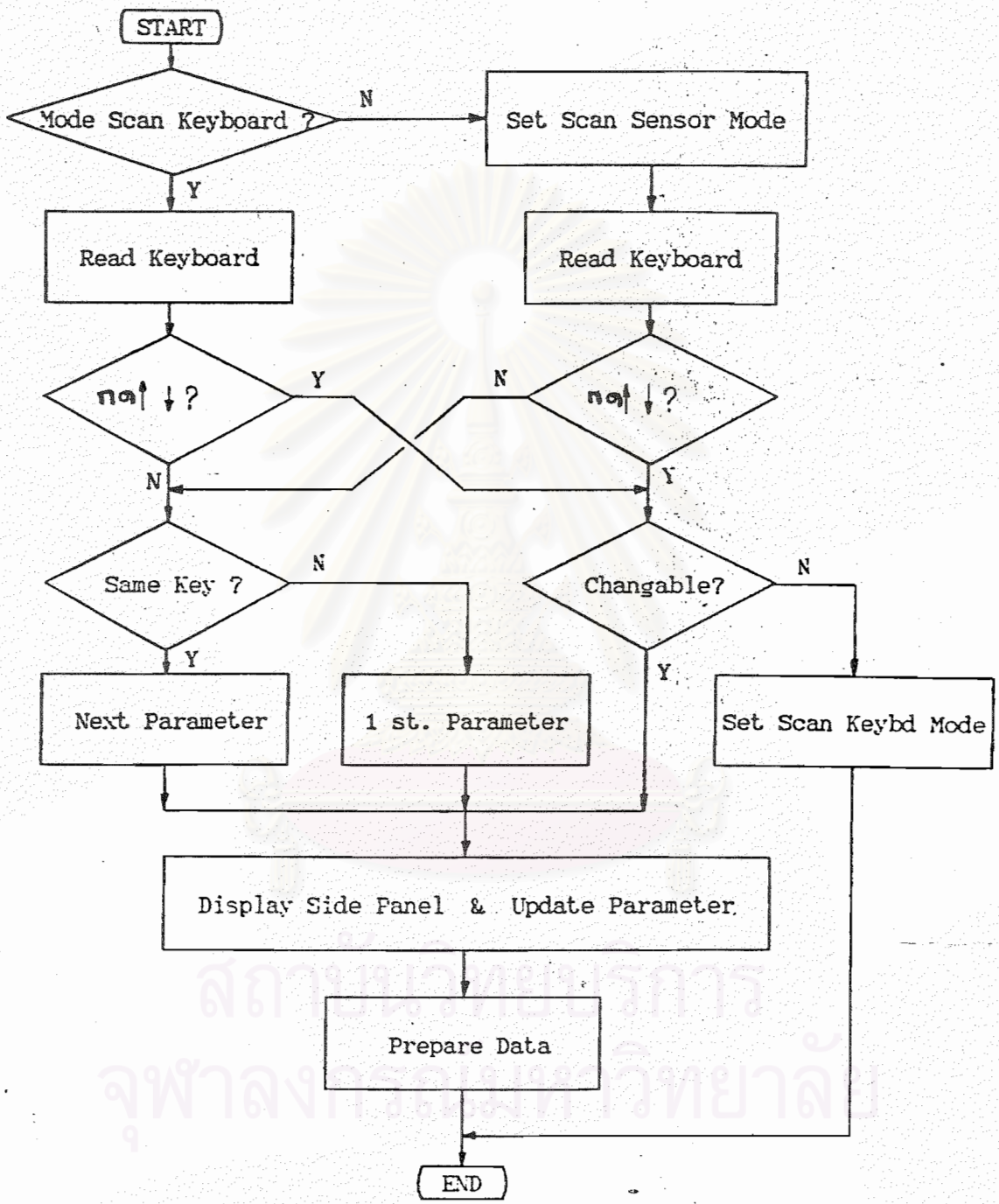
5.3.5.2 โปรแกรมรับค่าแป้นพิมพ์และแสดงผลด้านซ้าง แบ่งเป็นโปรแกรมย่อยได้ 3 โปรแกรม ดังนี้

(1) โปรแกรมตรวจสอบแป้นพิมพ์ด้านซ้าง เนื่องจากฮาร์ดแวร์ของแป้นพิมพ์ในส่วนนี้จะถูกควบคุมโดยไอซี 8279 ซึ่งหลักในการออกแบบส่วนนี้ ต้องการให้มีจำนวนแป้นพิมพ์ไม่มากนัก จึงจัดให้มีการใช้ 1 แป้นพิมพ์ในการแสดงผลค่าพารามิเตอร์ได้ 2-3 ตัว ซึ่งมีฟังก์ชันใกล้เคียงกัน เช่น ใช้แสดงผลค่า PV และ SV โดยกดที่แป้นพิมพ์เดียวกัน ซึ่งการกดครั้งแรกจะแสดงผลค่า PV เมื่อกดซ้ำอีกครั้งจะแสดงผลของ SV เป็นต้น สำหรับในการเปลี่ยนแปลงค่าพารามิเตอร์สามารถลดจำนวนแป้นพิมพ์ได้ โดยใช้แป้นพิมพ์เพียง 3 ตัว คือ Δ , ∇, $\Updownarrow$ เพิ่มหรือลดค่าตัวแปรแทนการกดค่าตัวเลข จากหลักการทำงานดังกล่าว โปรแกรมในส่วนนี้จะโปรแกรมให้ 8279 ทำงานในโหมด 2 โหมด คือ

- Encoded Scan Keyboard มีลักษณะการทำงานคือ 8279 จะสแกนอ่านแป้นพิมพ์ และเมื่อมีการกดแป้นพิมพ์ 8279 จะแปลงค่ารหัสของแป้นพิมพ์ลงบน RAM ภายใน 8279 และเมื่อ 8279 สแกนข้อมูลอีกครั้ง ถึงแม้แป้นพิมพ์เดิมยังคงถูกค้างไว้ 8279 ก็จะไม่ทราบว่ามีแป้นพิมพ์ยังคงกดค้างไว้ นอกจากจะมีการปล่อยแป้นพิมพ์ก่อนกดอีกครั้ง ซึ่งโปรแกรมจะใช้การทำงานในโหมดนี้ เพื่อใช้กับแป้นพิมพ์ที่ใช้ในการแสดงผล เพราะเมื่อใช้แป้นพิมพ์เดียวแทนการแสดงผลของพารามิเตอร์หลายตัว เมื่อมีการกดแป้นพิมพ์ค้างไว้ก็ไม่ทำให้การแสดงผลค่าพารามิเตอร์เปลี่ยนไปเป็นค่าพารามิเตอร์อื่น นอกจากจะปล่อยและกดแป้นนั้นซ้ำเพื่อเรียกพารามิเตอร์ถัดไปที่ใช้แป้นเดียวกัน

- Encode Scan Sensor Matrix ลักษณะการทำงานที่สำคัญของโหมดนี้คือ สามารถอ่านค่าแป้นพิมพ์ที่ถูกกดค้างไว้ได้ ในทุกรอบของการสแกนอ่านค่าแป้นพิมพ์ ดังนั้นจึงนำมาใช้กับแป้นพิมพ์ที่ใช้ในการเปลี่ยนแปลงค่าพารามิเตอร์

สำหรับโฟว์ชาร์ทแสดงการทำงานของโปรแกรมเขียนได้ ดังรูปที่ 5.17



รูปที่ 5.17 โฟว์ชาร์ทของโปรแกรมตรวจสอบแป้นพิมพ์ด้านข้าง

ตารางที่ 5.3 แสดงชื่อและหน้าที่ของแป้นฟิมน์

แป้นฟิมน์	จำนวน	ฟังก์ชั่น	หน่วย	แก้ไข
Pn	15	ค่าพารามิเตอร์ช่วยในการคำนวณ	–	/
Xn	1-5	อ่านค่าอนาลอคอินพุท	%	X
Yn	1-6	อ่านค่าอนาลอคเอาท์พุท	%	X
DI	1-4	อ่านค่าดิจิตอลอินพุท	0/1	X
DO	1-5	อ่านค่าดิจตอลเอาท์พุท	0/1	X
PV	1-2	อ่านค่า Process variable	%	X
SV	1-2	อ่านค่า Set point 1	%	X
		Set point 2	%	/
DV	1-2	อ่านค่าผลต่างของ PV และ SV	%	X
MV	1	อ่านค่า Manipulated variable	%	X
MH	1	กำหนดค่า MV high limit	%	/
ML	1	กำหนดค่า MV low limit	%	/
PH	1,2	กำหนดค่า PV high alarm	%	/
PL	1,2	กำหนดค่า PV low alarm	%	/
PB	1,2	กำหนดค่า Proportional band	%	/
Ti	1,2	กำหนดค่า Integral time	sec.	/
Td	1,2	กำหนดค่า Derivative time	sec.	/
FX	1	กำหนดค่าเอาท์พุทของแต่ละ Segment	%	/

(2) โปรแกรมจัดการแสดงผลด้านข้าง เป็นส่วนหนึ่งในโฟว์ชาร์ทรูปที่ 5.17 โปรแกรมส่วนแสดงผลด้านข้างแบ่งเป็น 3 โปรแกรม คือ

(ก) โปรแกรมแสดงผลแบบ floating point พารามิเตอร์ทุกตัวจะใช้โปรแกรมนี้ ยกเว้นพารามิเตอร์ของดิจิตอลอินพุทและเอาท์พุท และเนื่องจากฮาร์ดแวร์ในส่วนแสดงผลใช้รหัส ASCII ในการแสดงผล ดังนั้นก่อนการแสดงผลจำเป็นต้องมีการแปลงรหัสจาก binary เป็น ASCII ก่อนเสมอ สำหรับโปรแกรมส่วนนี้นอกจากทำการแสดงผลแล้ว ยังรวมถึงการเปลี่ยนแปลงค่าพารามิเตอร์เข้าไปด้วยสำหรับพารามิเตอร์ที่สามารถเปลี่ยนแปลงได้

(ข) โปรแกรมแสดงผลค่าดิจิตอลอินพุท โปรแกรมจะนำตัวแปรที่ได้จากโปรแกรมสแกนอ่านดิจิตอลอินพุท ซึ่งได้รับการแยกไบท์ (unpacked) สถานะของแต่ละอินพุทเรียบร้อยแล้ว มาใช้ในการแสดงผล

(ค) โปรแกรมแสดงผลค่าดิจิตอลเอาท์พุท โปรแกรมนำค่าที่ได้จากคำสั่งควบคุมมาแสดงผลได้เลย โดยไม่ต้องผ่านการ unpack เพราะแต่ละเอาท์พุทมี 1 ไบท์ ในการแทนค่าสถานะ

(3) โปรแกรมเตรียมข้อมูล เป็นโมดูลสุดท้ายในโฟว์ชาร์ทรูปที่ 5.17 โปรแกรมส่วนนี้จะทำการเตรียมข้อมูลสำหรับการคำนวณในภายหน้า สาเหตุที่มีโปรแกรมนี้ เนื่องจากต้องการเพิ่มความเร็วในการทำงานในแต่ละรอบของโปรแกรมวนรอบ เพราะในการทำงานจริงจะมีการเปลี่ยนแปลงค่าพารามิเตอร์ในขณะควบคุมไม่บ่อยนัก ดังนั้นในการทำงานวนรอบไม่จำเป็นต้องมาคำนวณค่าตัวแปรที่มีความสัมพันธ์กับพารามิเตอร์ทุกรอบ แต่จะคำนวณเมื่อมีการเปลี่ยนค่าพารามิเตอร์โดยแป้นพิมพ์เท่านั้น ดังนั้นจึงนำโปรแกรมเตรียมข้อมูลมาใส่ในส่วนโปรแกรมแป้นพิมพ์ สำหรับพารามิเตอร์ที่จำเป็นต้องมีการเตรียมข้อมูลมีดังนี้

(ก) Proportional band จากสมการที่ใช้ในการคำนวณการควบคุมแบบ PID ดังสมการที่ (5.1)

$$MV = \frac{100}{PB}\left(PV + \frac{1}{T_i s}E + \frac{T_D S}{1+\frac{T_d s}{N}}PV\right) \quad \ldots\ldots\ldots (5.1)$$

ให้ $MV = A_n + B_n + C_n$

โดย $A_n = K_p PV_n$

$$B_n = B_{n-1} + \frac{K_p T_s}{T_i} E_n$$

$$C_n = \frac{T_f C_{n-1}}{T_s+T_f} + \frac{K_p T_D}{6(T_s+T_f)}(PV_n - PV_{n-3} - 3PV_{n-2} + 3PV_{n-1})$$

ให้ $blconst = K_p T_s / T_i$ ..... (5.2)

$clconst1 = K_p T_d$ ..... (5.3)

$clconst2 = 6(T_f + T_s)$ ..... (5.4)

$T_f = T_d / 8$ ..... (5.5)

ซึ่งค่าคงที่ในสมการที่ (5.2-5.5) จะไม่เปลี่ยนแปลง ถ้าค่า PB,Ti และ Td ไม่เปลี่ยน จึงสามารถคำนวณค่าคงที่เหล่านี้ไว้ใช้ในการคำนวณในรอบต่อๆไป ดังนั้นถ้ามีการเปลี่ยนแปลงค่า PB จำเป็นจะต้องมีการคำนวณค่า blconst, clconst1 ใหม่ วิธีนี้จะทำให้การคำนวณสมการของ PID เร็วขึ้น เพราะในแต่ละรอบ สมการ PID ไม่จำเป็นต้องมีการคำนวณค่าคงที่เหล่านี้ใหม่โดยไม่จำเป็น

(ข) Integral Time ใช้หลักการในการพิจารณาเดียวกัน ดังนั้นเมื่อมีการเปลี่ยนแปลงค่า Ti จะต้องมีการเปลี่ยนแปลงค่า blconst ด้วย

(ค) Derivative Time เมื่อมีการเปลี่ยนแปลงค่า Td ค่าที่ต้องทำการคำนวณใหม่ คือ clconst1, tf, clconst2

(ง) Segment Interporation การทำงานของโปรแกรม Interporation จะใช้ค่าหลัก 10 ค่า เมื่อค่าอินพุทอยู่ระหว่างจุด 2 จุด ใน 10 ค่านี้ จะทำการคำนวณโดยอาศัยค่าความชันระหว่างจุด 2 จุดนั้นช่วยในการคำนวณค่า ดังนั้นเมื่อมีการเปลี่ยนแปลงค่าหลักใน Interporation จึงจำเป็นต้องมีการคำนวณค่าความชันใหม่ การทำเช่นนี้จะช่วยเพิ่มความเร็วในการทำ Interporation เพราะเมื่อมีการทำ Interporation อีก ก็ไม่จำเป็นต้องคำนวณหาค่าความชันซ้ำอีก

5.3.6 โปรแกรมย่อยบริการผู้ใช้

เป็นโปรแกรมย่อยที่เขียนเป็นโมดูล เพื่อสร้างฟังก์ชันให้ผู้ใช้นำมาเขียนโปรแกรมกำหนดรูปแบบการควบคุม ซึ่งการส่งผ่านข้อมูลของแต่ละฟังก์ชันจะใช้วิธีการของ stack และวิธีการโปรแกรมกำหนดรูปแบบจะใช้วิธีของ Reverse Polish Notation (RPN) ซึ่งมีหลักเกณฑ์การทำงานดังนี้ [9]

- นิพจน์ของ RPN จะทำการประมวลผลจากซ้ายมาขวา ไม่มีเครื่องหมายวงเล็บและไม่มีความสำคัญกว่า (Operator Precedence)
- เมื่อพบค่า Operand จะเก็บค่าลงบน stack
- เมื่อพบค่า Operator จะทำงานตาม Operator ที่พบโดยนำค่าบน stack 2 ตัวแรกมาเป็น Operand และเมื่อได้ผลลัพธ์จะเก็บไว้บน stack

ตัวอย่างการโปรแกรมวิธี RPN กับสมการพีชคณิต แสดงได้ดังตารางที่ 5.4

ตารางที่ 5.4

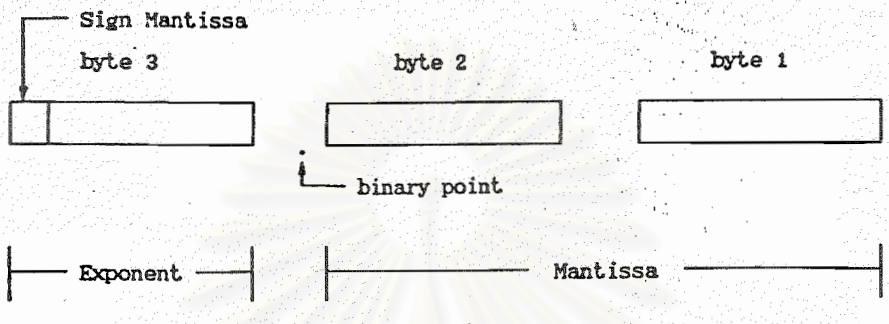
สมการพีชคณิต	RPN
A+B	AB+
A(B+C)	BC+A*
(A/B)+C	AB/C+

ตารางที่ 5.5 แสดงประเภทและหน้าที่ของ Function ทั้งหมดที่มีในเครื่องควบคุม

ประเภท	หน้าที่	nmemonic
คำนวณคณิตศาสตร์	บวก	ADD
	ลบ	SUB
	คูณ	MUL
	หาร	DIV
	ถอดรากที่สอง	SQR
	ค่าสัมบูรณ์	ABS
ทางตรรก	AND, OR, NOT	AND, OR, NOT
	กระโดดแบบไม่มีเงื่อนไข	GO
	กระโดดแบบมีเงื่อนไข	GIF
	เปรียบเทียบค่า	CMP
ฟังก์ชันพื้นฐาน	First Order Lag	LAG
	First Order Lead	LED
	Deadtime	DED
	Timer	TIM
	High & Low Selector	HSL, LSL
	Interporation	FX
P I D	Basic P I D	BPID
	Cascade P I D	CPID

5.3.6.1 โปรแกรมคำนวณทางคณิตศาสตร์

การคำนวณในการควบคุมจำเป็นต้องมีความละเอียดในการคำนวณสูง ดังนั้นโปรแกรมจึงใช้โครงสร้างของตัวเลขแบบมีจุดทศนิยม (ดังรูปที่ 5.18) ประกอบ



รูปที่ 5.18 โครงสร้างของตัวเลขทศนิยม

ด้วย ส่วน mantissa 2 ไบท์ แทนส่วนที่เป็นเศษส่วน เพราะส่วน mantissa จะมีจุดทศนิยมนำหน้าอยู่ และมีบิทเครื่องหมายของ mantissa อยู่ในไบท์เดียวกับส่วน exponent ทำให้ส่วน exponent มีขนาด 7 บิท ซึ่งจะไม่มีบิทเครื่องหมายของ exponent เพราะการแทนค่าจาก -64 ถึง 63 จะนำค่าไบแอส 64 มาบวกได้ค่าจาก 0 ถึง 127 ซึ่งทำให้การเปรียบเทียบค่า exponent เมื่อมีการทำบวก ลบ คูณ หาร ง่ายขึ้น เพราะไม่จำเป็นต้องสนใจกับบิทเครื่องหมายของ exponent [10] ตัวอย่างหน่วยความจำที่ใช้เก็บค่าตัวเลข แสดงดังรูปที่ 5.19

0	1000000	.1000000000000000	= 0.5
1	1000001	.1100000000000000	= -1.5

รูปที่ 5.19 ตัวอย่างหน่วยความที่เก็บตัวเลข

การคำนวณตัวเลขทศนิยมให้ได้ค่าที่ละเอียด และมีประสิทธิภาพ ตัวเลขทศนิยมทุกตัวที่ใช้ในการคำนวณจำเป็นต้องผ่านการ normalize ก่อน [7,10] ซึ่งทำได้โดยการเลื่อนส่วน mantissa ไปทางซ้ายทีละบิทจนกระทั่งบิทแรกทางขวาของจุดทศนิยมมีค่าเป็น 1 โดยการเลื่อนแต่ละครั้งต้องลดค่า exponent ด้วย ดังนั้นค่าของ mantissa ที่ normalize แล้วจะมีขนาดดังนี้ (ยกเว้น ค่า 0)

$$0.5 \leq f < 1$$

ตัวอย่างของตัวเลข unnormalize และ normalize (ดูรูปที่ 5.20)

unnormalize	0	1000001	0100000000000000	= 0.5
normalize	0	1000000	1000000000000000	= 0.5

รูปที่ 5.20 ตัวอย่างตัวเลข normalize

หลักการดังกล่าวข้างต้นถูกนำมาใช้ในการคำนวณทางคณิตศาสตร์ รายละเอียดของแต่ละโปรแกรมมีดังนี้

(1) บวกและลบ

mnemonic: ADD ; บวก
SUB ; ลบ

อัลกอริทึม: โปรแกรมบวก ลบ ตัวเลขทศนิยม แบ่งออกเป็นส่วนๆ ได้ 6 ส่วน ดังนี้ [10]

(ก) ตรวจสอบค่า Operand เป็น 0

(ข) การจัด (Align) ส่วน mantissa ของ Operand ที่มีค่า exponent เล็กกว่า โดยการเลื่อนค่า mantissa ไปทางขวาจนกระทั่ง exponent เท่ากัน

(ค) บวกหรือลบค่า mantissa

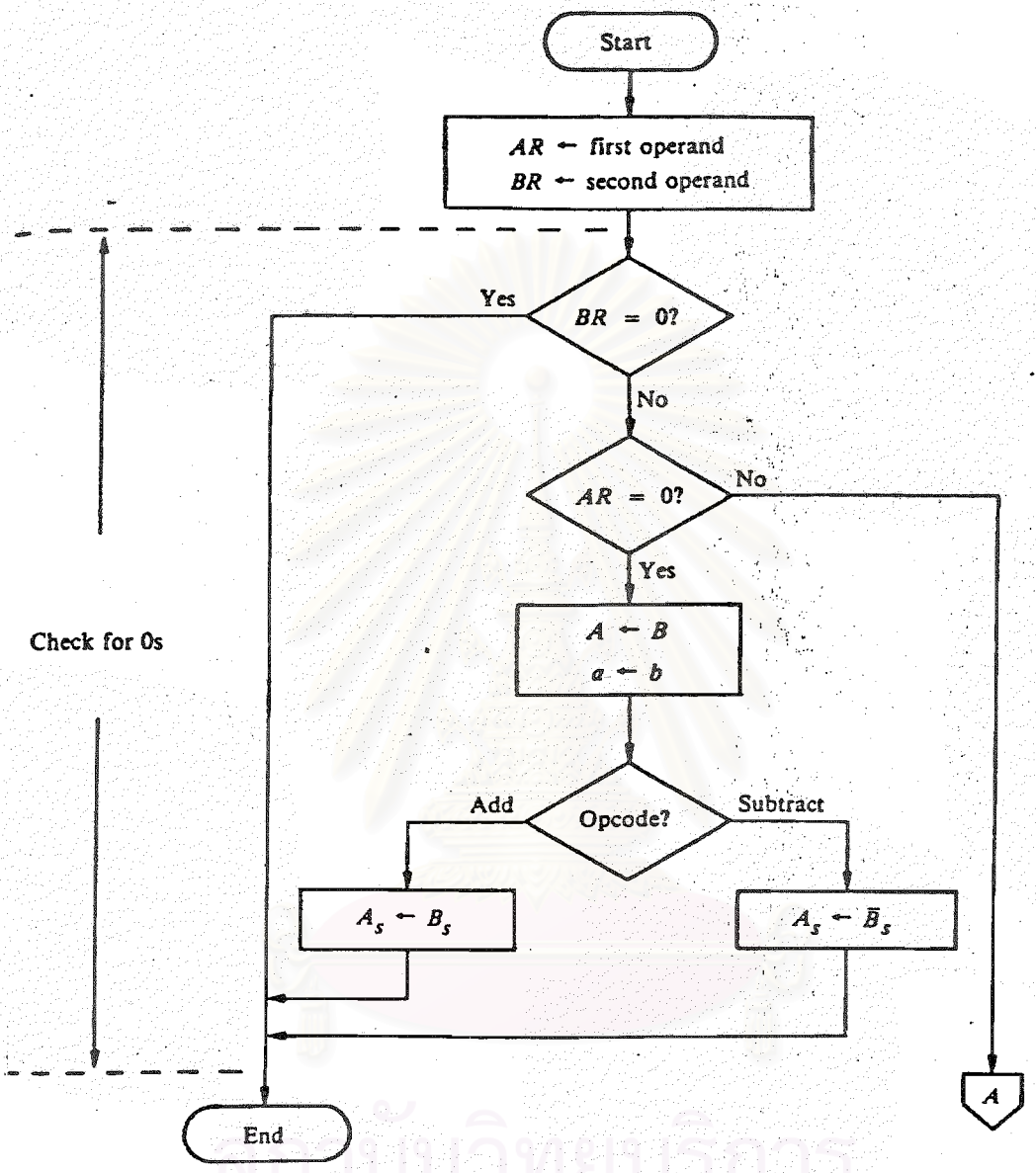
(ง) ให้ค่า exponent ของผลลัพธ์เท่ากับ ค่า exponent ของ Operand ที่มากกว่า

