หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 4

การสร้างโปรแกรมย่อยทางกราฟฟิกสำหรับงานวิควกรรม

4.1 แนวความคิดในการสร้างโปรแกรมย่อยทางกราฟฟิกสำหรับงานวิควกรรม[2,3]

โปรแกรมย่อยทางกราฟฟิกสำหรับงานวิควกรรมจะถูกแบ่งออกตามหน้าที่การทำงานใหญ่ๆ 3 ประการ (โปรดดูรูปที่ 4-1 ประกอบ) คือ

ประการแรก โปรแกรมย่อยที่มีหน้าที่เกี่ยวกับการจัดการอุปกรณ์ที่ใช้แสดงผลลัพธ์ เช่น การส่งคำสั่งพลอตไปยังจอภาพ การเปิดปิดแฟ้มข้อมูล การเปลี่ยนรูปแบบของข้อมูลสำหรับส่งไปพิมพ์ยังเครื่องพิมพ์แล้วบันทึกข้อมูลลงในแฟ้มข้อมูล หรือควบคุมการทำงานของโปรแกรมย่อยอื่นๆ โดยใช้ตัวแปรร่วมควบคุม (โปรดดูความหมายตัวแปรร่วมต่างๆ) โปรแกรมย่อยที่ทำหน้าที่ประการแรกจะมีดังนี้

- โปรแกรมย่อย INITPARA
- โปรแกรมย่อย INITSYST
- โปรแกรมย่อย ENDPLT
- โปรแกรมย่อย CLRBUF
- โปรแกรมย่อย GENFPLOT
- โปรแกรมย่อย BDRWLINE

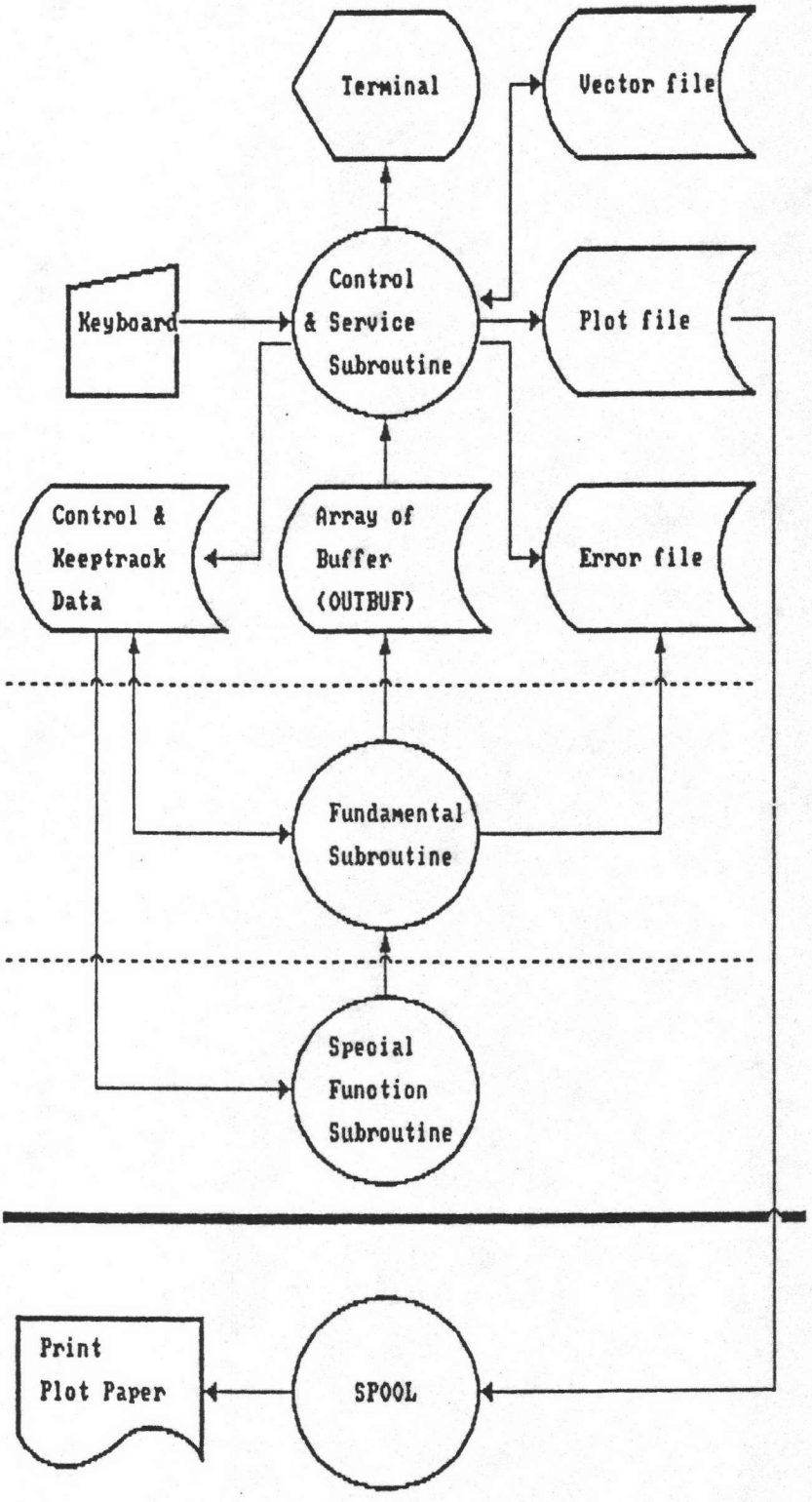
ประการที่สอง โปรแกรมย่อยทางกราฟฟิกที่มีหน้าที่การทำงานพื้นฐาน ซึ่งจำเป็นต้องมีการเรียกใช้ หรือมีการเรียกใช้บ่อยครั้ง โปรแกรมย่อยที่ทำหน้าที่ประการที่สองจะมีดังนี้

- โปรแกรมย่อย PLOTS
- โปรแกรมย่อย NEWPLT
- โปรแกรมย่อย PLOT
- โปรแกรมย่อย FACTOR

- โปรแกรมย่อย WHERE
- โปรแกรมย่อย ROTATE
- โปรแกรมย่อย MIRROR
- โปรแกรมย่อย SYMBOL
- โปรแกรมย่อย CSLANT
- โปรแกรมย่อย USRSYM
- โปรแกรมย่อย NUMBER
- โปรแกรมย่อย SCALE
- โปรแกรมย่อย AXIS
- โปรแกรมย่อย LINE
- โปรแกรมย่อย CIRCLE
- โปรแกรมย่อย OPMES

ประการที่สาม โปรแกรมย่อยทางกราฟฟิกที่มีหน้าที่พิเศษเฉพาะอย่าง เช่น โปรแกรมย่อยสำหรับสร้างรูปกราฟต่างๆ โปรแกรมย่อยสำหรับสร้างลูกศร ฯลฯ โปรแกรมย่อยที่ทำหน้าที่ประการที่สามจะมีดังนี้

- โปรแกรมย่อย RPLOT
- โปรแกรมย่อย RELPLT
- โปรแกรมย่อย DASHLN
- โปรแกรมย่อย AROHD
- โปรแกรมย่อย RCTNGL
- โปรแกรมย่อย RCTBLK
- โปรแกรมย่อย BAR
- โปรแกรมย่อย BAR2
- โปรแกรมย่อย PIE



รูปที่ 4-1 แสดงโครงสร้างโปรแกรมย่อยทางกราฟฟิกสำหรับงานวิศวกรรม

ความหมายของตัวแปรร่วมต่าง ๆ

(คำว่า เวกเตอร์ หมายถึง คำสั่งในการพลอตซึ่งอาจเป็นการลากเส้น หรือเพียงแต่ย้ายตำแหน่งการพลอต อย่างใดอย่างหนึ่ง)

(คำว่า บัฟเฟอร์ หมายถึง ที่ว่างซึ่งใช้เก็บเวกเตอร์การพลอตชั่วคราว)

(คำว่า พารามิเตอร์ หมายถึง ข้อมูลที่ส่งให้โปรแกรมย่อยเมื่อมีการเรียกใช้)

ตัวแปร CURX และ CURY

ใช้เก็บตำแหน่งล่าสุดที่มีการพลอตของผู้ใช้ ใช้หน่วยอ้างอิงตามหน่วยความยาวของผู้ใช้

ตัวแปร ERRDEV

ใช้เก็บเลขอ้างอิงแฟ้มข้อมูลเก็บข้อผิดพลาดการพลอต

ตัวแปร FILEPLOT

ใช้บอกว่าการแสดงผลลัพธ์ทางกราฟฟิกจะเกิดขึ้นบนอุปกรณ์อะไรระหว่าง จอภาพ กับ เครื่องพิมพ์

ตัวแปร HEADXHI และ HEADYHI

ใช้เก็บค่าที่มากที่สุดที่มีการพลอต สำหรับบันทึกลงที่หัวแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร HEADLENG

ใช้เก็บจำนวนเวกเตอร์การพลอตที่มีอยู่ในแฟ้มข้อมูล สำหรับบันทึกลงที่หัวแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร HEADRPIX และ HEADRPIY

ใช้เก็บจำนวนจุดต่อหนึ่งหน่วยความยาวที่ใช้ สำหรับบันทึกลงที่หัวแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร HEADXMAX และ HEADYMAX

ใช้เก็บค่าอ้างอิงตำแหน่งของแกนแนวราบและแนวดิ่งที่มากที่สุดที่การพลอตสามารถจะอ้างอิงได้ สำหรับบันทึกลงที่หัวแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร HEADSIZE

ใช้กำหนดขนาดของหัวแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร IXCURS และ IYCURS

ใช้เก็บตำแหน่งล่าสุดที่มีการพลอตบนตัวแปรร่วม PAPER

ตัวแปร LASTX และ LASTY

ใช้เก็บตำแหน่งล่าสุดที่มีการพลอตของผู้ใช้ ใช้หน่วยอ้างอิงตามหน่วยความยาวของผู้ใช้

ตัวแปร MIRROR

ใช้บอกว่าตำแหน่งแนวราบ หรือแนวดิ่งมีการสะท้อนหรือไม่

ถ้า MIRROR = 0 จะไม่มีการสะท้อนตำแหน่ง

ถ้า MIRROR = 1 ตำแหน่งแนวราบสะท้อน(กลับซ้ายเป็นขวา)

(Xnew = Xmax - Xinput)

ถ้า MIRROR = 2 ตำแหน่งแนวดิ่งสะท้อน(กลับบนเป็นล่าง)

(Ynew = Ymax - Yinput)

ถ้า MIRROR = 3 ตำแหน่งทั้งสองแนวสะท้อน

(Xnew = Xmax - Xinput)

(Ynew = Ymax - Yinput)

ตัวแปร OUTBUF

ใช้เก็บเวกเตอร์การพลอตชั่วคราว

ตัวแปร OUTBUFPT

ใช้เป็นดัชนีชี้ตำแหน่งที่เก็บเวกเตอร์ในบัฟเฟอร์

ตัวแปร OUTBUFSZ

ใช้กำหนดขนาดของบัฟเฟอร์

ตัวแปร OUTDEV

ใช้เก็บเลขอ้างอิงแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

ตัวแปร PAPER

ใช้แทนกระดาษสำหรับพลอตถูกใช้โดยโปรแกรมย่อย GENFPLOT และ BDRWLINE โดยการอ่านข้อมูลเวกเตอร์จากแฟ้มข้อมูลมาพลอตลงที่ตัวแปรนี้ก่อน แล้วหลังจากพลอตเสร็จก็จะทำการอ่านตัวแปรร่วมนี้บันทึกลงแฟ้มข้อมูล(Plot file)ที่จะนำไปพลอตต่อไป

ตัวแปร RPIX และ RPIY

ใช้เก็บจำนวนจุดต่อหนึ่งหน่วยความยาวที่ใช้

ตัวแปร SFACTR

ใช้เก็บอัตราย่อขยายขนาดภาพ

ตัวแปร SLANT

ใช้เก็บมุมเอียงของตัวอักษรอ้างอิงกับแกนแนวดิ่ง

ตัวแปร SYM

ใช้เก็บเวกเตอร์ของตัวอักษร (โปรดดูรูปที่ 4-2 ประกอบ)

ตัวแปร TCOS และ TSIN

ใช้เก็บค่า COS และ SIN ของมุมเอียงของภาพอ้างอิงกับแกนแนวราบ

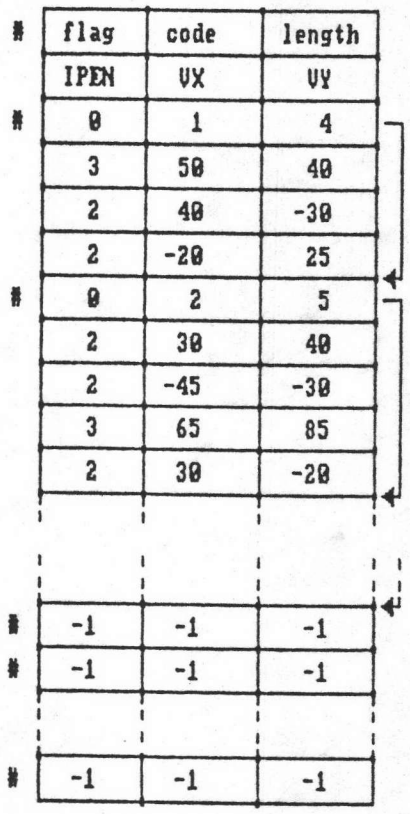
ตัวแปร VERTPLOT

ใช้บอกว่าต้องการพลอตภาพ โดยที่แกนแนวราบของภาพขนานกับแกนแนวราบของจอภาพ หรือตั้งฉากกับแกนแนวราบของจอภาพ

