บทที่ 5

ซอฟท์แวร์เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้

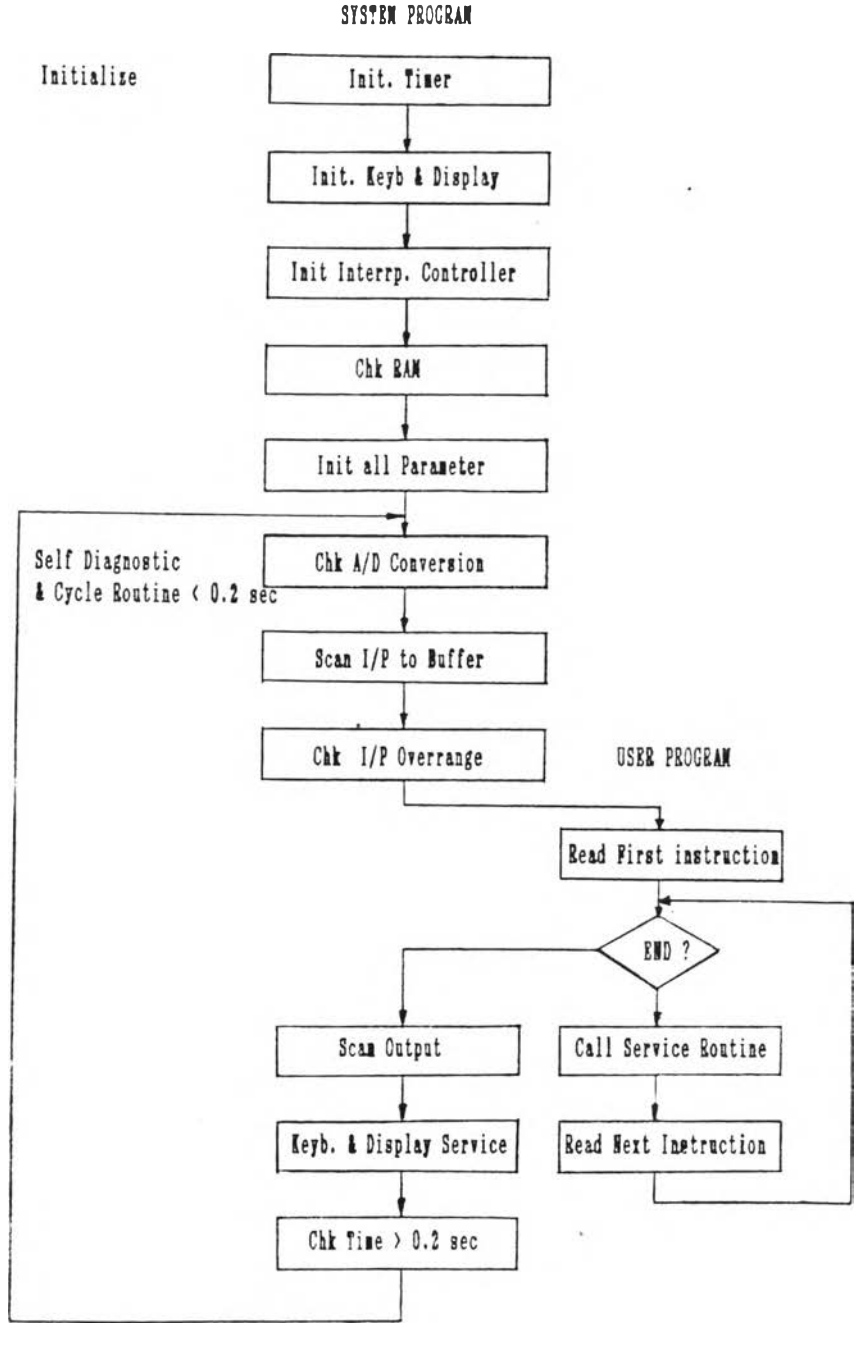
เครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ สามารถทำการควบคุมได้หลายรูปแบบ ซึ่งการกำหนดรูปแบบการควบคุมขึ้นกับโปรแกรมที่เขียนโดยผู้ใช้งาน ซึ่งการพิจารณาจากแนวความคิดการออกแบบใน บทที่ 3 ได้ลักษณะของโปรแกรมที่เลือกใช้ในการเขียนเพื่อกำหนดรูปแบบการควบคุม คือ โปรแกรมที่มีลักษณะภาษาแบบ แอสแซมบลี้ สำหรับภาษาที่ใช้ในการพัฒนาซอฟท์แวร์บนเครื่องควบคุมเชิงเลขชนิดโปรแกรมได้ 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 ซึ่งมีโฟลว์ชาร์ทแสดงการทำงานได้ 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 โฟลว์ชาร์ทการทำงานของเครื่องควบคุมเชิงเลข

(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
- Interpolation

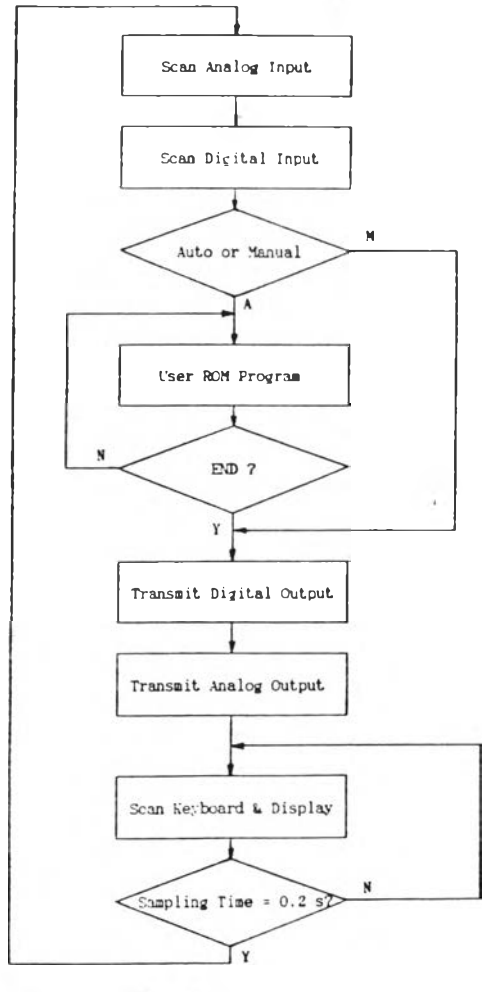
(4) **โปรแกรมควบคุมแบบ PID**
- Basic PID
- Cascade PID

5.2 การทำงานของโปรแกรมควบคุมระบบ

จากโฟลว์ชาร์ทรูปที่ 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 แสดงโฟลว์ชาร์ทการทำงานแบบวนรอบ

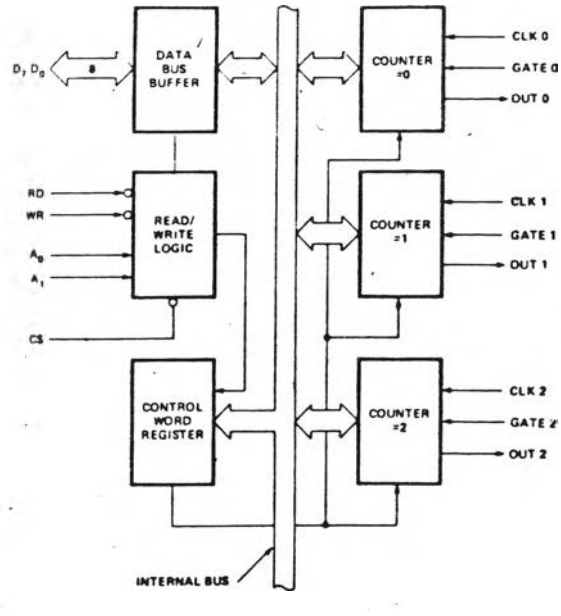


รูปที่ 5.2 โฟลว์ชาร์ทการทำงานแบบวนรอบ

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:

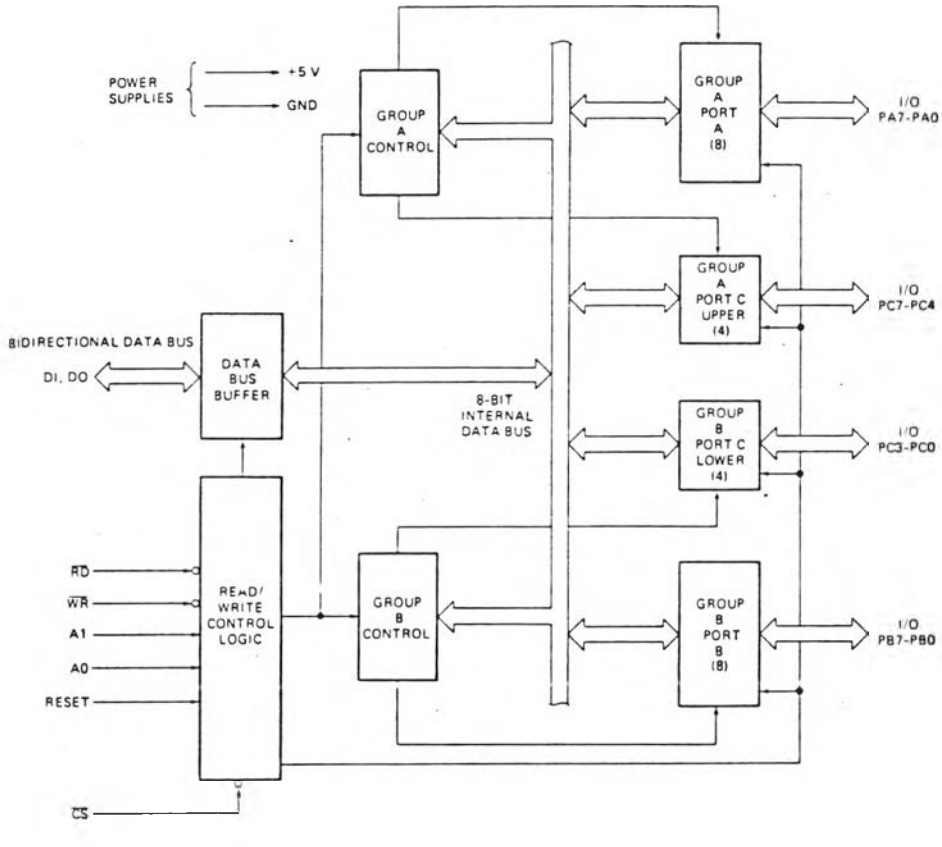
0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

รูปที่ 5.4 ฟอร์แมทคำสั่งควบคุมของ 8253

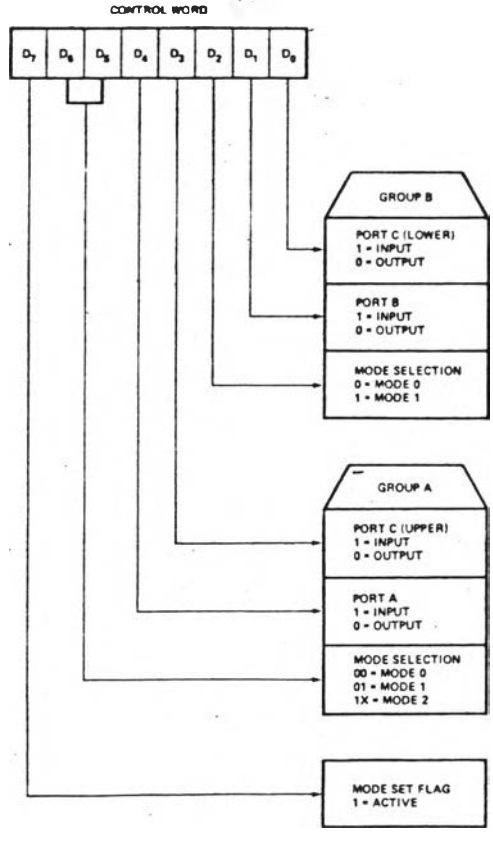
เครื่องควบคุมโปรแกรมให้ 8253 ทำหน้าที่นับเวลา เพื่อจับเวลาการสุ่มข้อมูลจากภายนอก โดย 8253 จะรับสัญญาณเข้ามีความถี่ 1.25 MHz เข้าที่ตัวนับที่ 0 และสร้างสัญญาณจัตุรัสขนาด 50 ms. เพื่ออินเตอร์รัพซีพียู การทำงานถูกกำหนดโดยการส่งข้อมูลจากซีพียู ไปยังพอร์ทเบอร์ 0C803H ของ 8253 3 ครั้ง ตามลำดับ ดังนี้

```
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



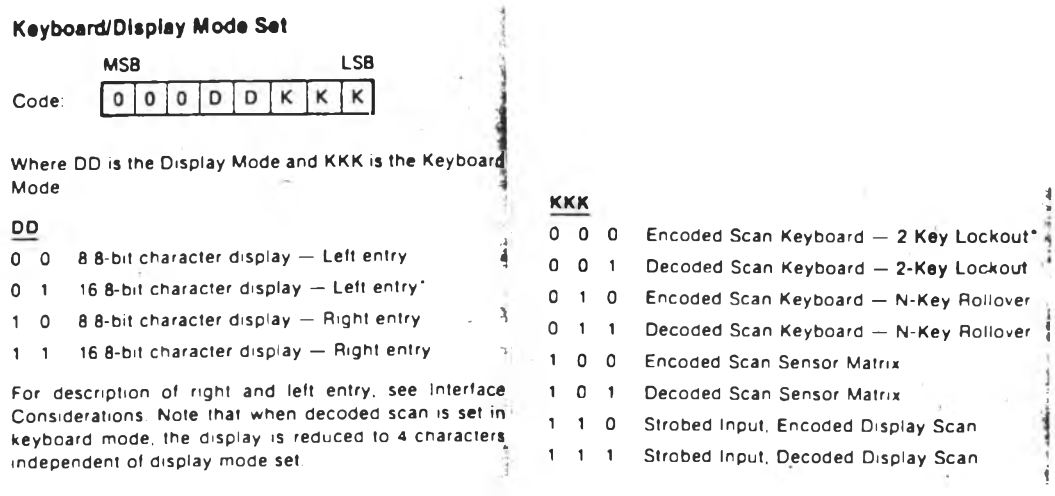
รูปที่ 5.5 โครงสร้างภายใน 8255A



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

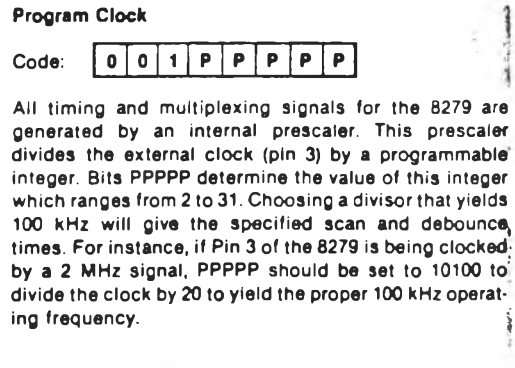
For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

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.