(จ) ทำ normalize ตัวเลขที่เป็นผลลัพธ์

(ฉ) ตรวจสอบการเกิด Overflow และ Underflow

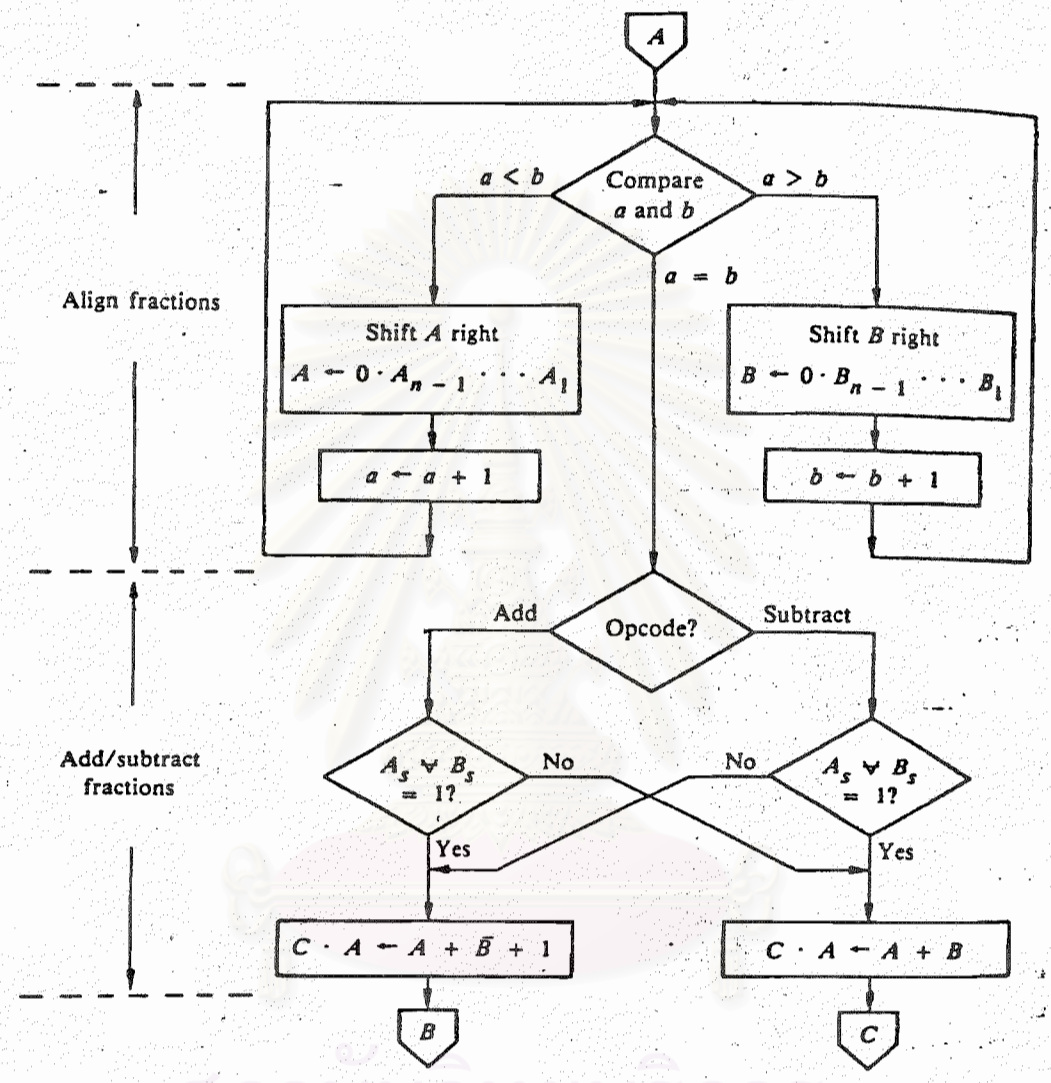
โฟว์ชาร์ทของโปรแกรมแสดงดังรูปที่ 5.21-5.24 อธิบายถึงการบวก ลบ ของค่า Operand 2 ตัว คือ

$$AR \leftarrow AR \pm BR$$



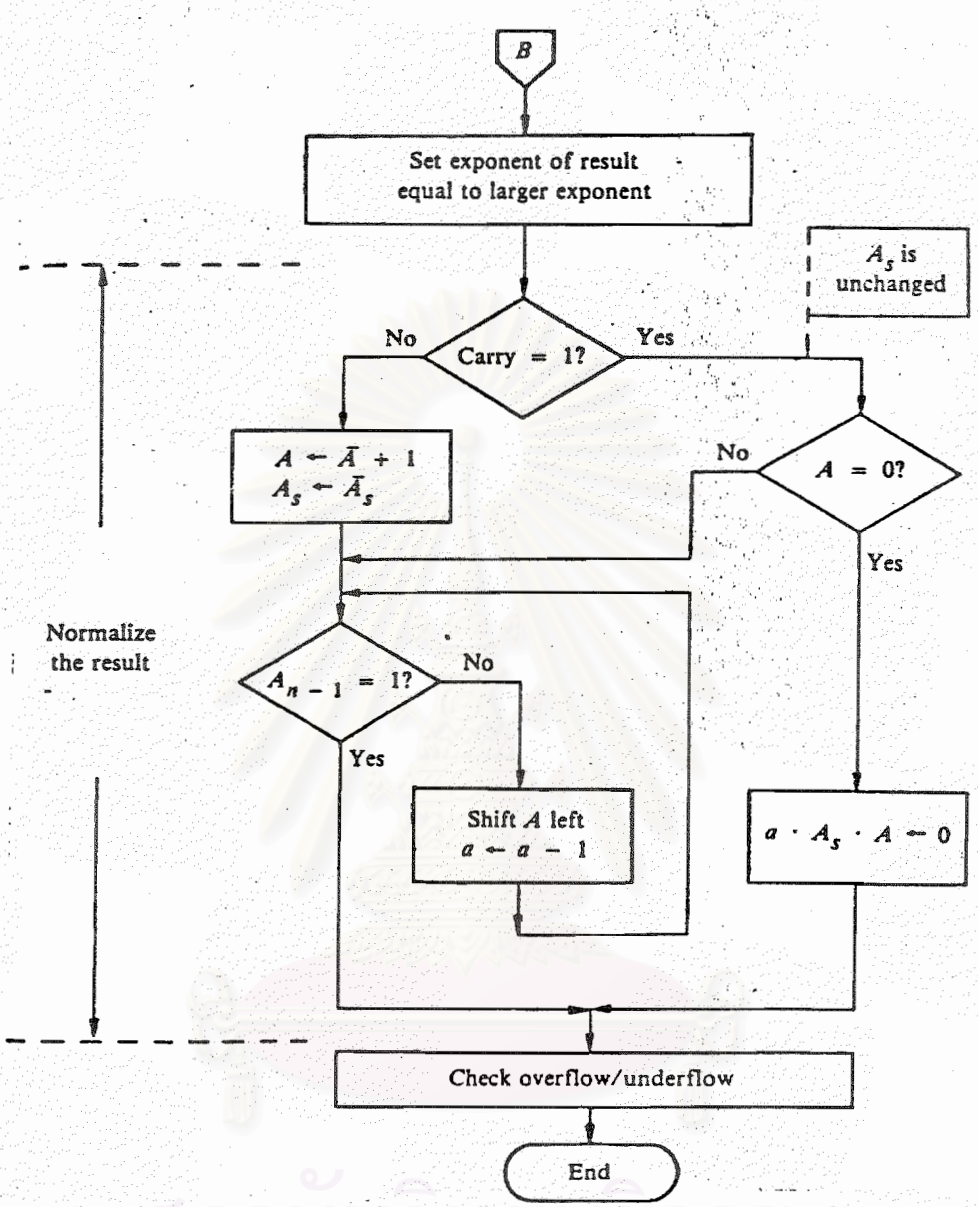
(a) การตรวจสอบค่า 0

รูปที่ 5.21 โฟว์ชาร์ทการบวกและลบเลขทศนิยม



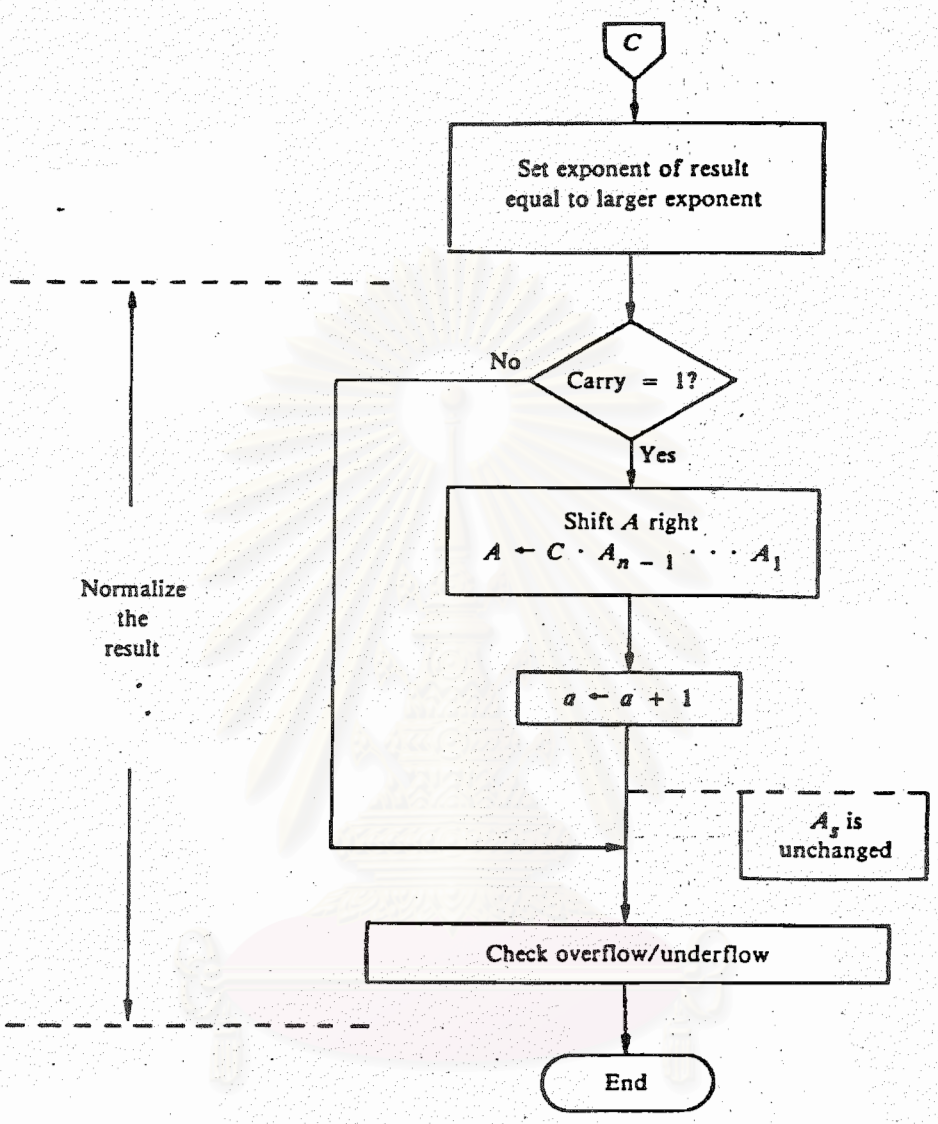
(b) align และบวกลบ mantissa

รูปที่ 5.22 โฟว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)



(c) หาค่า exponent และ normalize ค่าผลลัพธ์

รูปที่ 5.23 โฟว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)



(d) หาค่า exponent และ normalize ค่าผลลัพธ์

รูปที่ 5.24 โฟว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)

สัญญลักษณ์ที่ใช้ในโฟว์ชาร์ทมีดังนี้

AR = a.As.A

BR = b.Bs.B

a,b : ส่วน exponent ของ Operand AR,BR

As,Bs : บิทเครื่องหมายของ mantissa

A,B : ส่วน mantissa

V : exclusive OR

การทำงาน: ฟังก์ชันจะอ่านค่า Operand จากยอดของ stack S1 และ S2 โดยใช้ S2 เป็นตัวตั้งและใช้ S1 เป็นตัวบวก หรือ ลบ ผลลัพธ์จะเก็บบนยอดของ stack คือ S1

วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	LD P01	P01	X1		อ่านพารามิเตอร์
3	ADD	X1+P01			บวก

(2) คูณ

mnemonic: MUL

อัลกอริทึม: แบ่งเป็น 5 ส่วน (ดูรูปที่ 5.25-5.28) ดังนี้ [10]

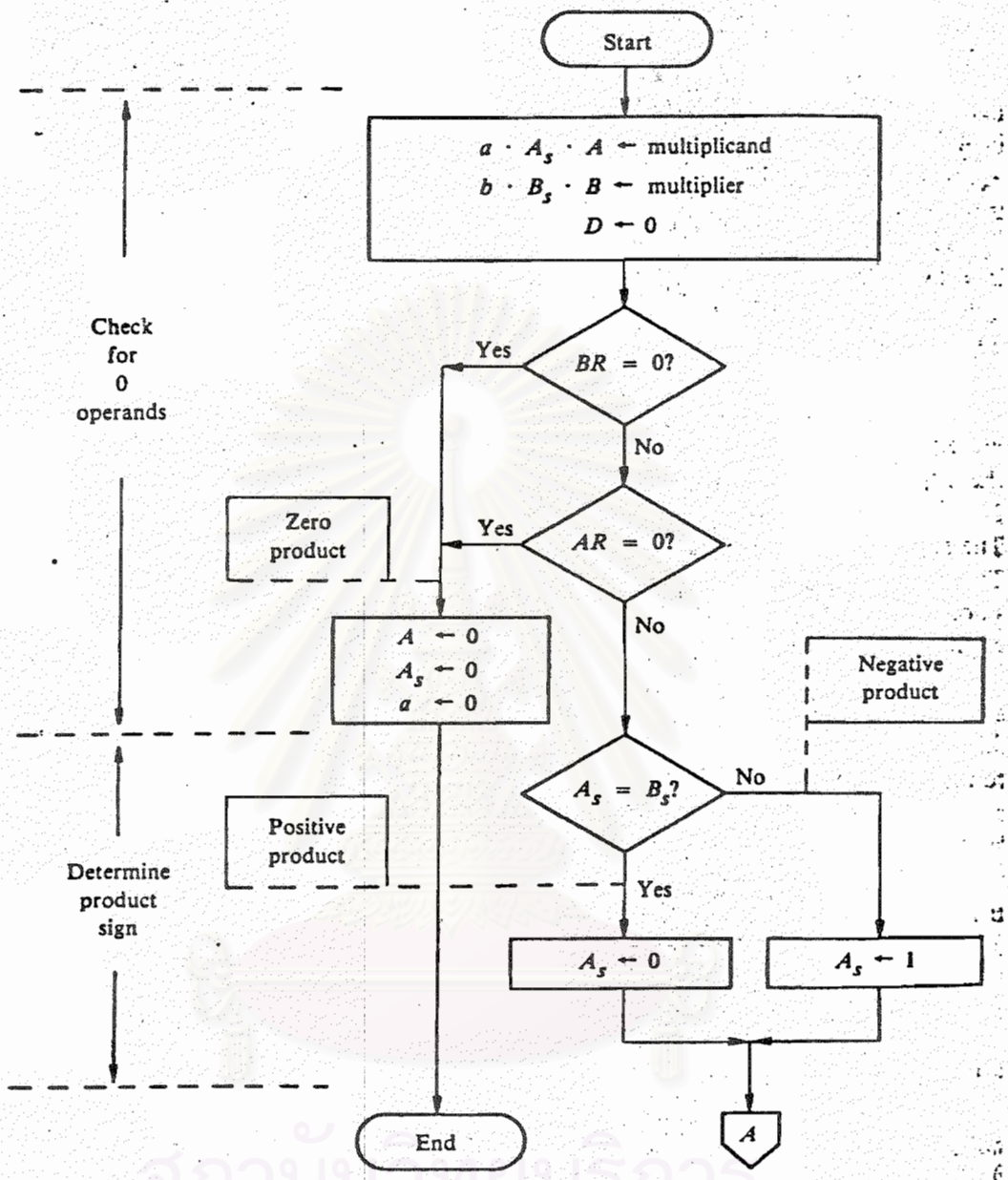
(ก) ตรวจสอบค่า Operand เป็น 0

(ข) หาเครื่องหมายของผลลัพธ์ที่ได้จากการคูณ

(ค) บวกค่า exponent

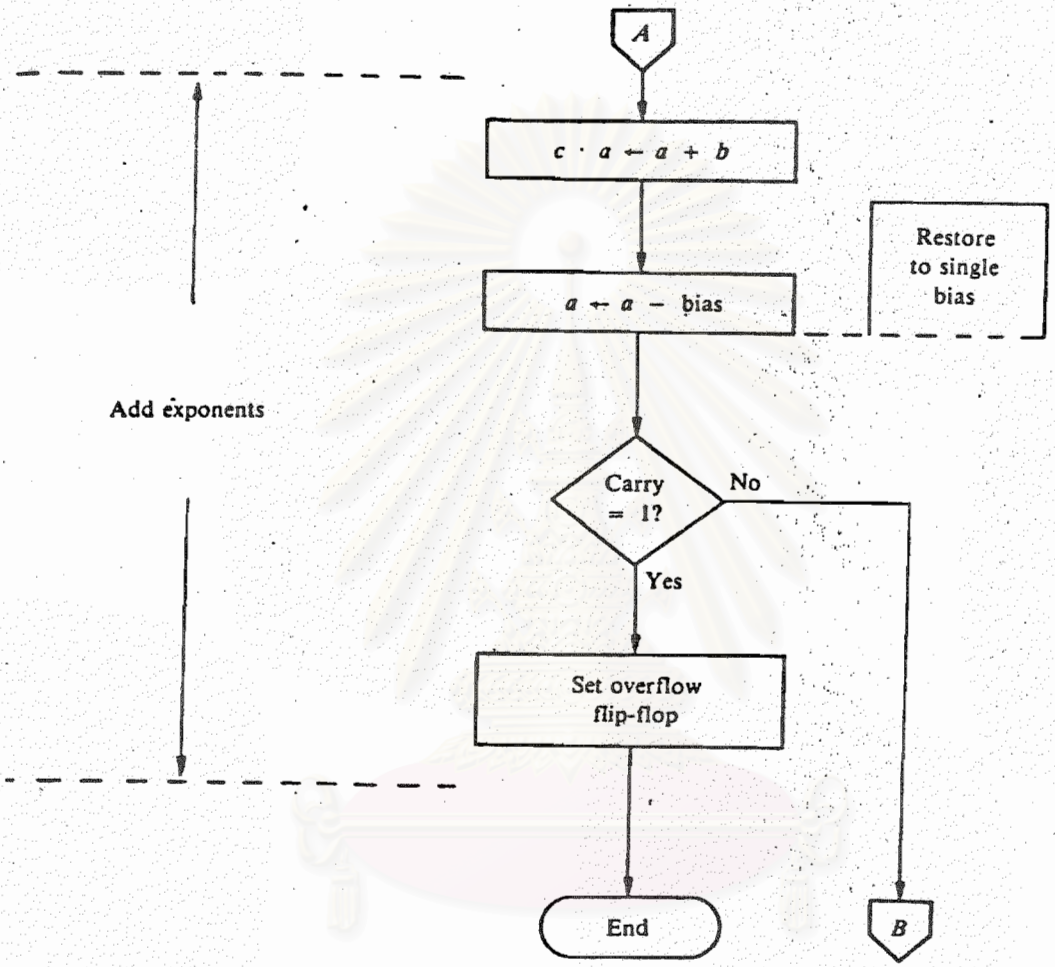
(ง) คูณค่า mantissa

(จ) normalize ผลลัพธ์



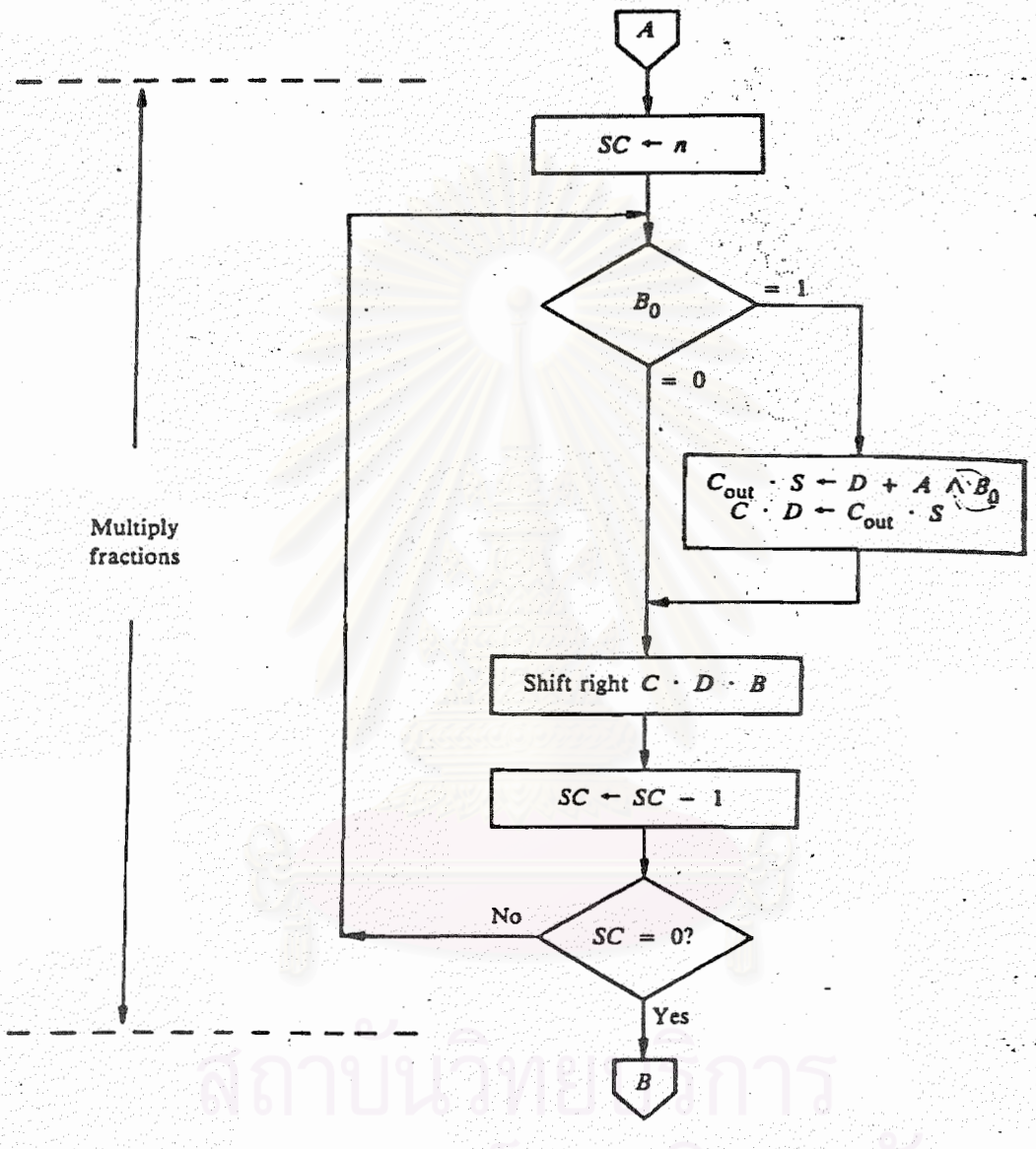
(a) ตรวจสอบค่า 0 และหาเครื่องหมายผลลัพธ์

รูปที่ 5.25 โฟว์ชาร์ทการคูณเลขทศนิยม



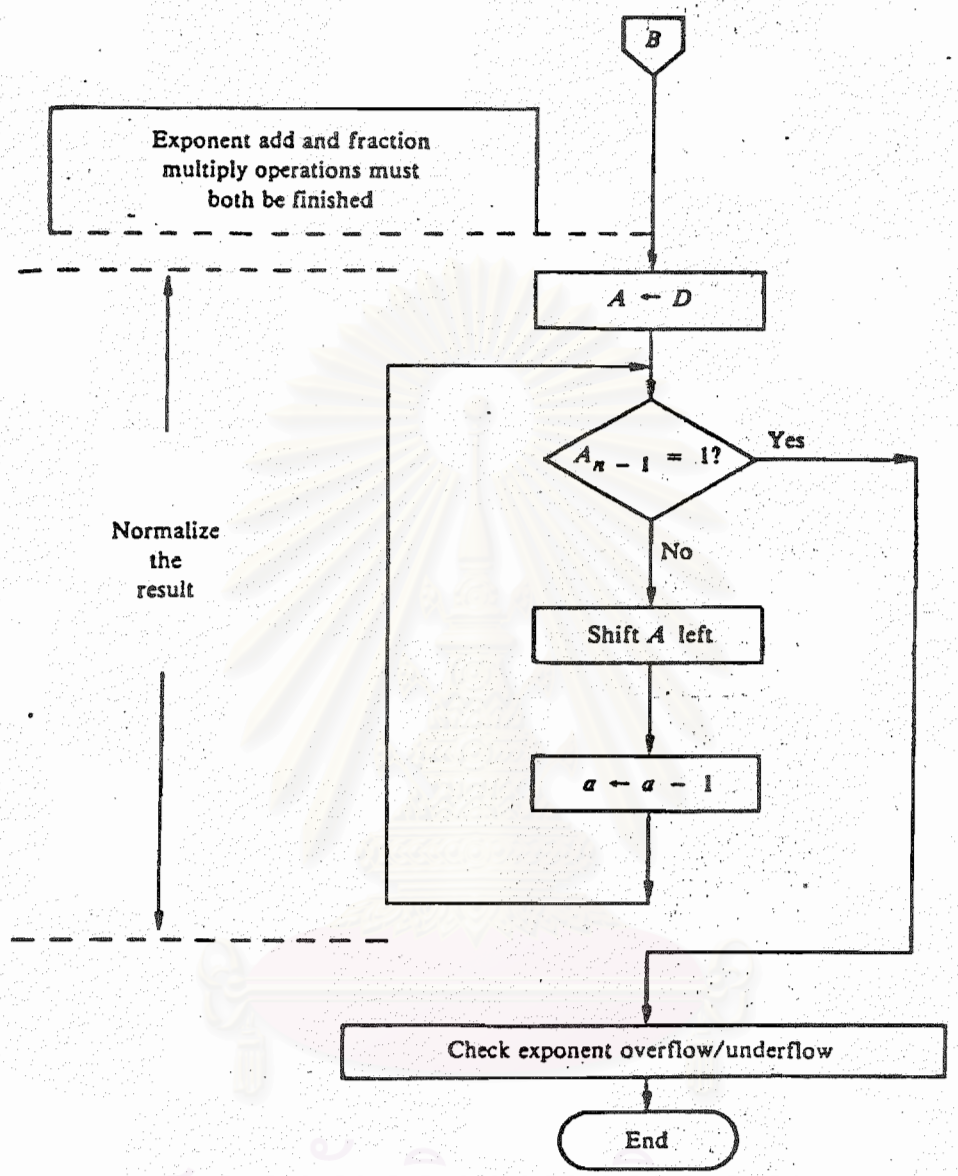
(b) บวกค่า exponent

รูปที่ 5.26 โฟว์ชาร์ทการคูณเลขทศนิยม (ต่อ)