ตัวแปร XORG และ YORG

ใช้เก็บตำแหน่งที่แทนจุดกำเนิดในการพลอตภาพของผู้ใช้

(อ้างอิงตามตำแหน่งจริงบนจอภาพ)



#	flag	code	length
	IPEN	VX	VY
#	0	1	4
	3	50	40
	2	40	-30
	2	-20	25
#	0	2	5
	2	30	40
	2	-45	-30
	3	65	85
	2	30	-20
#	-1	-1	-1
#	-1	-1	-1
#	-1	-1	-1

flag = 0 :Character exist
flag = -1 :Empty
flag = 9999 :Mark delete

code :Character code
length :length of Character vectors

IPEN :Pen instruction
VX :Vector-X
VY :Vector-Y

รูปที่ 4-2 แสดงโครงสร้างข้อมูลของตัวแปรร่วม SYM

4.2 การสร้างโปรแกรมย่อยที่มีหน้าที่เกี่ยวกับการจัดการอุปกรณ์ที่ใช้แสดงผลลัพธ์หรือควบคุมการทำงานของโปรแกรมย่อยอื่นๆ

4.2.1 การสร้างโปรแกรมย่อย INITPARA

จุดประสงค์ ใช้เริ่มต้นการเรียกใช้โปรแกรมย่อยทางกราฟฟิกสำหรับงานวิศวกรรม การทำงานของโปรแกรมย่อยนี้(โปรดดูรูปที่ 4-3 ประกอบ) เริ่มดังนี้คือ

- ถามความต้องการว่าต้องการแสดงผลลัพธ์ทางไหน ระหว่าง จอภาพ กับ แฟ้มข้อมูล โดยจะไปกำหนดค่าเริ่มต้นตัวแปรร่วม FILEPLOT ถ้าจริง จะแสดงผลลัพธ์ทางแฟ้มข้อมูล ถ้าเท็จ จะแสดงผลลัพธ์ทางจอภาพ และ

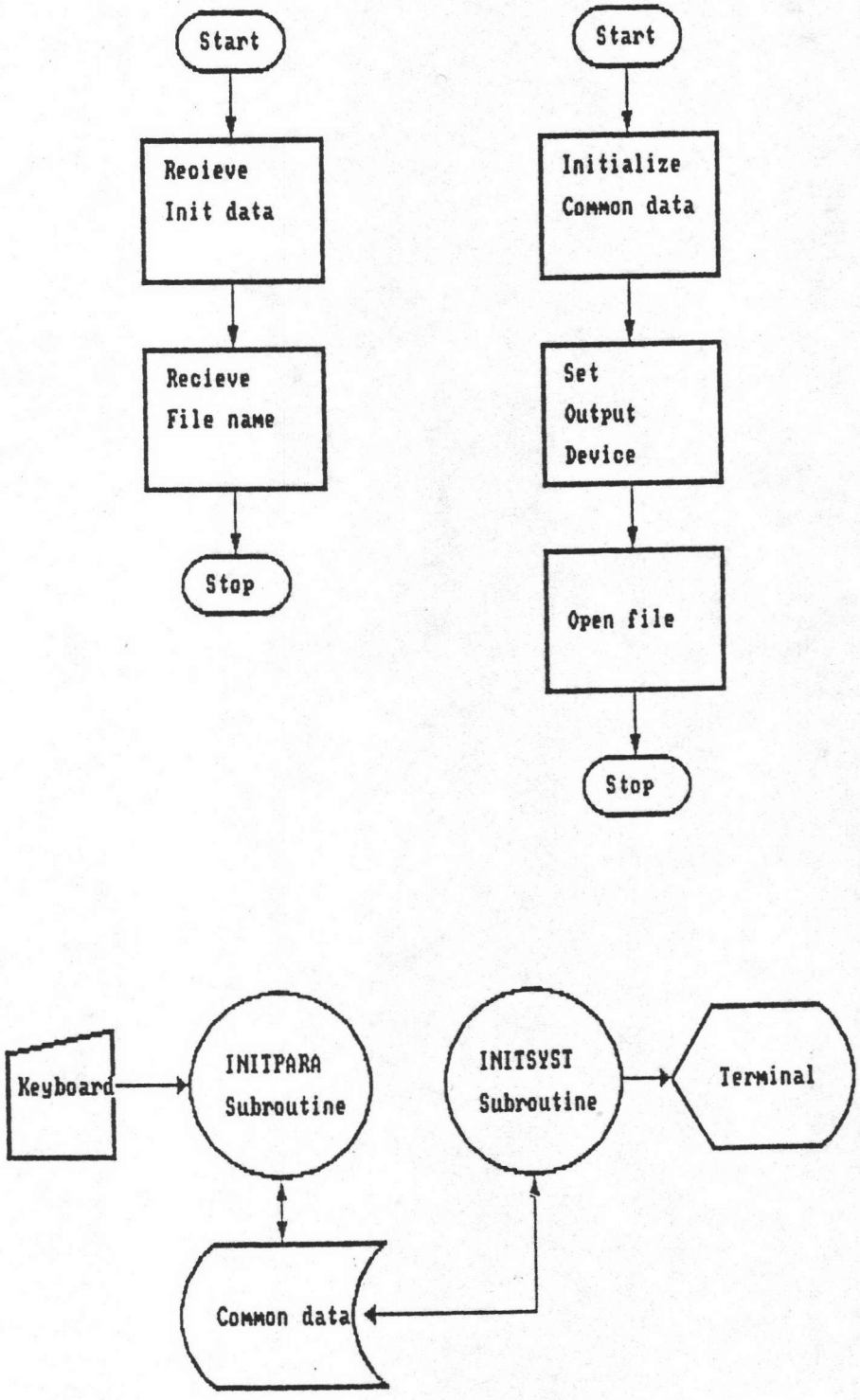
  ในกรณีนี้จะถามต่อว่าต้องการให้มีการพลอตทุกคำสั่ง PLOT ทันทีหรือไม่ โดยจะไปกำหนดตัวแปรร่วม INTERACT ถ้าจริงแสดงว่าต้องการ และ

  ในกรณีที่มีการแสดงผลลัพธ์ทางแฟ้มข้อมูลตัวแปรร่วม INTERACT จะถูกกำหนดให้เป็นเท็จ และจะมีการถามชื่อแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต
- ถามจำนวนจุดต่อหนึ่งหน่วยความยาวที่ผู้ใช้ต้องการ (ในที่นี้ ใช้ตัวย่อว่า RPI : Raster Per Inch โดย Inch ใช้แทนคำว่าหน่วยความยาว) โดยมีให้ 9 อัตรา ตั้งแต่ 5 จุด ถึง 800 จุดต่อหนึ่งหน่วยความยาว โดยจะไปกำหนดในตัวแปรร่วม RPIX และ RPIY ซึ่งมีค่าเท่ากันเสมอ
- ถามชื่อแฟ้มข้อมูลที่เก็บข้อผิดพลาดการพลอต (พลอตเกินขอบเขตที่กำหนด)
- ถามว่าต้องการให้แกนแนวราบของภาพขนานกับแนวราบของจอภาพหรือตั้งฉากกับแนวราบของจอภาพ โดยจะไปกำหนดตัวแปรร่วม VERTPLOT ถ้าจริง แสดงว่าต้องการให้แนวราบของภาพตั้งฉากกับแนวราบของจอภาพ ถ้าเท็จ แสดงว่าต้องการให้แนวราบของภาพขนานกับแนวราบของจอภาพ
- ทำการเรียกโปรแกรมย่อย INITSYST ต่อไป

4.2.2 การสร้างโปรแกรมย่อย INITSYST

จุดประสงค์ ใช้กำหนดค่าเริ่มต้นของตัวแปรร่วม ทำการเปิดแฟ้มข้อมูลและจัดการเกี่ยวกับจอภาพ (ถ้ามีความต้องการแสดงผลลัพธ์ทางจอภาพโดยตรวจสอบจากตัวแปรร่วม FILEPLOT) การทำงานของโปรแกรมย่อยนี้(โปรดดูรูปที่ 4-3 ประกอบ) เริ่มดังนี้คือ

- กำหนดตำแหน่งของจุดกำเนิด (XORG,YORG) ไว้ที่ตำแหน่ง (0,0)
- กำหนดตำแหน่งล่าสุดที่มีการพลอต(CURX,CURY) ไว้ที่ตำแหน่ง (0,0)
(ตำแหน่งอ้างอิงตามตำแหน่งจริงของจอภาพ)
- ตำแหน่งล่าสุดที่มีการพลอต(LASTX, LASTY) ไว้ที่ตำแหน่ง (0.,0.)
(ตำแหน่งอ้างอิงตามตำแหน่งที่ผู้ใช้ใช้)
- กำหนดอัตราย่อขยายขนาดภาพ (SFACTR) ให้มีค่าปกติ (เท่ากับ 1)
- กำหนดดัชนีของบัฟเฟอร์ที่ใช้เก็บคำสั่งพลอตไว้ตอนต้นของบัฟเฟอร์
(ตัวแปรร่วม OUTBUFPT = 1)
- กำหนดค่า COS และ SIN ของมุมเอียงภาพไว้ที่มุมศูนย์องศา อ้างอิงกับแกนแนวราบของจอภาพ (ตัวแปรร่วม TCOS และ TSIN)
- กำหนดขอบเขตของตำแหน่งของแกนแนวราบ และของแกนแนวดิ่งที่ใช้อ้างอิงได้มากที่สุด (ตัวแปรร่วม HEADXMAX และ HEADYMAX)
- ทำการเปิดแฟ้มข้อมูลที่เก็บข้อผิดพลาดการพลอต
(พลอตเกินขอบเขตที่กำหนด)
- ถ้าเป็นการแสดงผลลัพธ์ทางจอภาพต้องทำการเปลี่ยนโหมดของจอภาพจากโหมดตัวอักษรเป็นโหมดกราฟฟิก
- แต่ถ้าต้องการแสดงผลลัพธ์ทางแฟ้มข้อมูลก็จะทำการเปิดแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต
- จบการทำงานของโปรแกรมย่อยนี้



รูปที่ 4-3 แสดงการทำงานของโปรแกรมย่อย INITPARA และ INITSYST และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย INITPARA และ INITSYST กับส่วนอื่น

4.2.3 การสร้างโปรแกรมย่อย ENDPLT

จุดประสงค์ ใช้สำหรับปิดแฟ้มข้อมูลต่าง ๆ หรือจัดการกับจอภาพเมื่อมีความต้องการเลิกการเรียกใช้โปรแกรมย่อยทางกราฟฟิกสำหรับงานวิศวกรรม การทำงานของโปรแกรมย่อยนี้(โปรดดูรูปที่ 4-4 ประกอบ) เริ่มดังนี้คือ

- นำคำสั่งพลอตที่ค้างอยู่ในบัฟเฟอร์ออกมาพลอตก่อนให้หมด
- แล้วทำการตรวจสอบดูว่าอุปกรณ์ที่ใช้แสดงผลลัพธ์คืออะไร

ถ้าคือจอภาพ ต้องทำการเปลี่ยนโหมดจากโหมดกราฟฟิกมาเป็นโหมดตัวอักษร

ถ้าคือแฟ้มข้อมูล ต้องทำการปิดแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอต

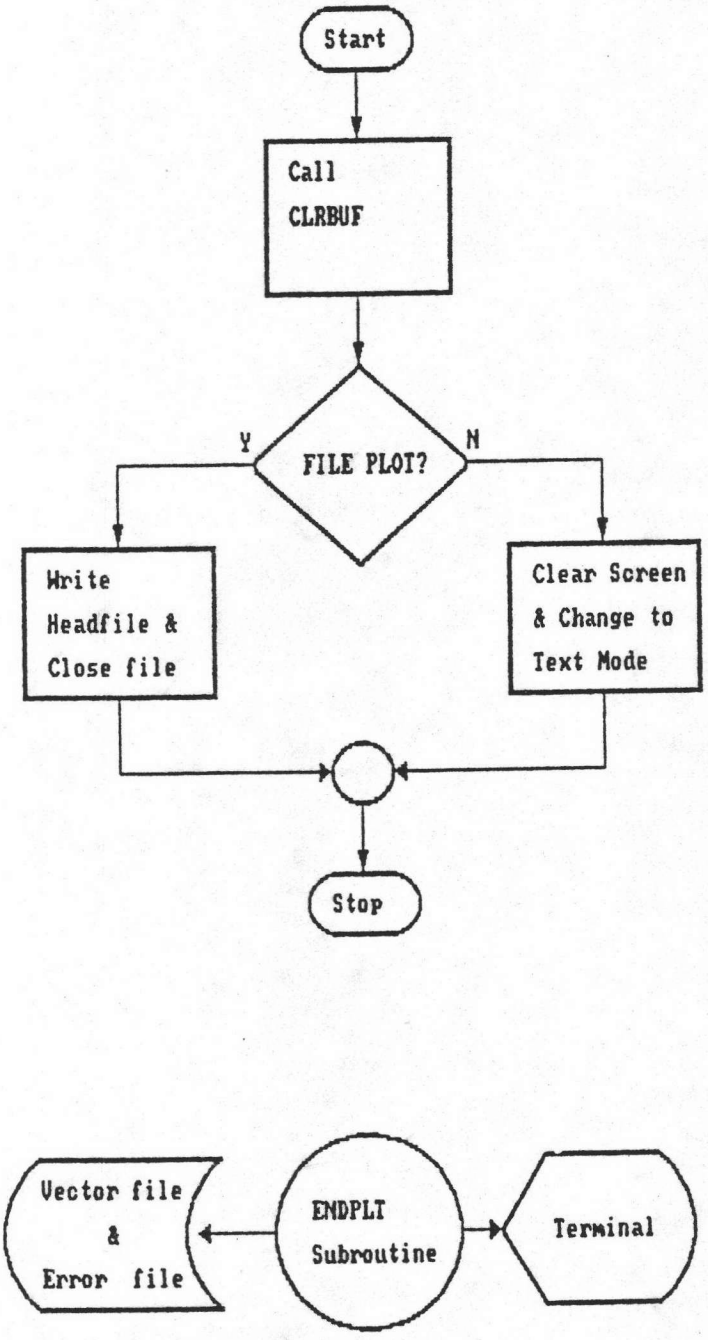
- หลังจากนี้ก็จะทำการปิดแฟ้มข้อมูลที่เก็บข้อผิดพลาดการพลอต (พลอตเกินขอบเขตที่กำหนด)
- จบการทำงานของโปรแกรมย่อยนี้