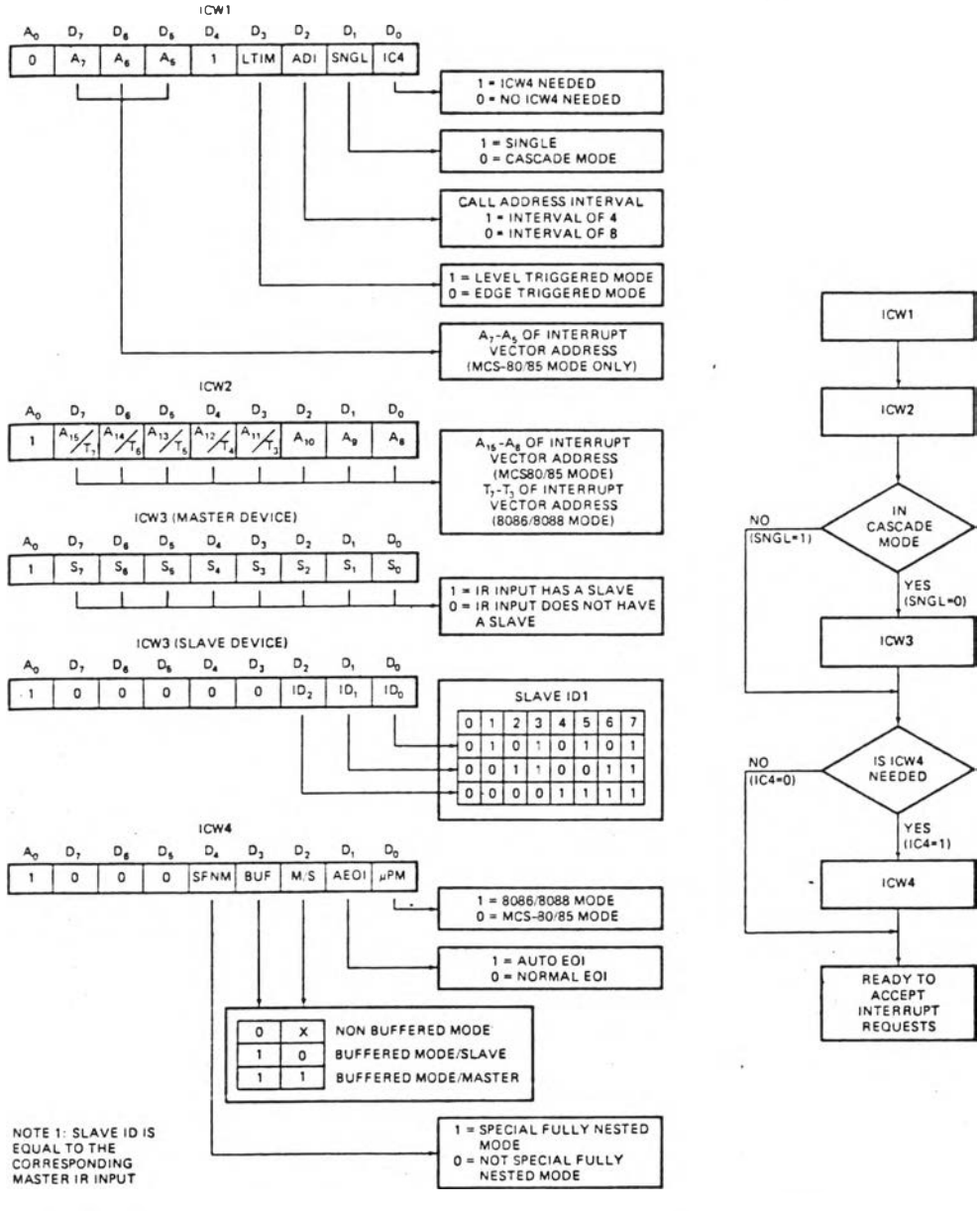
รูปที่ 5.8 ฟอร์แมทคำสั่งของ Internal Clock ของ 8279

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 สำหรับอินเตอร์รัพท์แรกบน 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

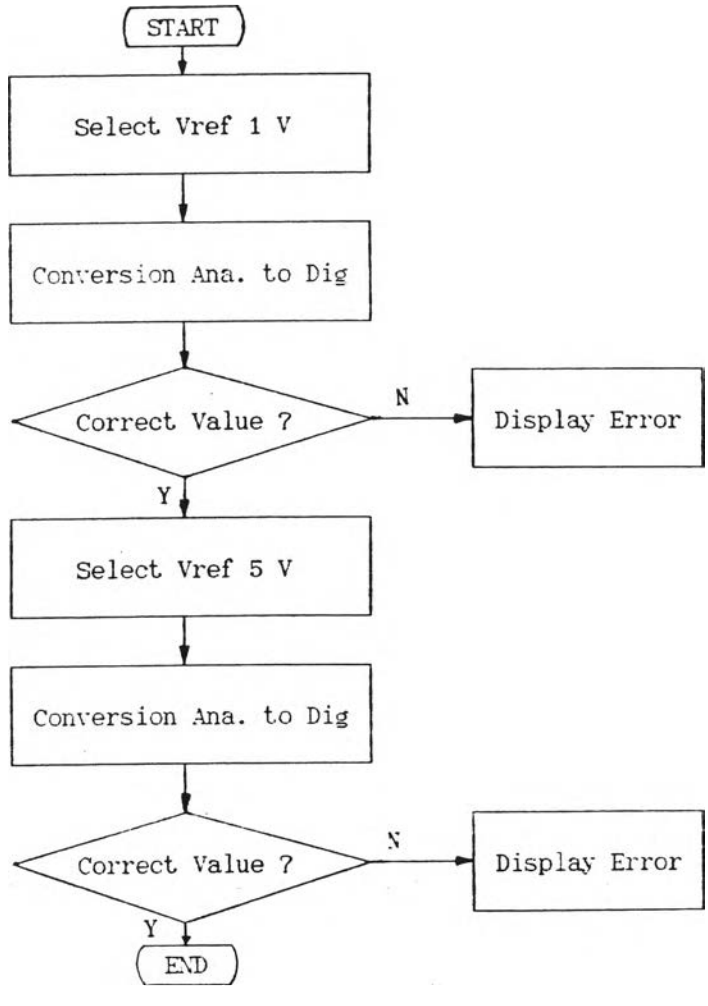
5.3.2 โปรแกรมกำหนดค่าเริ่มต้นของพารามิเตอร์ เครื่องควบคุมโปรแกรมโหมดการทำงานเริ่มต้นเป็นแบบ Manual ในโหมดนี้สัญญาณที่ส่งออกไปควบคุมโปรเซสถูกควบคุมโดยแป้นพิมพ์ สำหรับโปรแกรมกำหนดค่าเริ่มต้นของพารามิเตอร์ในการควบคุม โปรแกรมจะกำหนดค่าเริ่มต้นดังแสดงใน ตารางที่ 5.1 โดยพารามิเตอร์เหล่านี้สามารถแสดงผลและเปลี่ยนแปลงได้โดยแป้นพิมพ์

ตารางที่ 5.1

พารามิเตอร์	จำนวน	ความหมาย	หน่วย	ค่าเริ่มต้น
Pn	1-8	ค่าคงที่ช่วยในการคำนวณ	-	0.0
PB	1,2	ค่า Proportional band	%	100
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 Interpolation	%	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 โฟลว์ชาร์ทแสดงการทำงานของโปรแกรมแสดงดังรูปที่ 5.10

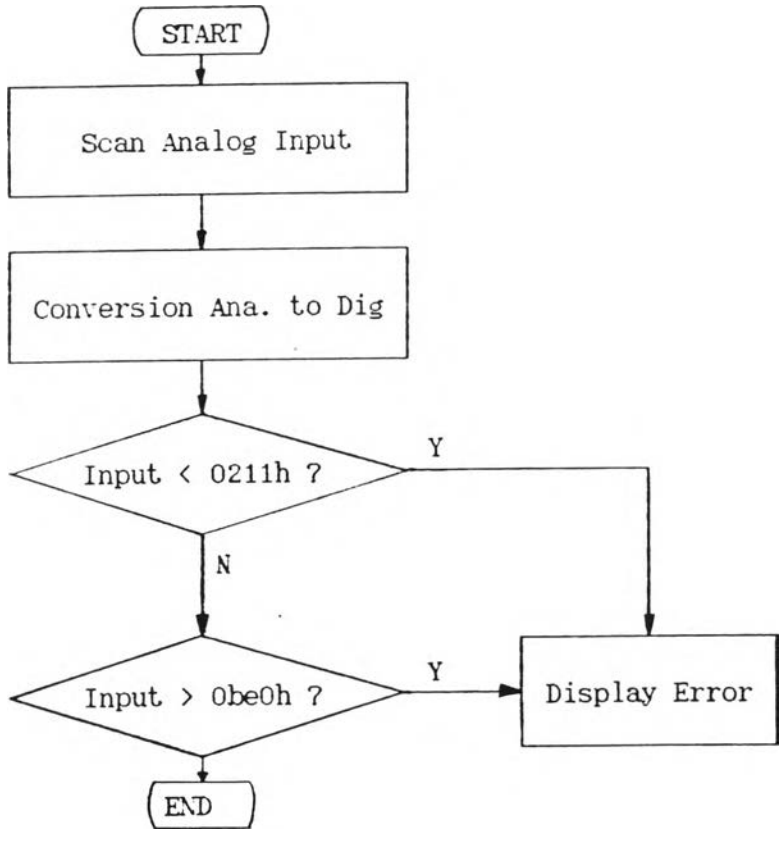


รูปที่ 5.10 โฟลว์ชาร์ทของโปรแกรมตรวจสอบ DAC

(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

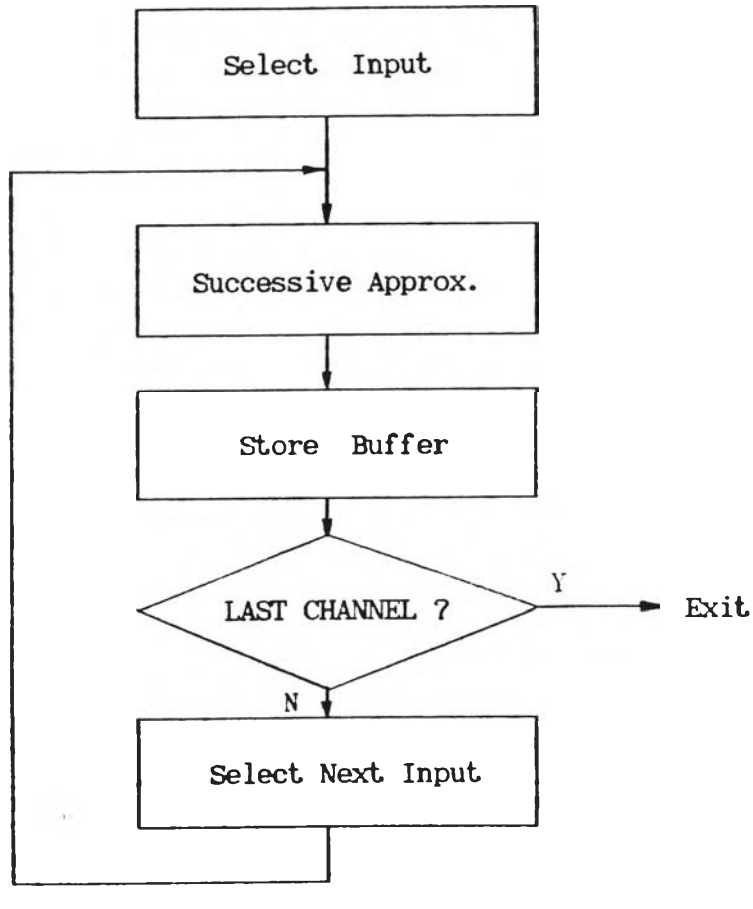
(4) **โปรแกรมตรวจสอบอินพุทเกินพิสัย** อินพุทของเครื่องควบคุมมีพิสัยอยู่ระหว่างค่า 1-5 V.(ค่าที่ได้จาก DAC คือ 0211H กับ 0BE0H) โปรแกรมจะทำการตรวจสอบค่าตัวแปรที่ได้จากโปรแกรมสแกนอินพุท ว่ามีค่าอยู่ในพิสัยหรือไม่ โฟลว์ชาร์ทแสดงการทำงานของโปรแกรมแสดงได้ดังรูปที่ 5.11



รูปที่ 5.11 โฟลว์ชาร์ทของโปรแกรมตรวจสอบอินพุทเกินพิสัย