(c) คูณค่า mantissa

รูปที่ 5.27 โฟว์ชาร์ทการคูณเลขทศนิยม (ต่อ)



(d) normalize ค่าผลลัพธ์

รูปที่ 5.28 โฟว์ชาร์ทการคูณเลขทศนิยม (ต่อ)

การทำงาน: ฟังก์ชันจะอ่านค่า Operand 2 ค่า จากยอดของ stack S1 , S2 มาคูณกัน ได้ผลลัพธ์เก็บไว้ที่ S1

วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	LD P01	P01	X1		อ่านพารามิเตอร์
3	MUL	X1*P01			คูณ

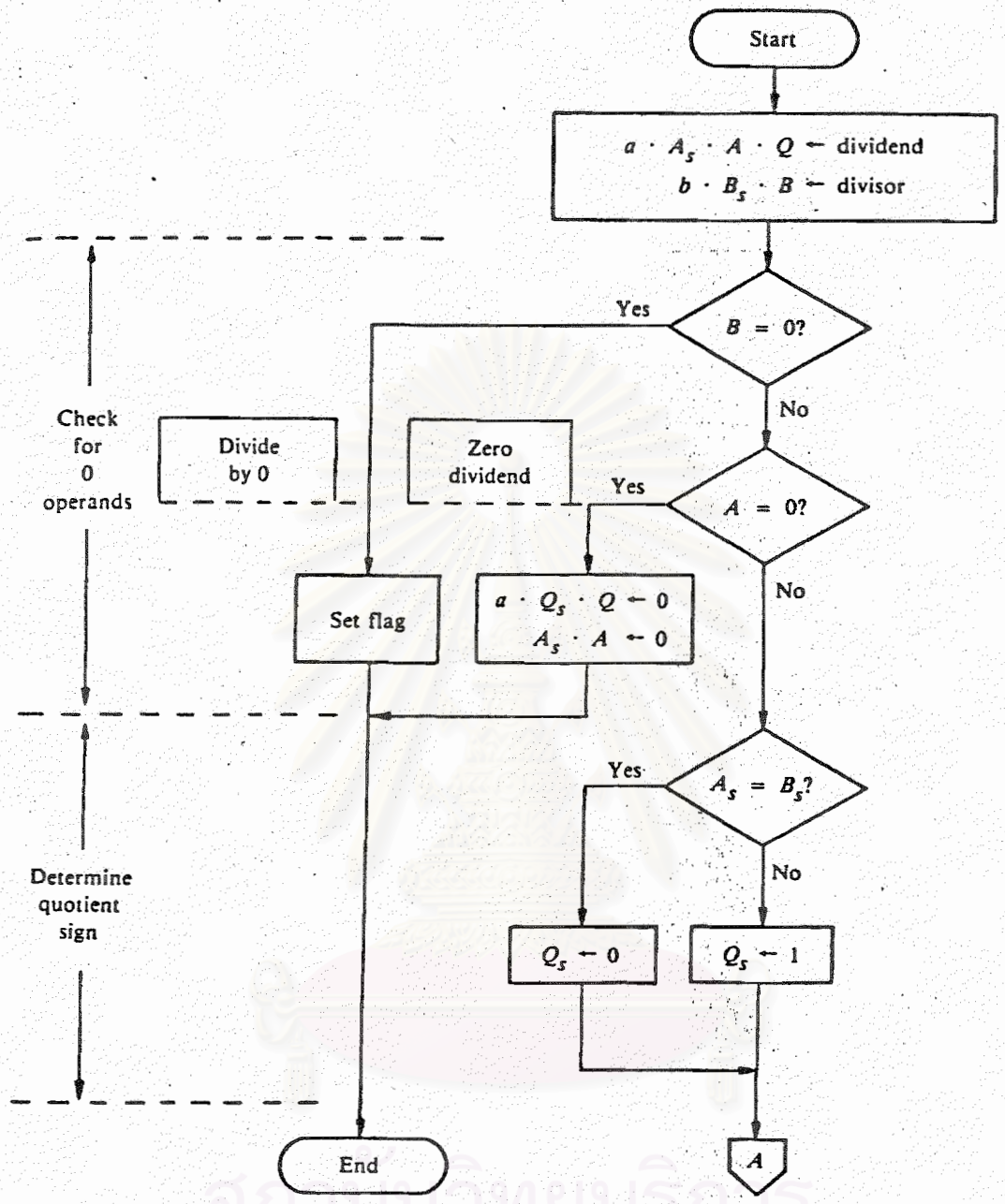
(3) หาร

mnemonic: DIV

อัลกอริทึม: แบ่งเป็น 5 ส่วน (ดูรูปที่ 5.29-5.32) ดังนี้ [10]

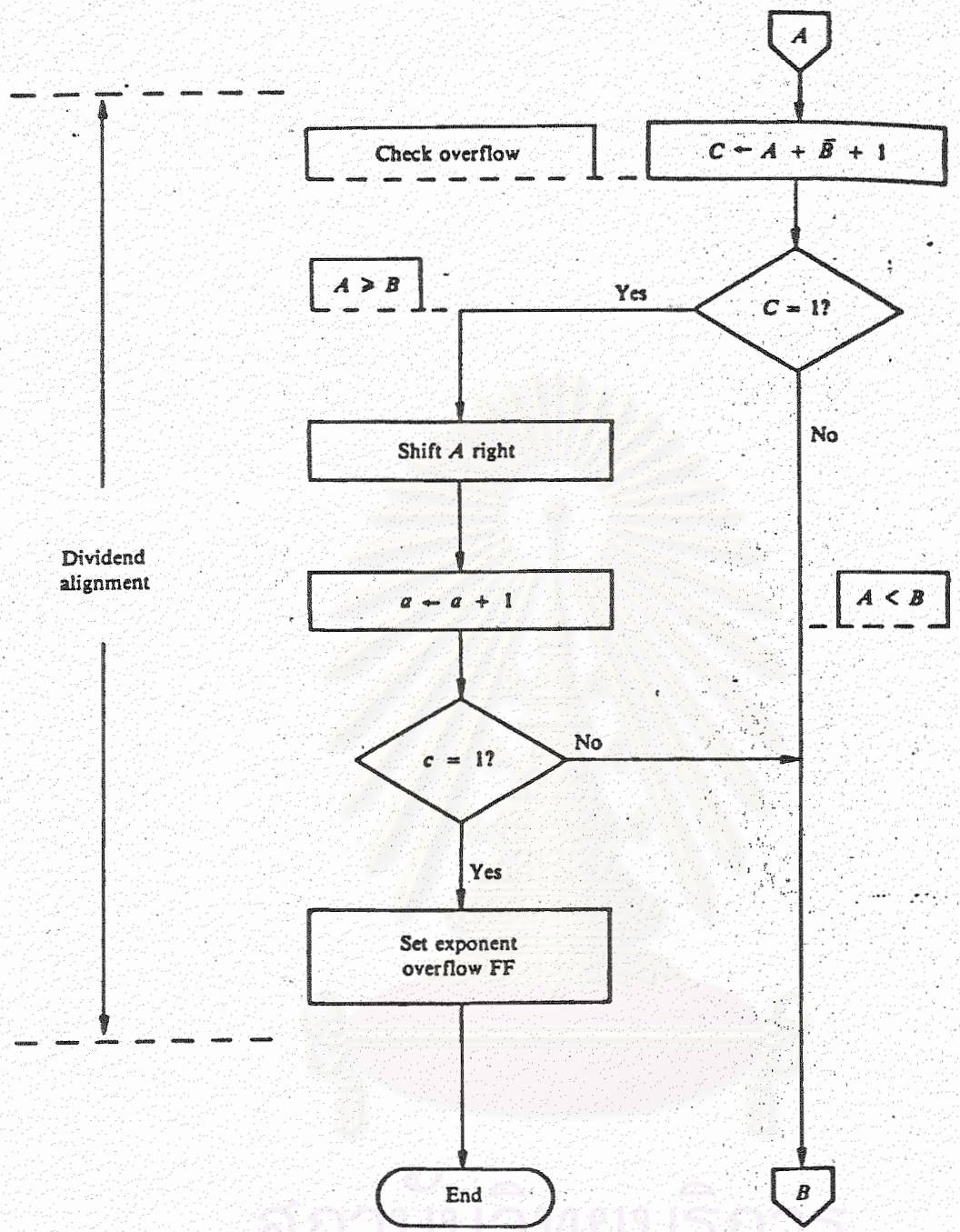
(ก) ตรวจสอบค่า Operand เป็น 0
(ข) หาเครื่องหมายของผลลัพธ์ที่ได้จากการหาร
(ค) จัดตัวตั้งสำหรับการหาร (Align the Dividend)
(ง) ลบค่า exponent
(จ) หารค่า mantissa

โปรแกรมหารจะใช้ Operand Q และ A คู่กันเพื่อเป็นตัวตั้ง ซึ่ง Q จะเป็นส่วนที่มีนัยสำคัญน้อยกว่า A ตารางที่ 5.6 แสดงค่าเริ่มต้น และผลลัพธ์ของ Operand ทั้งหมดที่ถูกใช้ในโปรแกรมมีดังนี้



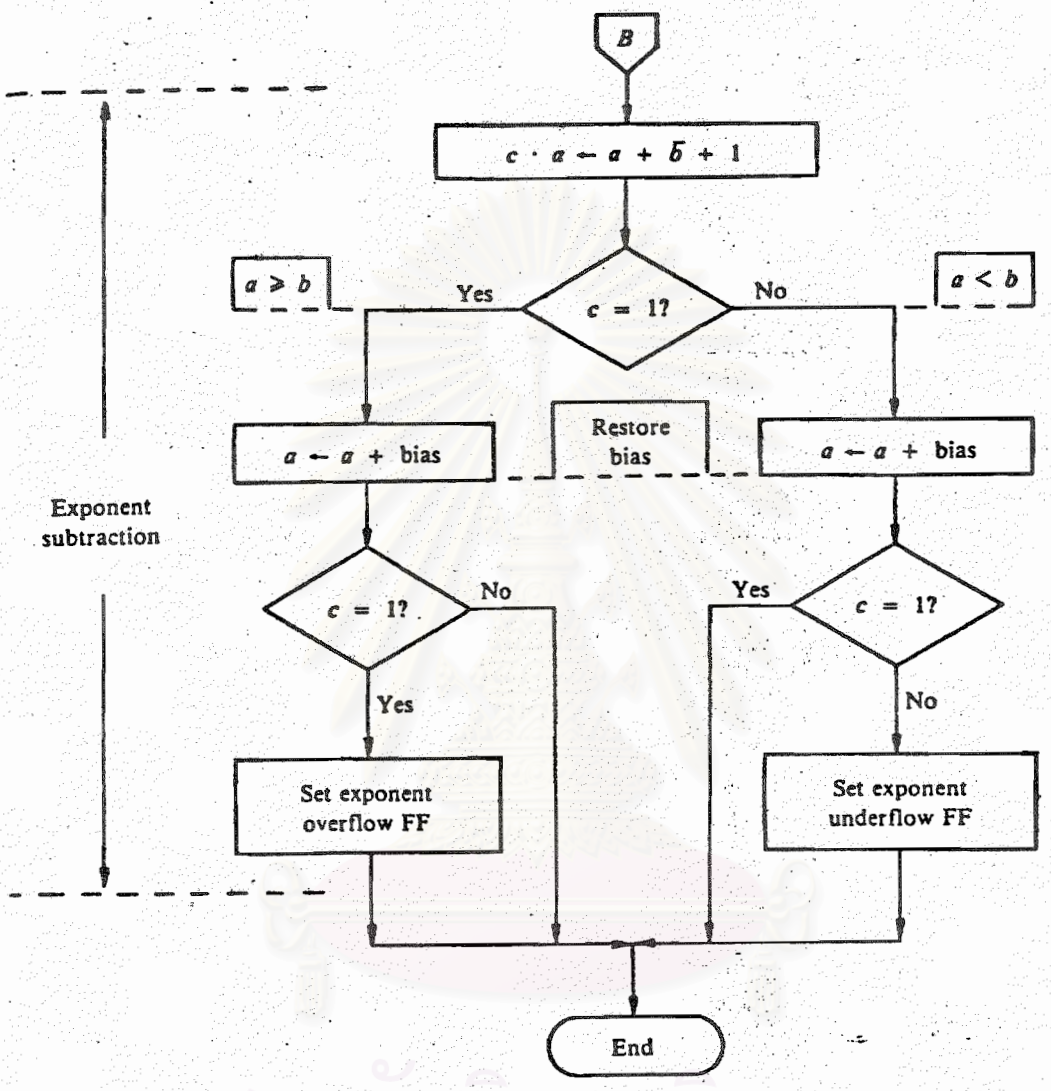
(a) ตรวจสอบค่า 0 และหาเครื่องหมายผลลัพธ์

รูปที่ 5.29 โฟว์ชาร์ทการหารเลขทศนิยม



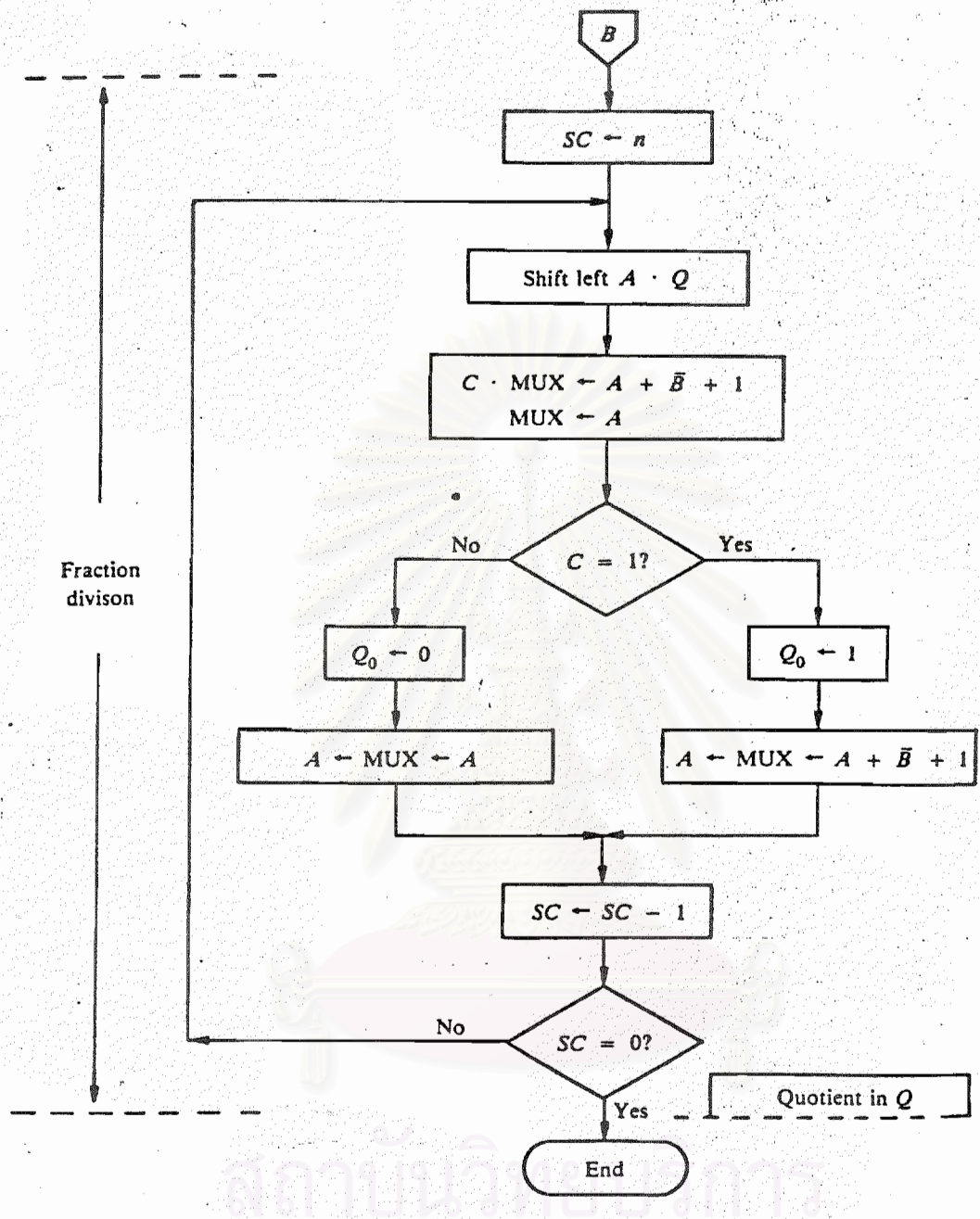
(b) align ค่าตัวตั้ง

รูปที่ 5.30 โฟว์ชาร์ทการหารเลขทศนิยม (ต่อ)



(c) ลบค่า exponent

รูปที่ 5.31 โฟว์ชาร์ทการหารเลขทศนิยม (ต่อ)



(d) หารค่า mantissa

รูปที่ 5.32 โฟว์ชาร์ทการคูณเลขทศนิยม (ต่อ)

ตารางที่ 5.6

Operand	ค่าเริ่มต้น	ผลลัพธ์
A	ตัวตั้งนัยสำคัญมาก	ค่าผลลัพธ์เศษ
Q	ตัวตั้งนัยสำคัญน้อย	ค่าผลลัพธ์
B	ตัวหาร	ตัวหาร

การทำงาน: ฟังก์ชันจะอ่านค่า Operand 2 ค่า จากยอดของ stack S1 , S2 มาหารกัน โดยค่า S2 เป็นตัวตั้ง และผลลัพธ์เก็บไว้ที่ S1

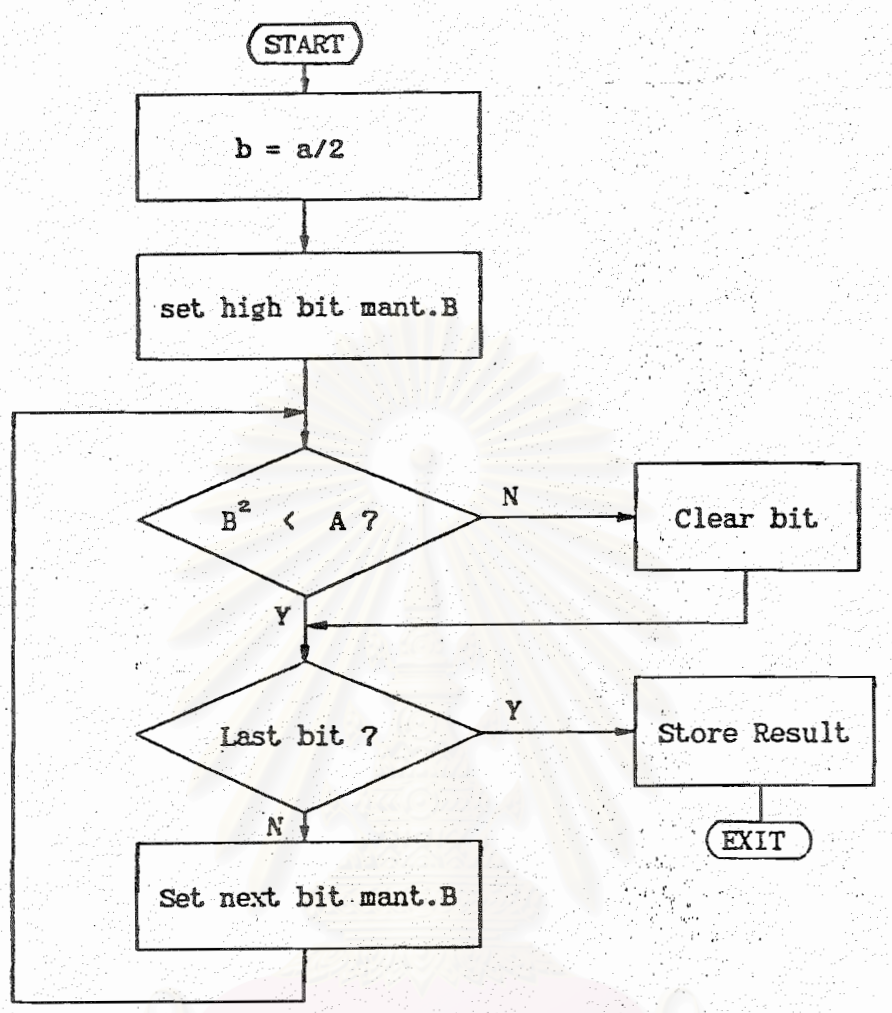
วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	LD P01	P01	X1		อ่านพารามิเตอร์
3	DIV	X1/P01			หาร

(4) ถอดรากที่สอง

mnemonic: SQR

อัลกอริทึม: เนื่องจากโครงสร้างของเลขทศนิยมที่ใช้ให้ค่าความถูกต้องหลังทศนิยมประมาณ 4-5 ตัว ดังนั้นถ้ามีการถอดรากที่สองค่าถูกต้องหลังทศนิยมจะมีจำนวนประมาณ 2 ตัว ซึ่งโปรแกรมถอดรากที่สองนี้ใช้อัลกอริทึมแบบ Successive ให้ค่าความถูกต้องของตัวเลขหลังทศนิยม 2 ตำแหน่ง มีโฟว์ชาร์ทแสดงการทำงานดังรูปที่ 5.33



รูปที่ 5.33 โฟว์ชาร์ทของฟังก์ชันถอดรากกำลังสอง (SQR)

การทำงาน: ฟังก์ชันจะอ่านค่า Operand 1 ค่า จากยอดของ stack คือ S1 และผลลัพธ์ของรากที่สองเก็บไว้ที่ S1

วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	SQR	X1			ถอดรากที่สอง

(5) ค่าสัมบูรณ์

mnemonic: ABS

อัลกอริทึม: เนื่องจากโครงสร้างของเลขทศนิยม มีบิทที่แสดงเครื่องหมายของตัวเลขอยู่ที่ส่วน exponent ดังนั้นการหาค่าสัมบูรณ์ทำได้โดยการเซ็ทบิทเครื่องหมายนี้ให้ เท่ากับ 0

การทำงาน: ฟังก์ชันจะอ่านค่า Operand 1 ค่า จากยอดของ stack คือ S1 และผลลัพธ์ของค่าสัมบูรณ์เก็บไว้ที่ S1

วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	ABS	X1			ค่าสัมบูรณ์

5.3.6.2 โปรแกรมทางตรรก

(1) AND,OR,NOT

mnemonic: AND
OR
NOT

การทำงาน: ฟังก์ชัน AND และ OR อ่านค่าจาก S1 และ S2 มาทำลอจิคแล้วให้ผลลัพธ์บน S1 สำหรับฟังก์ชัน NOT จะใช้ค่าและให้ผลลัพธ์บน S1 เท่านั้น

วิธีการโปรแกรม เมื่อ D1=1 D2=0

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD D1	1			
2	LD D2	0	1		
3	AND	0			
4	NOT	1			

(2) กระโดดแบบไม่มีเงื่อนไข

mnemonic: GO nn ; nn = เลขที่คำสั่งเป้าหมาย

อัลกอริทึม: โปรแกรมจะเปลี่ยน Instruction Pointerของโปรแกรมให้ชี้ไปยังตำแหน่งของคำสั่งที่ต้องการกระโดดไป

การทำงาน: ฟังก์ชัน GO จะกระโดดไปที่เลขที่คำสั่งที่ระบุในตอนท้ายของฟังก์ชัน โดยที่ค่าต่างๆภายใน stack จะไม่เปลี่ยนแปลง

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	GO 03	A	B	C	
2					
3		A	B	C	

(3) กระโดดแบบมีเงื่อนไข

mnemonic: GIF nn ; nn = เลขที่คำสั่งเป้าหมาย

การทำงาน: ฟังก์ชัน GO จะตรวจสอบค่าใน stack S1 ถ้ามีค่า 1 โปรแกรมจะกระโดดไปที่เลขที่คำสั่ง ที่ระบุในตอนท้ายของฟังก์ชัน โดยที่ค่าต่างๆภายใน stack จะเลื่อนขึ้นมา 1 ตำแหน่ง แต่ถ้ามีค่า 0 โปรแกรมจะทำคำสั่งถัดไป

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD D1	0/1	A	B	
2	GIF 04	A	B		S1=0 ⟵ 1 ⟵
3					
4		A	B		

(4) เปรียบเทียบค่า

mnemonic: CMP

การทำงาน: ฟังก์ชัน CMP นำค่าใน stack S1 และ S2 ทำการเปรียบเทียบ ถ้า S1 $\leq$ S2 โปรแกรมจะให้ค่า 1 ที่ S1 แต่ถ้า S1 > S2 จะให้ค่า 0 ที่ S1

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			
2	LD X2	X2	X1		
3	CMP	0/1			X2$\leq$X1, S1=1
					X2>X1, S1=0

5.3.6.3 โปรแกรมฟังก์ชันพื้นฐาน

(1) First Order Lag

mnemonic: LAGn ; n=1,4

อัลกอริทึม: สมการของฟังก์ชันในแบบต่อเนื่องเขียนได้ดังนี้

$$Y = (1-e^{-t/t_1}).X$$

t_1 = lag time (sec)