4.2.4 การสร้างโปรแกรมย่อย CLRBUF

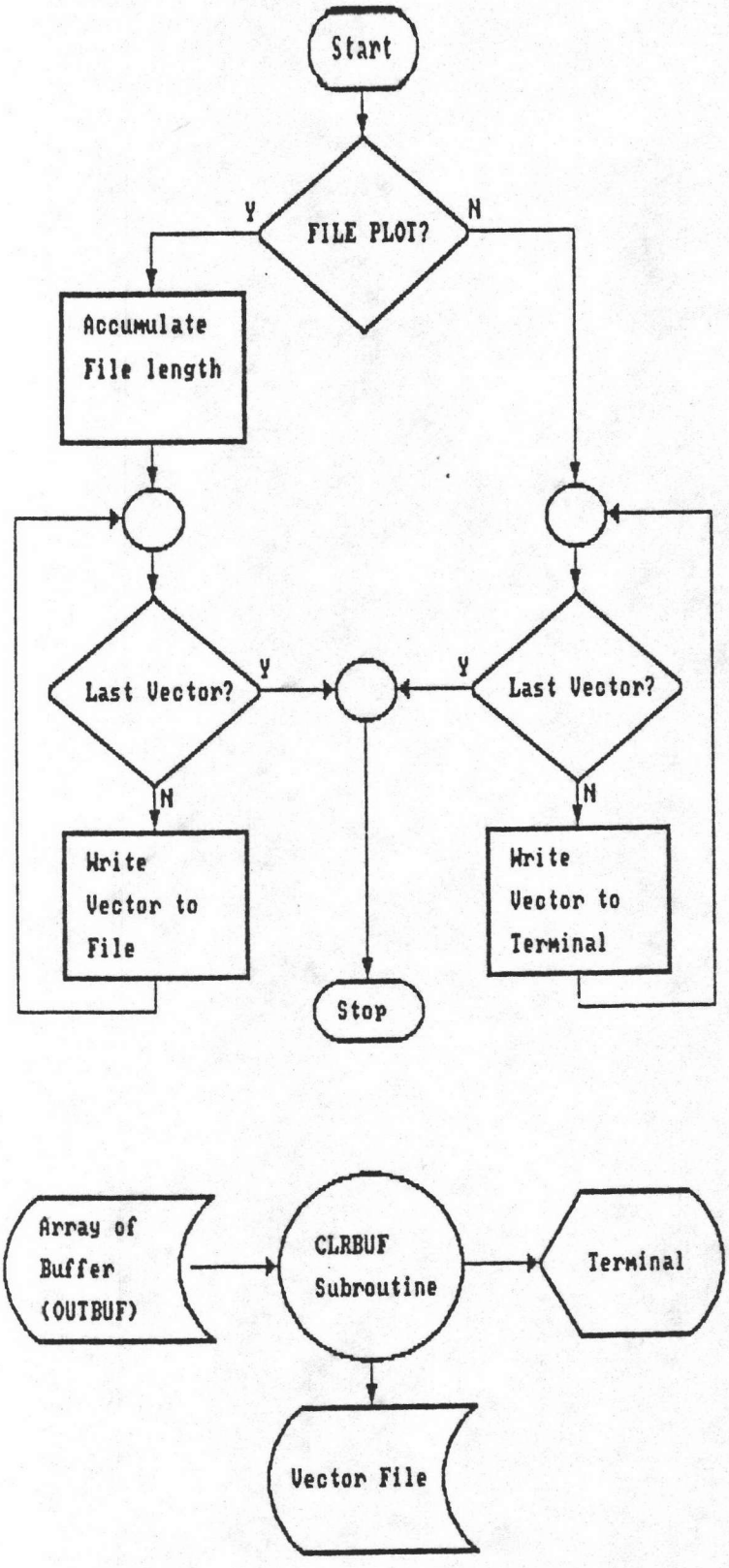
จุดประสงค์ใช้นำคำสั่งพลอตที่เก็บในบัฟเฟอร์ (ตัวแปรร่วม OUTBUF) ออกมาแสดงผลลัพธ์ตามที่กำหนดทำให้โปรแกรมย่อย PLOT มีความเป็นอิสระจากอุปกรณ์แสดงผลลัพธ์ การทำงานของโปรแกรมย่อยนี้(โปรดดูรูปที่ 4-5 ประกอบ) เริ่มดังนี้คือ

- ตรวจสอบว่ามีคำสั่งพลอตในบัฟเฟอร์หรือไม่ ถ้าไม่ก็จะจบการทำงาน แต่ถ้ามีก็จะทำงานต่อไป
- ตรวจสอบดูว่าต้องการแสดงผลลัพธ์กราฟฟิกทางไหน(ดูจากตัวแปรร่วม FILEPLOT)

ถ้าต้องการแสดงผลลัพธ์ทางแฟ้มข้อมูล ก็จะทำการอ่านข้อมูลจากบัฟเฟอร์มาใส่แฟ้มข้อมูลจนหมด และเก็บจำนวนของเวกเตอร์การพลอตที่มีไว้ด้วย



รูปที่ 4-4 แสดงการทำงานของโปรแกรมย่อย ENDPLT และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย ENDPLT กับส่วนอื่น



รูปที่ 4-5 แสดงการทำงานของโปรแกรมย่อย CLRBUF และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย CLRBUF กับส่วนอื่น

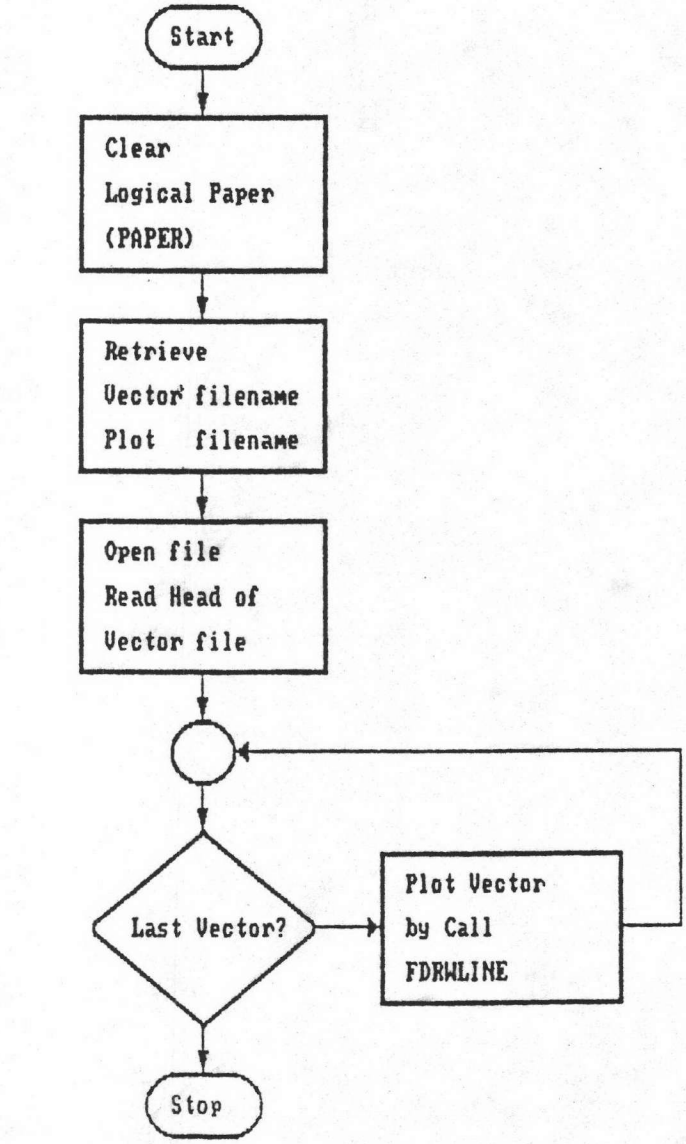
เพื่อบันทึกลงที่หัวแฟ้มข้อมูลสำหรับเก็บเวกเตอร์การพลอตตอนจบการเรียกใช้โปรแกรมย่อยทางกราฟฟิก

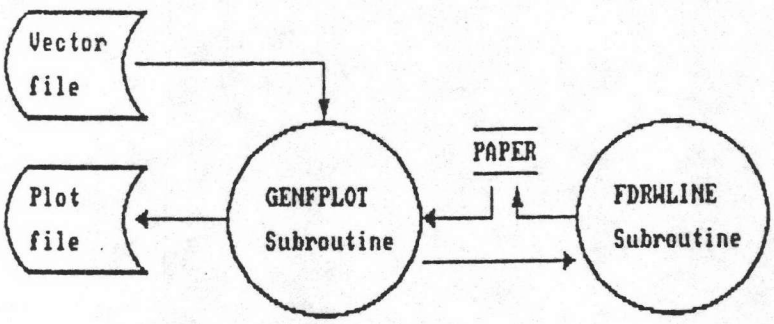
- แต่ถ้าต้องการแสดงผลลัพธ์ทางจอภาพก็จะนำคำสั่งพลอตส่งออกมาแสดงทางจอภาพจนหมด โดยเวกเตอร์ที่เก็บไว้ในบัฟเฟอร์ ตำแหน่งแนวดิ่งต้องมาทำการลบออกหนึ่งก่อน เพราะก่อนเก็บตำแหน่งแนวดิ่งจะมีการบวกเพิ่มไว้หนึ่ง เพื่อให้ตำแหน่งแนวดิ่งเวลามีค่าเท่ากับศูนย์จะกลายเป็นหนึ่งเก็บค่าลบได้ ซึ่งใช้ในกรณีมีการย้ายตำแหน่งอย่างเดียวโดยไม่มีการลากเส้น
- จบการทำงานของโปรแกรมย่อยนี้ เมื่อพลอตหมดแล้ว

4.2.5 การสร้างโปรแกรมย่อย GENFPLOT

จุดประสงค์ ใช้เปลี่ยนรูปแบบโครงสร้างของข้อมูลที่ใช้แทนรูปภาพ คือ จากที่เก็บเป็นเวกเตอร์การพลอตมาเป็นรูปแบบที่เครื่องพิมพ์จะนำมาพิมพ์ให้เป็นภาพที่มองเห็นได้ การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-6 ประกอบ) เริ่มดังนี้คือ

- กำหนดค่าเริ่มต้นให้กับกระดาษจำลอง(ตัวแปรร่วม PAPER) ให้เหมือนกระดาษเปล่า
- ถามชื่อแฟ้มข้อมูลที่เก็บเวกเตอร์การพลอตเพื่อจะได้อ่านข้อมูลเวกเตอร์การพลอตมาพลอตลงบนกระดาษจำลอง
- ถามชื่อแฟ้มข้อมูลสำหรับเก็บข้อมูลการพลอตภาพทุกทุกจุด(Plot file) ซึ่งอ่านมาจากกระดาษจำลองที่ถูกพลอต จากการเรียกใช้โปรแกรมย่อย BDRWLINE เมื่อพลอตจนครบทุกเวกเตอร์แล้ว
- จบการทำงานของโปรแกรมย่อยนี้ เมื่อพลอตจนครบทุกเวกเตอร์แล้ว





รูปที่ 4-6 แสดงการทำงานของโปรแกรมย่อย GENFPLOT และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย GENFPLOT กับส่วนอื่น

4.2.6 การสร้างโปรแกรมย่อย BDRWLINE

จุดประสงค์ ใช้พลอตเส้นตรงลงบนตัวแปรร่วม PAPER ซึ่งจำลองแทนกระดาษในการพลอต พารามิเตอร์ที่ใช้คือ จุดลองจุด (จุดเริ่มต้น และจุดลิ้นสุดของเส้นตรง) ในที่นี้แทนด้วย (Xs,Ys,) และ (Xe,Ye) ตามลำดับ การทำงานของโปรแกรมย่อยนี้เริ่มโดยแยกกรณีออกเป็น 9 กรณี คือ

กรณีที่ 1 พารามิเตอร์ที่รับเข้ามาเป็นจุด เพราะฉะนั้นจะทำการพลอตเพียงจุดจุดเดียว

Xs = Xe และ Ys = Ye

กรณีที่ 2 พารามิเตอร์ที่รับเข้ามาเป็นเส้นตรงขนานกับแกนแนวราบ (แกน-X) คือ

Xs <> Xe และ Ys = Ye

กรณีที่ 3 พารามิเตอร์ที่รับเข้ามาเป็นเส้นตรงขนานกับแกนแนวดิ่ง (แกน-Y) คือ

Xs = Xe และ Ys <> Ye

เพราะฉะนั้นจะทำการพลอตขนานกับแกนแนวดิ่ง โดยเริ่มที่ตำแหน่งแนวดิ่ง(ตำแหน่ง Y) ที่น้อยกว่าเพิ่มค่า Y ทีละหนึ่ง แล้วทำการพลอตโดยตำแหน่งแนวราบคงที่ (ตำแหน่ง X)