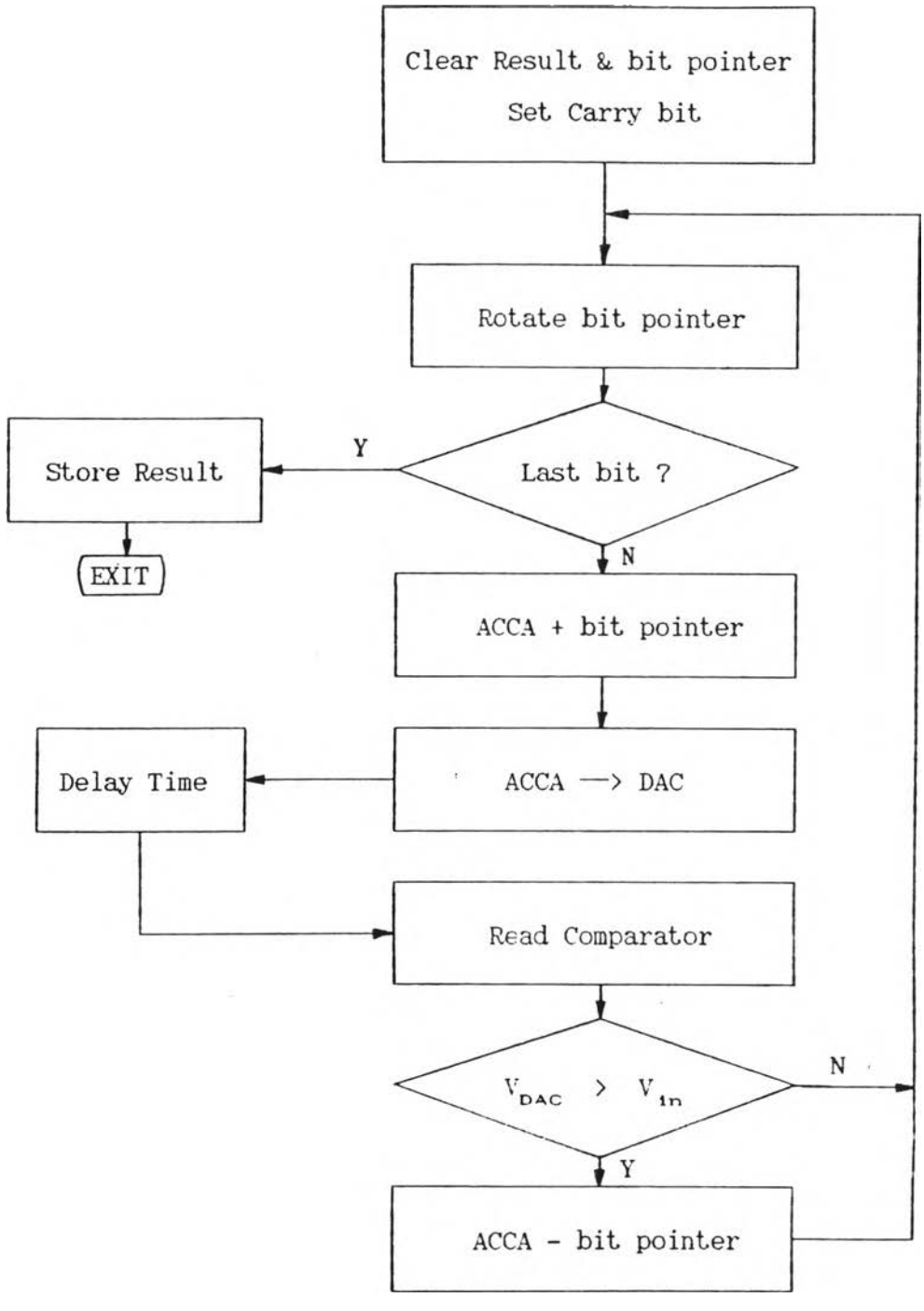
5.3.4 โปรแกรมสแกนอินพุทและเอาท์พุท

(1) **โปรแกรมสแกนอนาลอกอินพุท** เนื่องจากเครื่องควบคุมมีอินพุทและเอาท์พุทหลายช่องสัญญาณ ฮาร์ดแวร์ในส่วนอินเตอร์เฟสนี้จึงมีการออกแบบเพื่อลดจำนวนไอซี ADC และ DAC โดยทุกๆช่องสัญญาณของอินพุทและเอาท์พุทจะใช้ DAC เพียงตัวเดียว กับอนาลอกสวิทช์และตัวเปรียบเทียบ (Comparator) ทำงานร่วมกับโปรแกรมเพื่อแปลงสัญญาณอนาลอกเป็นสัญญาณเชิงเลข โดยอัลกอริทึมในการแปลงใช้วิธี Succesive Approximation และเก็บค่าไว้ที่บัฟเฟอร์สำหรับโปรแกรมอื่นเรียกใช้ โฟลว์ชาร์ทแสดงดังรูปที่ 5.12



รูปที่ 5.12 โฟลว์ชาร์ทของโปรแกรมสแกนอนาลอกอินพุท

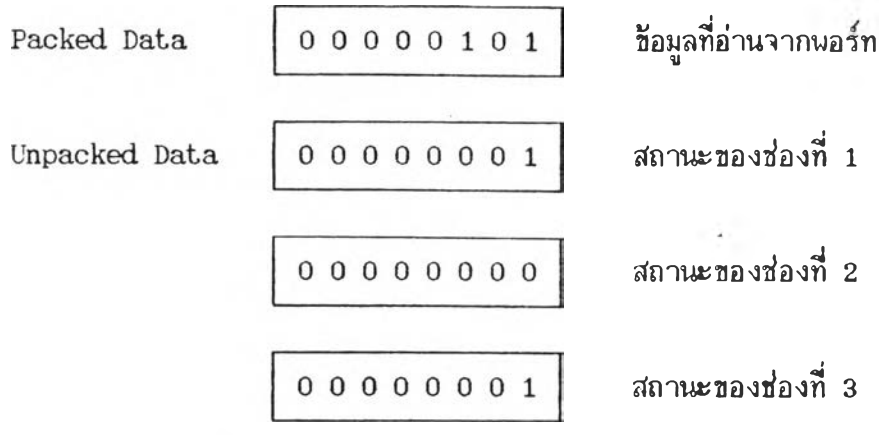
สำหรับอัลกอริทึมการแปลงสัญญาณแบบ Successive Approximation มีโฟลว์ชาร์ท แสดงดังรูปที่ 5.13 [8]



รูปที่ 5.13 โฟลว์ชาร์ทของ Successive Approximation

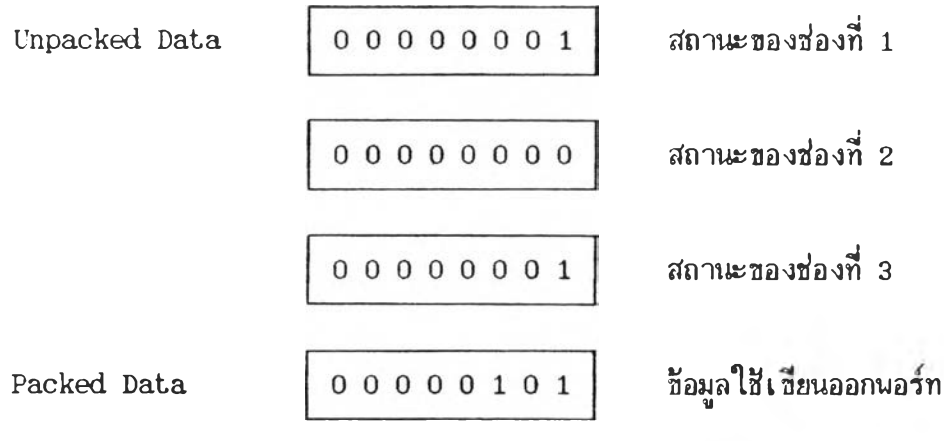
(2) โปรแกรมสแกนอนาลอคเอาท์พุท โปรแกรมจะส่งสัญญาณอนาลอคออกที่เอาท์พุท โดยการเลือกช่องสัญญาณของอนาลอคสวิทช์ก่อนส่งข้อมูลผ่าน DAC เพื่อแปลงค่าสัญญาณอนาลอคออกที่ช่องสัญญาณที่เลือกไว้

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



รูปที่ 5.14 การ unpack ดิจิตอลอินพุท

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



รูปที่ 5.15 การ pack ข้อมูลของดิจิตอลเอาท์พุท

5.3.5 โปรแกรมรับค่าจากแป้นพิมพ์และแสดงผล โปรแกรมแบ่งเป็น 2 ส่วน คือ

5.3.5.1 โปรแกรมรับค่าแป้นพิมพ์และแสดงผลด้านหน้า โปรแกรมส่วนนี้แบ่งเป็นโปรแกรมย่อยได้ 2 โปรแกรม ดังนี้