เขียนสมการใน s-domain ได้ดังนี้

$$Y(s) = X(s) / (1+t_1)$$

$$t_1 sY(s) + Y(s) = X(s)$$

สมการในรูป difference คือ

$$t_1 \frac{(Y_N - Y_{N-1})}{T_s} + Y_N = X_N$$

$$Y_N = Y_{N-1} + \frac{T_s}{(T_1+T_s)} (X_N+Y_{N-1}) \quad ..... \quad (5.6)$$

สมการ (5.6) เป็นสมการ discrete ที่ใช้ในการเขียนโปรแกรมหาค่าฟังก์ชันของ First Order Lag

การทำงาน: ฟังก์ชัน LAG นำค่าใน stack S1 เป็นค่าของ lag time และค่าใน S2 เป็นอินพุท

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			
2	LD P1	P1	X1		
3	LAG1	$X1(1-e^{-t/P1})$			

(2) First Order Lead

mnemonic: LEDn ; n=1,2

อัลกอริทึม: สมการของฟังก์ชันในแบบต่อเนื่องเขียนได้ดังนี้

$$Y = X.e^{-t/t2}$$

t_2 = lead time (sec)

เขียนสมการใน s-domain ได้ดังนี้

$$\frac{Y(s)}{X(s)} = \frac{t_2 s}{1+t_2 s}$$

$$(1+t_2 s)Y(s) = t_2 sX(s)$$

สมการในรูป difference คือ

$$(T_s+t_2)Y_N - t_2 Y_{N-1} = t_2(X_N - X_{N-1})$$

$$Y_N = \frac{t_2}{(t_2+T_s)} (Y_{N-1}+X_N-X_{N-1}) \quad ..... (5.7)$$

สมการ (5.7) เป็นสมการ discrete ที่ใช้ในการเขียนโปรแกรมหาค่าฟังก์ชันของ First Order Lead

การทำงาน: ฟังก์ชัน LED นำค่าใน stack S1 เป็นค่าของ lead time และค่าใน S2 เป็นอินพุท

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			
2	LD P1	P1	X1		
3	LED1	$X1(e^{-t/P1})$			

(3) Dead time

mnemonic: DEDn ; n=1,2

อัลกอริทึม: โปรแกรมจะจัดบัฟเฟอร์สำหรับหน่วงเวลา 20 ตัว เพื่อเก็บค่าอินพุทที่ถูกหน่วงเวลาในการให้เอาท์พุท ตามค่า deadtime ที่ระบุ โดยโปรแกรมจะอ่านข้อมูลทุกๆช่วงเวลาเท่ากับ (dead time)/20 วินาที เข้าที่บัฟเฟอร์ตัวแรก และเลื่อนค่าให้บัฟเฟอร์ตัวที่ 20 ออกมาเป็นเอาท์พุท กรณีที่ค่าช่วงเวลาในการ sampling น้อยกว่า (deadtime)/20 ค่าเอาท์พุทที่ได้จากฟังก์ชันคำนวณได้จากการ Interpolation ระหว่างค่าของเอาท์พุทปัจจุบันกับค่าบัฟเฟอร์ตัวที่ 20 ซึ่งทำให้สัญญาณที่ออกมาราบเรียบ [2]

กรณีที่ค่า deadtime มีค่าระหว่าง 1-3 วินาที จำนวนของบัฟเฟอร์ที่โปรแกรมใช้จะลดลงด้วย เพราะถ้าใช้ 20 ตัว การอ่านค่าเข้าบัฟเฟอร์แต่ละครั้งจะใช้เวลาน้อยกว่า 0.2 วินาที ซึ่งการสแกนอ่านค่าอินพุทของเครื่องใช้เวลามากกว่า ดังนั้นจึงเลือกจำนวนบัฟเฟอร์ 5-15 ตัวใช้กับ deadtime ที่มีค่าระหว่าง 1-3 วินาที

การทำงาน: ฟังก์ชันอ่านค่า deadtime จาก stack S1 และอ่านอินพุทจาก S2 ผลลัพธ์ของฟังก์ชันจะเก็บไว้ใน S1

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุท 1
2	LD P1	P1	X1		อ่าน dead time
3	DED1	$X1_{t-P1}$			

(4) Timer

mnemonic: DEDn ; n=1,2

การทำงาน: ฟังก์ชันอ่านข้อมูลจาก stack S1 ถ้า S1=0 ฟังก์ชันให้ค่า 0 ที่ S1 และถ้า S1=1 จะให้ค่าเวลาหน่วยเป็นวินาทีบน stack S1

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD D1	D1			อ่านดิจิตอลที่ 1
2	TIM1	time			

(5) Selector

mnemonic: HSL ; High selector
LSL ; Low selector

การทำงาน: ฟังก์ชันอ่านข้อมูลจาก stack S1 และ S2 โดยฟังก์ชันจะให้ค่าผลลัพธ์ที่มากกว่าที่ S1 สำหรับ HSL แต่จะให้ค่าที่น้อยกว่าบน S1 สำหรับฟังก์ชัน LSL

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุทที่ 1
2	LD X2	X2	X1		อ่านอินพุทที่ 2
3	HSL	X2			X2 > X1

(6) Interpolation

mnemonic: FX

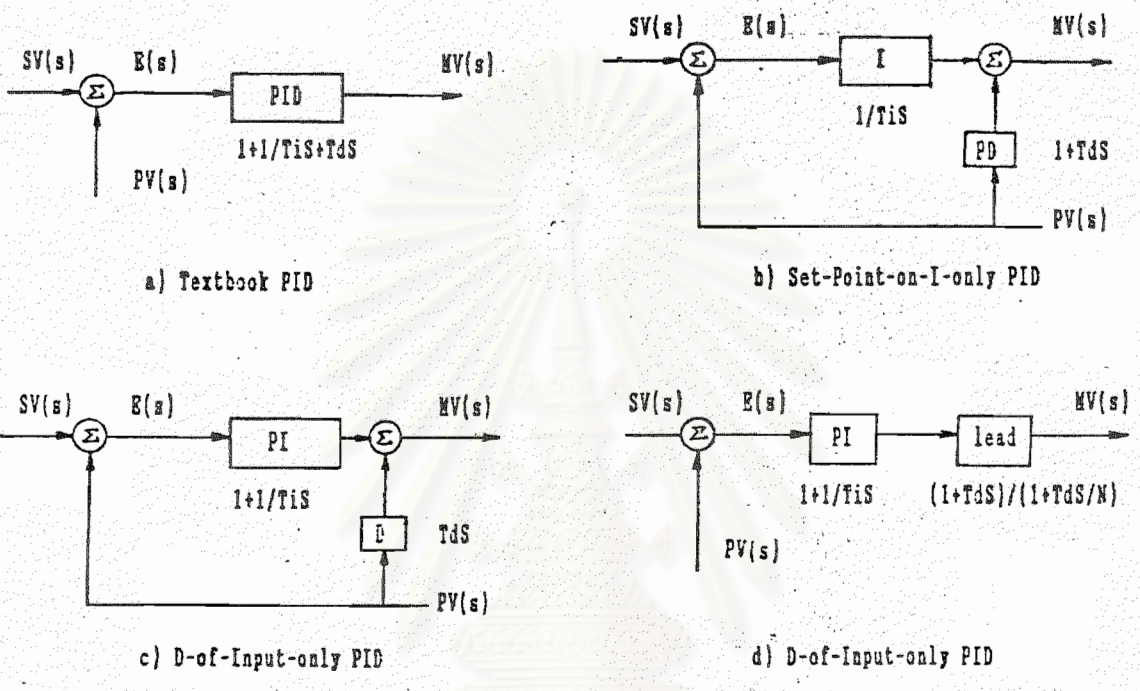
การทำงาน: ฟังก์ชันอ่านค่าอินพุทจาก stack S1 เพื่อคำนวณหาผลลัพธ์โดยวิธี Interporation จากค่าหลัก และค่าความชันของแต่ละ segment ซึ่งได้มาจากโปรแกรมสแกนแป้นนิมน์ด้านข้าง

วิธีการโปรแกรม

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่านอินพุทที่ 1
2	FX	ผลลัพธ์			Interpolation

5.3.6.4 โปรแกรมควบคุมแบบ PID

สมการของการควบคุมแบบ PID มีหลายแบบ ซึ่งสามารถแสดงในรูปของ s-domain ด้วยบล็อคไดอะแกรม ดังรูปที่ 5.34 [1,6]



รูปที่ 5.34 บล็อคไดอะแกรมของการควบคุม PID แบบต่างๆ

SV(S) - ค่าเป้าหมาย (Set Point Variable)
PV(S) - สัญญาณโปรเซส (Process Variable)
E(S) - ผลต่างระหว่างค่าเป้าหมายและสัญญาณป้อนกลับ
MV(S) - สัญญาณควบคุม (Manipulated Signal)

ซึ่งในการควบคุมอุตสาหกรรม (Industrial Process Control) เครื่องควบคุมจะทำการปรับค่าสัญญาณควบคุม เพื่อทำให้ค่าสัญญาณโปรเซสมีค่าเท่ากับ ค่าเป้าหมายตลอดเวลา ในงานควบคุมบางประเภทค่าเป้าหมายจะมีค่าคงที่ (Fixed Set Point Control) แต่ในงานควบคุมอีกประเภทค่าเป้าหมายจะเปลี่ยนแปลงตามเวลา (Follow-up Control) ซึ่งจากโครงสร้างของสมการทั้งหมดสามารถเลือกแบบที่เหมาะสมกับการควบคุมทั้งสองประเภทได้ดังนี้

รูปแบบ (b) เหมาะกับการควบคุมที่ค่าเป้าหมายคงที่ สมการที่ใช้คำนวณ คือ

$$MV = \frac{100}{PB}\left(PV + \frac{1}{T_I s}E + \frac{T_D S}{1+\frac{T_d s}{N}}PV\right) \quad \ldots\ldots\ldots (5.8)$$

รูปแบบ (c) เหมาะกับการควบคุมที่ค่าเป้าหมายเปลี่ยนแปลงตามเวลา สมการคือ

$$MV = \frac{100}{PB}\left(E + \frac{1}{T_I s}E + \frac{T_D S}{1+\frac{T_d s}{N}}PV\right) \quad \ldots\ldots\ldots (5.9)$$

ในส่วนของ derivative จะมีสมการของ Low Pass Filter ที่มีความถี่ cut off สูงกว่า $1/T_D$ เท่ากับ N เท่า รวมอยู่ด้วย เพื่อบั่นทอนสัญญาณรบกวนที่ความถี่สูง [6,11]

ซึ่งการหาสมการในรูป discrete เพื่อใช้กับไมโครโปรเซสเซอร์ จะประมาณค่าอนุพันธ์โดยใช้วิธีแบบ "Four point central difference" [8] ดังสมการที่ (5.10)

$$\frac{\Delta PV}{T_s} = \frac{1}{6T_s}(PV_n - PV_{n-3} + 3PV_{n-1} - 3PV_{n-2}) \quad \ldots\ldots (5.10)$$

ฟังก์ชันการควบคุมแบบ PID แบ่งออกเป็น 2 โปรแกรม คือ

(1) **Basic PID**

mnemonic: BPID

อัลกอริทึม: ใช้สมการ (5.8) ในการคำนวณหาสัญญาณควบคุม ซึ่งสามารถหาสมการ discrete ได้ดังนี้

ให้ $MV = A + B + C$

$A_n = K_p PV_n(s)$

โดย $K_p = 100/PB$

$A_n = K_p PV_n \quad \ldots\ldots\ldots (5.11)$

$$B = \frac{K E(s)}{T_i s}$$

$$\frac{B_n - B_{n-1}}{T_s} = \frac{K_p E_n}{T_i}$$

$$B_n = B_{n-1} + \frac{K_p T_s}{T_i} E_n \quad \ldots\ldots\ldots \quad (5.12)$$

$$C_n = \frac{T_D SPV}{1+T_f s}$$

ให้ $T_f = T_D/8$

$$C_n + T_f\frac{(C_n - C_{n-1})}{T_s} = \frac{K_p T_D}{6T_s}(PV_n - PV_{n-3} - 3PV_{n-2} + 3PV_{n-1})$$

$$C_n = \frac{T_f C_{n-1}}{T_s+T_f} + \frac{K_p T_D}{6(T_s+T_f)}(PV_n - PV_{n-3} - 3PV_{n-2} + 3PV_{n-1}) \quad ..(5.13)$$

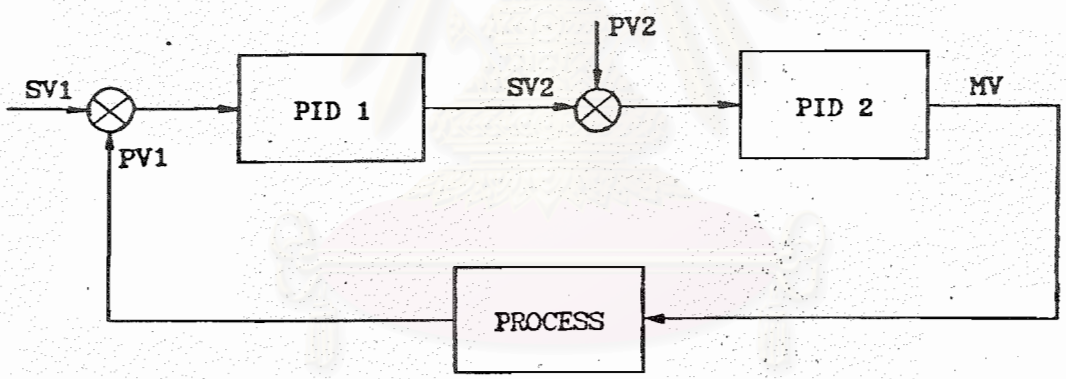
สมการที่ (5.11-5.13) จะถูกนำมาใช้เพื่อคำนวณหาค่าสัญญาณควบคุมของการควบคุมแบบ PID ซึ่งโฟว์ชาร์ทของฟังก์ชัน BPID แสดงได้ดังรูปที่ 5.35 โดยก่อนการคำนวณโปรแกรมจะตรวจสอบค่าสัญญาณโปรเซส ซึ่งถ้ามากกว่าค่า PH (Process variable high alarm) โปรแกรมจะเซ็ทค่ารีจีสเตอร์ FL0 และถ้าค่าน้อยกว่า PL (Process variable low alarm) รีจีสเตอร์ FL1 จะถูกเซ็ท ซึ่งค่ารีจีสเตอร์เหล่านี้สามารถถูกส่งออกไปควบคุมรีเลย์เพื่อเตือนภัยได้ หลังจากตรวจสอบค่า PV โปรแกรมจะคำนวณสมการ PID และจำกัดค่า MV (Manipulated Variable) ให้มีค่าเท่ากับ MH หรือ ML กรณีที่มีค่ามากกว่าค่า MH (Manipulated High Limit) หรือน้อยกว่าค่า ML (Manipulated Low Limit) ตามลำดับ ซึ่งพารามิเตอร์ PH,PL,MH และ ML ถูกกำหนดค่าโดยใช้แป้นพิมพ์ด้านข้าง

การทำงาน: ฟังก์ชันอ่านค่า Process variable จาก stack S1 และให้ค่าผลลัพธ์บน S1

(2) Cascade PID

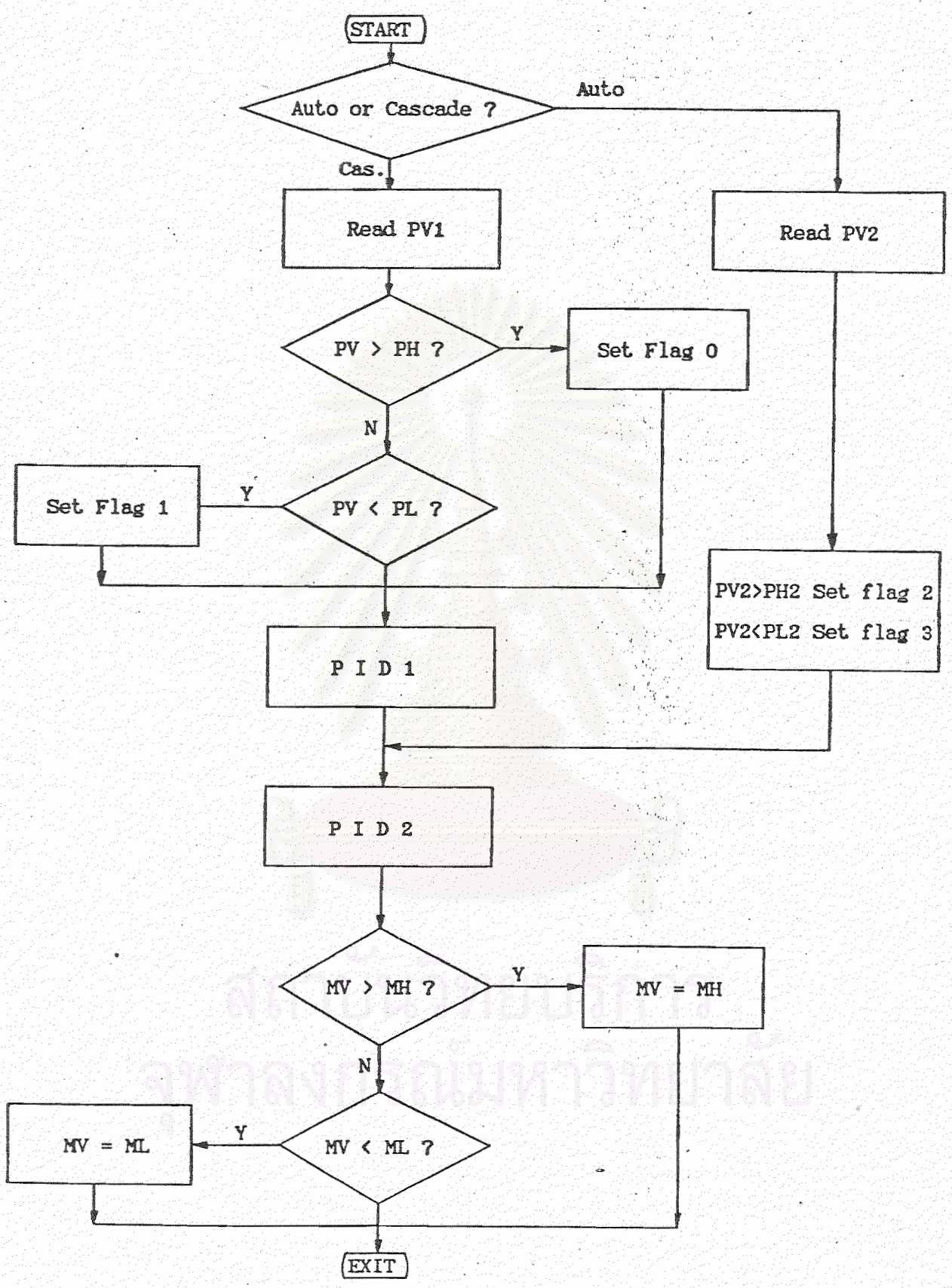
mnemonic: CPID

อัลกอริทึม: บล็อคไดอะแกรมของโปรแกรมควบคุมแบบ Cascade แสดงดังรูปที่ 5.36 ซึ่งการควบคุมแบบ Cascade จะประกอบด้วยลูปการควบคุมแบบ PID 2 ลูป โดยในลูปแรกจะเป็นการควบคุมแบบค่าเป้าหมายคงที่ และลูปที่ 2 เป็นการควบคุมแบบค่าเป้าหมายเปลี่ยนแปลงตามเวลา ดังนั้นจึงใช้สมการที่ (5.8) ในการคำนวณลูปที่ 1 และสมการที่ (5.9) ในลูปที่ 2 ซึ่งการคำนวณหาสมการ discrete ใช้วิธีเดียวกับโปรแกรม BPID



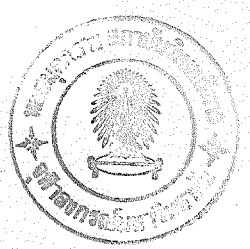
รูปที่ 5.36 บล็อคไดอะแกรมการควบคุมแบบ Cascade

โฟว์ชาร์ทของโปรแกรม CPID (ดูรูปที่ 5.37) มีการตรวจสอบค่า PV และ MV เหมือนกับฟังก์ชัน BPID แต่ CPID จะมีโหมดการทำงานมากกว่า โดยแบ่งเป็นแบบ Auto และ Cascade การทำงานในโหมด Auto ฟังก์ชันจะตัดการคำนวณในลูปที่ 1 ออก ทำให้การควบคุมในลูปที่ 2 (ลูปใน) ทำงานเป็นอิสระ เพื่อประโยชน์ในการปรับหาค่า PB, Ti และ Td ที่เหมาะสมสำหรับลูปที่ 2 ก่อนที่จะต่อเข้ากับลูปที่ 1 เพื่อทำงานในโหมด Cascade



รูปที่ 5.37 โฟว์ชาร์ทของฟังก์ชัน CPID

การทำงาน: ฟังก์ชันอ่านค่า PV1 จาก S1 และค่า PV2 จาก S2 โดยหลังจากการคำนวณจะให้ผลลัพธ์บน S1

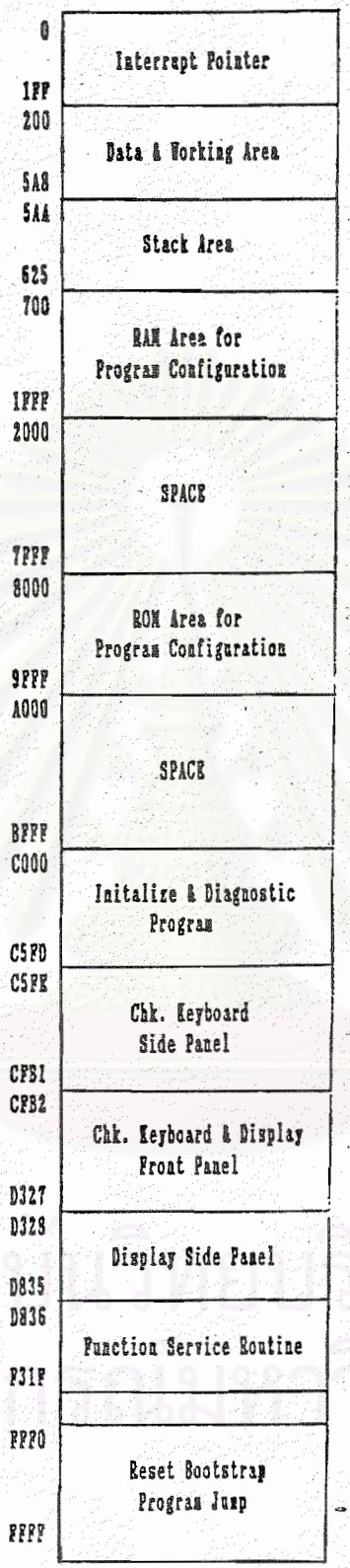


วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X2	X2			อ่าน PV2
2	LD X1	X1	X2		อ่าน PV1
2	CPID	MV			คำนวณสมการ CPID
3	ST Y1	MV			ส่งค่า MV ช่องที่ 1

5.4 SYSTEM MEMORY MAPPING

โปรแกรมควบคุมระบบของเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ ใช้หน่วยความจำสำหรับเก็บโปรแกรมควบคุมระบบประมาณ 16 KB (ไอซี 2764 2 ตัว) โปรแกรมกำหนดรูปแบบการควบคุมใน ROM ที่ตำแหน่ง 8000 (ไอซี 2764 1 ตัว) กรณีโปรแกรกำหนดรูปแบบอยู่ใน RAM จะใช้ตำแหน่ง 700 ใช้ไอซี 6264 ซึ่งเป็นพื้นที่เหลือจากการใช้ RAM ในการเก็บข้อมูล, Working Area และ Interrupt Pointer รายละเอียดการจัด MEMORY MAP แสดงได้ดังรูปที่ 5.38



รูปที่ 5.38 SYSTEM MEMORY MAP

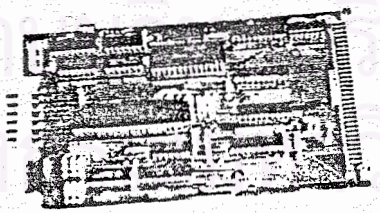
บทที่ 6

การสร้างเครื่องควบคุมและการทดสอบ

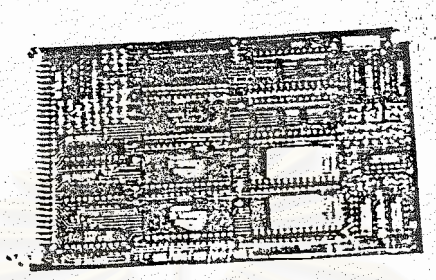
6.1 การสร้างเครื่องควบคุม

(1) บอร์ดหลักของระบบ (System main board) ใช้ card system SDA-88 [12] มีลักษณะของฮาร์ดแวร์เป็น rack โดยจะแบ่งวงจรออกเป็นบอร์ดตามหน้าที่การทำงาน ซึ่งแต่ละบอร์ดจะเชื่อมโยงกันผ่านบัส UROCON ประกอบด้วยบอร์ด 3 บอร์ด ดังนี้

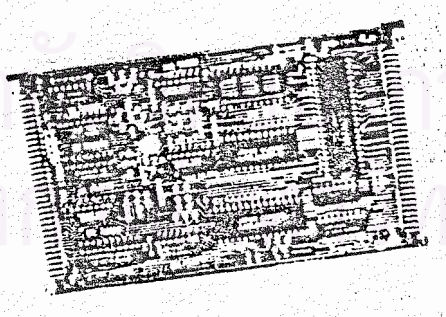
- CPU Board (รูปที่ 6.1)
- Memory Board (รูปที่ 6.2)
- Timer & Communication (รูปที่ 6.3)