กรณีที่ 4 พารามิเตอร์ที่รับเข้ามาเป็นเส้นตรงที่มีความชันเท่ากับเท่ากับ 1 คือ

(Ye - Ys) / (Xe - Xs) = 1

เพราะฉะนั้นจะทำการพลอตโดยเริ่มจากตำแหน่งแนวดิ่งและแนวราบที่น้อยกว่า แล้วเพิ่มตำแหน่งแนวราบและแนวดิ่ง (ตำแหน่ง Y) ทีละหนึ่งแล้วทำการพลอตด้วยทุกครั้ง

กรณีที่ 5 พารามิเตอร์ที่รับเข้ามาเป็นเส้นตรงที่มีความชันเท่ากับ -1 คือ

(Ye - Ys) / (Xe - Xs) = -1

เพราะฉะนั้นจะทำการพลอตโดยเริ่มจากตำแหน่งแนวราบทีละหนึ่ง และลดตำแหน่งแนวดิ่งทีละหนึ่ง พร้อมทำการพลอตไปด้วย จนกระทั่งถึงตำแหน่งแนวราบที่มากกว่า และตำแหน่งแนวดิ่งที่น้อยกว่า

กรณีที่ 6 พารามิเตอร์ที่เข้ามาเป็นเส้นตรงที่มีความชันระหว่าง 0 กับ 1 คือ

0 < (Ys-Ye) / (Xs-Xe) <1 และ

Xs <> Xe และ Ys <> Ye

กรณีที่ 7 พารามิเตอร์ที่เข้ามาเป็นเส้นตรงที่มีความชันระหว่าง 1 กับ inf.
(inf. means infinity)

1 < (Ys-Ye)/(Xs-Xe) < inf. และ

Xs <> Xe และ Ys <> Ye

กรณีที่ 8 พารามิเตอร์ที่เข้ามาเป็นเส้นตรงที่มีความชันระหว่าง -1 กับ 0

-1 < (Ys-Ye)/(Xs-Xe) < 0 และ

Xs <> Xe และ Ys <> Ye

กรณีที่ 9 พารามิเตอร์ที่เข้ามาเป็นเส้นตรงที่มีความชันระหว่าง -1 กับ -inf.

-inf. < (Ys-Ye)/(Xe-Xe) < -1 และ

Xs<> Xe และ Ys <> Ye

ทั้งสี่กรณีหลังนี้จะทำงานคล้ายกันจะกล่าวถึงกรณีที่ 6 เป็นหลักก่อน เพราะฉะนั้นในกรณีนี้จะใช้แนวราบ (แนว-X) เป็นแนวในการเพิ่มตำแหน่งแนวราบทีละหนึ่งหน่วยโดยเริ่มที่ตำแหน่งแนวราบที่น้อยกว่าก่อน (ต้องเปรียบเทียบ ระหว่าง Xs กับ Xe ค่าตัวไหนน้อยกว่าก็เลือกตัวนั้น) ตำแหน่งต่อไปที่จะต้องเลือกมี 2 ตำแหน่งดังรูปข้างล่างที่จะ

. (Xi + 1,Yi + 1)

(Xi,Yi). . (Xi + 1,Yi)

ทำให้เส้นตรงที่พลอตใกล้เคียงเส้นตรงที่ถูกกำหนดมามากที่สุด โดยตัดสินจากค่า (Ys-Ye) / (Xs-Xe) ซึ่งคืออัตราการเปลี่ยนแปลงของแนวดิ่งเมื่อแนวราบเปลี่ยนไปหนึ่งหน่วยความยาว เพราะฉะนั้นจะใช้ค่านี้เทียบว่าใกล้ศูนย์หรือใกล้หนึ่งมากกว่ากัน ถ้าใกล้ศูนย์มากกว่าก็จะเลือกตำแหน่ง (Xi +1,Yi) ถ้าใกล้หนึ่งมากกว่าก็จะเลือกตำแหน่ง (Xi +1,Yi +1) ความใกล้นี้วัดได้โดยการนำค่า (Ys-Ye)/(Xs-Xe) มาลบกับ ค่า 1/2 ถ้ามากกว่าศูนย์ก็คือใกล้ ตำแหน่ง (Xi+1/Yi+1) มากกว่า ถ้าน้อยกว่าศูนย์ก็คือใกล้ตำแหน่ง (Xi+1, Yi) มากกว่า และในกรณีใกล้ตำแหน่ง (Xi+1, Yi+1) จุด 2 จุด ต่อไปที่ต้องเลือกคือจุด (Xi+2, Yi+1) หรือจุด (Xi+2, Yi+2) ความใกล้ที่ใช้ตัดสินใจก็จะเป็นดังสมการข้างล่างนี้ คือ

NEAR = (2((Ye-Ys) / (Xe-Xs)) - 1/2 -1)

ค่า - 1 ตัวหลังเกิดการเพิ่มตำแหน่งแนวดิ่งอีก คือ Yi + 1 เพราะฉะนั้น ความใกล้จะอยู่ในรูปสมการดังนี้ (โปรดดูรูปที่ 3-7 ประกอบ)

1/2 คือค่าที่ตัดสินเลือกในครั้งแรก

ความใกล้ = n ((Ye-Ys) / (Xe-Xs)) - 1/2 -1 (p)

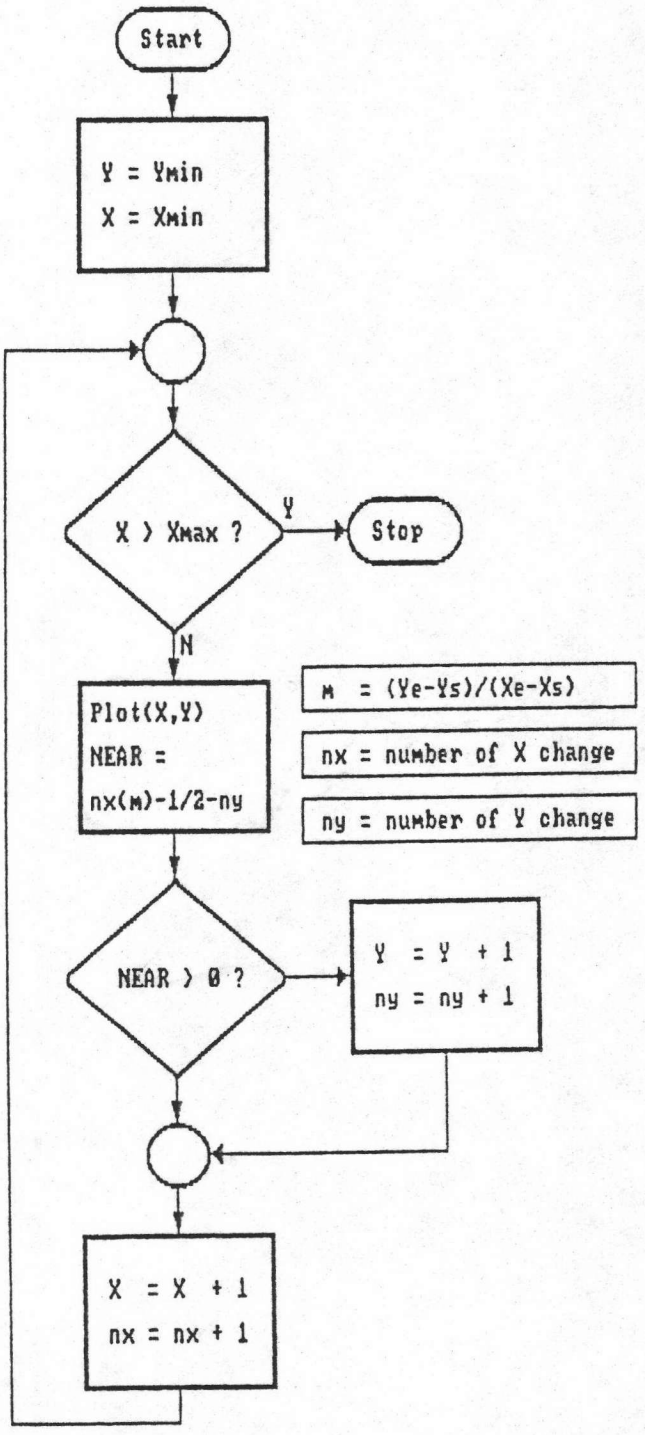
n คือ จำนวนครั้งที่ตำแหน่งแนวดิ่ง เพิ่มขึ้นหนึ่ง (แนว - ข)

แต่ทั้งเพื่อลดเวลาที่ต้องเสียในการคูณ เราสามารถเปลี่ยนการทำงานจากการคูณเป็นการบวกแทน[4] โดยอาศัยหลักที่ว่าการคูณคือการบวกหลายๆครั้ง (โปรดดูรูปที่ 3-8 ประกอบ)

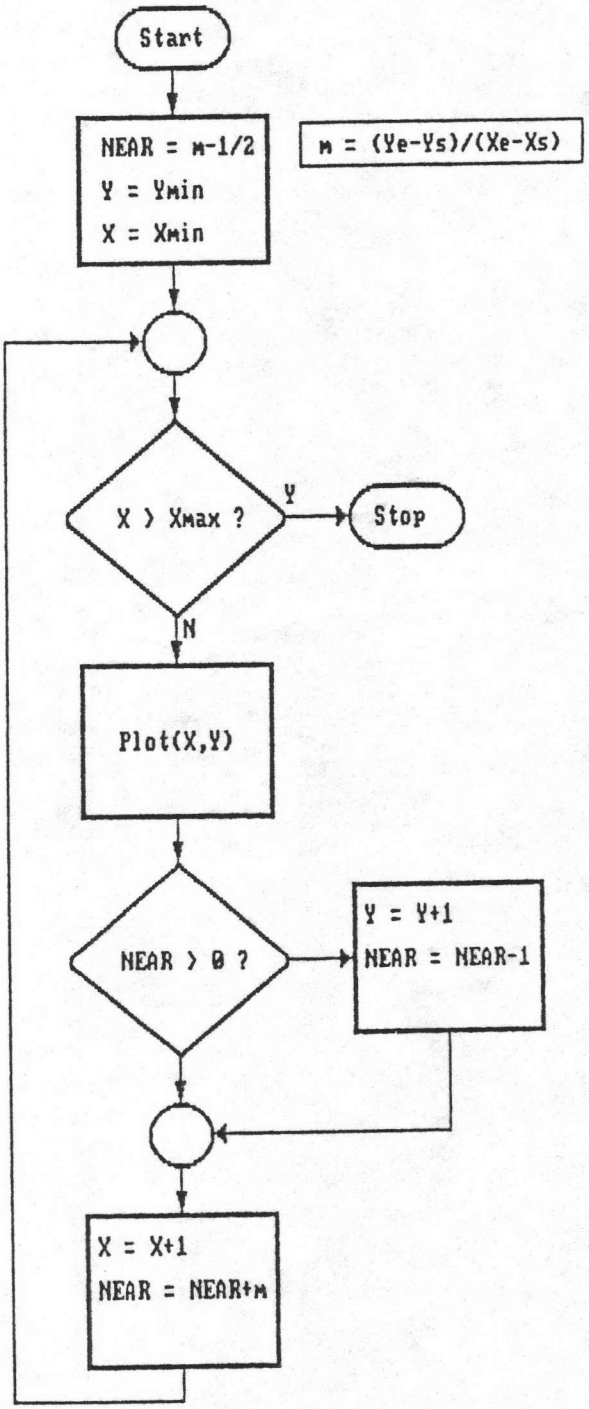
การทำงานในกรณีที่ 7 จะเหมือนกับกรณีที่ 6 เมื่อเรามองตำแหน่งแนวราบ (แนว-X) สลับกับตำแหน่งแนวดิ่ง (แนว-Y) เพราะฉะนั้น การทำงานก็จะสลับค่าทั้งลองเวลาพลอตจะใช้ตำแหน่งแนวราบเป็นแนวดิ่ง ตำแหน่งแนวดิ่งเป็นแนวราบ (Plot point (Y,X))

การทำงานในกรณีที่ 8 จะต่างกันกรณีที่ 6 คือแทนที่ตำแหน่งแนวดิ่งจะเพิ่มขึ้นทีละหนึ่งเมื่อมีความใกล้มากกว่าก็จะเป็นลดลงทีละหนึ่งแทน เพราะฉะนั้นการทำงานแทน เพราะฉะนั้น การทำงานแทนจะเปลี่ยนตำแหน่งแนวดิ่งโดยเพิ่มอีก 1 (Yi + 1) ก็จะเป็น (Yi-1) แทน ตำแหน่งแรกในการทำงานเป็นตำแหน่งมากกว่าระหว่าง Xs กับ Xe

การทำงานในกรณีที่ 9 จะต่างกับกรณีที่ 7 คือแทนที่ตำแหน่งแนวราบจะเพิ่มขึ้นทีละหนึ่งเมื่อมีความใกล้มากกว่า ก็จะเป็นลดลงทีละหนึ่งแทน เพราะฉะนั้นการพลอตแทนที่จะเปลี่ยนตำแหน่งแนวราบ โดยเพิ่มอีก 1 (Xi+1) ก็จะเป็น (Xi-1) (กรณีที่ 9 อาจเทียบกับได้กับกรณีที่ 8 โดยตำแหน่งแนวดิ่งกับตำแหน่งแนวราบต้องสลับกัน เวลามีการทำงาน) และตำแหน่งแรกในการทำงานเป็นตำแหน่งมากกว่าระหว่าง Xs กับ Xe