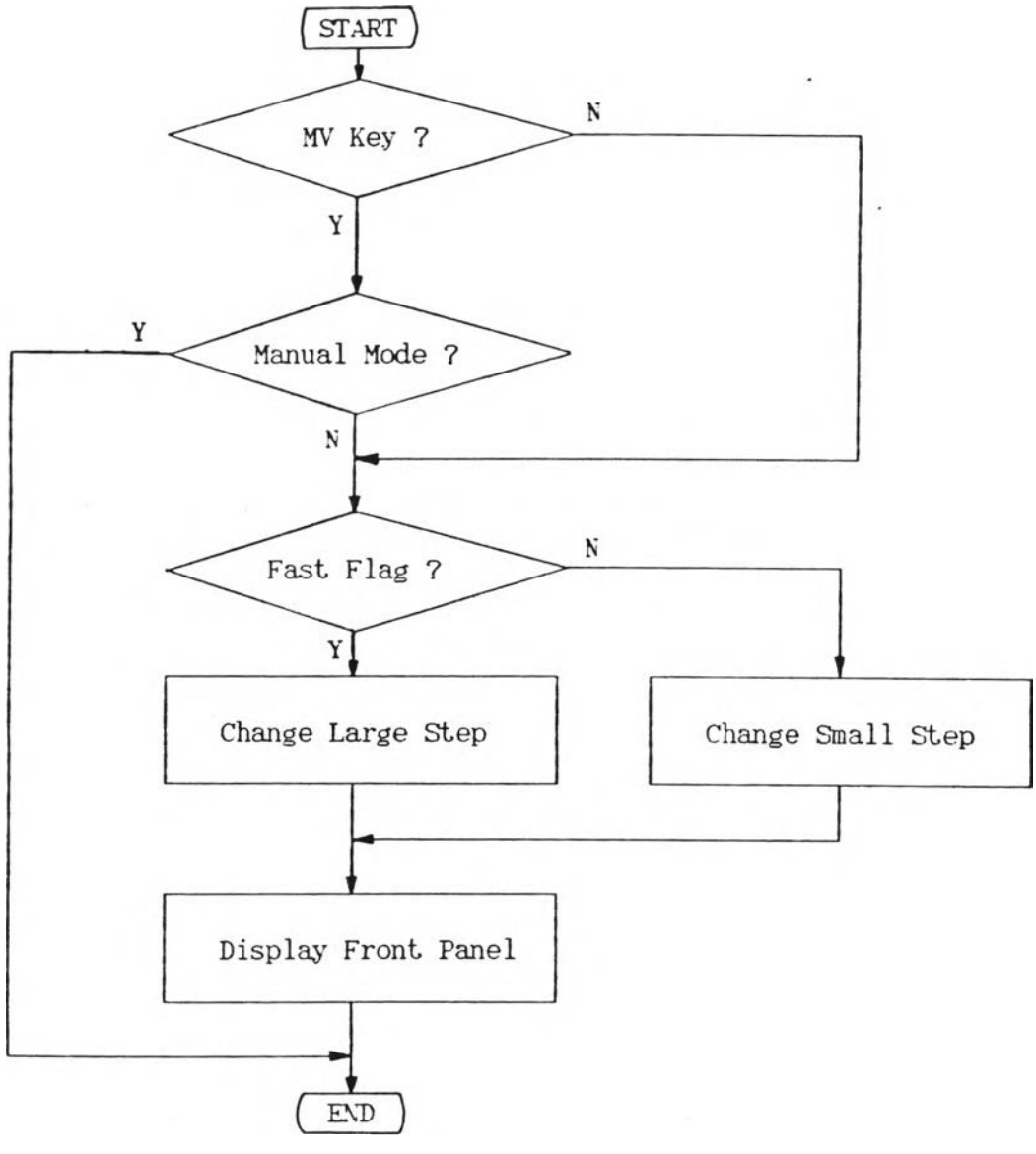
(1) โปรแกรมตรวจสอบแป้นพิมพ์ด้านหน้า ด้านหน้าของเครื่องควบคุมมี 10 แป้นพิมพ์ แต่ละตัวมีหน้าที่ดังตารางที่ 5.2

ตารางที่ 5.2

แป้นพิมพ์	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)

แป้นพิมพ์ 8 ตัวแรกในตารางที่ 5.2 ต่อกับอินพุทพอร์ทที่ AC00 การทำงานของโปรแกรมตรวจสอบแป้นพิมพ์ เมื่อมีการกดแป้นพิมพ์แต่ละตัวมีดังนี้

- 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 โฟลว์ชาร์ทแสดงการทำงานแสดงดังรูปที่ 5.16



รูปที่ 5.16 โฟลว์ชาร์ทของโปรแกรมการเปลี่ยนแปลงค่า MV,SV

(2) โปรแกรมแสดงผลด้านหน้า ส่วนแสดงผลด้านหน้าประกอบด้วย

- LED 7 ส่วน 4 ตัว เพื่อแสดงค่าตัวแปรโปรเซส (PV), ค่าเป้าหมาย (SV), ค่าตัวแปรควบคุม (MV) ขึ้นอยู่กับแป้นพิมพ์ 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

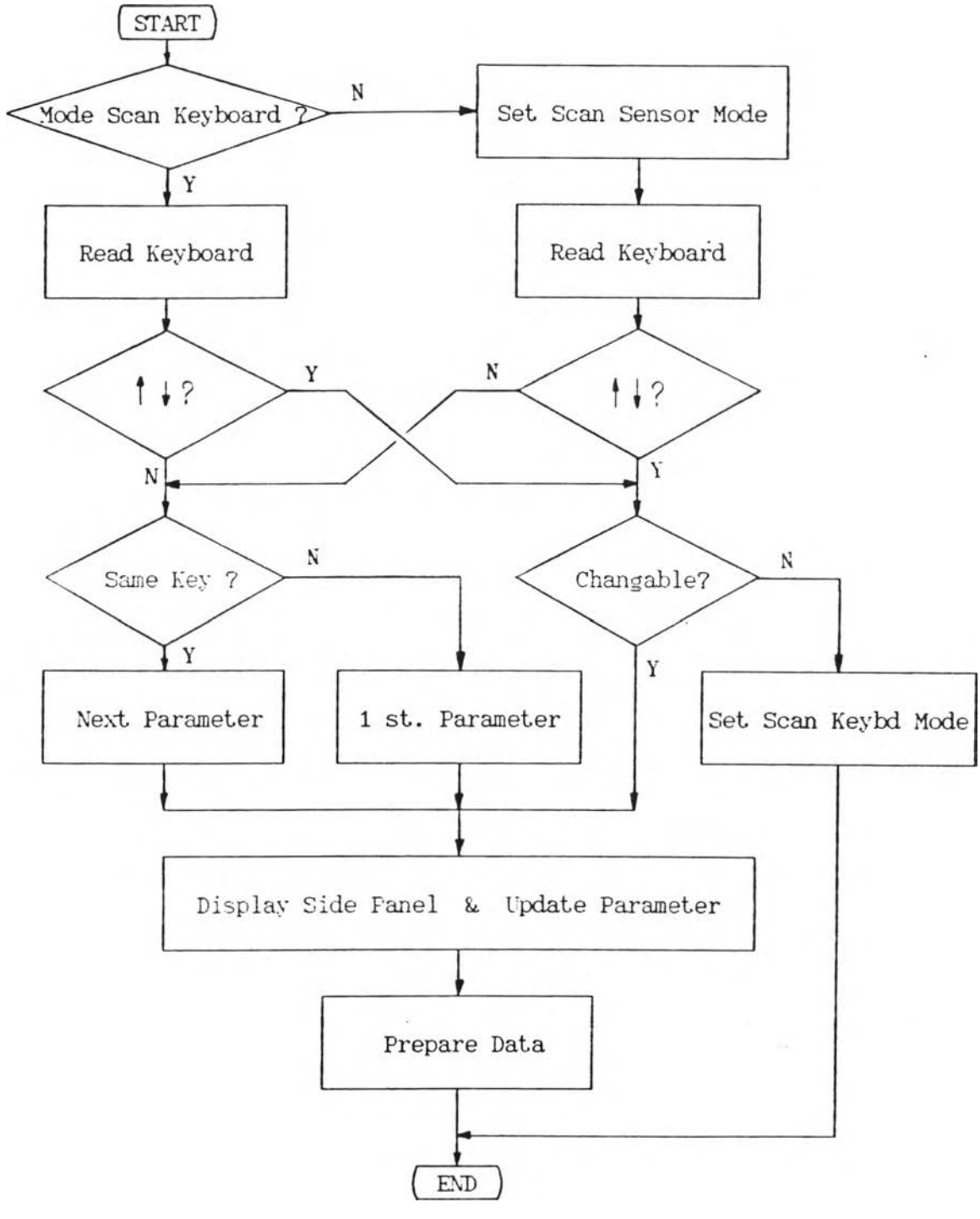
5.3.5.2 โปรแกรมรับค่าแป้นพิมพ์และแสดงผลด้านข้าง แบ่งเป็นโปรแกรมย่อยได้ 3 โปรแกรม ดังนี้

(1) โปรแกรมตรวจสอบแป้นพิมพ์ด้านข้าง 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 $\triangle$, $\triangledown$, $\Leftrightarrow$ 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

สำหรับโฟลว์ชาร์ทแสดงการทำงานของโปรแกรมเขียนได้ ดังรูปที่ 5.17



รูปที่ 5.17 โฟลว์ชาร์ทของโปรแกรมตรวจสอบแป้นพิมพ์ด้านเข้าง

ตารางที่ 5.3 แสดงชื่อและหน้าที่ของแป้นพิมพ์