รูปที่ 6.1 แผ่นวงจรซีพียู

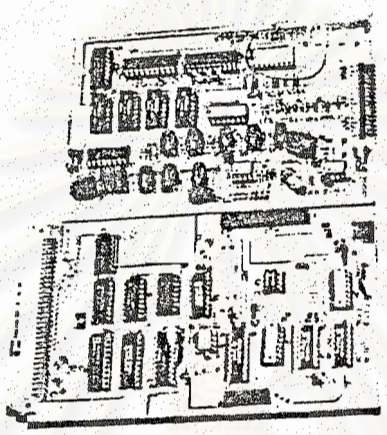


รูปที่ 6.2 แผ่นวงจรหน่วยความจำ



รูปที่ 6.3 แผ่นวงจร Timer & Communication

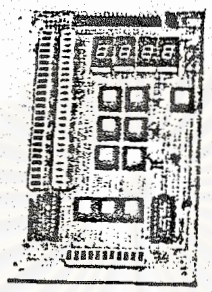
(2) วงจรอินพุทและเอาท์พุท อินเตอร์เฟสกับสัญญาณภายนอกทั้งแบบอนาลอค และดิจิตอล ได้ออกแบบเป็นแผ่นวงจรพิมพ์ ดังรูปที่ 6.4



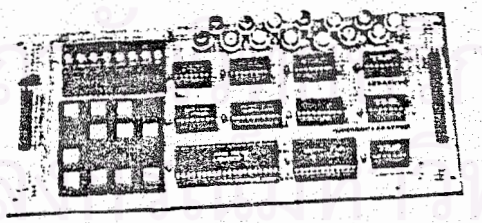
รูปที่ 6.4 แผ่นวงจรอินพุทและเอาท์พุทของเครื่องควบคุมเชิงเลข

(3) วงจรแป้นพิมพ์และแสดงผล ออกแบบและต่อวงจรด้วยการเดินสาย Wire Wrap แบ่งเป็น 2 บอร์ด คือ

- บอร์ดแสดงผลและแป้นพิมพ์ด้านหน้า (รูปที่ 6.5)
- บอร์ดแสดงผลและแป้นพิมพ์ด้านข้าง (รูปที่ 6.6)



รูปที่ 6.5 แผ่นวงจรส่วนแสดงผลและแป้นพิมพ์ด้านหน้า



รูปที่ 6.6 แผ่นวงจรแสดงผลและแป้นพิมพ์ด้านข้าง

6.2 การทดสอบซอฟแวร์ของเครื่องควบคุม

การพัฒนาซอฟแวร์มีขั้นตอน ดังนี้

(1) เขียนโปรแกรมควบคุมระบบโดยใช้ภาษา PLM-86 [13] และ ASSEMBLY [14] ซึ่งจะได้โปรแกรม file.plm และ file.asm ตามลำดับ การเขียนโปรแกรมจะแบ่งเขียนเป็นโมดูล เพื่อง่ายต่อการทดสอบและแก้ไขโปรแกรม

(2) ใช้ Compiler PLM86 แปลง file.plm เป็น file.obj
ใช้ Compiler ASM86 แปลง file.asm เป็น file.obj

(3) การแปลง file.obj เป็น file.h (Intel Hex 16 บิท) ใช้ iAPX 86,88 Family Utilities [15] ดังนี้

- LINK86 รวมหลายๆ file.obj เป็น 1 โมดูล ได้ file.lnk
- LOC86 กำหนดตำแหน่งของ Segment Register ต่างๆของโปรแกรม โดยเปลี่ยน file.lnk เป็น file.loc
- OH86 เปลี่ยน file.loc เป็น file.h

(4) ใช้โปรแกรม INTELH ซึ่งเขียนขึ้นเองเพื่อเปลี่ยน file.h เป็น file.hex (Intel Hex 8 บิท) เนื่องจากการเขียนโปรแกรมลงบน ROM โดยใช้ EPROM Programmer CLK3000 และ การโหลด file.hex ลงบนโปรแกรม DEBUG ต้องใช้ไฟล์ที่มี ฟอร์แมท Intel Hex 8 บิท

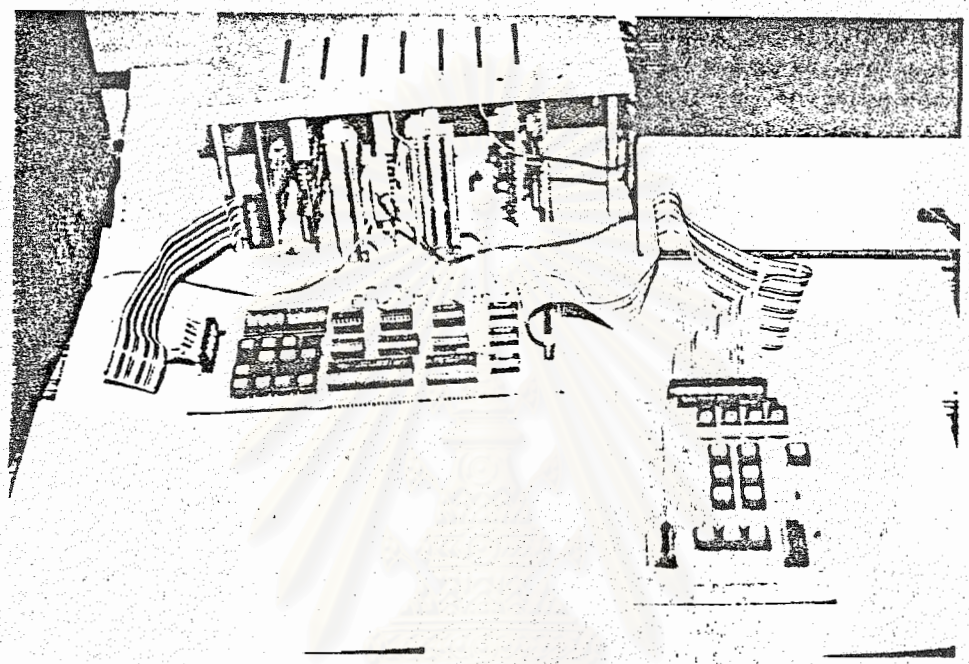
(5) การทดสอบซอฟแวร์ขณะพัฒนาโปรแกรม แบ่งการทดสอบเป็น 2 ลักษณะ ดังนี้

- ทดสอบบนไมโครคอมพิวเตอร์ โดยไม่ต้องใช้ฮาร์ดแวร์ของเครื่องควบคุม เช่น โปรแกรมการบวก ลบ คูณ หาร เลขทศนิยม การทดสอบทำโดยใช้โปรแกรม DEBUG อ่าน file.hex มาทดสอบ

- ทดสอบบนฮาร์ดแวร์ของเครื่องควบคุม บางโปรแกรมจำเป็นต้องใช้ฮาร์ดแวร์ เช่น โปรแกรมสแกนอนาลอคอินพุท เป็นต้น การทดสอบทำโดยใช้โปรแกรม DLFAST ซึ่งเขียนขึ้นเอง โหลด file.hex จากไมโครคอมพิวเตอร์ผ่าน Serial Comm. ลงใน RAM ของเครื่องควบคุม และทำการทดสอบโปรแกรม

6.3 การทดสอบเครื่องควบคุม

เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ ที่ใช้ทดสอบแสดงดังรูปที่ 6.7



รูปที่ 6.7 เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ใช้ในการทดสอบ

6.3.1 การทดสอบฟังก์ชันการทำงาน

ทดสอบโดยโหลดแต่ละโมดูลของฟังก์ชันลงในหน่วยความจำของเครื่องควบคุม โดยใช้โปรแกรม DLFAST และสั่งให้โปรแกรมฟังก์ชันนั้นทำงานและตรวจสอบผลการทดสอบจากหน่วยความจำ

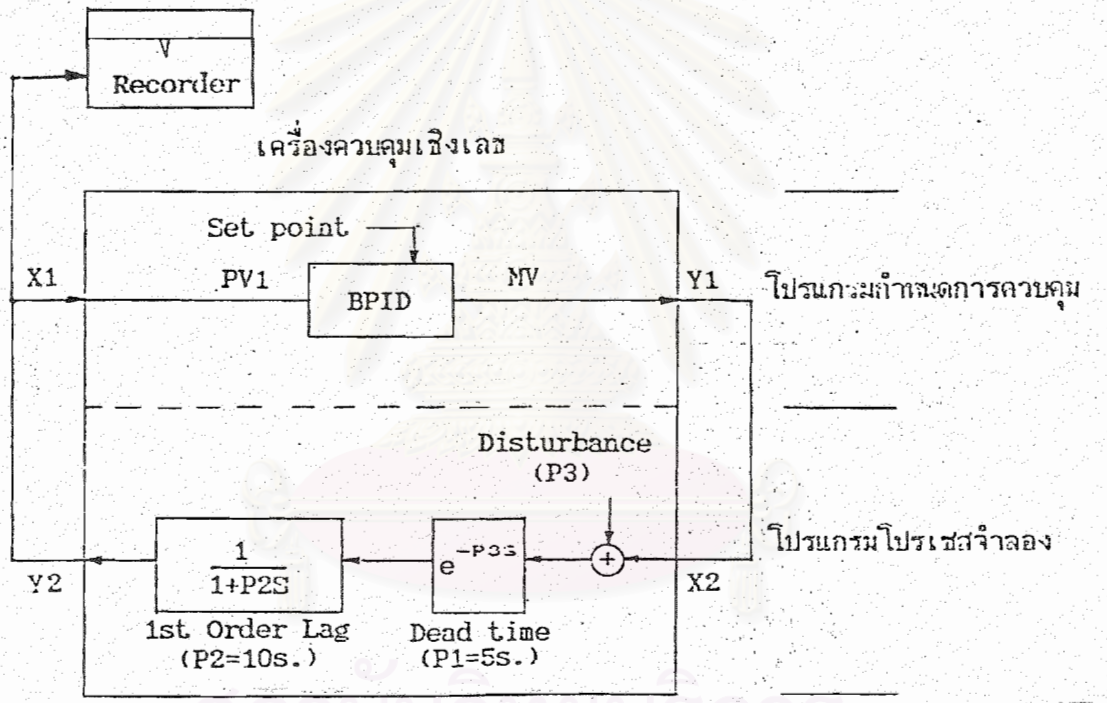
ผลการทดลอง ฟังก์ชันที่เขียนขึ้นสามารถทำงานได้ถูกต้องตามต้องการ

6.3.2 การทดสอบกับโปรเซส

แบ่งการทดสอบออกเป็น 2 ส่วน คือ

(1) ทดสอบกับโปรเซสจำลองที่ถูกสร้างโดยโปรแกรม (Process Simulation Program)

ไดอะแกรมที่ใช้ในการทดสอบแสดงดังรูปที่ 6.8



รูปที่ 6.8 ไดอะแกรมของโปรเซสจำลองที่สร้างจากโปรแกรมเพื่อใช้ทดสอบ

การทดสอบทำได้โดยใช้ฟังก์ชันภายในเครื่องควบคุม เขียนโปรแกรมสร้างโปรเซสจำลองที่มี dead time = 5 sec., lag time = 10 sec. โปรแกรมกำหนดรูปแบบการควบคุมและโปรแกรมสร้างโปรเซสจำลองเขียนได้ ดังรูปที่ 6.9 และเขียนเป็นภาษา Assembly ได้ดังรูปที่ 6.10 นำมาแปลงเป็น file.hex เขียนลง ROM โดยใช้ EPROM Programmer CLK3000 เพื่อควบคุมการทำงาน

```
---------------------------------------------------------------

Control Configuration

                LD      X1      ; read PV1
                BPID            ; PID Control
                ST      Y1      ; out control signal

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

Process Simulation Program

                LD      X2      ; read PV2
                LD      P3      ; read disturbance
                ADD             ; add disturbance
                LD      P1      ; read dead time
                DED1            ; dead time function
                LD      P2      ; read lag time
                LAG1            ; 1st order lag function
                ST      Y2      ; output channel 2

---------------------------------------------------------------
```

รูปที่ 6.9 โปรแกรม mnemonic กำหนดรูปแบบการควบคุมและสร้างโปรเซสจำลอง

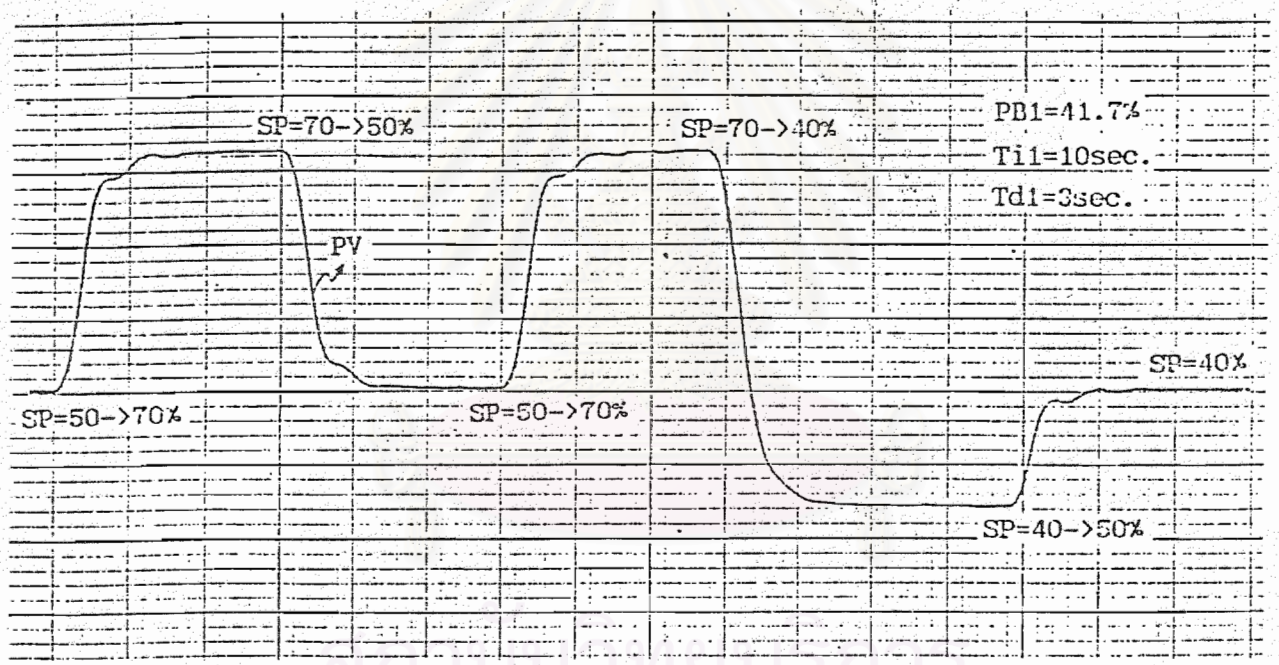
```
NAME user_pgm
ASSUME CS:CODE,DS:DATA
DATA   SEGMENT  WORD  PUBLIC  'DATA'
DATA   ENDS

CODE   SEGMENT  WORD  PUBLIC  'CODE'
userp  proc  far
       push  bp
       push  ax
       push  bx
       mov   al,0h
       push  ax
       mov   bx,785Ah
       call  bx              ; ld x1
       mov   ax,88cah
       call  bx              ; bpid
       mov   al,0h
       push  ax
       mov   bx,78f0h
       call  bx              ; st y1
       mov   al,1h
       push  ax
       mov   bx,785ah
       call  bx              ; ld x2
       mov   ax,21h
       push  ax
       mov   bx,7834h
       call  bx              ; ld p3
       mov   bx,7fdah
       call  bx              ; add
       mov   ax,1bh
       push  ax
       mov   bx,7834h
       call  bx              ; ld p1
       mov   ax,1b8h
       push  ax
       mov   ax,1eah
       push  ax
       mov   bx,7e6ah
       call  bx              ; div1
       mov   ax,1eh
       push  ax
       mov   bx,7884h
       call  bx              ; ld p2
       mov   ax,19ch
       push  ax
       mov   bx,7a0eh
       call  bx              ; lag1
       mov   al,1h
       push  ax
       mov   bx,78f0h        ; st y2
       call  bx
       pop   bx
       pop   ax
       pop   bp
       ret
userp  ENDP
CODE   ENDS
       END
```

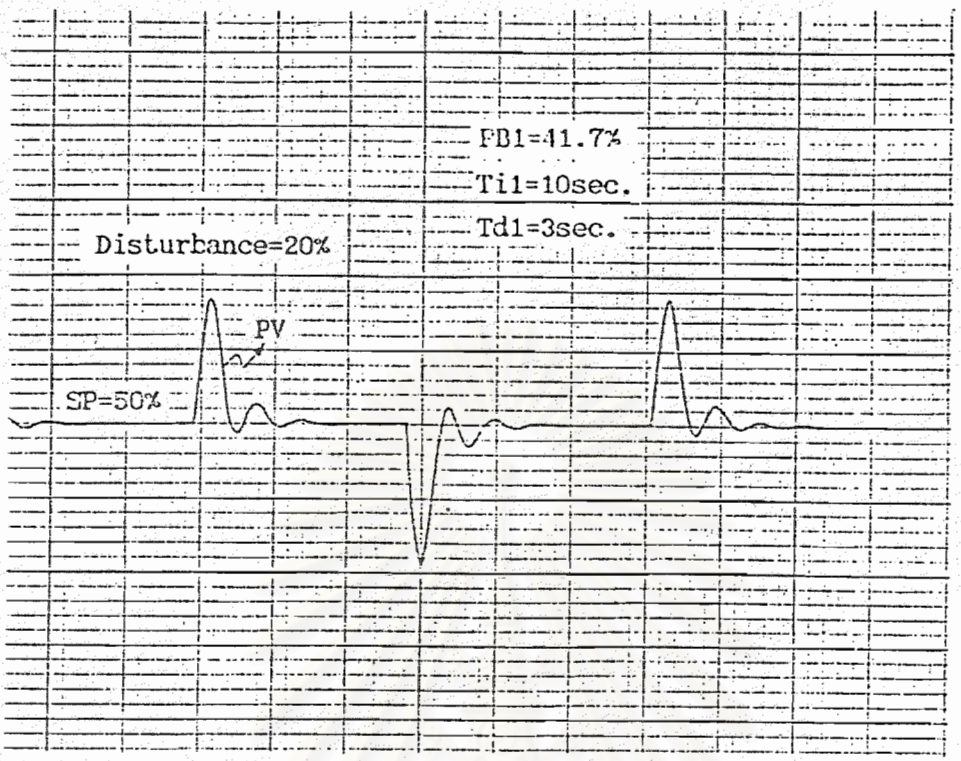
รูปที่ 6.10 โปรแกรม assembly สำหรับรูปแบบการควบคุมและสร้างโปรเซสลำดับ

ขั้นตอนการทดลองมีดังนี้

- Power On เครื่องควบคุมจะอยู่ในโหมด Manual โปรแกรมควบคุมระบบและโปรแกรมโปรเซสจำลองยังไม่ทำงาน
- ตั้งค่า PB1 = 41.7 %, Ti1 = 10 s., Td1 = 3 s.
- เปลี่ยนโหมดเป็น Auto โดยกดแป้นฟิมพ์ A ด้านหน้าเครื่อง โปรแกรมควบคุมและโปรเซสจำลองเริ่มทำงาน
- เปลี่ยนค่า Set point โดยกดแป้นฟิมพ์ SV เพื่อเปลี่ยนค่า Set point ได้กราฟผลการทดสอบดังรูปที่ 6.11
- เพิ่มค่า Disturbance โดยเปลี่ยนค่าพารามิเตอร์ P3 จากแป้นฟิมพ์ด้านข้าง ผลการทดสอบดังรูปที่ 6.12



รูปที่ 6.11 กราฟผลตอบสนองของโปรเซสเมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย



รูปที่ 6.12 กราฟผลตอบสนองของโปรเซสเมื่อมีสิ่งรบกวนระบบ

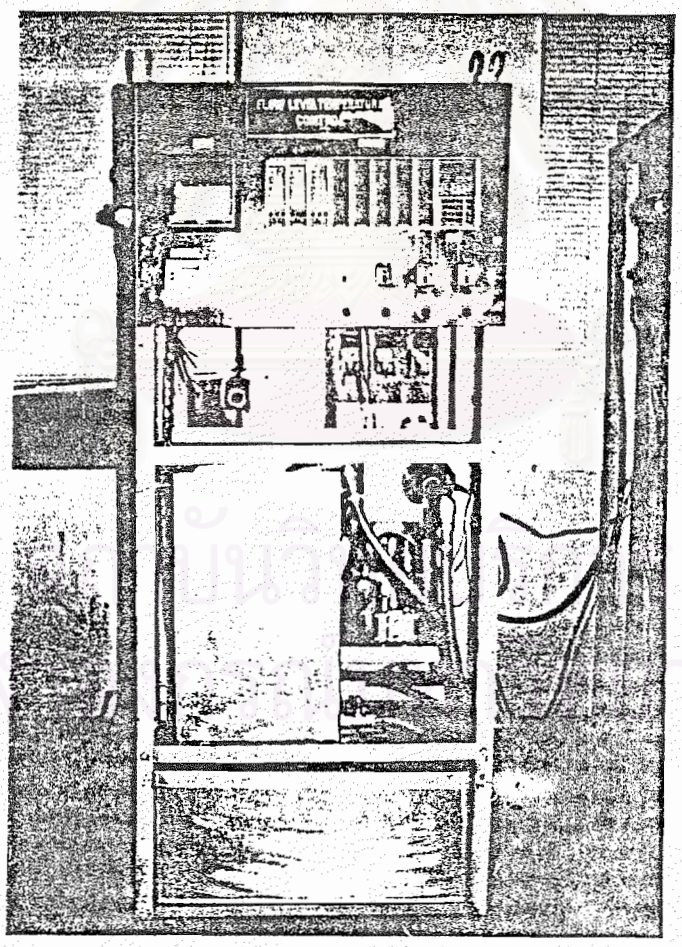
ผลการทดสอบ

การทดสอบกับโปรเซสจำลองที่สร้างจากโปรแกรม เครื่องควบคุมสามารถควบคุมการทำงานได้ เมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย หรือมีสิ่งรบกวนระบบ

(2) การทดสอบกับระบบจำลองทางอุตสาหกรรม

ทดสอบกับระบบจำลองของการไหล ระดับ และอุณหภูมิ (ดูรูปที่ 6.13) ซึ่งส่วนประกอบสำคัญที่ใช้ในการทดสอบคือ

- Ultrasonic Level Sensor
- Level Transmitter
- Control Valve
- Differential Pressure Transmitter
- Orifice
- Pump
- Tank

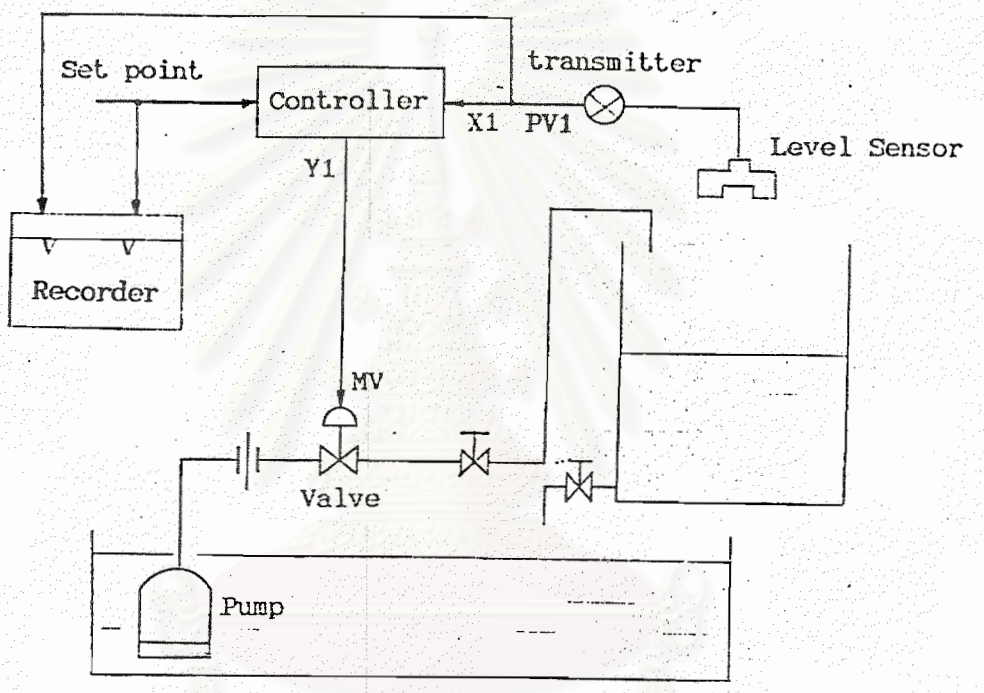


รูปที่ 6.13 ระบบจำลองการไหล ระดับ และอุณหภูมิ

การทดสอบจะใช้เครื่องควบคุมเชิงเลข ควบคุมระดับน้ำในถังของระบบจำลองทางอุตสาหกรรม โดยใช้รูปแบบวิธีการควบคุม 2 แบบ ดังนี้

(ก) การควบคุมแบบป้อนกลับแบบง่ายๆ

มีไดอะแกรมที่ใช้ในการทดลองดังรูปที่ 6.14



รูปที่ 6.14 ไดอะแกรมที่ใช้ในการทดสอบการควบคุมแบบง่ายๆ

ใช้ฟังก์ชันภายในเครื่องควบคุมเขียนโปรแกรมบนไมโครคอมพิวเตอร์ เพื่อกำหนดรูปแบบการควบคุมได้ดังรูปที่ 6.15 และเขียนเป็นภาษา Assembly ได้ดังรูปที่ 6.16 นำมาแปลงเป็น file.hex เขียนลง ROM

```
LD      SV1     ; read set point
ST      Y2      ; out to recorder
LD      X1      ; read process variable
BPID            ; PID control
ST      Y1      ; out control signal
```