รูปที่ 4-7 แสดงการสร้างเส้นตรงที่มีการคำนวณโดยการคูณ



รูปที่ 4-8 แสดงการสร้างเส้นตรงที่ไม่มีการคำนวณโดยการคูณ

4.3 การสร้างโปรแกรมย่อยที่มีหน้าที่การทำงานพื้นฐาน

4.3.1 การสร้างโปรแกรมย่อย PLOTS

จุดประสงค์ สร้างขึ้นมาให้ผู้ใช้ที่มีความต้องการใช้โปรแกรมย่อยทางกราฟฟิก เรียกใช้ในตอนเริ่มต้น เพื่อจะได้กำหนดค่าเริ่มต้นให้ตัวแปรร่วมที่ใช้ในโปรแกรมย่อยทางกราฟฟิกอื่นๆ โดยเรียกเพียงครั้งเดียวตอนเริ่มต้น การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-9 ประกอบ) เริ่มโดยเรียกใช้โปรแกรมย่อย INITPARA และโปรแกรมย่อย INITSYST โปรแกรมย่อยนี้มีหน้าที่เพียงช่วยให้ผู้เขียนโปรแกรมเรียกใช้งานได้ง่าย และอีกประการหนึ่งวิธีการเรียกใช้โปรแกรมย่อยนี้ จะเหมือนกับโปรแกรมย่อยทางกราฟฟิกซึ่งแสดงผลลัพธ์ทางเครื่องพลอตเตอร์

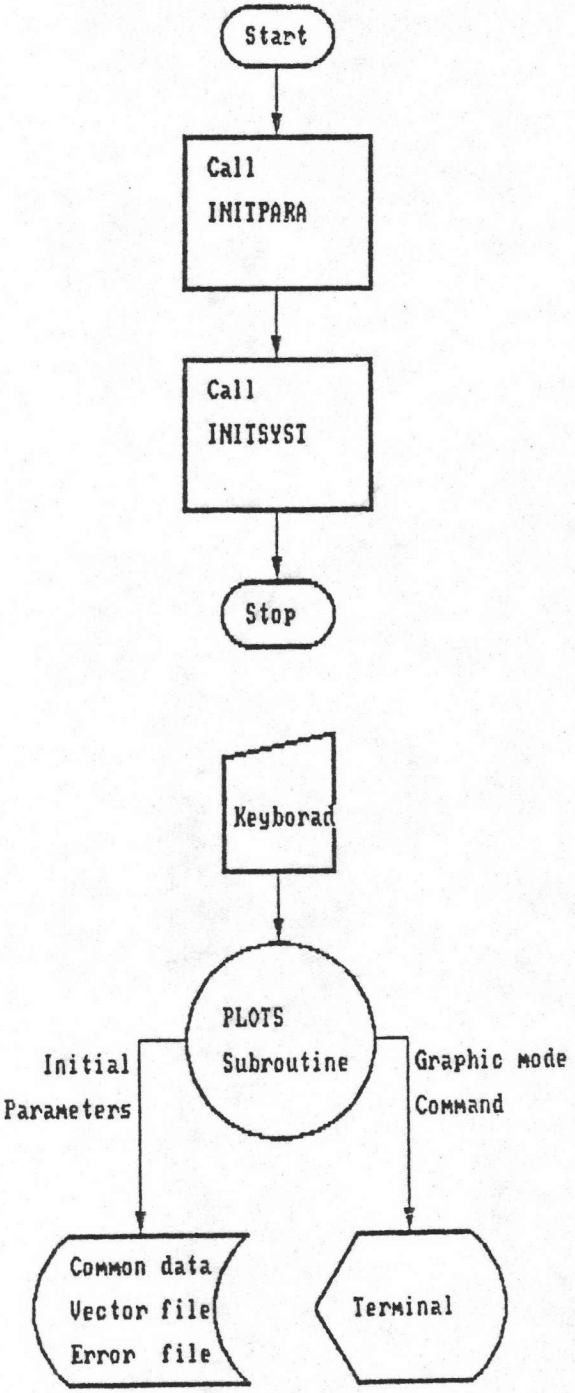
4.3.2 การสร้างโปรแกรมย่อย NEWPLT

จุดประสงค์สร้างขึ้นมาให้ผู้ใช้ที่มีความต้องการใช้โปรแกรมย่อยทางกราฟฟิกเรียกใช้เมื่อมีการพลอตภาพหลายภาพในโปรแกรมเดียวกันการทำงานของโปรแกรมย่อยนี้จะเพียงไปเรียกใช้โปรแกรมย่อย ENDPLT เพื่อจบการพลอตภาพ หลังจากนี้จะเรียกใช้โปรแกรมย่อย INITPARA และ INITSYST เพื่อเริ่มการพลอตภาพใหม่

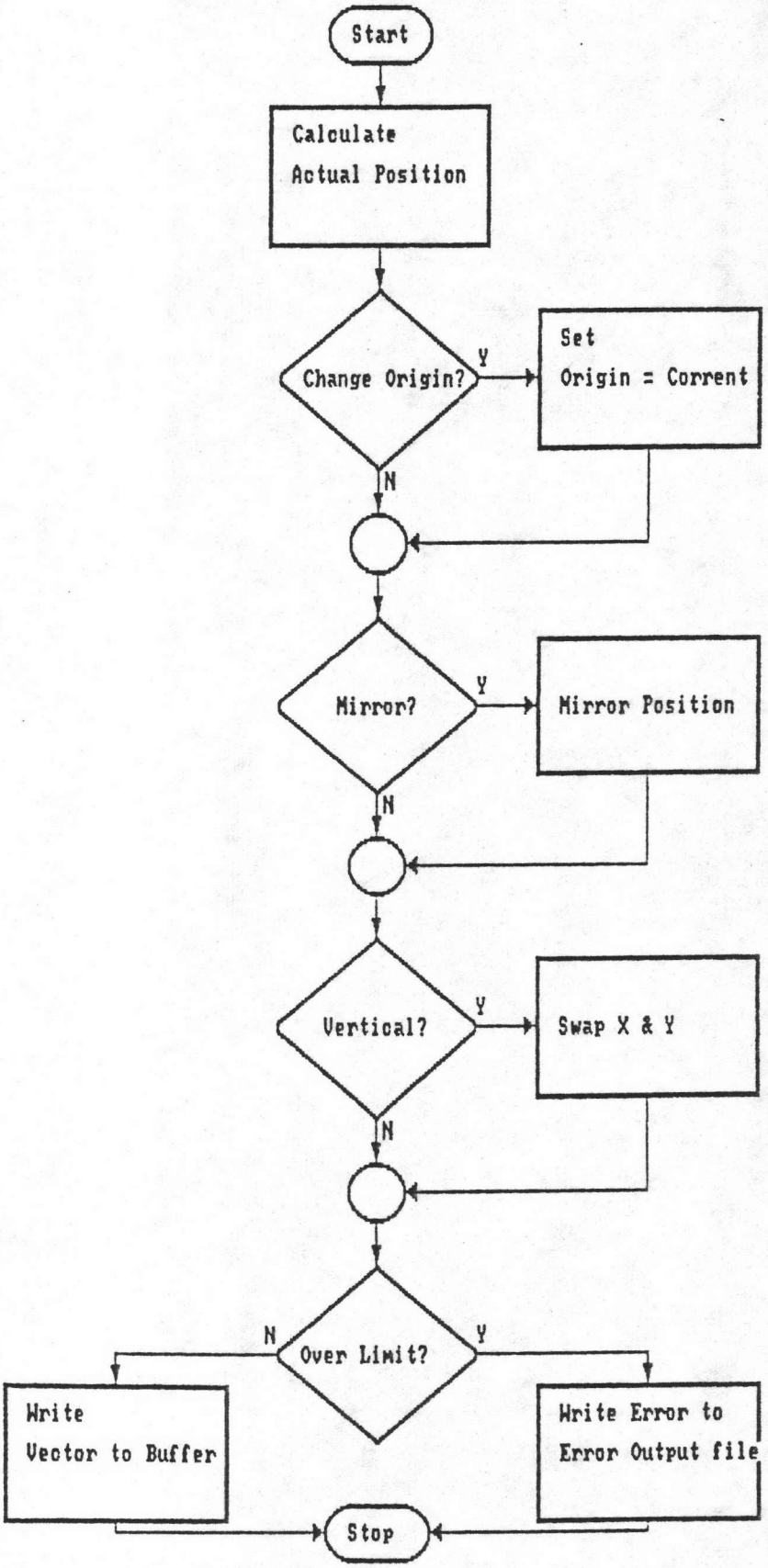
4.3.3 การสร้างโปรแกรมย่อย PLOT

จุดประสงค์เพื่อใช้พลอตภาพ เส้นทุกเส้นต้องทำการผ่านโปรแกรมย่อยนี้ มีอยู่ 2 คำสั่งคือ คำสั่งแรกทำการย้ายตำแหน่งพลอตอย่างเดียว อีกคำสั่งจะทำการย้ายตำแหน่งพลอตพร้อมกับลากเส้นไปด้วย การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-10 และ 4-11 ประกอบ) เริ่มดังนี้คือ

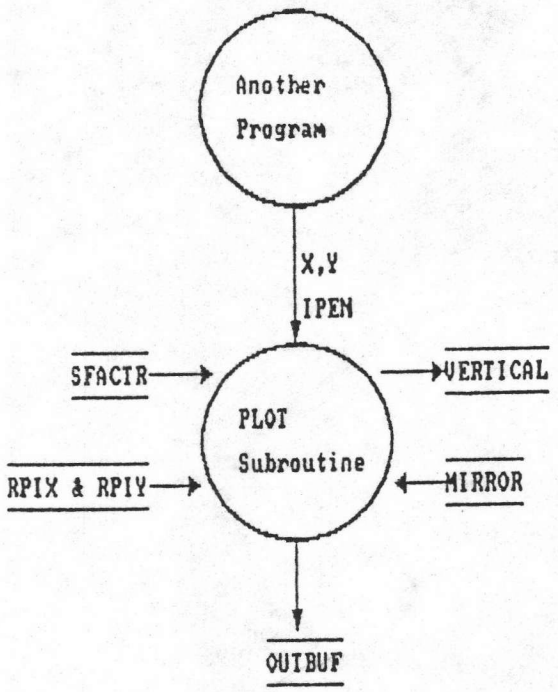
- นำค่าพารามิเตอร์(PARAMETER)ที่ส่งเข้ามาซึ่งประกอบด้วย ตำแหน่งที่จะทำการพลอต (X,Y) อ้างอิงกับจุดกำเนิดที่ถูกกำหนดไว้ (เก็บค่าไว้ที่ตัวแปรร่วม (XORG,YORG)) และคำสั่งการพลอต (IPEN) ที่กำหนดว่ามี



รูปที่ 4-9 แสดงการทำงานของโปรแกรมย่อย PLOTS และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย PLOTS กับส่วนอื่น



รูปที่ 4-10 แสดงการทำงานของโปรแกรมย่อย PLOT



รูปที่ 4-11 แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย PLOT กับส่วนอื่น

การลากเส้นหรือไม่ และมีการเปลี่ยนจุดกำเนิดหรือไม่ นำตำแหน่ง(X,Y) มาคำนวณหาตำแหน่งจริงบนจอภาพ โดยคำนวณกับอัตราย่อขยายขนากภาพ (ตัวแปรร่วม SFACTR) และจำนวนจุดต่อหนึ่งหน่วยความยาว(ตัวแปรร่วม RPIX RPIY) และจุดกำเนิด (ตัวแปรร่วม (XORG,YORG)) จนได้ตำแหน่งจริงบนจอภาพ

- ทำการตรวจดูคำว่าให้มีการย้ายจุดกำเนิดหรือไม่
  ถ้ามีจะทำการย้ายจุดกำเนิดมายังจุดปัจจุบันนี้
- ทำการตรวจลอบดูว่าต้องการให้มีการละท้อนภาพกับแกนแนวไหนหรือไม่ (ตรวจดูจากตัวแปรร่วม MIRROR ซึ่งจะถูกกำหนดได้ 2 วิธีคือ วิธีแรกถูกกำหนดให้เองตอนเริ่มต้น วิธีที่สองโดยโปรแกรมย่อย MIRROR)
- ทำการตรวจว่าต้องการให้สลับแกนภาพหรือไม่ (เพื่อให้รูปตะแคง แกนแนวราบของรูปจะขนานกับแกนแนวดิ่งของจอภาพ) จะได้ทำการสลับตำแหน่งแนวดิ่งกับแนวราบให้ (SWAP X and Y) (ตรวจจากตัวแปรร่วม VERTICLE) โดยค่านี้จะถูกกำหนดนี้จะถูกกำหนดจากผู้ใช้ตอนโปรแกรมเริ่มทำงาน (โปรดดูบทที่ 2)
- ทำการตรวจลอบ ตำแหน่งที่ได้นี้ว่า เกินขอบเขตที่จะพลอตได้หรือไม่ (มุมล่างซ้ายคือตำแหน่ง (0,0) มุมขวาบนต้องดูจากตัวแปรร่วมคือตำแหน่ง (HEADXMAX,HEADYMAX)) ถ้าเกินตำแหน่งนี้จะถูกพิมพ์ลงในแฟ้มข้อมูลที่เก็บข้อผิดพลาดการพลอต(Error file) แต่ถ้าถูกคำสั่งการพลอตนี้จะถูกเก็บไว้ในบัฟเฟอร์ก่อน (ตัวแปรร่วม OUTBUF) ซึ่งค่านี้จะถูกนำไปใช้ต่อโดยโปรแกรมย่อย CLRBUF อีกที เหตุที่ไม่ทำการพลอตไปยังอุปกรณ์แสดงผลลัพธ์เลยก็เพื่อแยกให้ โปรแกรมย่อยส่วนหนึ่งที่ผู้ใช้เรียกใช้ไม่ต้องขึ้นกับลักษณะของอุปกรณ์แสดงผลลัพธ์ (Machine Independence) การย้ายผลลัพธ์จากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์หนึ่งผู้ใช้ไม่ต้องยุ่งยาก โปรแกรมย่อยทางกราฟฟิกนี้จะช่วยจัดการให้