แป้นพิมพ์	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** เป็นส่วนหนึ่งในโฟลว์ชาร์ทรูปที่ 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** เป็นโมดูลสุดท้ายในโฟลว์ชาร์ทรูปที่ 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 Interpolation** การทำงานของโปรแกรม Interpolation จะใช้ค่าหลัก 10 ค่า เมื่อค่าอินพุทอยู่ระหว่างจุด 2 จุด ใน 10 ค่านี้ จะทำการคำนวณโดยอาศัยค่าความชันระหว่างจุด 2 จุดนั้นช่วยในการคำนวณค่า ดังนั้นเมื่อมีการเปลี่ยนแปลงค่าหลักใน Interpolation จึงจำเป็นต้องมีการคำนวณค่าความชันใหม่ การทำเช่นนี้จะช่วยเพิ่มความเร็วในการทำ Interpolation เพราะเมื่อมีการทำ Interpolation 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

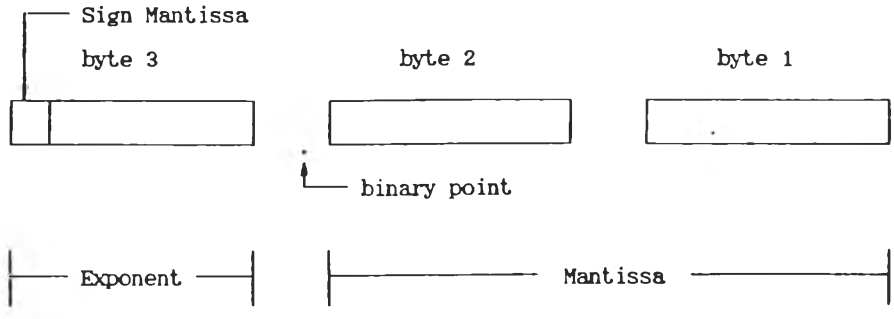
สมการพีชคณิต	RPN
A+B	AB+
A(B+C)	BC+A*
(A/B)+C	AB/C+

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 ตัวอย่างหน่วยความจำที่เก็บตัวเลข

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

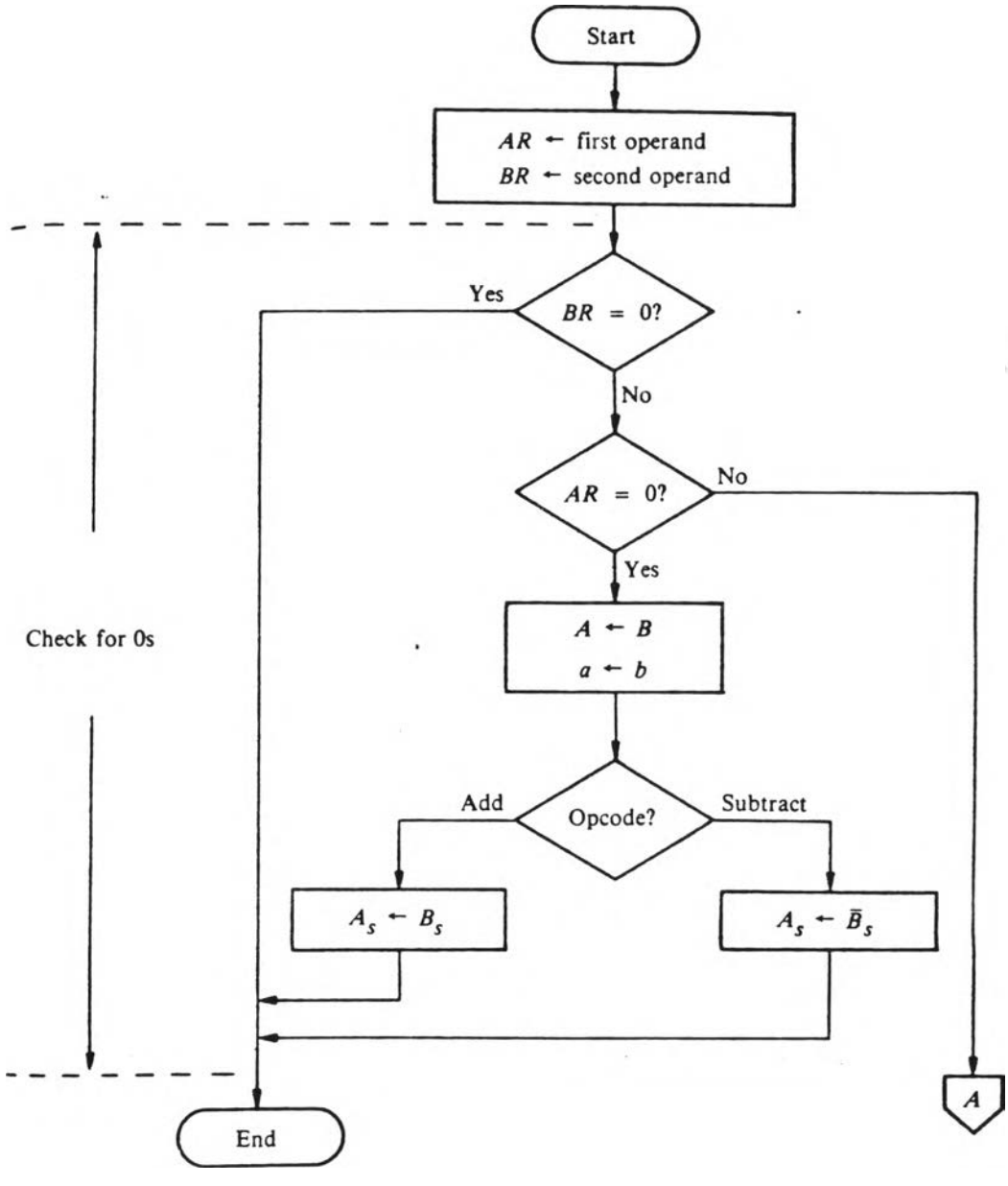
(ง) ให้ค่า exponent ของผลลัพธ์เท่ากับ ค่า exponent ของ Operand ที่มากกว่า

(จ) ทำ normalize ตัวเลขที่เป็นผลลัพธ์

(ฉ) ตรวจสอบการเกิด Overflow และ Underflow

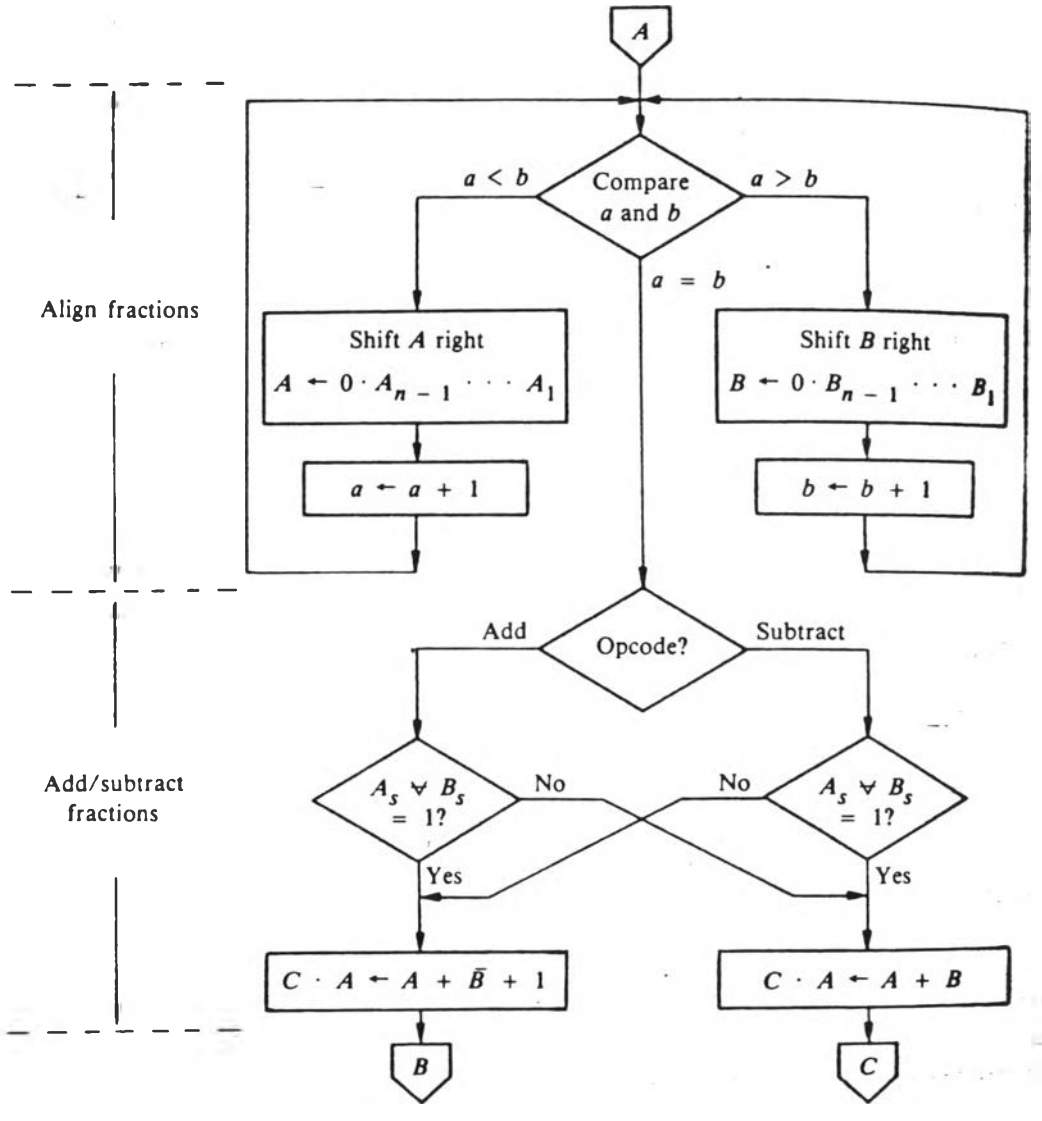
โฟลว์ชาร์ทของโปรแกรมแสดงดังรูปที่ 5.21-5.24 อธิบายถึงการบวก ลบ ของค่า Operand 2 ตัว คือ

$$AR \leftarrow AR \pm BR$$



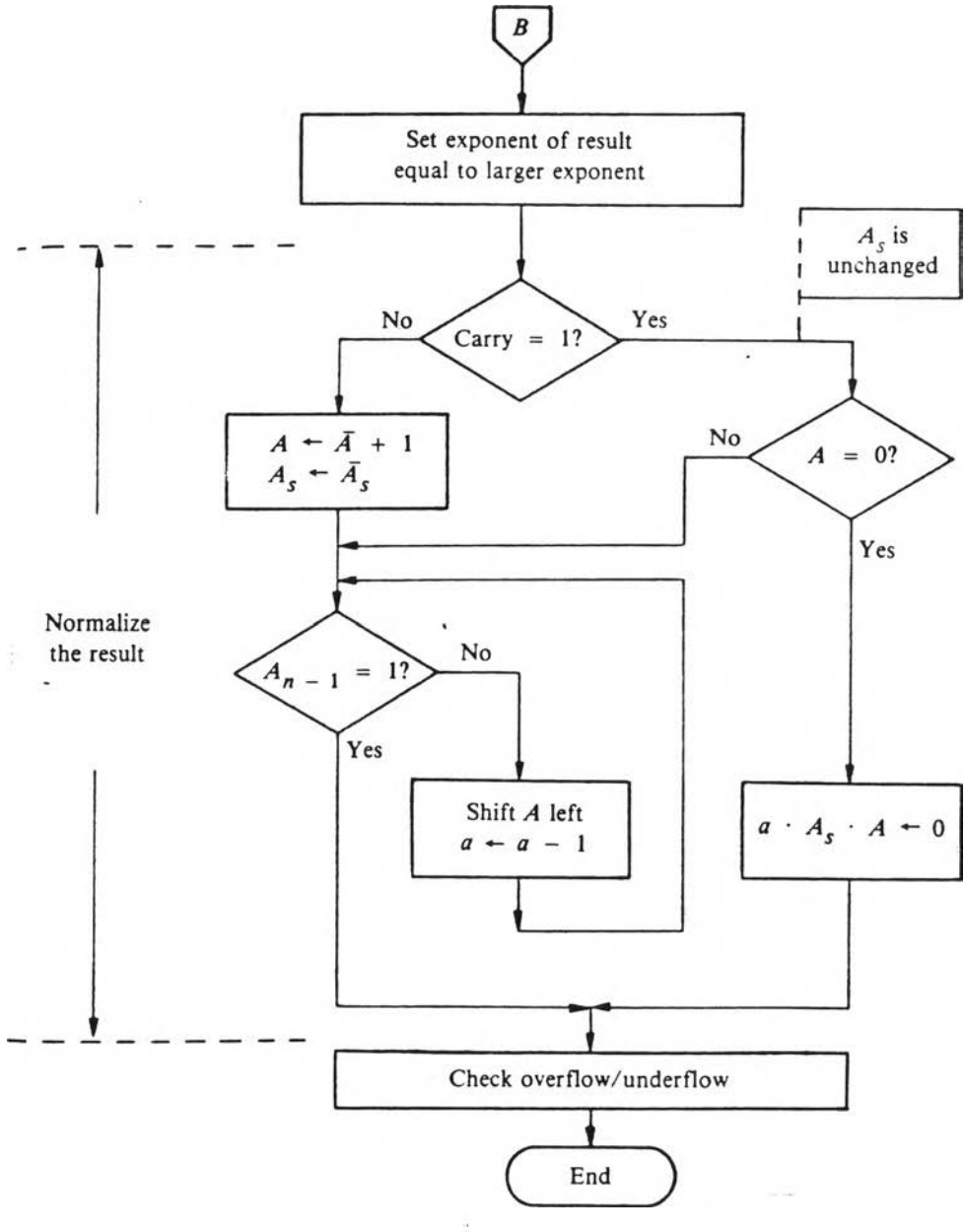
(a) การตรวจสอบค่า 0

รูปที่ 5.21 โฟลว์ชาร์ทการบวกและลบเลขทศนิยม



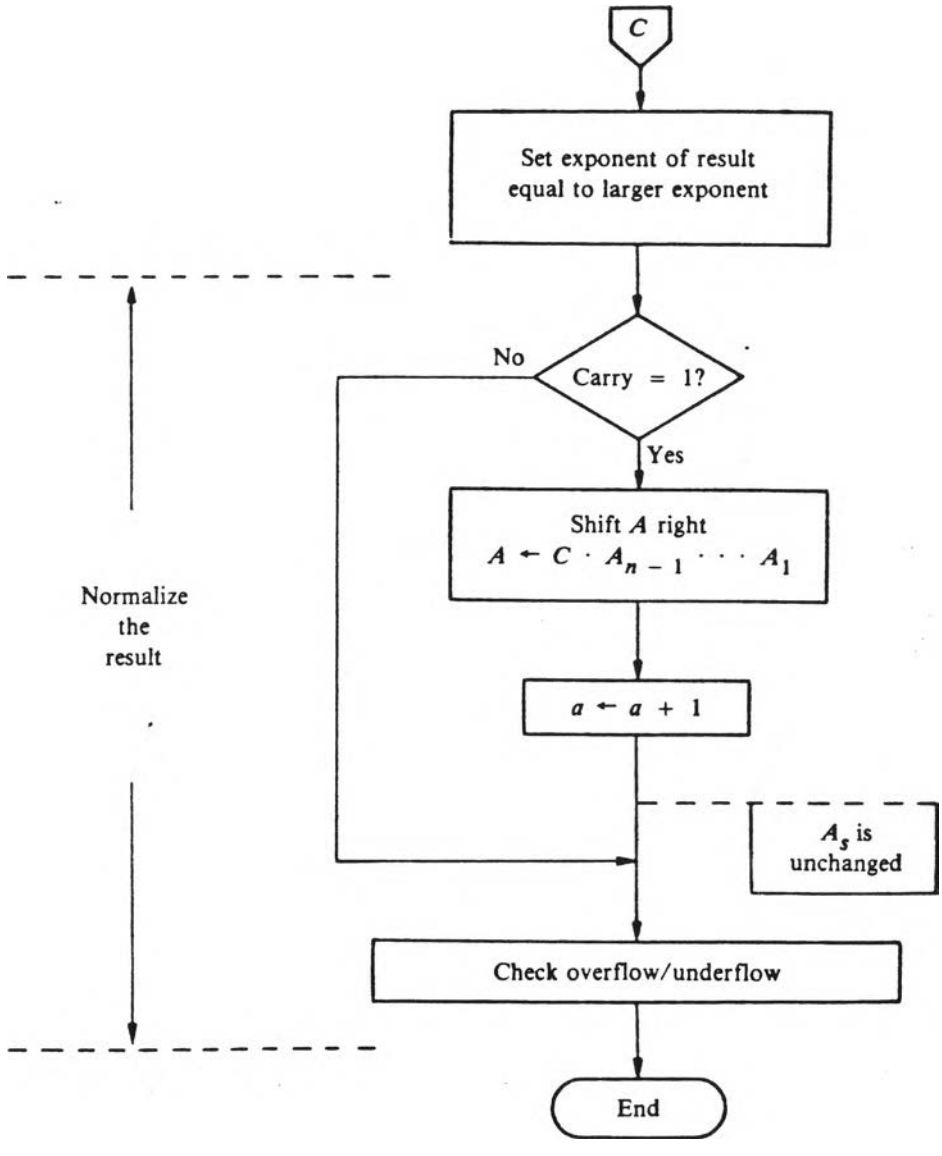
(b) align และบวกลบ mantissa

รูปที่ 5.22 โฟลว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)



(c) หาค่า exponent และ normalize ค่าผลลัพธ์