รูปที่ 6.15 โปรแกรม mnemonic กำหนดรูปแบบการควบคุม

```
NAME user_rom
ASSUME CS:CODE,DS:DATA
DATA   SEGMENT  WORD  PUBLIC  'DATA'
DATA   ENDS

CODE   SEGMENT  WORD  PUBLIC  'CODE'
userp  proc  far
       push  bp
       push  ax
       push  bx
       mov   ax,7eh
       push  ax
       mov   bx,7884h
       call  bx            ; ld sv1
       mov   al,1h
       push  ax
       mov   bx,78f0h
       call  bx            ; st y2
       mov   al,0h
       push  ax
       mov   bx,785ah
       call  bx            ; ld x1
       mov   bx,82cah
       call  bx            ; bpid
       mov   al,0h
       push  ax
       mov   bx,78f0h
       call  bx            ; st y1
       pop   bx
       pop   ax
       pop   bp
       ret
userp  ENDP
CODE   ENDS
       END
```

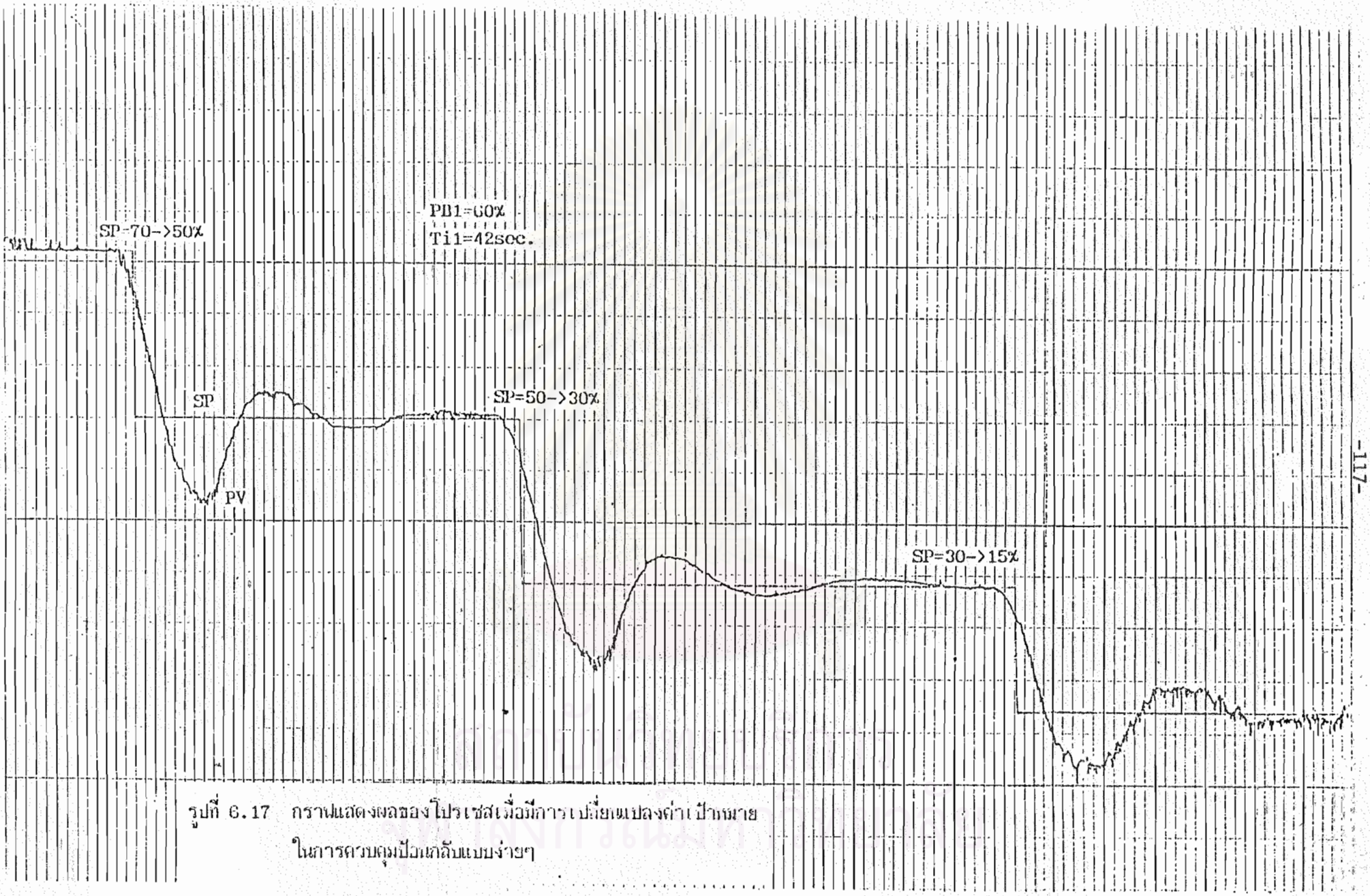
รูปที่ 6.16 โปรแกรม assembly กำหนดรูปแบบการควบคุม

ขั้นตอนการทดลอง

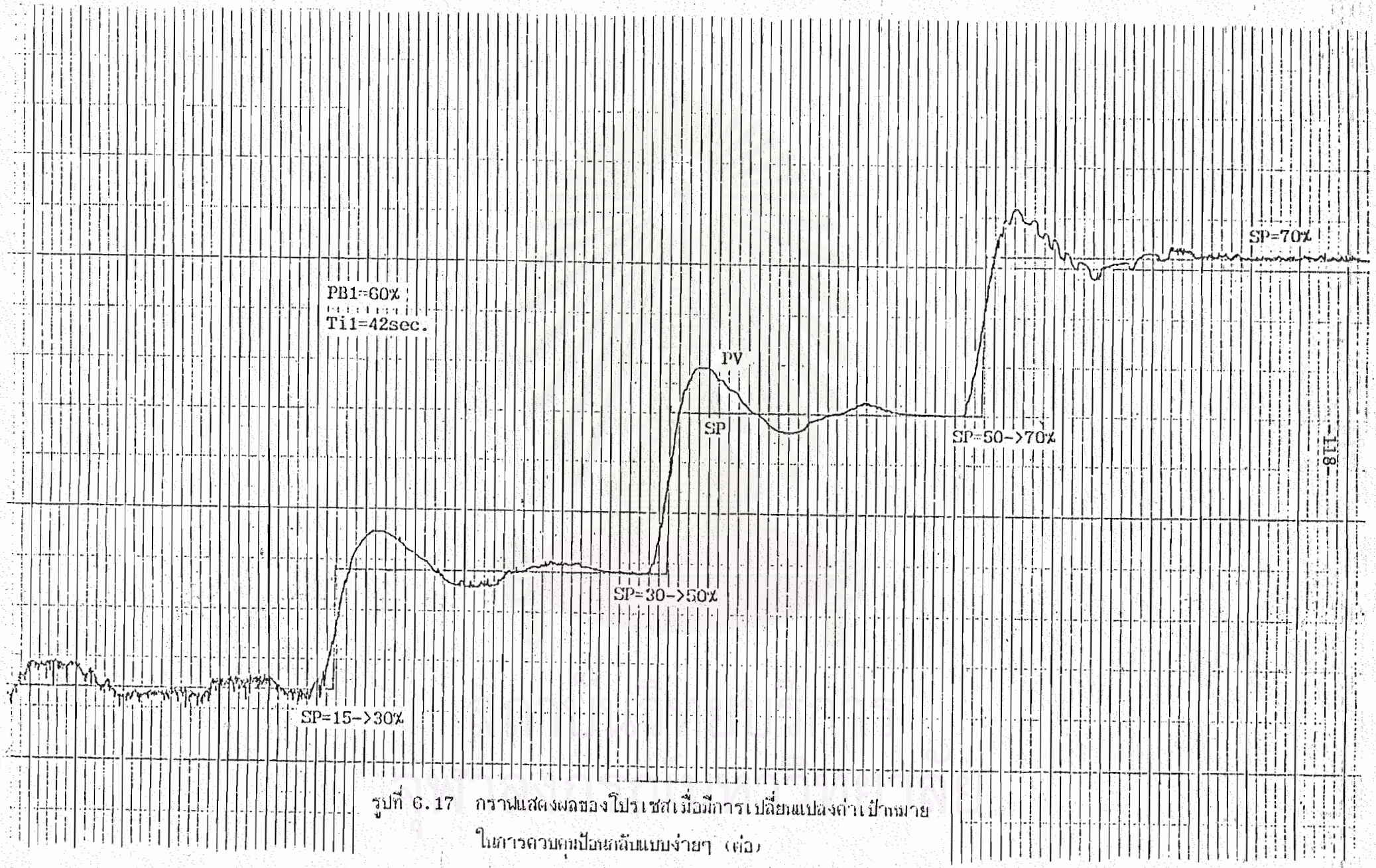
- Power On เครื่องควบคุมจะอยู่ในโหมด Manual
- ตั้งค่า PB1 = 60 %, Ti1 = 42 sec.
- เปลี่ยนโหมดเป็น Auto โดยกดแป้นพิมพ์ A ด้านหน้าเครื่อง
- เปลี่ยนค่า Set point โดยกดแป้นพิมพ์ SV เพื่อตรวจสอบผลการตอบสนองของเครื่องควบคุมเมื่อมีการเปลี่ยนแปลงค่า Set point

ผลการทดลอง

เครื่องควบคุมเชิงเลขสามารถควบคุมระดับน้ำให้เท่ากับค่าเป้าหมายของระดับน้ำที่ต้องการได้ถูกต้อง ดังรูปที่ 6.17



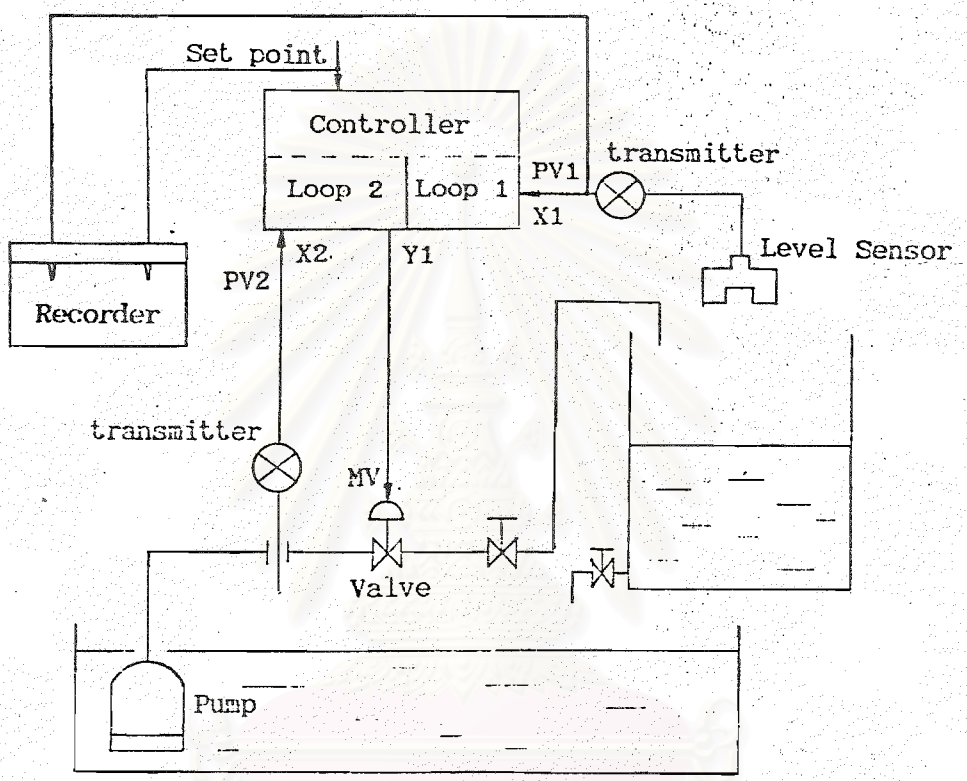
รูปที่ 6.17 กราฟแสดงผลของโปรเซสเมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย
ในการควบคุมป้อนกลับแบบง่ายๆ



รูปที่ 6.17 กราฟแสดงผลของโปรเซสเมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย ในการควบคุมป้อนกลับแบบง่ายๆ (ต่อ)

(ข) การควบคุมแบบ Cascade

มีไดอะแกรมที่ใช้ในการทดลองดังรูปที่ 6.18



รูปที่ 6.18 ไดอะแกรมที่ใช้ในการทดสอบการควบคุมแบบ Cascade

ใช้ฟังก์ชันภายในเครื่องควบคุมเขียนโปรแกรมบนไมโครคอมพิวเตอร์ เพื่อกำหนดรูปแบบการควบคุมได้ดังรูปที่ 6.19 และเขียนเป็นภาษา Assembly ได้ดังรูปที่ 6.20 นำมาแปลงเป็น file.hex เขียนลง ROM

```
LD      SV1     ; read set point
ST      Y2      ; out to recorder
LD      X2      ; read flow PV2
SQR             ; square root function
LD      X1      ; read level PV1
CPID            ; Cascade control
ST      Y1      ; out control signal
```

รูปที่ 6.19 โปรแกรม mnemonic กำหนดรูปแบบการควบคุม

```
        NAME user_rom
ASSUME CS:CODE,DS:DATA
DATA  SEGMENT  WORD  PUBLIC  'DATA'
DATA  ENDS
CODE  SEGMENT  WORD  PUBLIC  'CODE'
userp  proc  far
       push  bp
       push  ax
       push  bx
       mov   ax,7eh
       push  ax
       mov   bx,7884h      ; LD SV1
       call  bx
       mov   al,1h
       push  ax
       mov   bx,73f0h
       call  bx            ; ST Y2
       mov   al,1h
       push  ax
       mov   bx,785Ah
       call  bx            ; LD X2
       mov   bx,80A6h
       call  bx            ; SQR
       mov   al,0h
       push  ax
       mov   bx,785ah
       call  bx            ; LD X1
       mov   bx,8550h
       call  bx            ; CPID
       mov   al,0h
       push  ax
       mov   bx,73f0h
       call  bx            ; ST Y1
       pop   bx
       pop   ax
       pop   bp
       ret
userp  ENDP
CODE   ENDS
       END
```

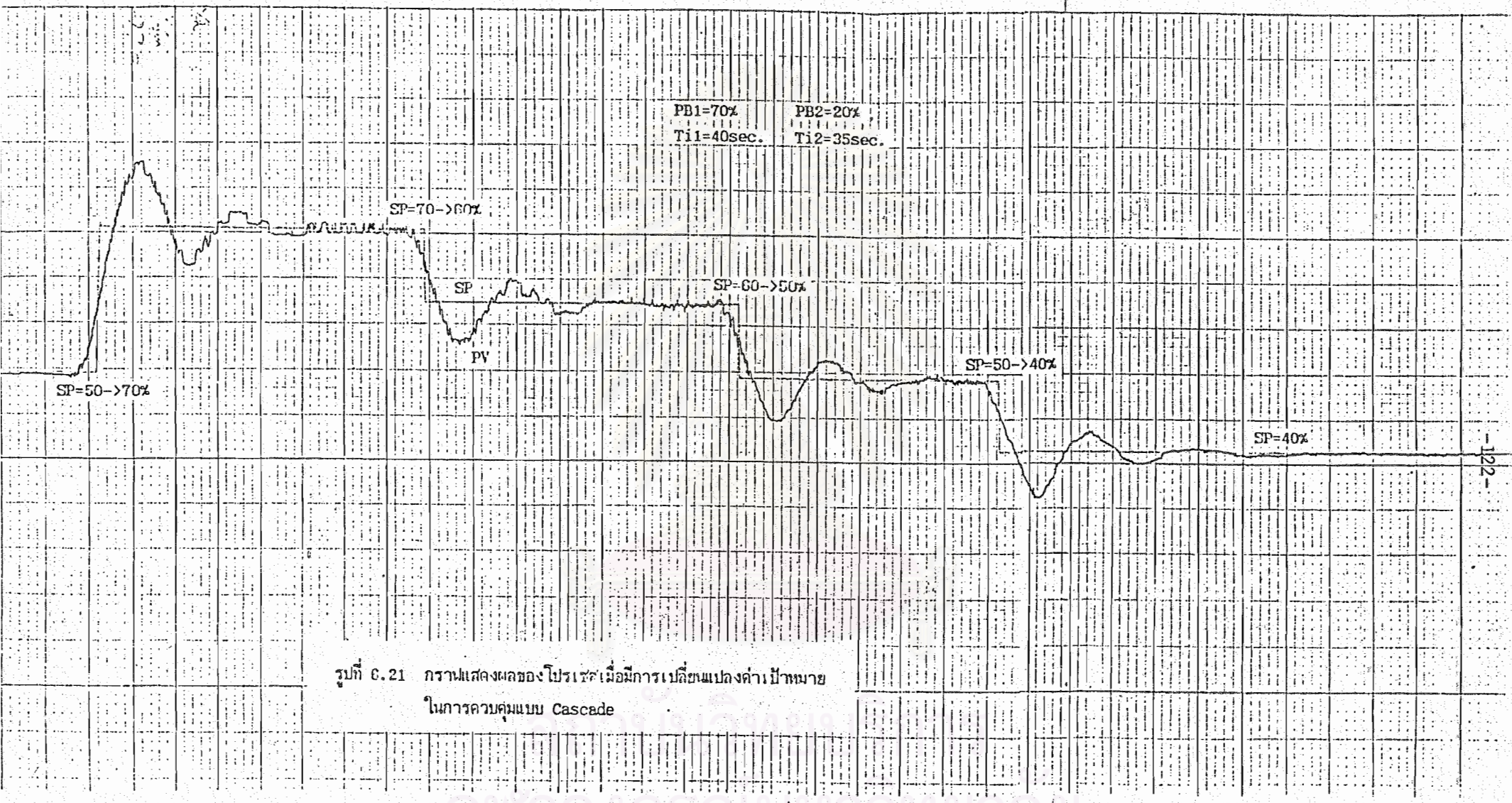
รูปที่ 6.20 โปรแกรม assembly กำหนดรูปแบบการควบคุม

ขั้นตอนการทดสอบ

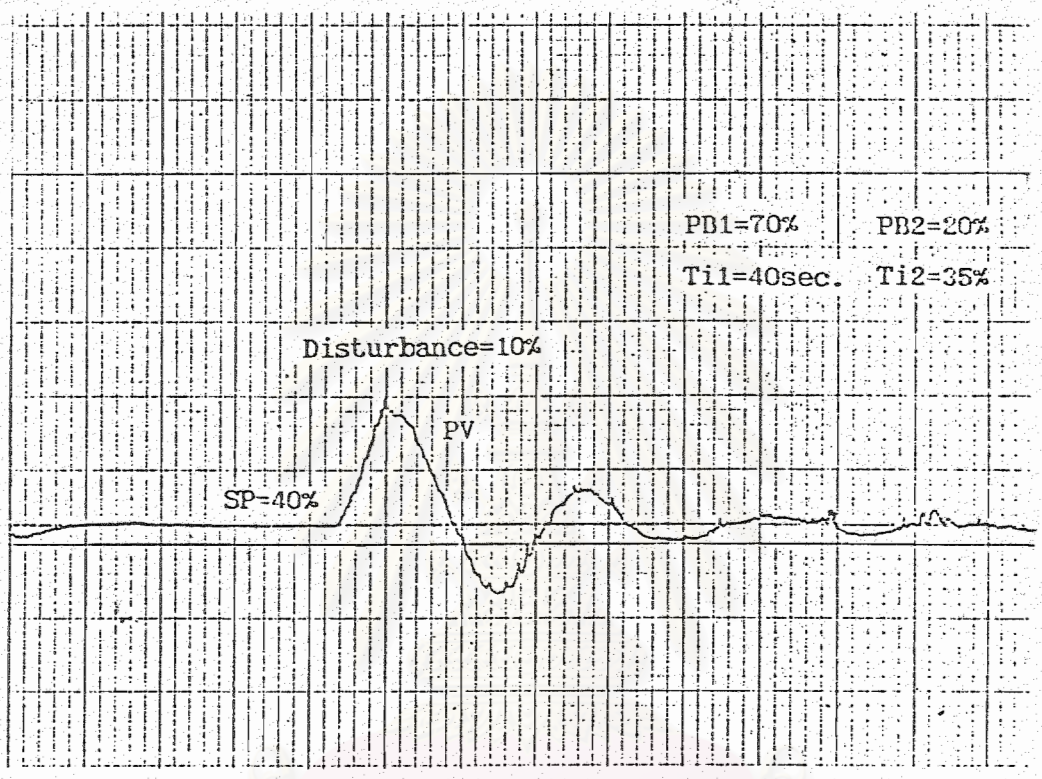
- Power On เครื่องควบคุมจะอยู่ในโหมด Manual
- ตั้งค่า PB2 = 20 %, Ti2 = 35 sec. ใช้แป้นเพิมพ์ด้านข้างเครื่อง
- เปลี่ยนโหมดเป็น Auto โดยกดแป้นเพิมพ์ A ด้านหน้าเครื่อง เครื่องควบคุมจะตัดลูปที่ 1 (ลูปใน) ออกจากระบบ เพื่อทำการปรับ (Tuning)ค่า PB,Ti และ Td ที่เหมาะสมของลูปที่ 2 (ลูปนอก)
- ตั้งค่า PB1 = 70 %, Ti1 = 40 sec.
- กดแป้นเพิมพ์ C เพื่อเปลี่ยนโหมดเป็น Cascade control
- เปลี่ยนค่า Set point โดยกดแป้นเพิมพ์ SV เพื่อตรวจสอบผลการตอบสนองของเครื่องควบคุมเมื่อมีการเปลี่ยนแปลงค่า Set point ดูรูปที่ 6.21
- ใส่สิ่งรบกวนระบบเข้าไป โดยเติมน้ำเข้าไปในถัง โดยที่ค่าเป้าหมายของระดับน้ำเท่าเดิม ได้ผลการตอบสนองดังรูปที่ 6.22

ผลการทดสอบ

เครื่องควบคุมเชิงเลขสามารถควบคุมระดับน้ำให้เท่ากับค่าเป้าหมายของระดับน้ำที่ต้องการได้ถูกต้อง เมื่อมีการเปลี่ยน Set point หรือเมื่อมี Process Disturbance



รูปที่ 6.21 กราฟแสดงผลของโปรเซสเมื่อมีการเปลี่ยนแปลงค่าเป้าหมาย ในการควบคุมแบบ Cascade



รูปที่ 6.22 กราฟแสดงผลของโปรเซสเมื่อมีสิ่งรบกวนระบบในการควบคุมแบบ Cascade

บทที่ 7

สรุปผลและข้อเสนอแนะ

การทำวิจัยครั้งนี้ ได้ทำการออกแบบและสร้างตัวควบคุมเชิงเลขชนิดโปรแกรมได้สำหรับงานควบคุมทางอุตสาหกรรม ซึ่งการออกแบบประกอบด้วยด้านฮาร์ดแวร์และซอฟแวร์ ตัวควบคุมเชิงเลขที่ได้ นำไปใช้ทดสอบการทำงานกับระบบจำลองทางอุตสาหกรรมของการไหล ระดับ และอุณหภูมิ ภายในห้องปฏิบัติการวัดคุมทางอุตสาหกรรม ได้ผลการทดสอบเป็นที่น่าพอใจ

ตัวควบคุมเชิงเลขที่สร้างขึ้นมีคุณลักษณะดังนี้

- ระบบการควบคุม — ใช้ไมโครโปรเซสเซอร์ขนาด 16 บิท เบอร์ 8088 ความเร็ว 5 MHz.
- หน่วยความจำ — ใช้ EPROM ขนาด 16 กิโลไบท์ สำหรับโปรแกรมควบคุมระบบ และ EPROM ขนาด 8 กิโลไบท์ สำหรับโปรแกรมกำหนดรูปแบบการควบคุม และ RAM ขนาด 8 กิโลไบท์
- อนาลอกอินพุท — สัญญาณแรงดันมาตรฐาน 1-5 V_{dc} 5 จุด
- ดิจิตอลอินพุท — 3 จุด ชนิด DC กระแสไหลเข้า (Current Sink type) ต่อผ่าน Opt.o-couple ทำงานด้วยระดับแรงดัน 24 V_{dc}
- อนาลอกเอาท์พุท — แบ่งเป็นสัญญาณกระแส 4-20 mA_{dc} 1 จุด และชนิดแรงดัน 1-5 V_{dc} 2 จุด
- ดิจิตอลเอาท์พุท — 3 จุด ชนิดรีเลย์ หน้าสัมผัสกระแสได้ 0.5 A_{dc}
- ส่วน Tuning — มีแป้นพิมพ์สำหรับแสดงผลและเปลี่ยนค่าพารามิเตอร์ 12 แป้นพิมพ์
- โปรแกรมกำหนดรูปแบบ — เขียนบนไมโครคอมพิวเตอร์ โดยใช้ภาษา Assembly เพื่อสร้าง Hex file ลงบน ROM เพื่อกำหนดรูปแบบการควบคุม

- ความเร็วในการทำงาน ในแต่ละรอบการทำงานของโปรแกรมกำหนดรูปแบบจะต้องไม่เกิน 200 ms.
- ความเร็วของฟังก์ชันPID ใช้เวลาน้อยกว่า 8 ms.