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

4.3.4 การสร้างโปรแกรมย่อย FACTOR

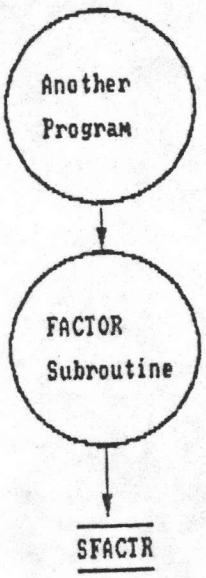
จุดประสงค์ใช้กำหนดอัตราย่อขยายขนาดภาพ เพื่อให้ได้ภาพที่มีขนาดตามต้องการของผู้ใช้ โดยรับค่าพารามิเตอร์ที่กำหนดอัตรายย่อขยายขนาดมาแล้วนำไปเก็บไว้ในตัวแปรร่วม SFACTR (โปรดดูรูปที่ 4-12 ประกอบ)

4.3.5 การสร้างโปรแกรมย่อย WHERE

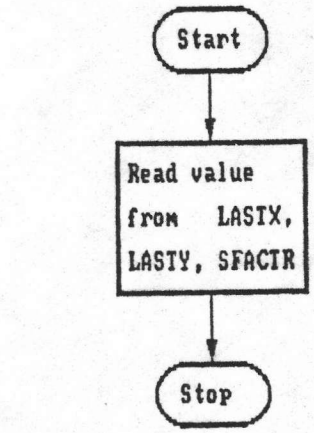
จุดประสงค์ใช้ตรวจสอบตำแหน่งล่าสุดที่มีการพลอต และอัตราย่อขยายขนาดภาพ การทำงานของโปรแกรมย่อยนี้เริ่มโดยอ่านค่าตัวแปรร่วม LASTX LASTY SFACTR ใส่ลงในพารามิเตอร์ แล้วส่งออกไป (โปรดดูรูปที่ 4-13 ประกอบ)

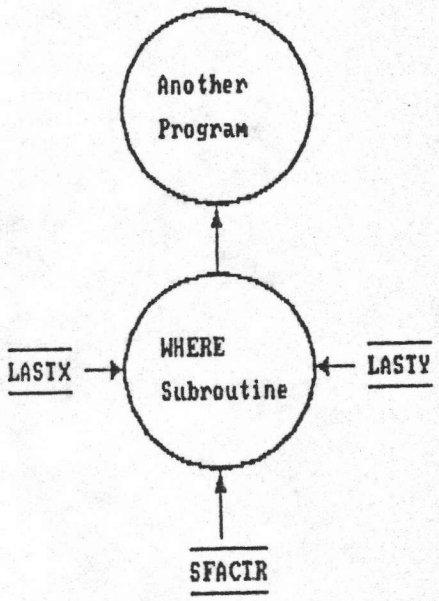
4.3.6 การสร้างโปรแกรมย่อย ROTATE

จุดประสงค์ ใช้กำหนดมุมเอียงของภาพ แต่การพลอตต้องกระทำผ่านโปรแกรมย่อย RELPLT แทนโปรแกรมย่อย PLOT (โปรดดูวิธีการเรียกใช้โปรแกรมย่อย ROTATE ในบทที่ 2) การทำการทำงานของโปรแกรมย่อยนี้ทำโดยนำค่าพารามิเตอร์ที่รับเข้ามาไปคำนวณ หาค่า SIN เก็บไว้ในตัวแปรร่วม TSIN และหาค่า COS เก็บไว้ในตัวแปรร่วม TCOS เพื่อใช้ในการเอียงภาพ โดยนำค่าตัวแปรร่วมทั้งสองไปใช้ในโปรแกรมย่อย RELPLT ต่อไป



รูปที่ 4-12 แสดงการทำงานของโปรแกรมย่อย FACTOR และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย FACTOR กับส่วนอื่น





รูปที่ 4-13 แสดงการทำงานของโปรแกรมย่อย WHERE และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย WHERE กับส่วนอื่น

4.3.7 การสร้างโปรแกรมย่อย MIRROR

จุดประสงค์ ใช้สะท้อนตำแหน่งที่ใช้พลอต (จากซ้ายเป็นขวา หรือจากบนเป็นล่าง หรือทั้งสองอย่าง) โปรแกรมย่อยนี้ทำงานโดยนำพารามิเตอร์มากำหนดตัวแปรร่วม MIRROR (ใช้ตรวจสอบในโปรแกรมย่อย PLOT) ซึ่ง

ถ้า MIRROR = 0 ตำแหน่ง (X,Y) จะเป็นปกติ

ถ้า MIRROR = 1 ตำแหน่งแนวราบจะสะท้อน คือ Xnew = Xmax - X

ถ้า MIRROR = 2 ตำแหน่งแนวดิ่งจะสะท้อน คือ Ynew = Ymax - X

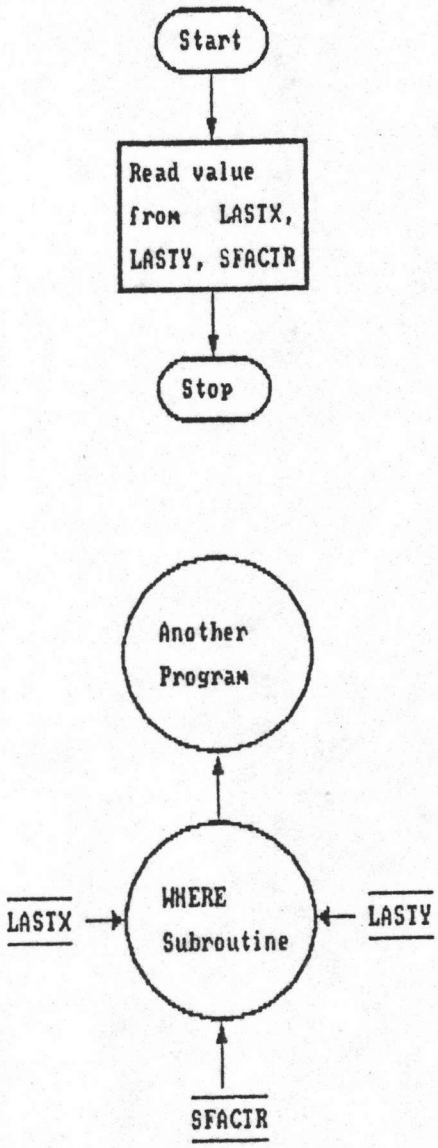
ถ้า MIRROR = 3 สะท้อนทั้งสองแกน

คือ Xnew = Xmax - X และ Ynew = Ymax - X

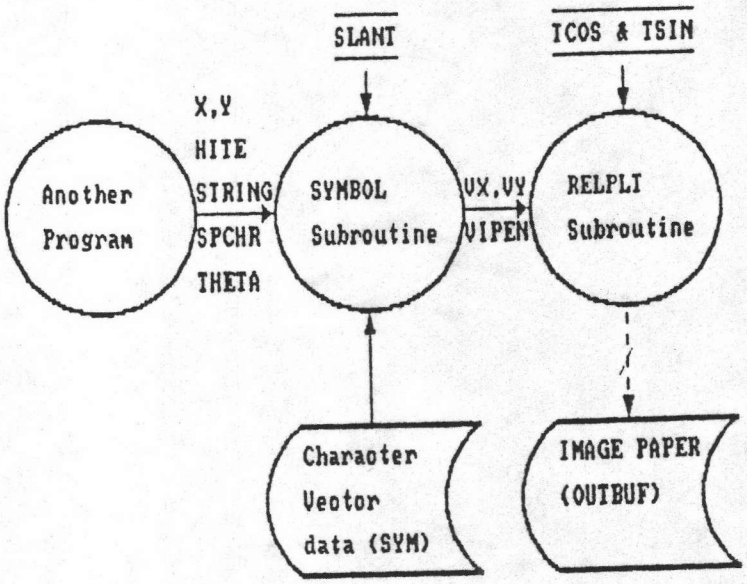
4.3.8 การสร้างโปรแกรมย่อย SYMBOL

จุดประสงค์ใช้พลอตตัวอักษรตามต้องการผู้ใช้ ตัวอักษรเหล่านี้จะถูกเก็บไว้ในลักษณะของเวกเตอร์(Stroke method)เก็บไว้ที่ตัวแปรร่วม SYM การทำงานของโปรแกรมย่อยนี้(โปรดดูรูปที่ 4-14 และ 4-15 ประกอบ) เริ่มดังนี้คือ

- ตรวจสอบว่าผู้ใช้ต้องการพลอตตัวอักษร ASCII หรือตัวอักษรพิเศษและตัวอักษรของผู้ใช้ แล้วทำการกำหนดตำแหน่งที่จะทำการพลอตตัวอักษร
- ทำการคำนวณหาความกว้างของตัวอักษร และช่องไฟจากความสูงของตัวอักษรที่กำหนดมา
- รหัสของตัวอักษรที่ส่งเข้ามาจะถูกนำมาใช้หาตำแหน่งที่เก็บเวกเตอร์ของตัวอักษร ถ้าหาไม่พบจะมีแต่การย้ายตำแหน่งพลอตไปเท่ากับความกว้างของตัวอักษร ถ้าเป็นตัวอักษร ASCII ตัวอักษรที่จะทำการพลอตอาจมีหลายตัวได้ ก็จะทำการพลอตตัวอักษรทุกตัวเรียงกันไป แต่ถ้าเป็นตัวอักษรพิเศษจะทำการพลอตตัวเดียว
- จบการทำงานของโปรแกรมย่อยนี้



รูปที่ 4-14 แสดงการทำงานของโปรแกรมย่อย SYMBOL



รูปที่ 4-15 แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย SYMBOL กับส่วนอื่น

4.3.9 การสร้างโปรแกรมย่อย CSLANT

จุดประสงค์ใช้กำหนดมุมเอียงของตัวอักษร ที่พลอตผ่านโปรแกรมย่อย SYMBOL โปรแกรมนี้จะไปกำหนดตัวแปรร่วม SLANT ตามพารามิเตอร์ ที่กำหนดเข้ามา เป็นอันจบการทำงาน (โปรดดูรูปที่ 4-16 ประกอบ)

4.3.10 การสร้างโปรแกรมย่อย USRSYM

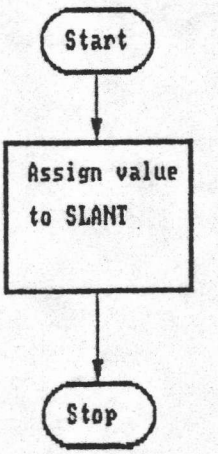
จุดประสงค์ให้ผู้ใช้สามารถสร้างตัวอักษรของตัวเอง ขึ้นมาใช้ได้โดย โปรแกรมย่อยนี้จะทำงาน 2 ลักษณะ คือ

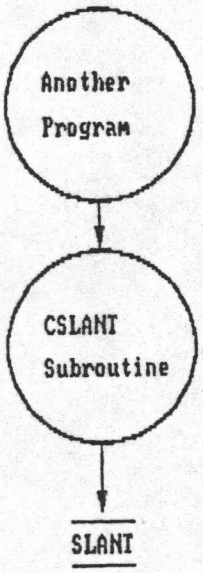
2. ทำการลบตัวอักษรที่สร้างไว้ทิ้ง โดยทำการลบเป็นชุด ชุดแรกรหัสตั้งแต่ 288 ถึง 383 หรือชุดที่สอง รหัสตั้งแต่ 384 ถึง 479 เป็นการลบโดยทำตำหนิ (Mark) ไว้ แล้วจึงทำการลบจริงๆ ตอนสุดท้ายก่อนจบการทำงาน