รูปที่ 5.23 โฟลว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)



(d) หาค่า exponent และ normalize ค่าผลลัพธ์

รูปที่ 5.24 โฟลว์ชาร์ทการบวกและลบเลขทศนิยม (ต่อ)

สัญญลักษณ์ที่ใช้ในโฟลว์ชาร์ทมีดังนี้

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]

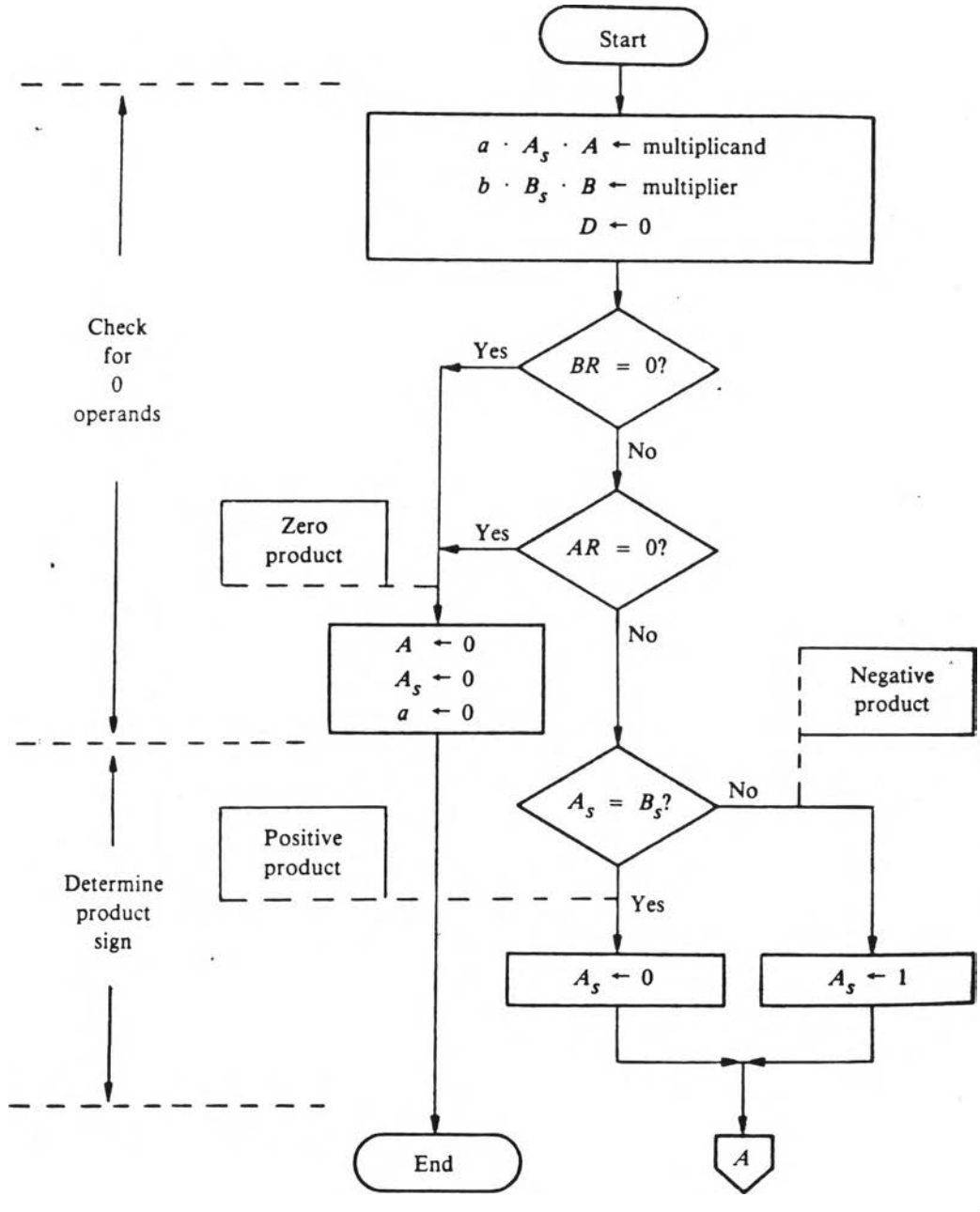
(ก) ตรวจสอบค่า Operand เป็น 0

(ข) หาเครื่องหมายของผลลัพธ์ที่ได้จากการคูณ

(ค) บวกค่า exponent

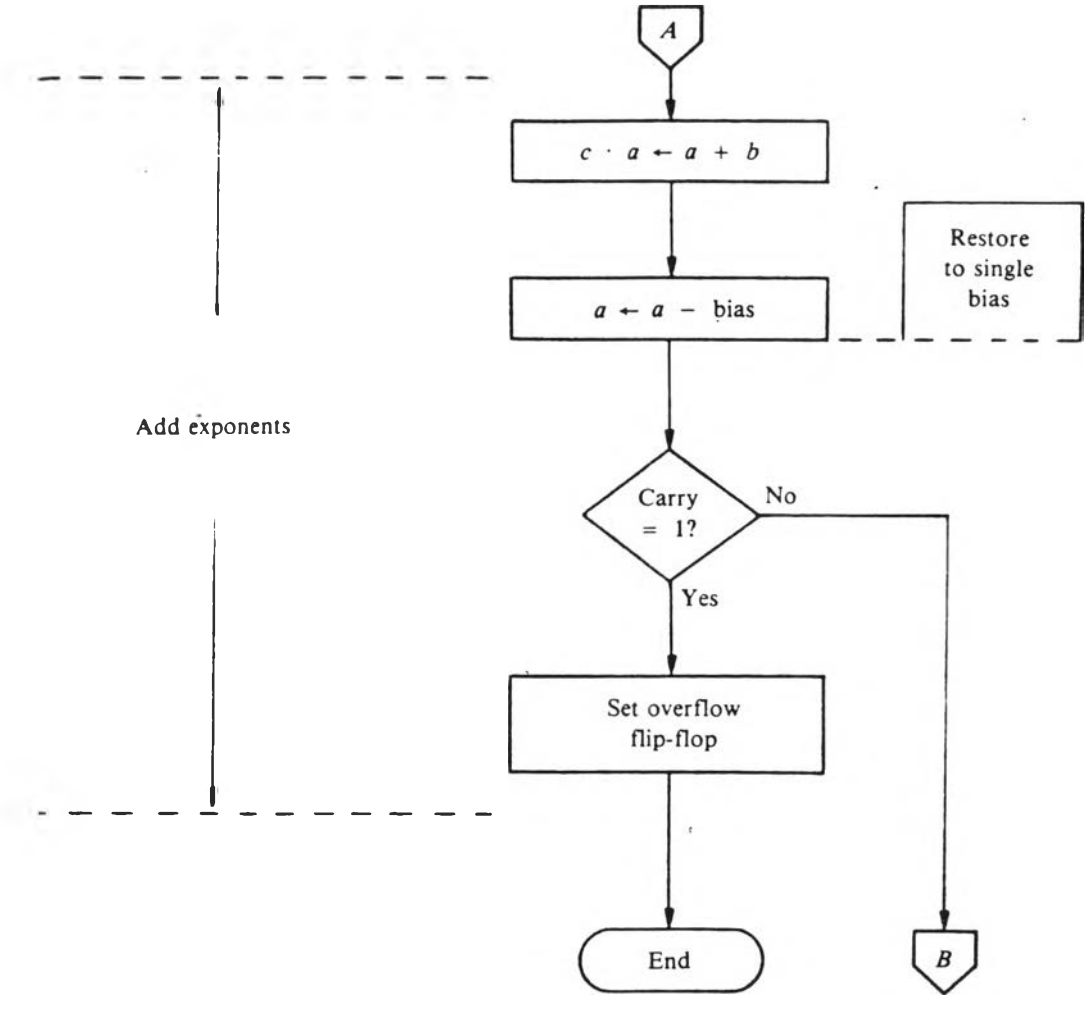
(ง) คูณค่า mantissa

(จ) normalize ผลลัพธ์



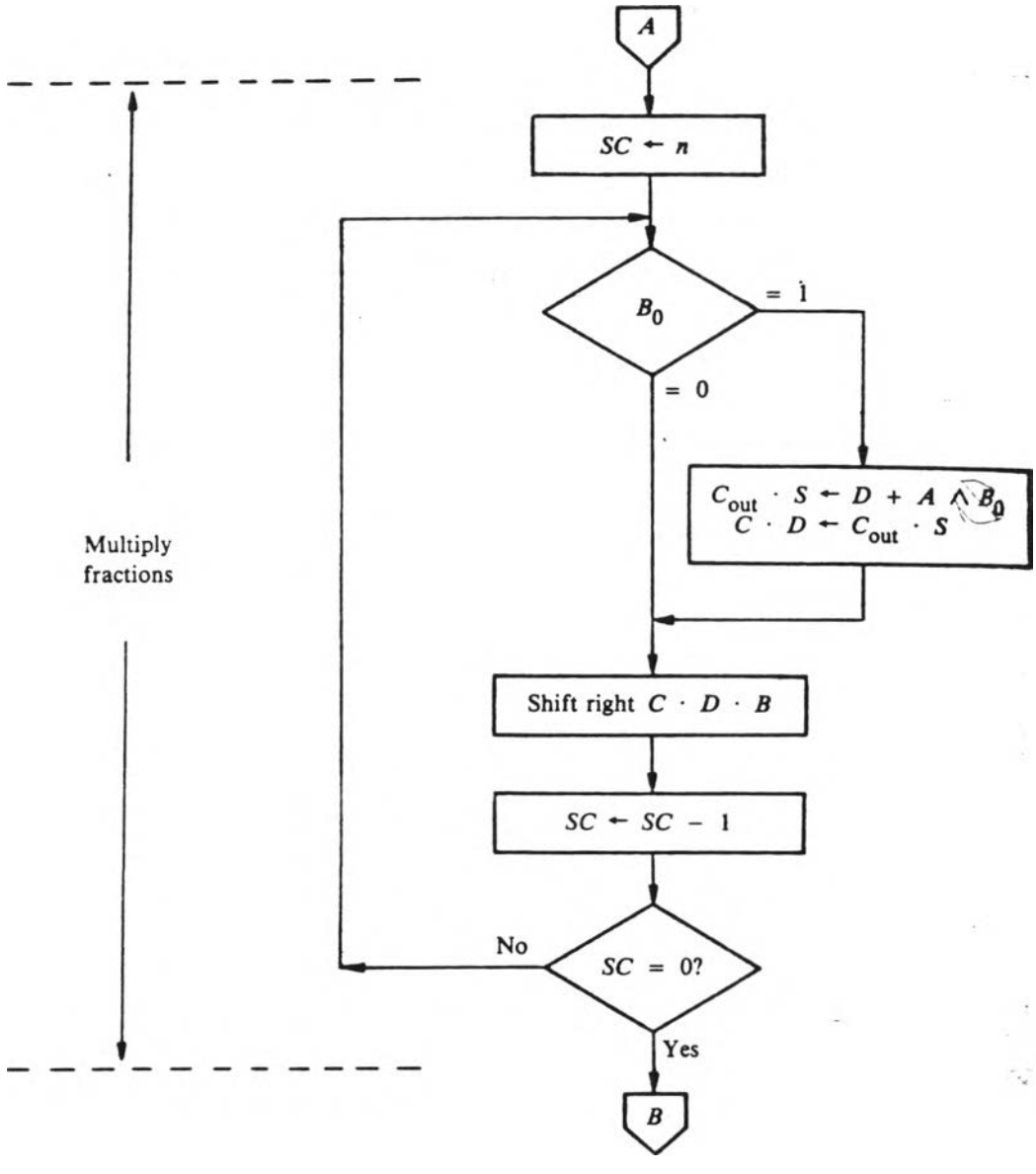
(a) ตรวจสอบค่า 0 และหาเครื่องหมายผลลัพธ์

รูปที่ 5.25 โฟลว์ชาร์ทการคูณเลขทศนิยม



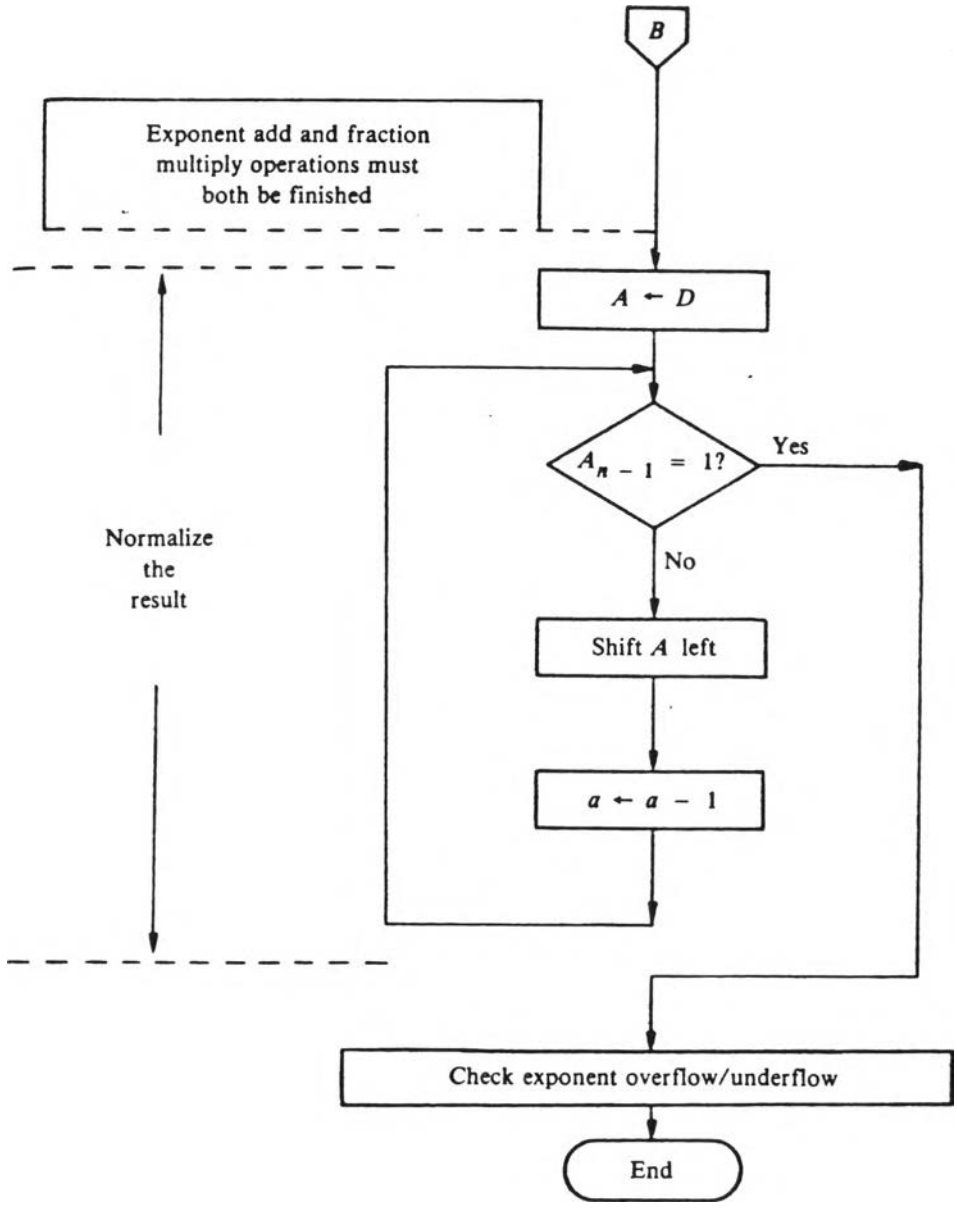
(b) บวกค่า exponent

รูปที่ 5.26 โฟลว์ชาร์ทการคูณเลขทศนิยม (ต่อ)



(c) คูณค่า mantissa

รูปที่ 5.27 โฟลว์ชาร์ทการคูณเลขทศนิยม (ต่อ)



(d) normalize ค่าผลลัพธ์

รูปที่ 5.28 โฟลว์ชาร์ทการคูณเลขทศนิยม (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

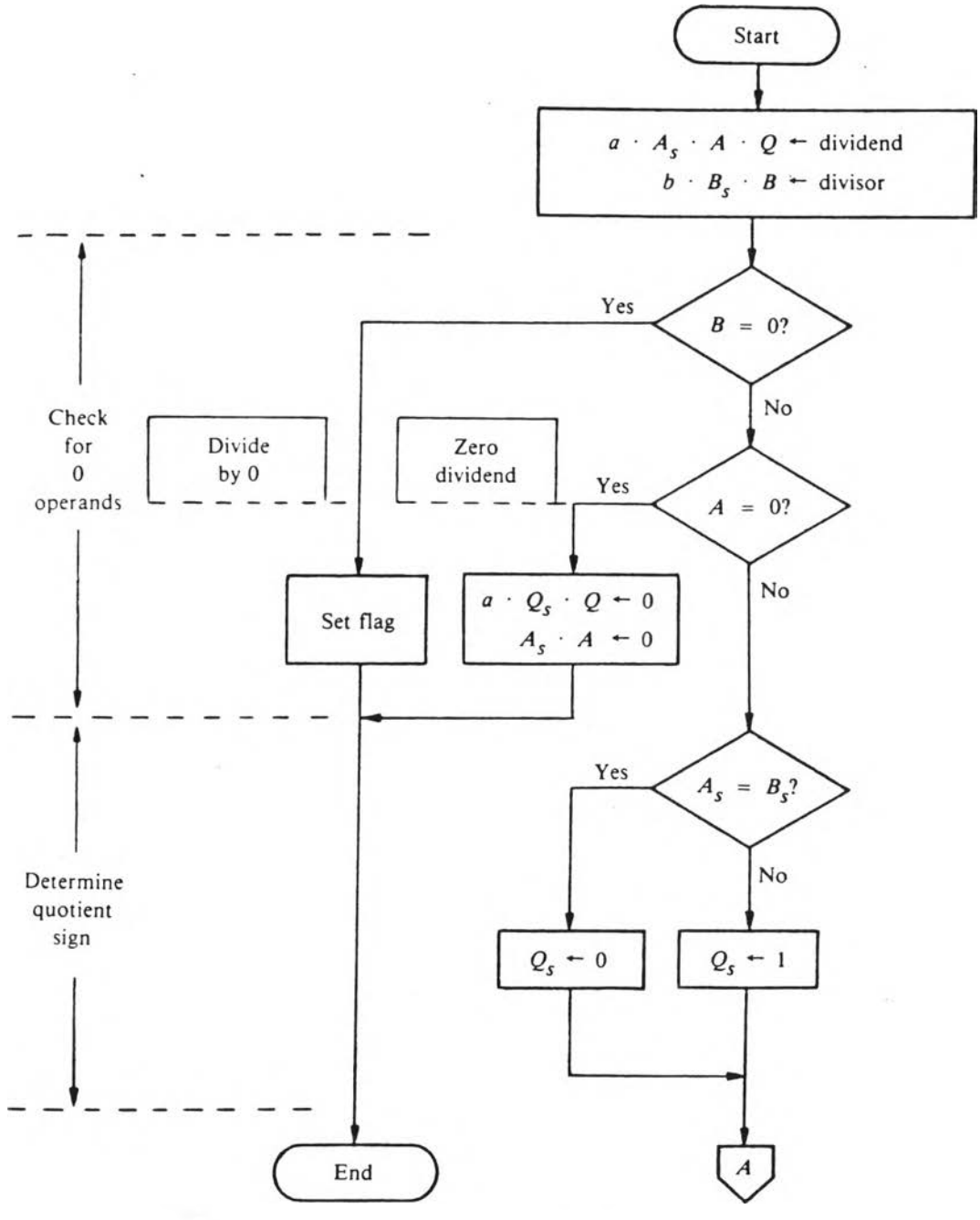
(3) หาร

mnemonic: DIV

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

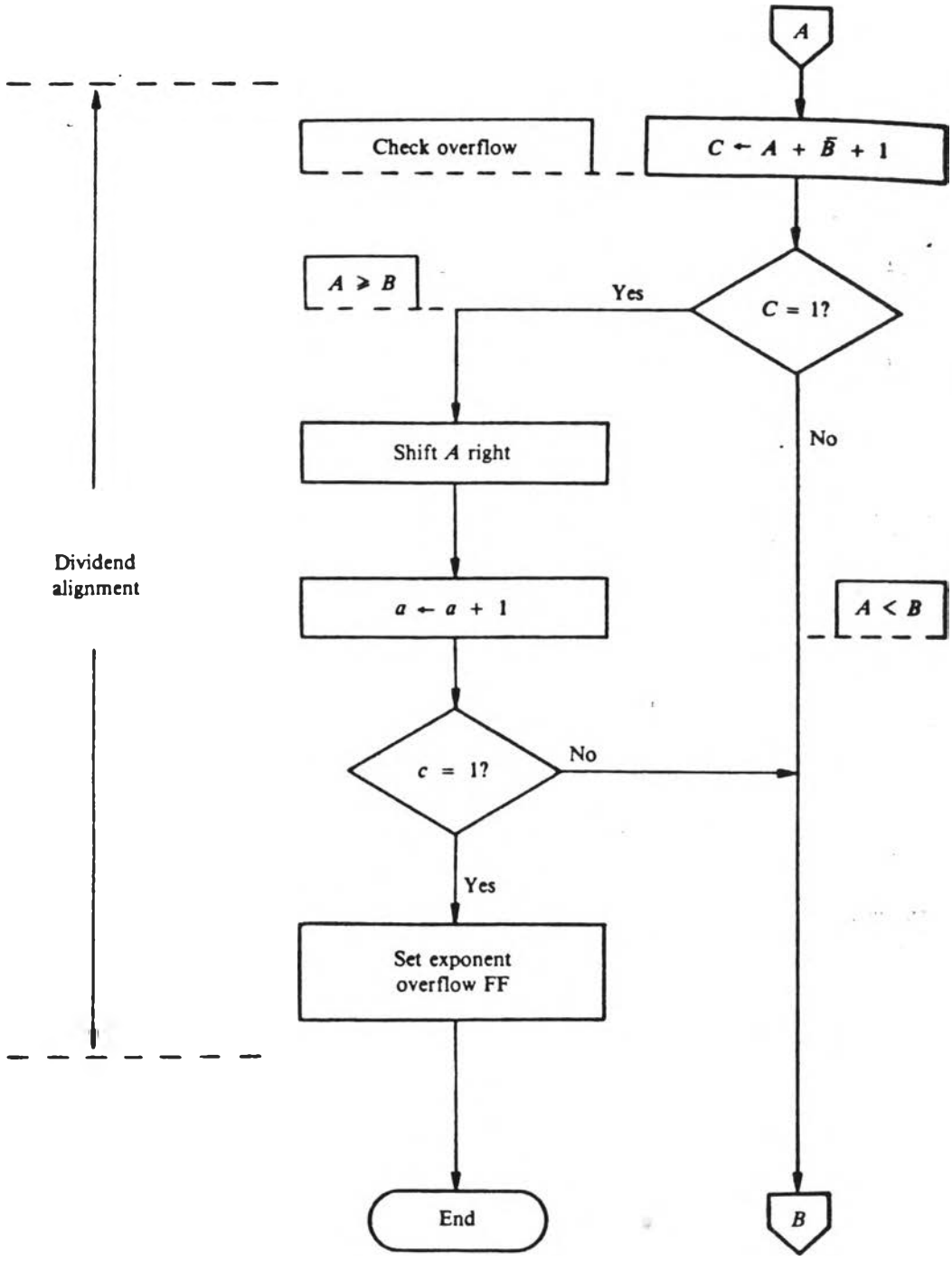
(ก) ตรวจสอบค่า Operand เป็น 0
(ข) หาเครื่องหมายของผลลัพธ์ที่ได้จากการหาร
(ค) จัดตัวตั้งสำหรับการหาร (Align the Dividend)
(ง) ลบค่า exponent
(จ) หารค่า mantissa

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



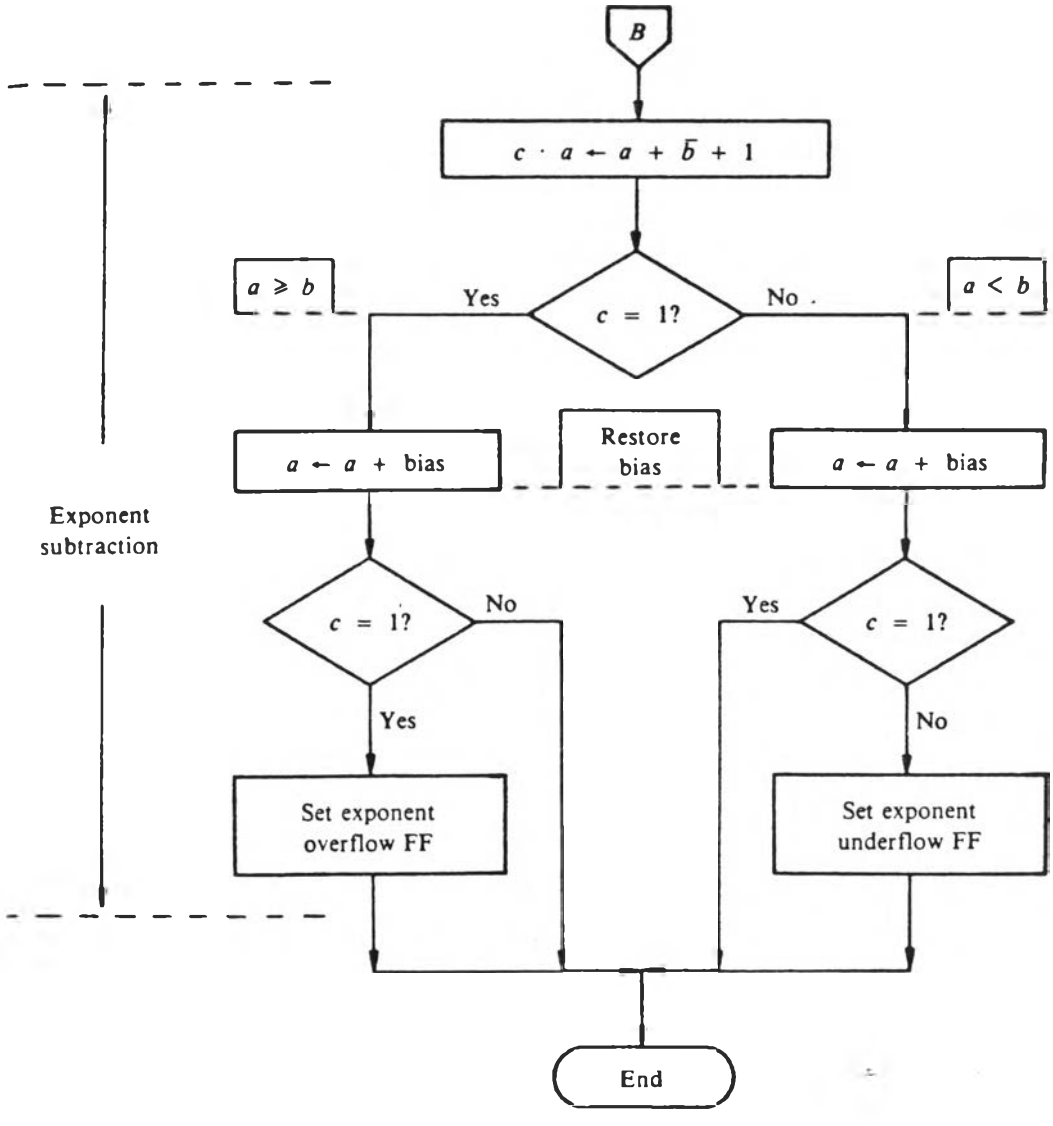
(a) ตรวจสอบค่า 0 และหาเครื่องหมายผลลัพธ์

รูปที่ 5.29 โฟลว์ชาร์ทการหารเลขทศนิยม



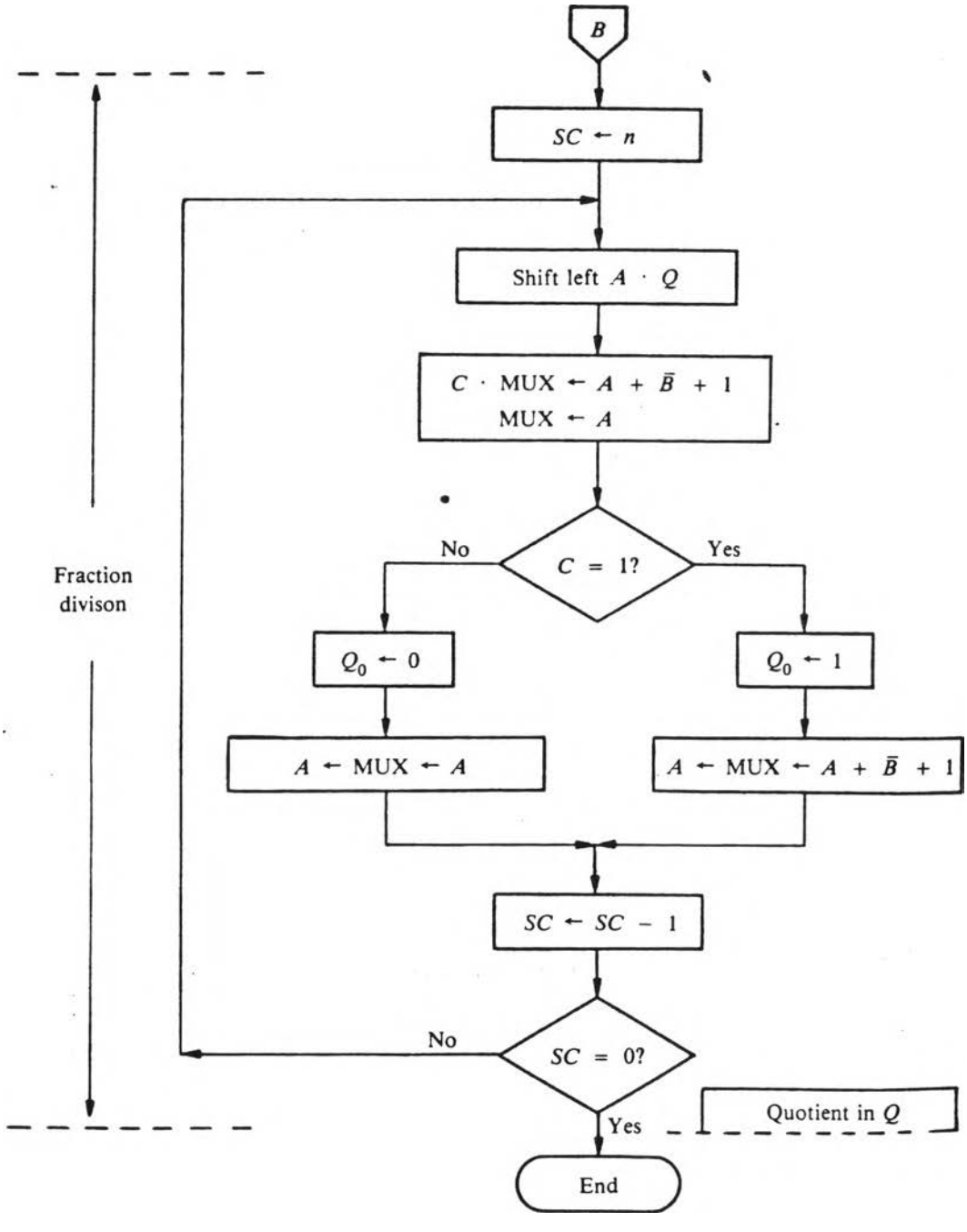