จากผลการทดลองเราสามารถสรุปได้ว่าตัวควบคุมเชิงเลขสามารถทำงานได้ตามเป้าหมายทุกประการ โดยมีความเร็วในการทำงานสูงมาก สามารถดัดแปลงนำไปควบคุมระบบควบคุมที่ซับซ้อนจำพวก Cascade Control Ratio Control ฯลฯ ได้มากกว่า 8 loops

แม้ว่างานวิจัยครั้งนี้จะสิ้นสุดโดยประสบผลสำเร็จตามเป้าหมายทุกประการก็ตามแต่ก็ยังมีงานพัฒนาต่อ ซึ่งคณะผู้วิจัยจะดำเนินการวิจัยต่อไปอีกซึ่งได้แก่ การออกแบบวงจรนิมม์ใหม่ให้มีขนาดเล็กลง การทดสอบความทนทานกับงานควบคุมจริงในโรงงานอุตสาหกรรมเพื่อหาจุดอ่อนของวงจรนำมาแก้ไขดัดแปลง การเขียนโปรแกรมบนไมโครคอมพิวเตอร์เพื่อเก็บข้อมูลการควบคุม การเขียน Mnemonic Compiler เพื่อความสะดวกในการใช้งาน นอกจากนั้นคณะผู้วิจัยยังจะพัฒนาการใช้ไมโครคอมพิวเตอร์สำหรับงานปรับตั้งค่า PID (PID Controller Tuning) อีกด้วย

เอกสารอ้างอิง

1. Michael P.Lukas, Distributed Control Systems: Their Evaluation and Design, 1986
2. YEW, Model SLPC,SLMC and SPLR Programmable Instruments Functions and Applications, Technical Information
3. FUJI, Compact Controller F User's Manual
4. TOSHIBA, One-loop Controller 211D User's Manual
5. ดร.สมบูรณ์ จงชัยกิจ, เอกสารประกอบการบรรยายเรื่อง การควบคุมแบบอัตโนมัติและการควบคุมแบบ PID, สมาคมเทคโนโลยี(ไทย-ญี่ปุ่น)
6. Karl J.Astrom and Bjorn Wittenmark, Computer Controlled System Theory and Design, Prentice-Hall, Inc. 1984.
7. Nicholas J.Krikelis and Spilios D.Fassois, Microprocessor Implementation of PID Controllers and Lead-Lag Compensators,
   IEEE trans. of Industrial Electronics, Vol.IE-31,NO.1, Feb 1984
8. Michael Andrews, Programming Microprocessor Interfaces for Control and Instrumentation, Prentice-Hall, 1982
9. John Zarrella, Language Translators, Microcomputer Applications, California, 1982
10. Joseph J.F. Cavanagh, Digital Computer Arithmetic Design and Implementation, McGraw-Hill, 1985
11. Robert J.Bibbero, Microprocessors in Instruments and Control, John Wiley & Sons, 1977
12. ALS, ALS CARD SYSTEM USER'S MANUAL 8088 SYSTEM, INDIA
13. Intel Corporation, PL/M-86 USER'S GUIDE, 1985
14. Intel Corporation, 8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL, 1980
15. Intel Corporation, iAPX 86,88 Family Utilities USER'S GUIDE, 1982

16. ดร.สมบูรณ์ จงชัยกิจ, รศ.กฤษดา วิศวชีรานนท์และอำนวย แสงวิโรจนรัฒน์, ตัวควบคุมเชิงเลขสำหรับงานควบคุมทางอุตสาหกรรมแบบต่อเนื่อง, การประชุมวิชาการ 9 สถาบัน ครั้งที่ 11, พ.ศ. 2532

ภาคผนวก ก

3.3.6 Bus connector

3.3.6.1 Bus connector (64 pin euroconnector) details - signal flow is referred to the processor card.

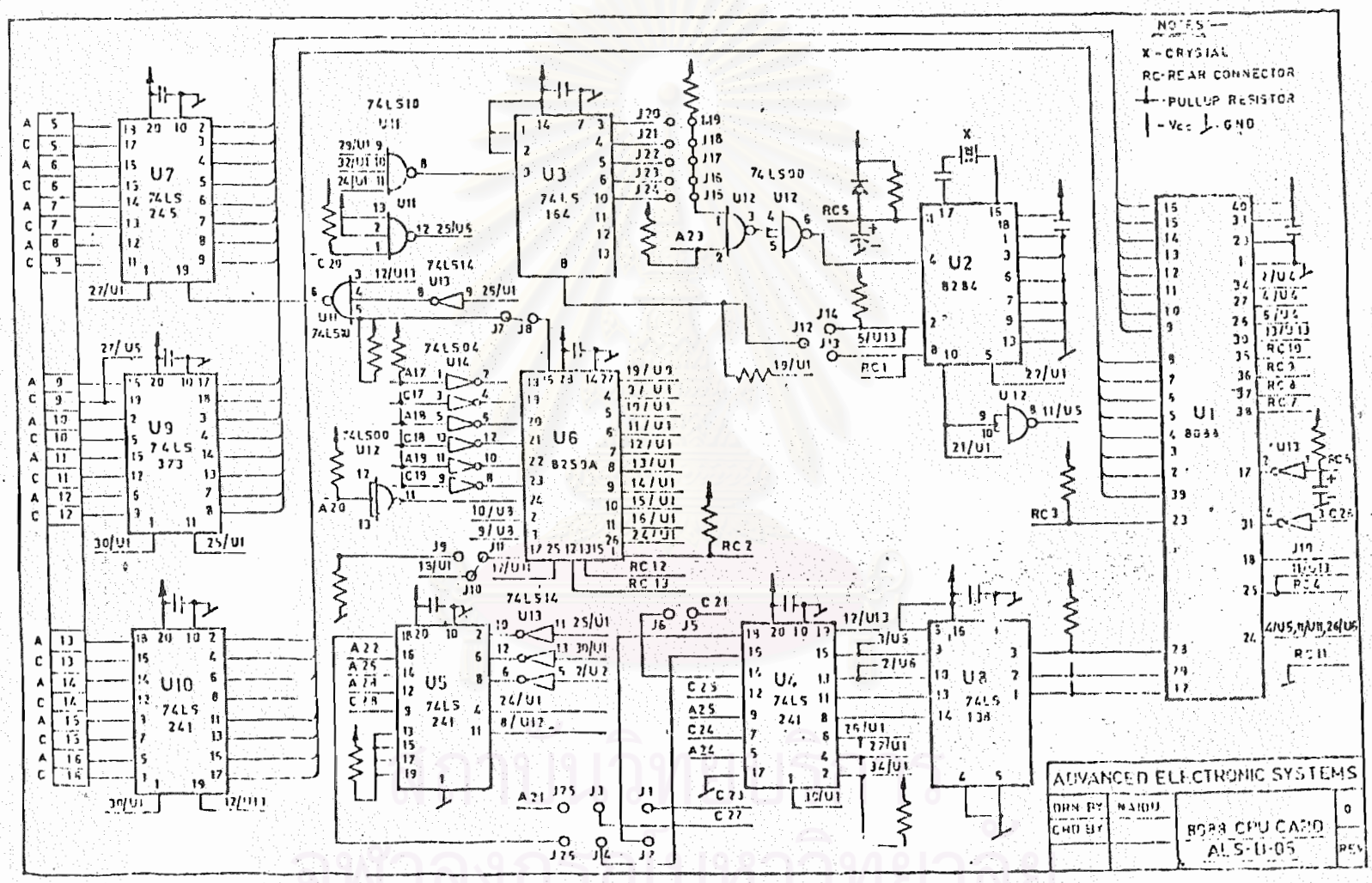
		ROW 'a'			ROW 'c'		
	PIN NO.	MNEMONIC	SIGNAL FLOW	DESCRIPTION	MNEMONIC	SIGNAL FLOW	DESCRIPTION
LOGIC POWER BUS	1	GND	IN	LOGIC GROUND	GND	IN	LOGIC GROUND
	32	GND	IN	LOGIC GROUND	GND	IN	LOGIC GROUND
	2	+5V	IN	LOGIC POWER	+5V	IN	LOGIC POWER
	31	+5V	IN	LOGIC POWER	+5V	IN	LOGIC POWER
	3	-12V	IN	ANALOG POWER	-12V	IN	ANALOG POWER
	4	+12V	IN	ANALOG POWER	+12V	IN	ANALOG POWER
	29	A.GND	IN	ANALOG COM	A.GND	IN	ANALOG COM
DATA BUS	5	DØ	IN/OUT	LOW-ORDER DATA BUS	D1	IN/OUT	LOW-ORDER DATA BUS
	6	D2	IN/OUT	LOW-ORDER DATA BUS	D3	IN/OUT	LOW-ORDER DATA BUS
	7	D4	IN/OUT	HIGH-ORDER DATA BUS	D5	IN/OUT	HIGH-ORDER DATA BUS
	8	D6	IN/OUT	HIGH-ORDER DATA BUS	D7	IN/OUT	HIGH-ORDER DATA BUS

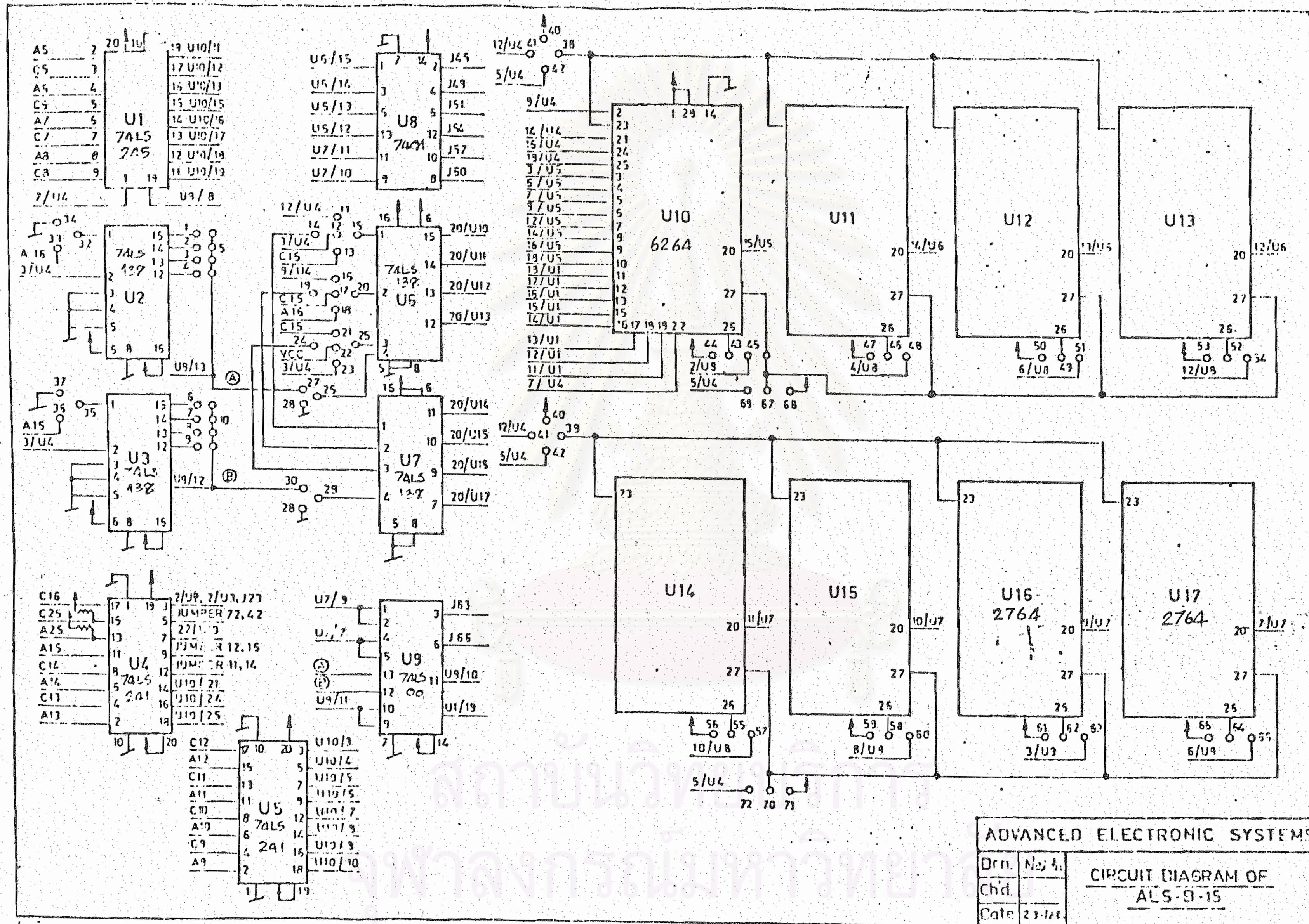
	PIN NO.	ROW 'a' MNEMONIC	SIGNAL FLOW	DESCRIPTION	ROW 'c' MNEMONIC	SIGNAL FLOW	DESCRIPTION
ADDRESS BUS	9	AØ	OUT	LOW-ORDER ADDRESS BUS	A1	OUT	LOW-ORDER ADDRESS BUS
	10	A2	OUT	-do-	A3	OUT	-do-
	11	A4	OUT	-do-	A5	OUT	-do-
	12	A6	OUT	-do-	A7	OUT	-do-
	13	A8	OUT	HIGH-ORDER ADDRESS BUS	A9	OUT	HIGH-ORDER ADDRESS BUS
	14	A10	OUT	-do-	A11	OUT	-do-
	15	A12	OUT	-do-	A13	OUT	-do-
	16	A14	OUT	-do-	A15	OUT	-do-

	PIN NO.	ROW 'a' MNEMONIC	SIGNAL FLOW	DESCRIPTION	ROW 'c' MNEMONIC	SIGNAL FLOW	DESCRIPTION
CONTROL BUS	17	$\overline{INT0}$	IN	Interrupt line	$\overline{INT1}$	IN	Interrupt line
	18	$\overline{INT2}$	IN	-do-	$\overline{INT3}$	IN	-do-
	19	$\overline{INT4}$	IN	-do-	$\overline{INT5}$	IN	-do-
	20	$\overline{INT6}$	IN	-do-	$\overline{INT7}$	IN	-do-
	21	$\overline{CCLK}$	OUT	Constant clock	--	--	reserved
	22	$\overline{INTA}$	OUT	Interrupt acknowledge	$\overline{INH2}$	OUT	Inhibit 2
	23	$\overline{XACK}$	IN	Reddy input	$\overline{INH1}$	OUT	Inhibit 1
	24	$\overline{IORC}$	OUT	I/o Read	$\overline{IOWC}$	OUT	I/o write
	25	$\overline{MRDC}$	OUT	Memory read	$\overline{MWTC}$	OUT	Memory write
	26	$\overline{BUSY}$	OUT	Bus busy	$\overline{BREQ}$	IN	Bus request
	27	$\overline{BPRN}$	IN	Bus priority in	$\overline{BPRO}$	OUT	Bus priority out
	28	$\overline{BCLK}$	OUT	Bus clock	$\overline{RESET}$	OUT	Initialisation signal
	30	--	--	reserved	--	--	reserved

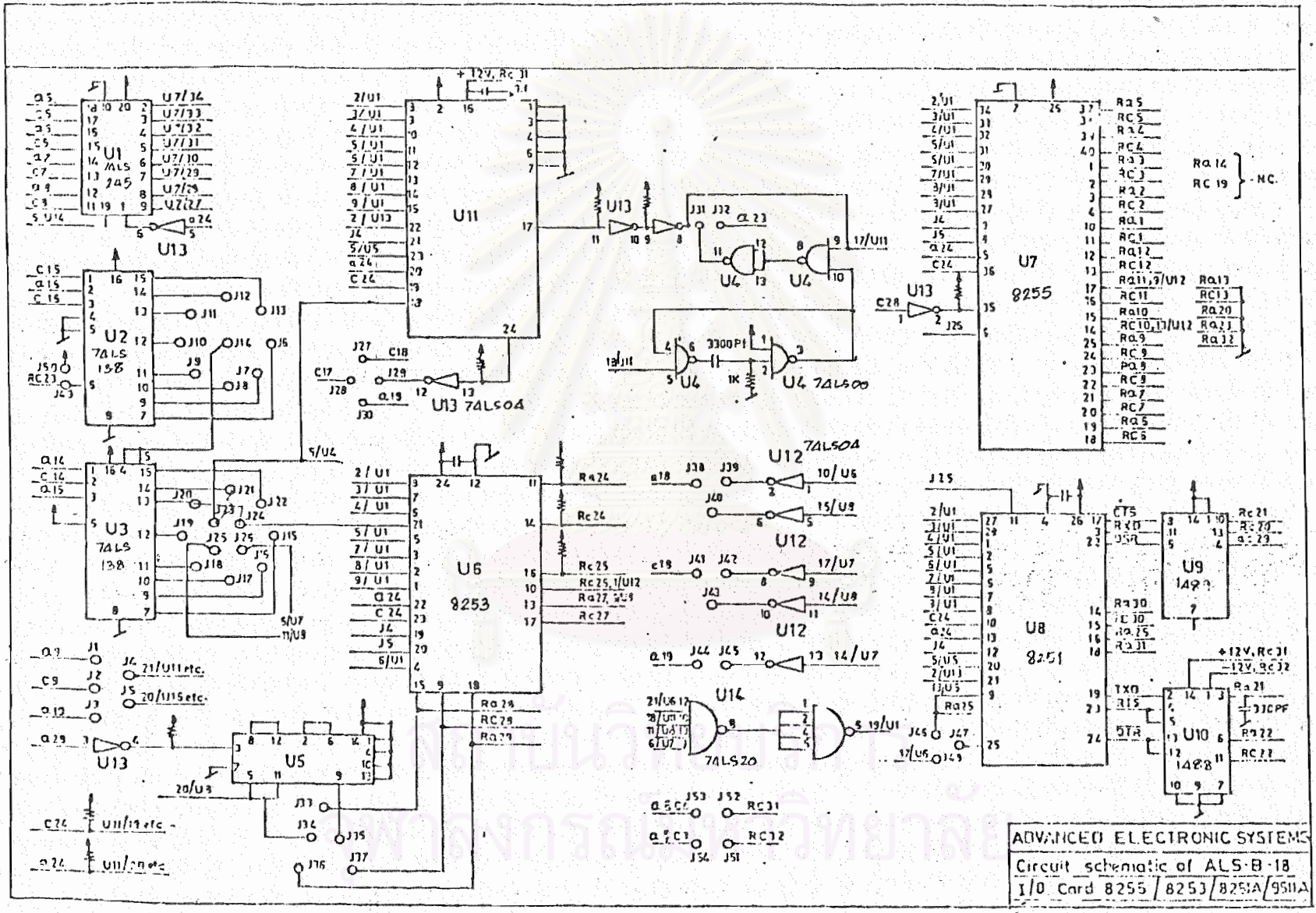
Note: 1) A bar over the mnemonic indicates that the signal is active when it is at a logic 'LO' level.