1. การสร้างตัวอักษร

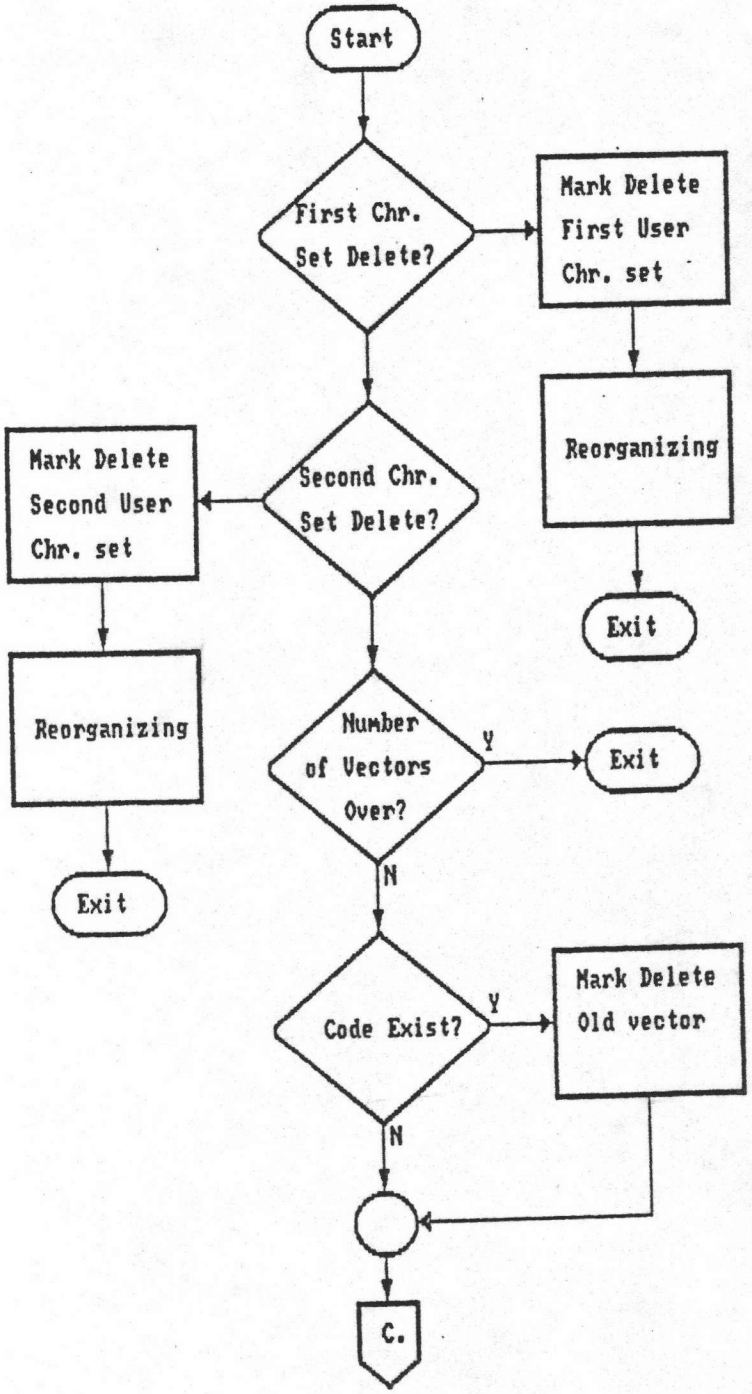
- ตรวจลอบก่อนว่ามีที่เหลือพอ สำหรับเก็บเวกเตอร์ของตัวอักษรที่กำหนดเข้ามาหรือไม่ ถ้าไม่จะจบการทำงาน แต่ถ้ามีที่พอก็จะทำงานต่อไป
- ตรวจลอบว่ามีตัวอักษรรหัสนี้อยู่หรือไม่ ถ้ามีต้องลบทิ้งก่อน หลังจากนี้จะทำการเก็บเวกเตอร์ของตัวอักษรนี้ไว้ที่ที่ว่างตอนท้ายของตัวแปรร่วมที่เก็บเวกเตอร์ของตัวอักษร(ตัวแปรร่วม SYM)แล้วทำการจัดการตัวแปรร่วมนี้ใหม่ โดยรวมส่วนที่มีการใช้งานให้ติดกันไว้ในตอนต้น
- จบการทำงานการทำงานของโปรแกรมย่อยนี้

(โปรดดูรูปที่ 4-17 และ 4-18 ประกอบ)

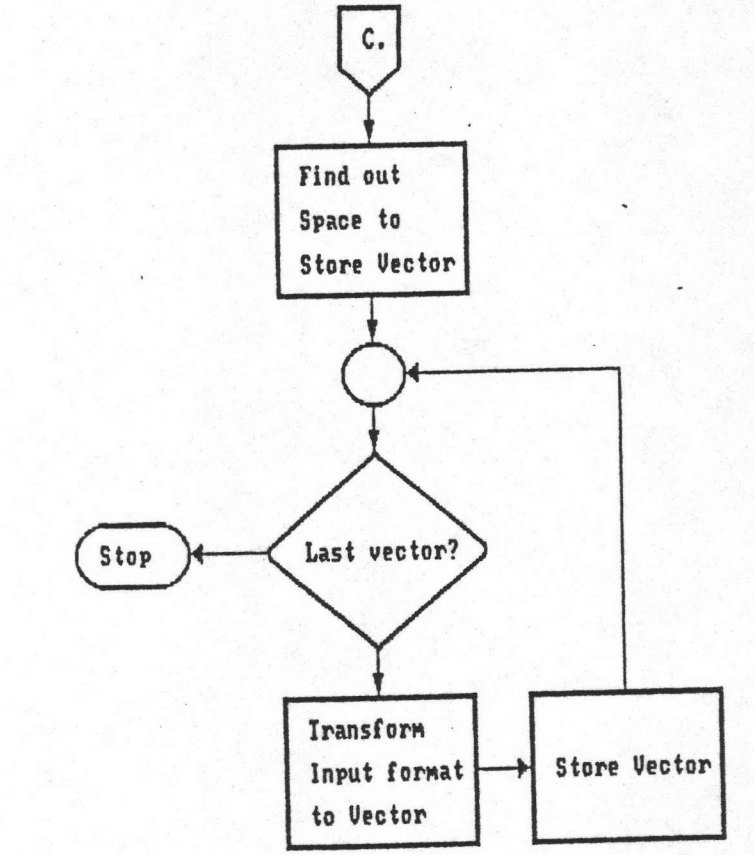


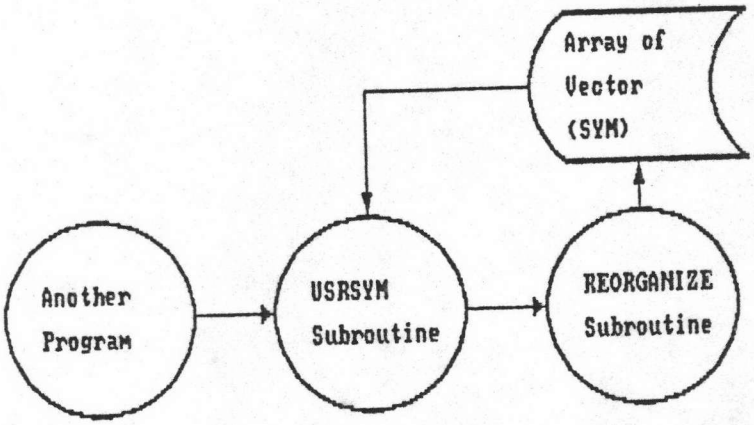


รูปที่ 4 -16 แสดงการทำงานของโปรแกรมย่อย CSLANT และ
แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย CSLANT กับส่วนอื่น



รูปที่ 4-17 แสดงการทำงานของโปรแกรมย่อย USRSYM





รูปที่ 4-18 แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย USRSYM กับส่วนอื่น

4.3.11 การสร้างโปรแกรมย่อย NUMBER

จุดประสงค์ ใช้เปลี่ยนรูปแบบของข้อมูลที่ใช้ในการคำนวณ มาเป็นรูปแบบที่สามารถแสดงผลลัพธ์ได้ (ตัวอักษร) การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-19 ประกอบ) เริ่มโดยนำจำนวนที่ต้องการแสดงผลลัพธ์ มาคำนวณหาว่าต้องใช้ตัวอักษร (ที่เป็นตัวเลขและจุดและเครื่องหมายลบ) กี่ตัว ในการแสดงผลลัพธ์ สมการในการคำนวณหาจะเป็นดังสมการข้างล่างนี้

NCHARS = + IFIX (ALOG10 (ABS (RNUM)))

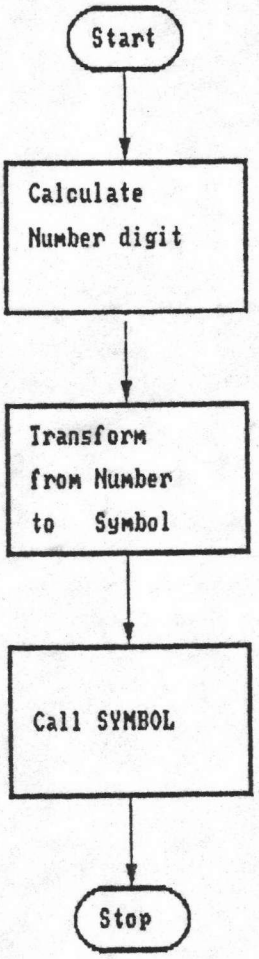
+ NDEC : จำนวนจุดหลังจุดทศนิยมที่กำหนดเข้ามา

+ 2 : สำหรับจุดทศนิยมและเครื่องหมายเลข

NCHARS : จำนวนตัวอักษรที่ต้องใช้

RNUM : ค่าตัวเลขที่ต้องการแสดงผลลัพธ์

หลังจากคำนวณหาจำนวนตัวอักษรได้แล้วก็จะทำการเปลี่ยนรูปแบบข้อมูล เป็นแบบตัวอักษร และส่งตัวอักษรที่ได้นี้ส่งไปยังโปรแกรมย่อย SYMBOL ต่อไปเป็นอันจบการทำงานของโปรแกรมย่อยนี้



รูปที่ 4-19 แสดงการทำงานของโปรแกรมย่อย NUMBER และ
แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย NUMBER กับส่วนอื่น

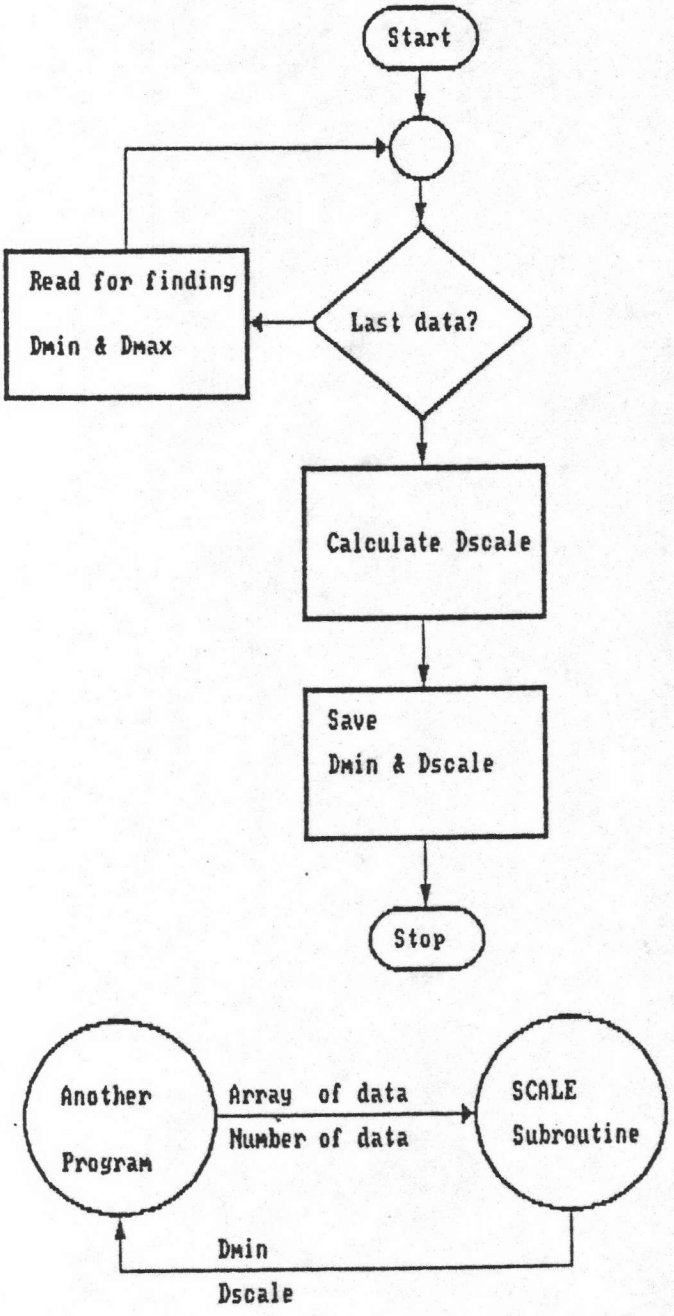
4.3.12 การสร้างโปรแกรมย่อย SCALE

จุดประสงค์เพื่อทำการคำนวณหาค่าที่น้อยที่สุดของข้อมูล และอัตราส่วนของค่าของข้อมูลต่อหนึ่งหน่วยความยาว ซึ่งจะใช้คำนวณหาค่ากำกับแกนกราฟ การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-20 ประกอบ) เริ่มโดยนำข้อมูลในชุดข้อมูลทั้งหมดมาหา ค่ามากที่สุด (Dmax) และ ค่าที่มีค่าน้อยที่สุด (Dmin) ซึ่งจะถูกเก็บไว้ที่ตอนท้ายของชุดข้อมูล หลังจากหาเจอแล้วจะนำมาคำนวณหาอัตราส่วนค่าของข้อมูลต่อหนึ่งหน่วยความยาว โดยใช้ค่าพารามิเตอร์ที่บอกความยาวของแกนกราฟร่วมด้วย ตามลมการต่อไปนี้