(b) align ค่าตัวตั้ง

รูปที่ 5.30 โฟลว์ชาร์ทการหารเลขทศนิยม (ต่อ)



(c) ลบค่า exponent

รูปที่ 5.31 โฟลว์ชาร์ทการหารเลขทศนิยม (ต่อ)



(d) หารค่า mantissa

รูปที่ 5.32 โฟลว์ชาร์ทการหารเลขทศนิยม (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

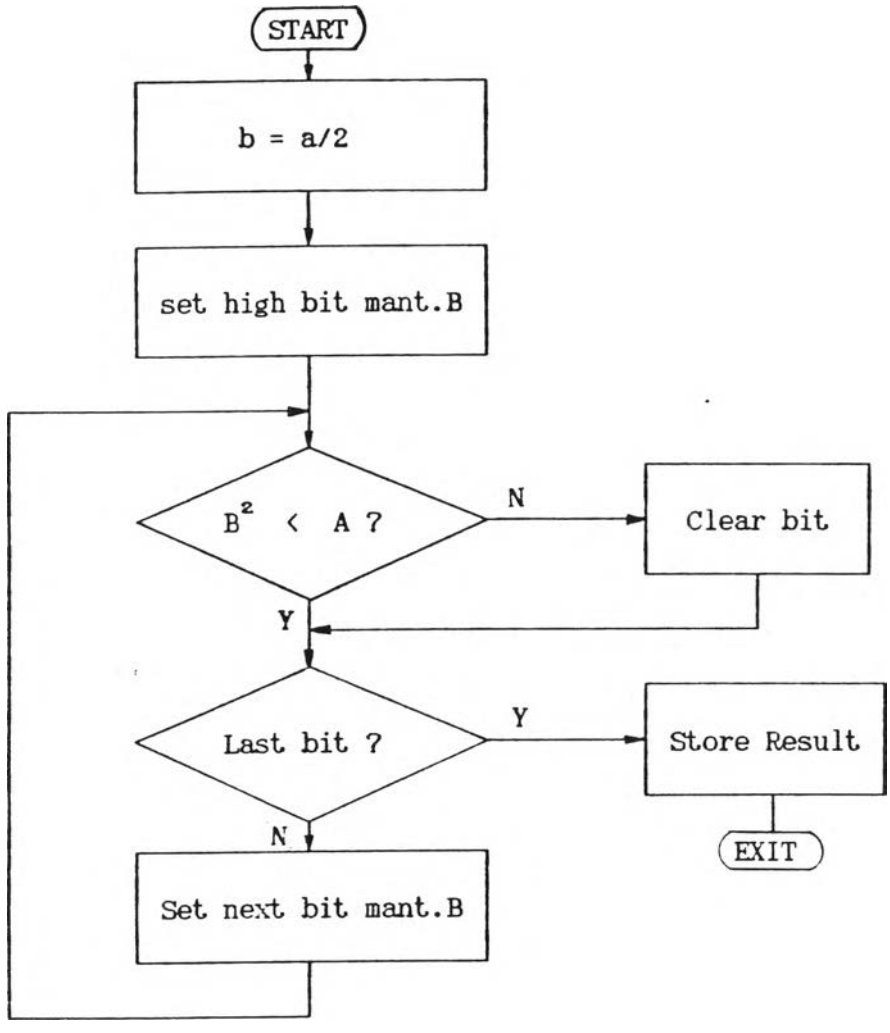
วิธีการโปรแกรม:

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 มีโฟลว์ชาร์ทแสดงการทำงานดังรูปที่ 5.33



รูปที่ 5.33 โฟลว์ชาร์ทของฟังก์ชันถอดรากกำลังสอง (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

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

$$t_2 = \text{lead time (sec)}$$

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

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

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

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: TIMn ; 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

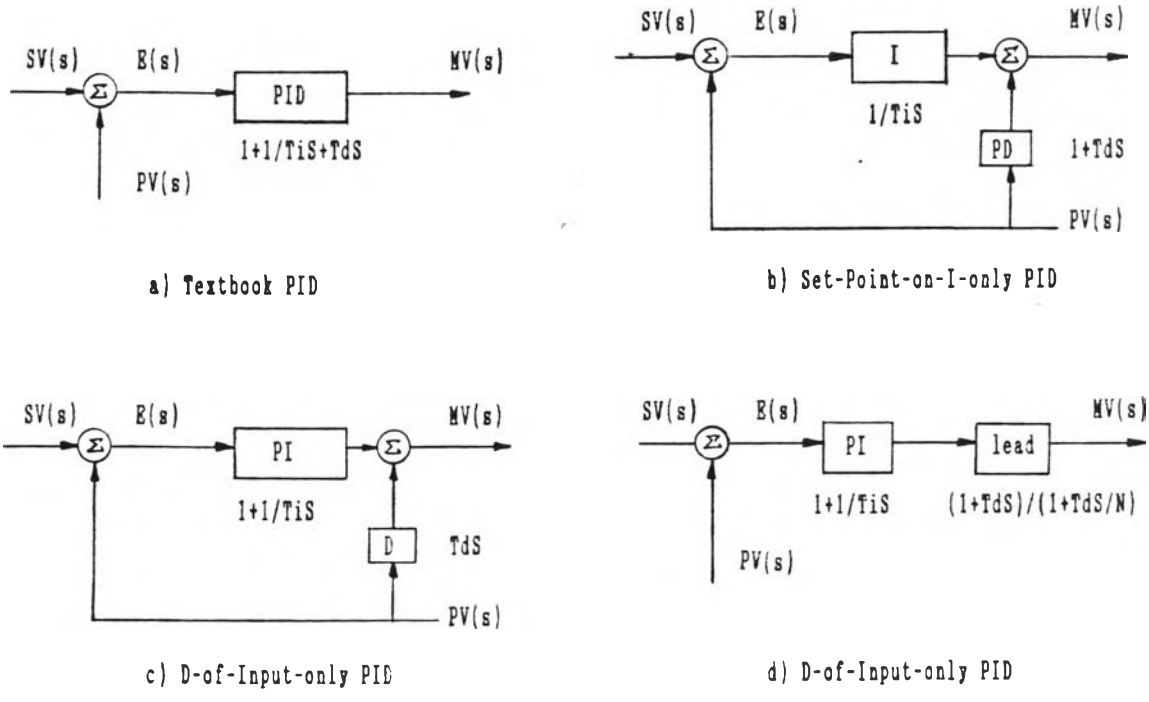
การทำงาน: ฟังก์ชันอ่านค่าอินพุทจาก stack S1 เพื่อคำนวณหาผลลัพธ์โดยวิธี Interporation จากค่าหลัก และค่าความชันของแต่ละ segment ซึ่งได้มาจากโปรแกรมสแกนแป้นพิมพ์ด้านข้าง

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)

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

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 $K_p = 100/PB$

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

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