ภาคผนวก ข



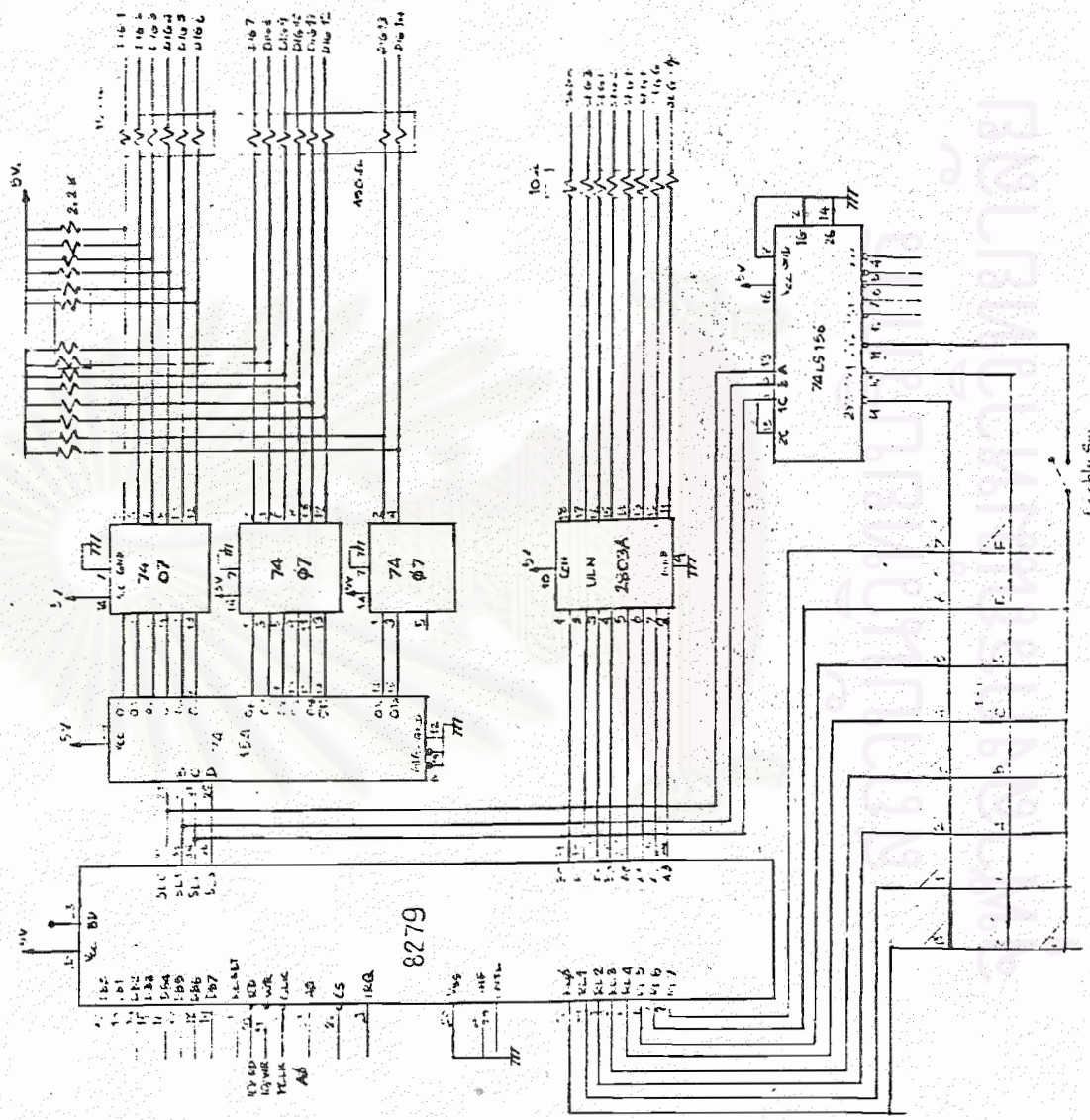


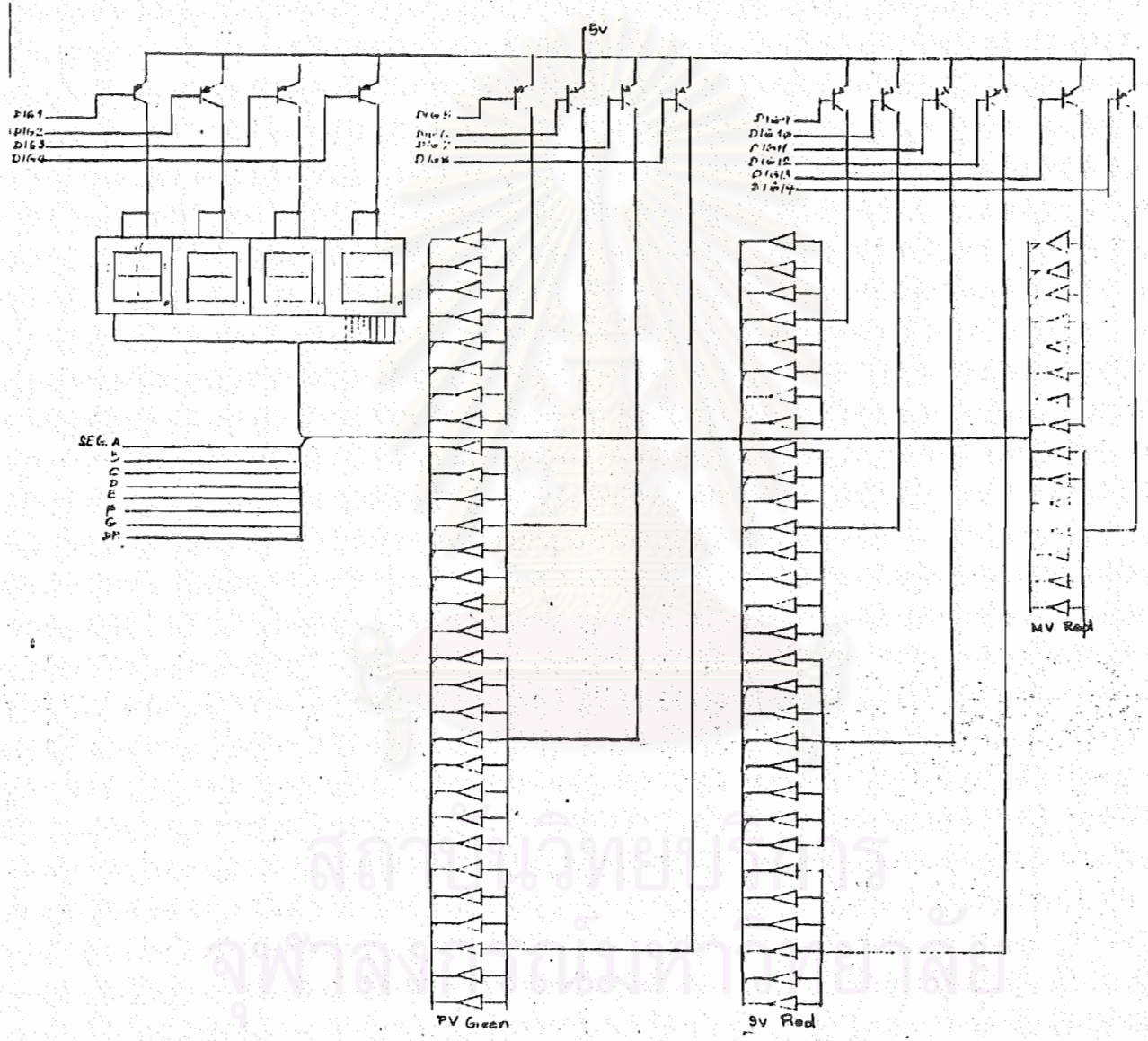
ข.2 วงจรส่วนหน่วยความจำ

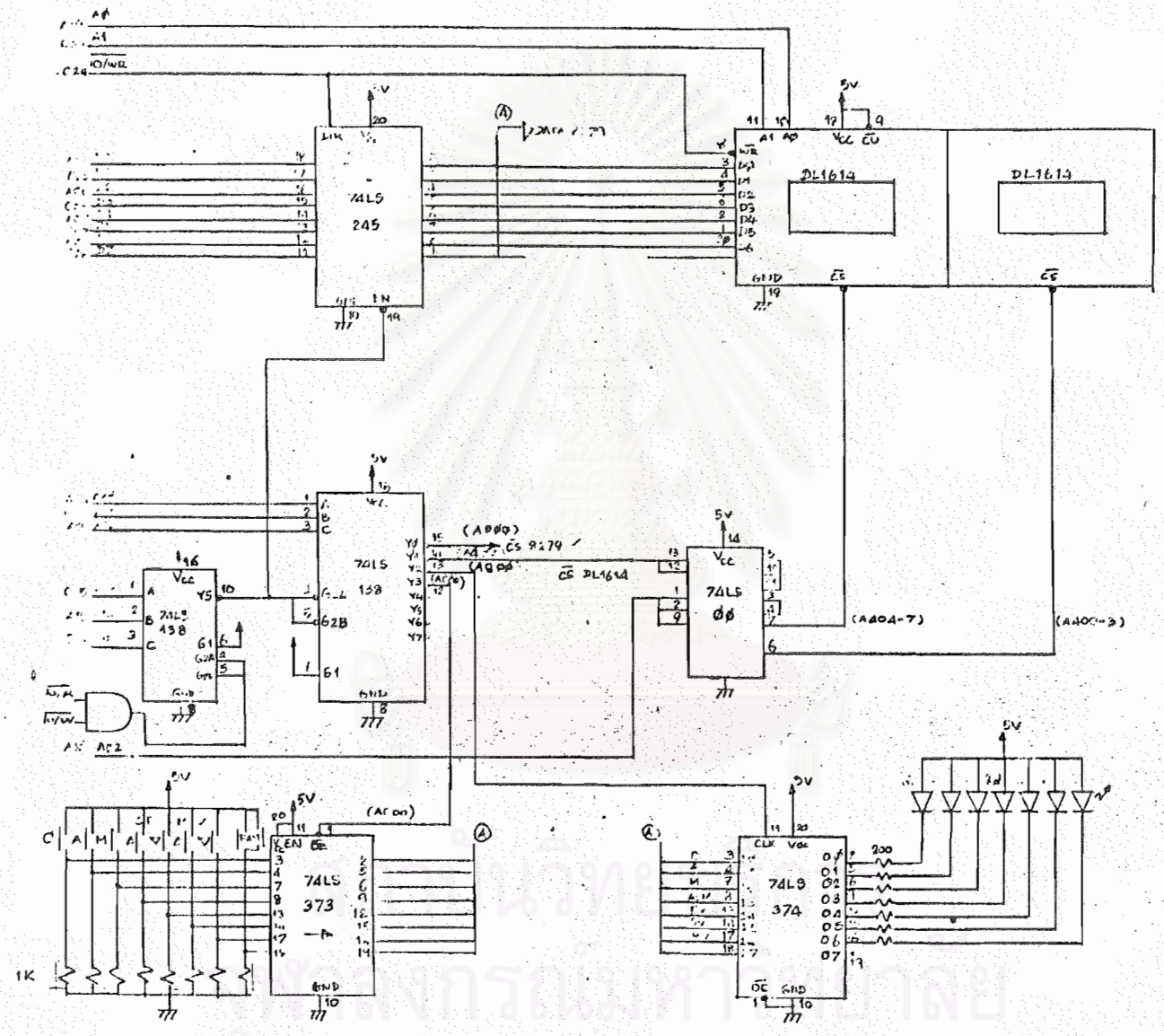


ข.3 วงจรไฟฟ้าของแผ่นวงจรพิมพ์

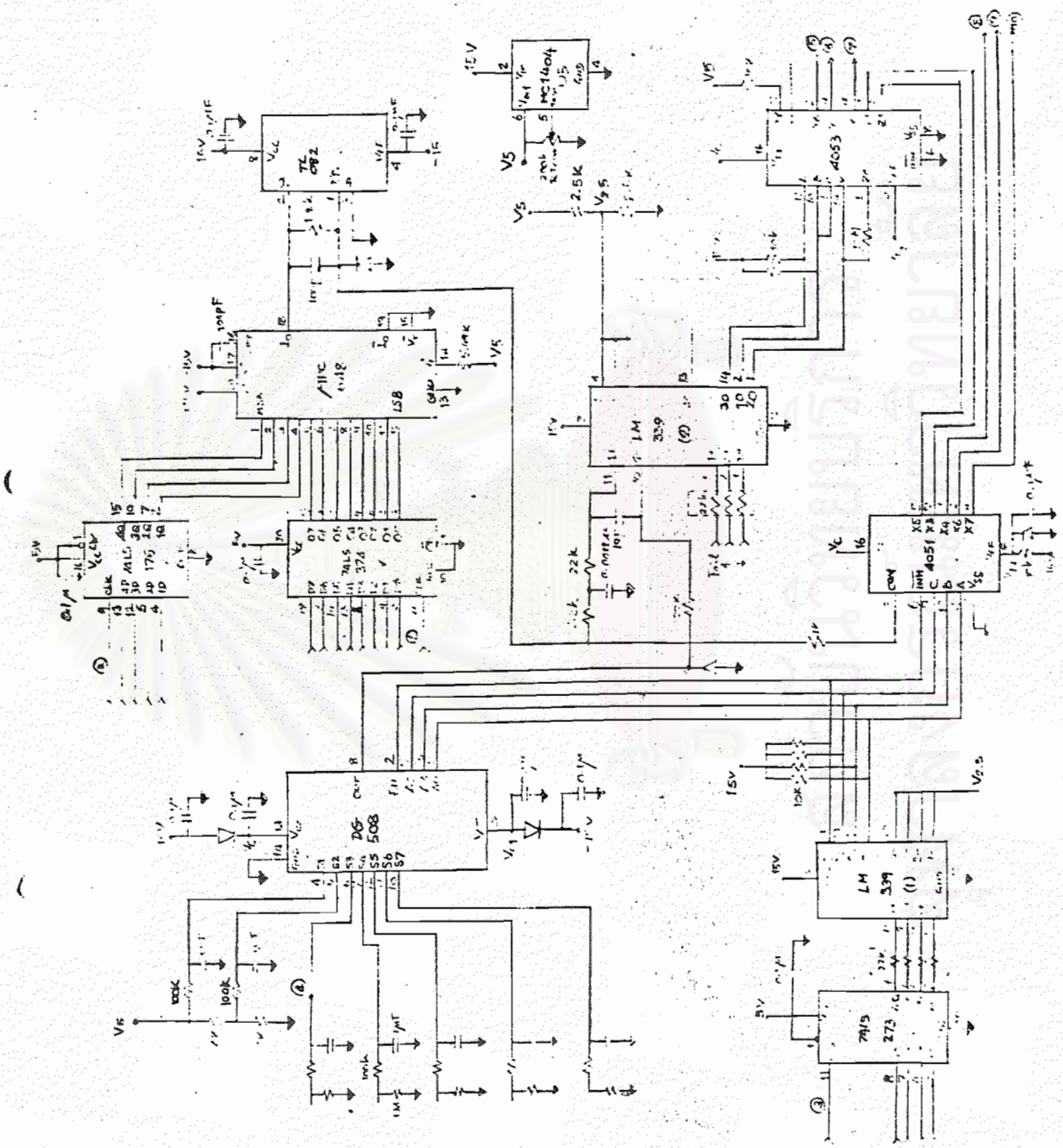
ข.4 วงจรส่วนแสดงผลและแป้นพิมพ์

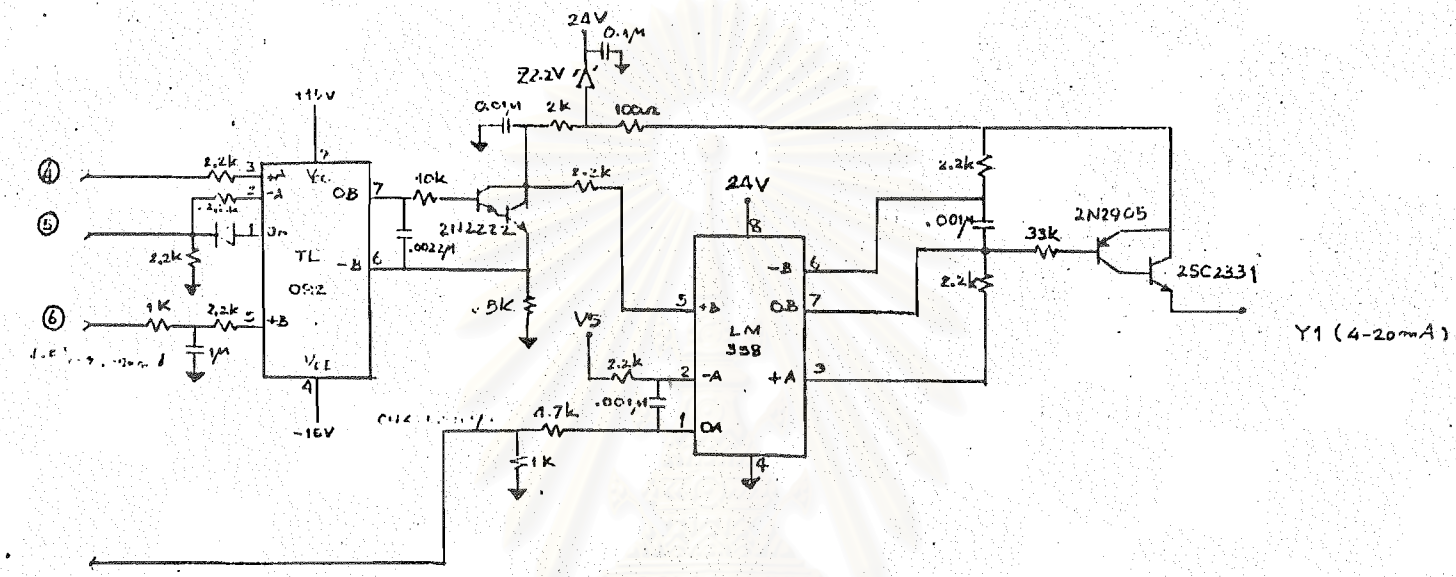


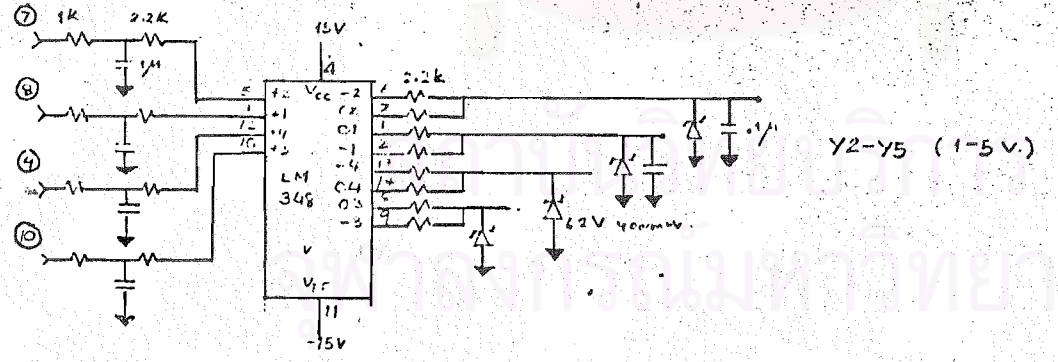

5v
DIG 1
DIG 2
DIG 3
DIG 4
SEG. A
B
C
D
E
F
G
DP
MV Red
PV Green
SV Red

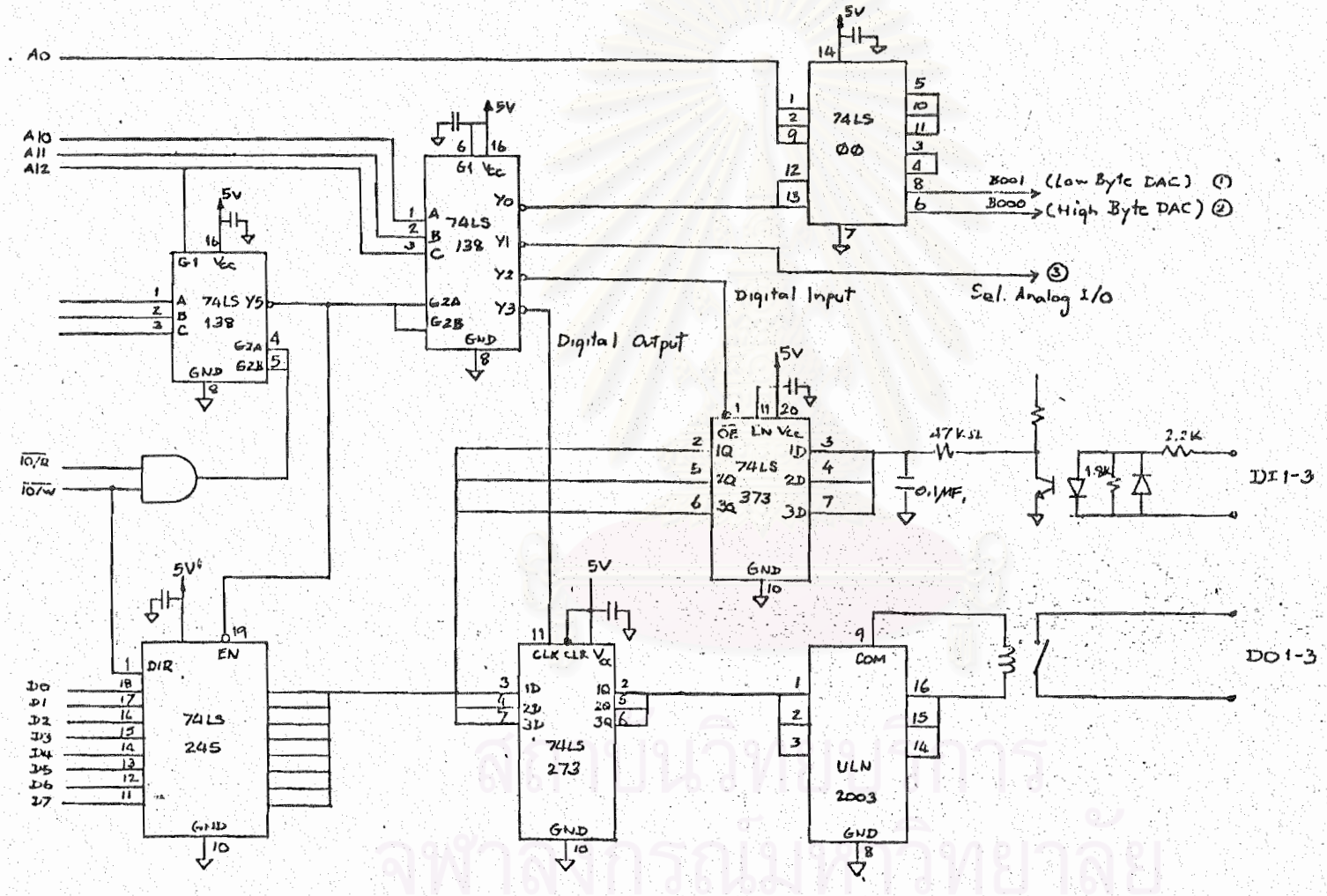

74LS 245
DL1614
DL1614
74LS 138
74LS 138
74LS ØØ
74LS 373
74LS 374
(A8ØØ)
(A4Ø4-7)
(A4ØØ-3)
(AØØØ)
1K
200

ง.5 วงจรส่วนอินพุทและเอาท์พุท




24V
0.1μ
Z2.2V
0.01μ
2k
100Ω
+15V
2.2k
10k
2.2k
2N2222
.0022μ
24V
2.2k
.001μ
33k
2N2905
2SC2331
4
5
6
1K
1μ
.8K
V5
LM
358
2.2k
.001μ
1.7k
1K
-15V
Y1 (4-20mA)


7
8
9
10
1K
2.2K
1μ
15V
2.2k
LM
348
.1μ
6.2V 40mW
-15V
Y2-Y5 (1-5 V.)



ภาคผนวก ค

วิธีการโปรแกรมกำหนดรูปแบบการควบคุมลง ROM

เนื่องจากตัวควบคุมเชิงเลขชนิดโปรแกรมได้ ที่สร้างขึ้นยังไม่มี Compiler บนไมโครคอมพิวเตอร์ที่ทำหน้าที่เปลี่ยนรหัส mnemonic เป็นไฟล์ Intel Hex เพื่อเขียนลง EPROM โดยใช้ EPROM Programmer CLK3000 บนไมโครคอมพิวเตอร์

ดังนั้นจึงจำเป็นต้องทราบวิธีการเปลี่ยนรหัส mnemonic เป็นไฟล์ Intel Hex โดยวิธีการเขียนโปรแกรมภาษาแอสแซมบลี้ ซึ่งเป็นวิธีที่ใช้ในการเขียนโปรแกรมทดสอบตัวควบคุมที่สร้างขึ้น ขั้นตอนทั้งหมดมีรายละเอียดดังนี้

ค.1 เขียนรหัส mnemonic รายละเอียดดูในบทที่ 5

ค.2 เขียนแอสแซมบลี้ จากรหัส mnemonic นำมาเขียนเป็นภาษาแอสแซมบลี้ได้ โดยสามารแบ่งประเภทของฟังก์ชันออกเป็น 4 ประเภท ได้ดังนี้

ค.2.1 ฟังก์ชันที่ไม่มีการส่งผ่านพารามิเตอร์ สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ค.1

Mnemonic	Assembly	Mnemonic	Assembly
ADD	ld bx,07fdah call bx	SUB	ld bx,0800dh call bx
MUL	ld bx,00040h call bx	DIV	ld bx,08073h call bx
ABS	ld bx,080beh call bx	SQR	ld bx,080a6h call bx
AND	ld bx,081cah call bx	OR	ld bx,081ech call bx
NOT	ld bx,0820eh call bx	CMP	ld bx,08220h call bx
HSL	ld bx,080d0h call bx	LSL	ld bx,0814dh call bx
FX	ld bx,07937h call bx		
PPID	ld bx,082cah call bx	CPID	ld bx,08550h call bx

รูปที่ ค.1 แสดงการแปลงฟังก์ชันที่ไม่มีการส่งผ่านพารามิเตอร์เป็นแอสแซมบลี้

ค.2.2 ฟังก์ชันที่มีการส่งผ่านค่าตัวแปร สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ค.2

Mnemonic	Assembly	Mnemonic	Assembly

LD DIn	mov al,n-1
	push ax
	mov bx,08272h
	call bx
		LD DOn	mov al,n-1
			push ax
			mov bx,082aeh
			call bx

LD FLn	mov al,n-1
	push ax
	mov bx,08290h
	call bx		

LD Xn	mov al,n-1
	push ax
	mov bx,0785ah
	call bx
		ST Yn	mov al,n-1
			push ax
			mov bx,078f0h
			call bx

**** n = channel no. & flag no.

รูปที่ ค.2 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตัวแปรเป็นแอสแซมบลี้

ค่า n ในรูปที่ ค.2 คือค่าของช่องสัญญาณของอินพุทเอาท์พุททั้งอนาลอคและดิจิตอล หรือ ค่าของสถานะ flag ของฟังก์ชัน PID โดยที่ค่า n ในรหัส mnemonic จะต้องถูกลบด้วย '1 ในแอสแซมบลี้ เช่น ต้องการอ่านค่าอนาลอคอินพุทของช่องสัญญาณที่ 1 เขียนคำสั่งได้ดังนี้

```
mnemonic :      LD   X1
assembly :      mov  al,0
                push ax
                mov  bx,0785ah
                call bx
```

ค.2.3 ฟังก์ชันที่มีการส่งผ่านค่าตำแหน่ง แบ่งเป็น 2 ประเภท คือ

1. ส่งผ่านค่าตำแหน่งหนึ่งค่า สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ค.3

Mnemonic	Assembly	Mnemonic	Assembly

LD Pn	mov ax,addr
	push ax
	mov bx,07884h
	call bx
		ST Pn	mov ax,addr
			push ax
			mov bx,078b2h
			call bx

LEDn	mov ax,addr
	push ax
	mov bx,07b04h
	call bx
		LAGn	mov ax,addr
			push ax
			mov bx,07a0eh
			call bx

TIMn	mov ax,addr
	push ax
	mov bx,07f76h
	call bx
		GIF	mov ax,addr
			push ax
			mov bx,08bceh
			call bx

รูปที่ ค.3 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตำแหน่งหนึ่งค่าเป็นแอสแซมบลี้

ค่า addr ในรูปที่ ค.3 คือค่าตำแหน่งของตัวแปรที่ใช้กับฟังก์ชัน โดยค่าตำแหน่งของแต่ละตัวแปรแสดงได้ดังรูปที่ ค.4

Variable	Address	Variable	Address
P1	01bh	P2	01eh
P3	021h	P4	024h
P5	027h	P6	02ah
P7	02dh	P8	030h
P9	033h	P10	036h
P11	039h	P12	03ch
P13	03fh	P14	042h
P15	045h		
LED1	0182h	LED2	018fh
LAG1	019ch	LAG2	01a3h
LAG3	01aah	LAG4	01b1h
TIM1	02a2h	TIM2	02a6h
TIM3	02aah	TIM4	02aeh

รูปที่ ค.4 แสดงตำแหน่งของตัวแปรที่ใช้ในการส่งผ่านค่าตำแหน่งหนึ่งค่า

ตัวอย่างเช่น ต้องการอ่านค่า P1 เขียนคำสั่งดังนี้

```
mnemonic :   LD   P1
assembly :   mov  ax,01bh
             push ax
             mov  bx,07884h
             call bx
```

ตัวอย่าง ต้องการเรียกฟังก์ชัน LAG ตัวที่ 2

```
mnemonic :   LAG2
assembly :   mov   ax,01a3h
             push  ax
             mov   bx,07a0eh
             call  bx
```

2. ส่งผ่านค่าตำแหน่งสองค่า สามารถแปลงแต่ละฟังก์ชันเป็นแอสแซมบลี้ได้ ดังรูปที่ ค.5

```
--------------------------------
 Mnemonic        Assembly
--------------------------------
 DEDn        mov  ax,addr1
             push ax
             mov  ax,addr2
             push ax
             mov  bx,07c6dh
             call bx
--------------------------------
```

รูปที่ ค.5 แสดงการแปลงฟังก์ชันที่มีการส่งผ่านค่าตำแหน่งสองค่าเป็นแอสแซมบลี้

ค่าตำแหน่งของ addr1 และ addr2 ของฟังก์ชันแสดงได้ดังรูปที่ ค.6

Variable	Address1	Address2
DED1	01b8h	01cah
DED2	0206h	0218h
DED3	0254h	0266h

รูปที่ ค.6 แสดงตำแหน่งของตัวแปรที่ใช้กับฟังก์ชัน dead time

ตัวอย่าง การเรียกใช้ฟังก์ชัน dead time ที่ 1 เขียนคำสั่งได้ดังนี้

```
mnemonic :   DED1
assembly :   mov  ax,01b8h
             push ax
             mov  ax,01cah
             push ax
             mov  bx,07c6dh
             call bx
```

เมื่อได้ assembly แล้วนำมาเขียนโปรแกรม ซึ่งรูปร่างของโปรแกรมแสดงได้ดังรูปที่ ค.7

```
NAME  user_rom                          ------------------
ASSUME CS:CODE,DS:DATA
DATA   SEGMENT WORD PUBLIC 'DATA'
DATA   ENDS                              โปรแกรมส่วนนี้
CODE   SEGMENT WORD PUBLIC 'CODE'          คงที่
userp  proc  far
       push  ax
       push  bx                          ------------------
       mov   al,0h
       push  ax
       mov   bx,0785ah                  โปรแกรมส่วนนี้ได้จาก
       call  bx  ; LD X1                การแปลง mnemonic เป็น
        ............                      แอสแซมบลี้
        ............
       mov   bx,082cah
       call  bx  ; BPID                  ------------------
       pop   bx
       pop   ax                          โปรแกรมส่วนนี้
       ret                                 คงที่
usep   ENDP
CODE   ENDS
       END                               ------------------
```

รูปที่ ค.7 รูปร่างของโปรแกรม assembly ที่ใช้สร้าง Intel Hex

ค.3 Compile & Link ทำการ compile โดยใช้ compiler ASM86 และ link โดยใช้ iAPX 86,88 Utility มีขั้นตอนดังนี้

1. >ASM86 file.asm
2. >LINK86 file.obj to file.lnk
3. >LOC86 file.lnk to file.loc AD(SM(CODE(08000H)))
4. >OH86 file.loc to file.h

เมื่อหมดขั้นตอนนี้จะได้ file.h ซึ่งเป็นฟอร์แมทของ Intel hex 16 บิท

ค.4 แปลง Intel hex 16 บิท เป็น 8 บิท เนื่องจาก EPROM Programming ทำงานกับไฟล์ที่มีฟอร์แมทเป็น Intel hex 8 บิท ทำการแปลงโดยใช้โปรแกรม INTELH ซึ่งเขียนขึ้นเองดังนี้

```
>INTELH <ret.>
>input file:file.h
>output file:file.hex
```

ค.5 เขียน EPROM โดยใช้ EPROM Programmer CLK3000 เขียน file.hex บน EPROM นำไปใส่ในบอร์ดหน่วยความจำตำแหน่ง 08000H เพื่อกำหนดรูปแบบการควบคุมให้กับตัวควบคุมเชิงเลข