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

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

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

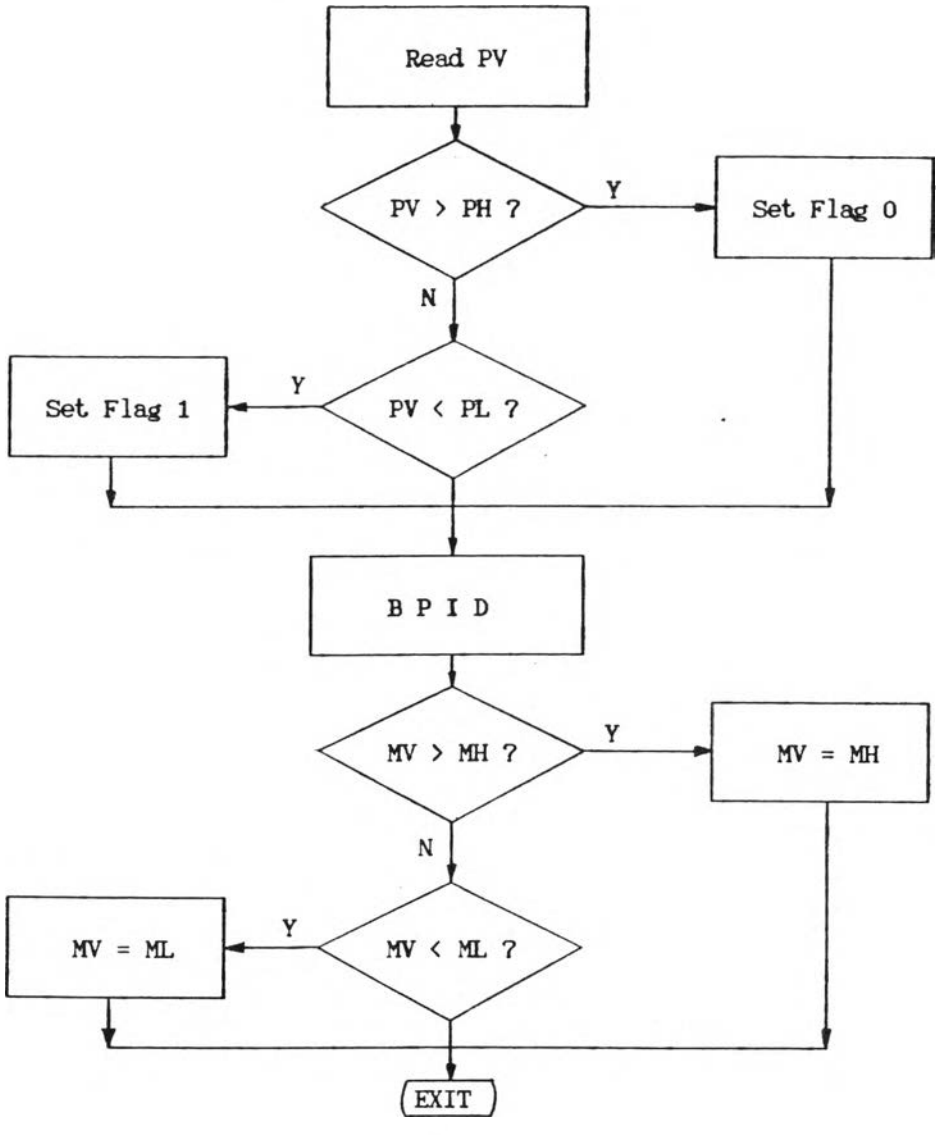
ให้ $T_f = T_D/8$

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

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

สมการที่ (5.11-5.13) จะถูกนำมาใช้เพื่อคำนวณหาค่าสัญญาณควบคุมของการควบคุมแบบ PID ซึ่งโฟว์ชาร์ทของฟังก์ชัน BPID แสดงได้ดังรูปที่ 5.35 โดยก่อนการคำนวณโปรแกรมจะตรวจสอบค่าสัญญาณโปรเซส ซึ่งถ้ามากว่าค่า 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 ถูกกำหนดค่าโดยใช้แป้นนิพม์ด้านข้าง

การทำงาน: ฟังก์ชันอ่านค่า Process variable จาก stack S1 และให้ค่าผลลัพธ์บน S1



รูปที่ 5.35 โฟลว์ชาร์ทของฟังก์ชัน BPID

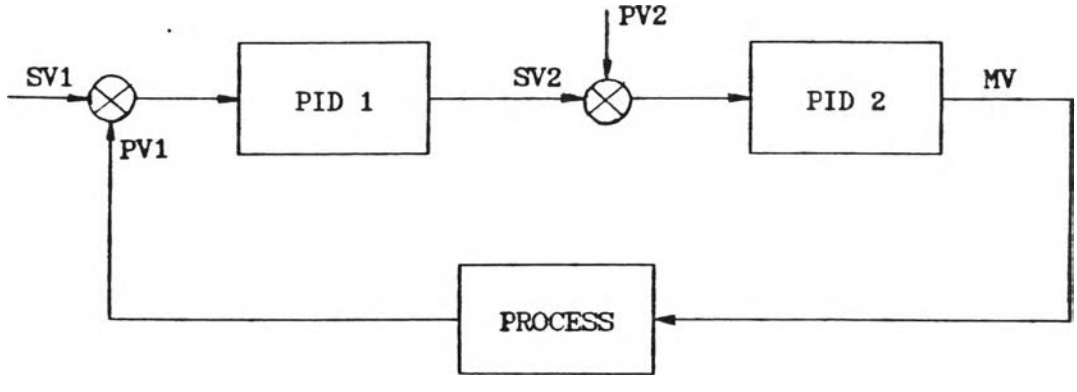
วิธีการโปรแกรม:

คำสั่งที่	ฟังก์ชัน	S1	S2	S3	คำอธิบาย
1	LD X1	X1			อ่าน PV
2	BPID	MV			คำนวณสมการ BPID
3	ST Y1	MV			ส่งค่า MV ช่องที่ 1
4	LD FLO	0/1			อ่านค่า limit alarm
5	ST D1	0/1			alarm output

(2) Cascade PID

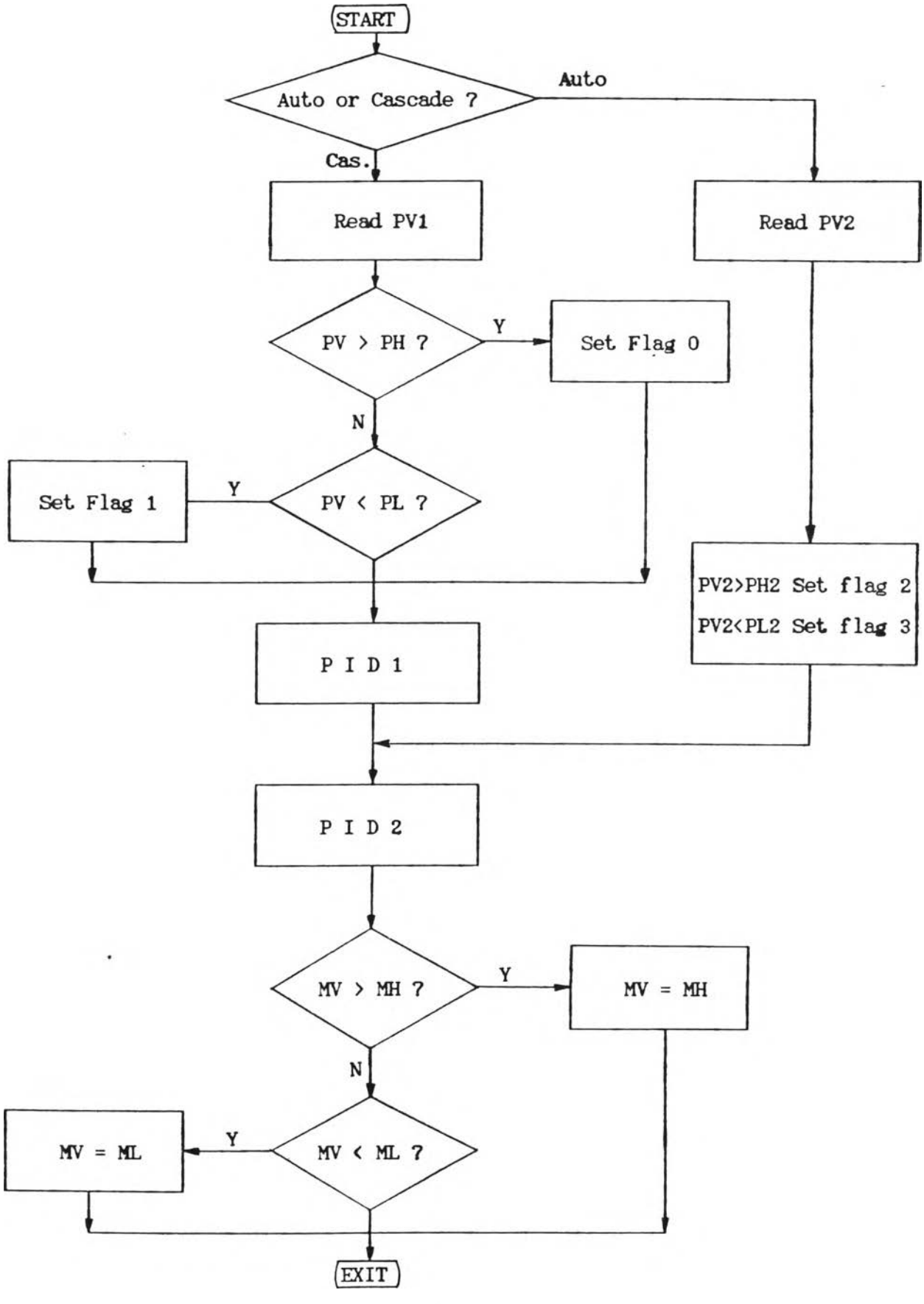
mnemonic: CPID

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

โฟลว์ชาร์ทของโปรแกรม 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 โฟลว์ชาร์ทของฟังก์ชัน CPID

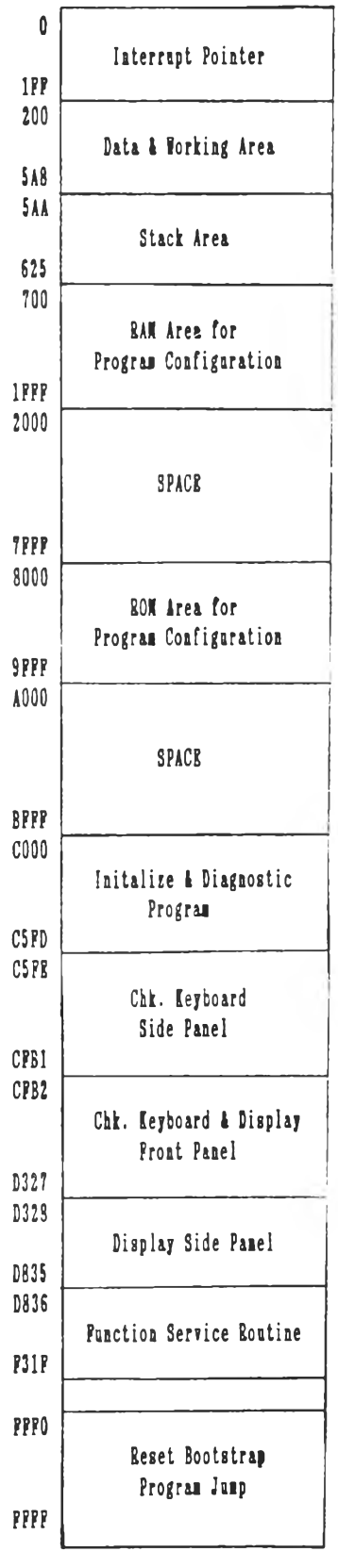
การทำงาน: ฟังก์ชันอ่านค่า PV1 จาก S1 และค่า PV2 จาก S2 โดยหลังจากการคำนวณจะให้ผลลัพธ์บน S1

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