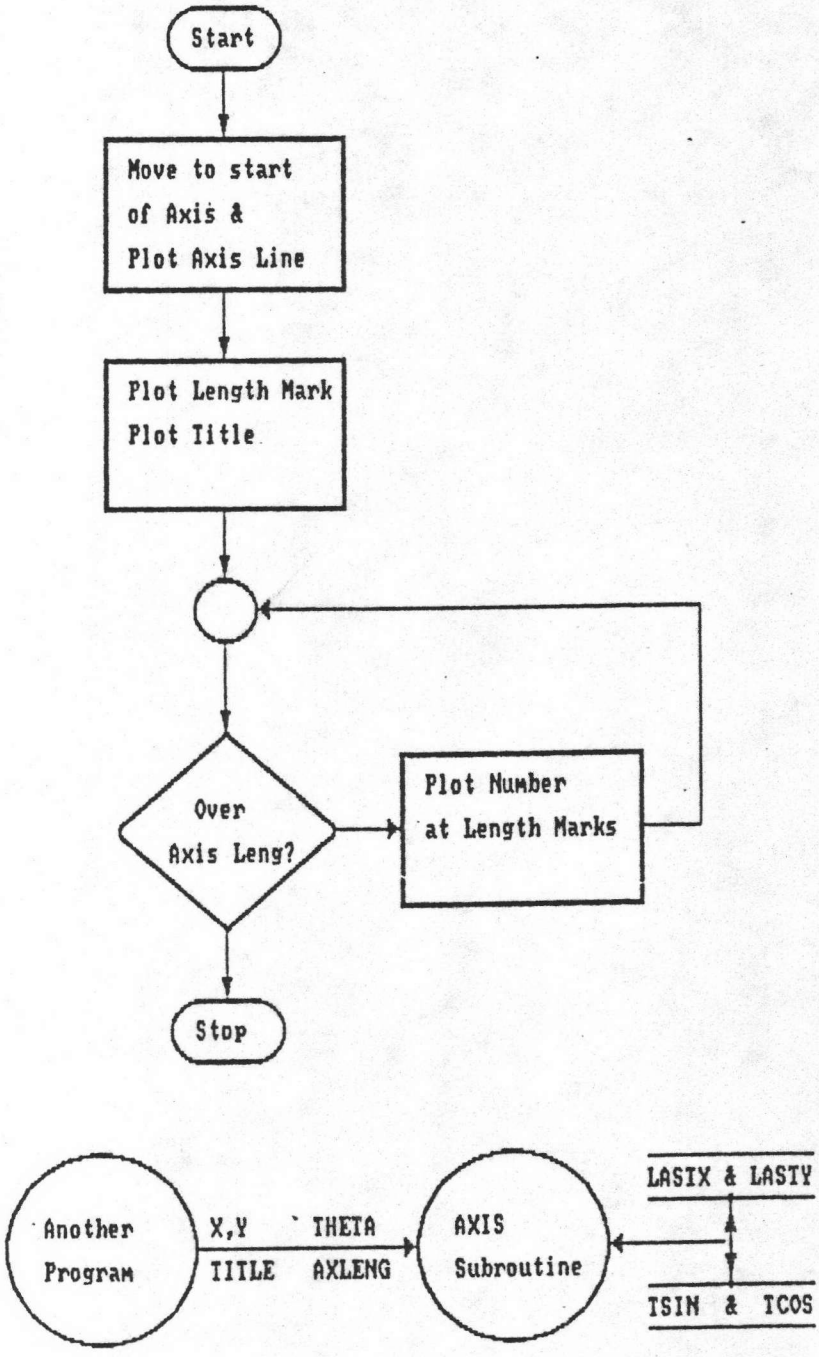
```
Dmin    -> ARRAY (NPTS*INC+1)
Dscale  = (Dmax-Dmin)/AXLENG
(AXLENG :  AXis LENGth)
Dscale  -> ARRAY (NPTS*INC+INC+1)
```

4.3.13 การสร้างโปรแกรมย่อย AXIS

จุดประสงค์ใช้สร้างแกนกราฟกำกับรูปกราฟเพื่อใช้อ้างอิง ในการอ่านข้อมูลจากกราฟ โดยมีค่าของข้อมูลกำกับแกนกราฟ การทำงานของโปรแกรมนี้ (โปรดดูรูปที่ 4-21 ประกอบ) เริ่มโดยย้ายตำแหน่งเริ่มต้นแกน และลากเส้นแกนกราฟ หลังจากนั้นจะขีดทุกหนึ่งหน่วยความยาวจนครบ พารามิเตอร์ที่ส่งเข้ามาจะกำหนดตำแหน่งเริ่มต้นแกนกราฟ คำอธิบายกำกับแกนกราฟ มุมของแกนกราฟ ความยาวแกนกราฟ และจะมีการใช้ตัวแปรร่วม TCOS และ TSIN สำหรับกำหนดมุมเอียงแกนกราฟ และตัวแปรร่วม LASTX และ LASTY



รูปที่ 4 -20 แสดงการทำงานของโปรแกรมย่อย SCALE และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย SCALE กับส่วนอื่น



รูปที่ 4-21 แสดงการทำงานของโปรแกรมย่อย AXIS และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย AXIS กับส่วนอื่น

4.3.14 การสร้างโปรแกรมย่อย LINE

จุดประสงค์ของโปรแกรมย่อยนี้ ใช้ลากเส้นตามข้อมูลในชุดข้อมูลโดยต้องบอกค่าที่น้อยที่สุดในชุดข้อมูล และอัตราย่อขยายที่เหมาะสมกับความยาวของแกนที่จะใช้ลากเส้น โปรแกรมย่อยนี้จะเริ่มต้นด้วยการนำค่าที่น้อยที่สุดของข้อมูลและอัตราย่อขยายจากตอนท้ายของชุดข้อมูลออกมาก่อน แล้วนำมาหาค่าที่ต้องใช้คูณกับทุกค่าข้อมูลก่อนส่งไปพลอต หลังจากนี้จะเริ่มนำค่าข้อมูลในชุดข้อมูลออกมาพลอต โดยมีการตรวจลอบดูว่ามีการพลอตตัวอักษร หรือพลอตแต่เส้นอย่างเดียวเส้นอย่างเดียว หรือพลอตทั้งสองอย่างจนกระทั่งครบทุกข้อมูล(โปรดดูรูปที่ 4-22 และ 4-23 ประกอบ)

คำนวณหาตำแหน่งตอนท้ายของชุดข้อมูลที่เก็บค่าที่น้อยที่สุดได้จากสมการนี้

Address of Xmin & Ymin = NPTS * INC +1

NPTS คือ จำนวนจุดที่ใช้สร้างเส้น

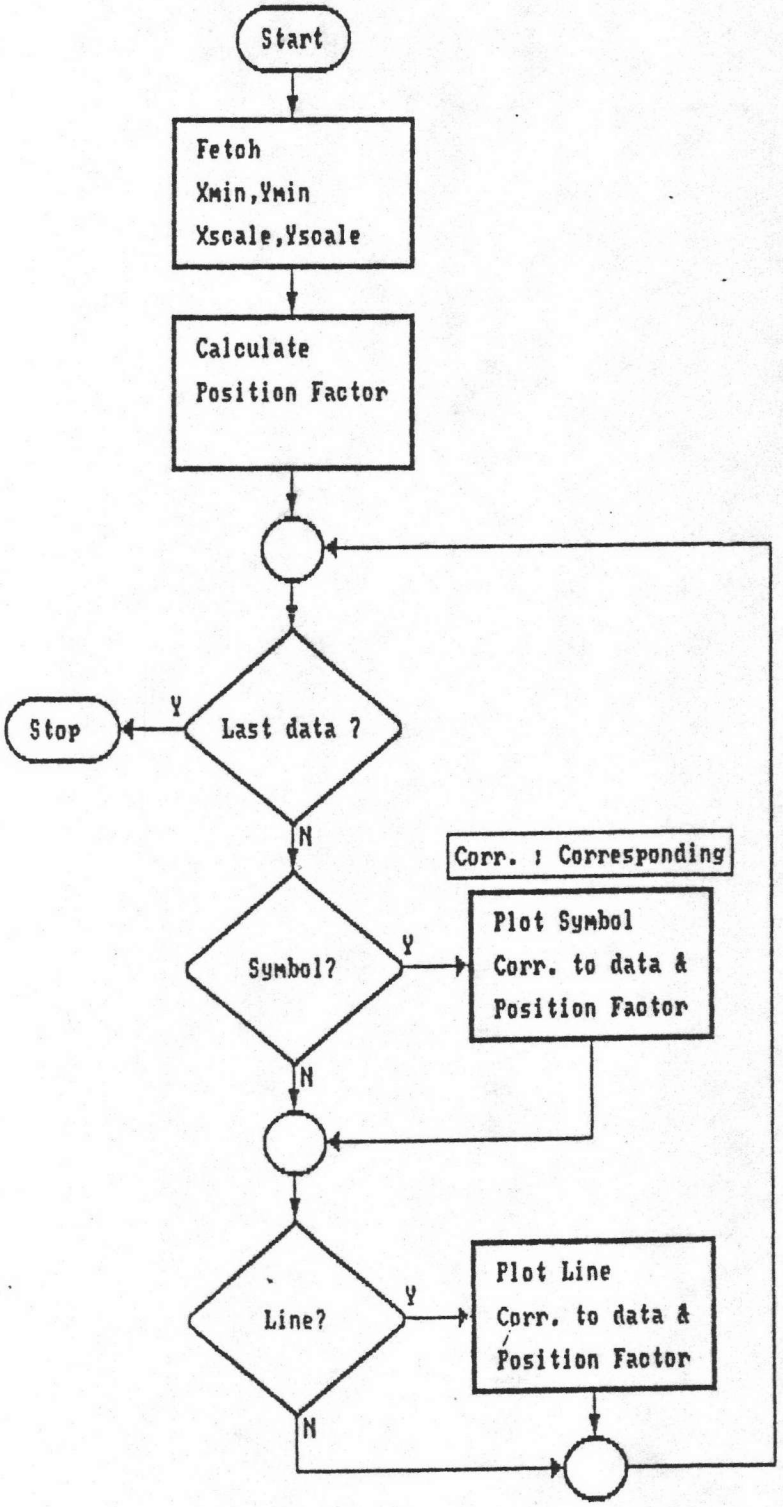
INC คือ ค่าที่กำหนดระยะห่างของตำแหน่งระหว่างข้อมูลแต่ละข้อมูล

คำนวณหาตำแหน่งตอนท้ายของชุดข้อมูลที่เก็บอัตราส่วนค่าของข้อมูลต่อหนึ่งหน่วยความยาวของแกนกราฟได้จากสมการต่อไปนี้

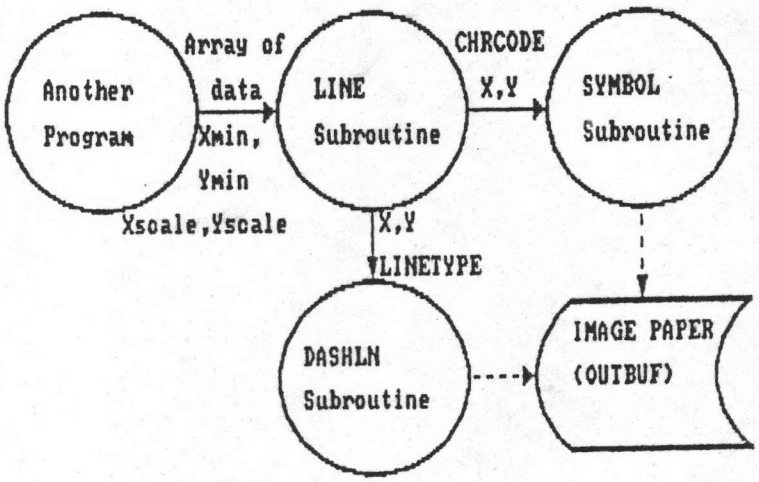
Address of xscale & Yscale =NPTS*INC + INC + 1

4.3.15 การสร้างโปรแกรมย่อย CIRCLE

จุดประสงค์ของโปรแกรมย่อยนี้ใช้พลอตวงกลมหรือส่วนหนึ่งของวงกลมโดยมีรูปแบบของการส่งพารามิเตอร์หลายวิธี ซึ่งพารามิเตอร์เหล่านี้จะถูกนำมาจัดอยู่ในรูปที่มีพารามิเตอร์ที่เหมือนกัน คือ ตำแหน่งจุดศูนย์กลางของวงกลม (Xc,Yc) รัศมีวงกลม (R) มุมเริ่มต้นที่มีการพลอต (As) มุมเส้นสุดที่มีการพลอต (As) มุมสิ้นสุดที่มีการพลอต (As) ทิศทางของการพลอตวงกลม โปรแกรมนี้จะเริ่มทำงานโดยตรวจสอบว่าโหมดของพารามิเตอร์ที่ส่งเข้ามามีลักษณะเช่นไรก่อน แล้วไปทำการแปลงรูปของพารามิเตอร์ให้อยู่ในรูปพารามิเตอร์เดียวกันก่อน แล้วส่งไปพลอตรูปวงกลมต่อไป ใช้วิธีใช้การสร้างเส้นตรงเล็กๆหลายๆเส้นมาต่อกัน ถ้าแบ่งเป็นเส้นตรงเล็กๆมากภาพก็จะละเอียดมาก ถ้าแบ่งเป็นเส้นตรงเล็กๆ น้อยภาพจะหยาบ



รูปที่ 4-22 แสดงการทำงานของโปรแกรมย่อย LINE



รูปที่ 4 -23 แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย LINE กับส่วนอื่น

วิธีการที่ใช้[5]คือ ให้ (X,Y) เป็นจุดบนเส้นโค้ง

A คือมุมที่เส้นรัศมีกวาดไป

X = RcosA + Xc

Y = RsinA + Yc

จากสมการจะเห็นได้ว่าพารามิเตอร์ คือ A และสมการดิฟเฟอเรนเชียล คือ

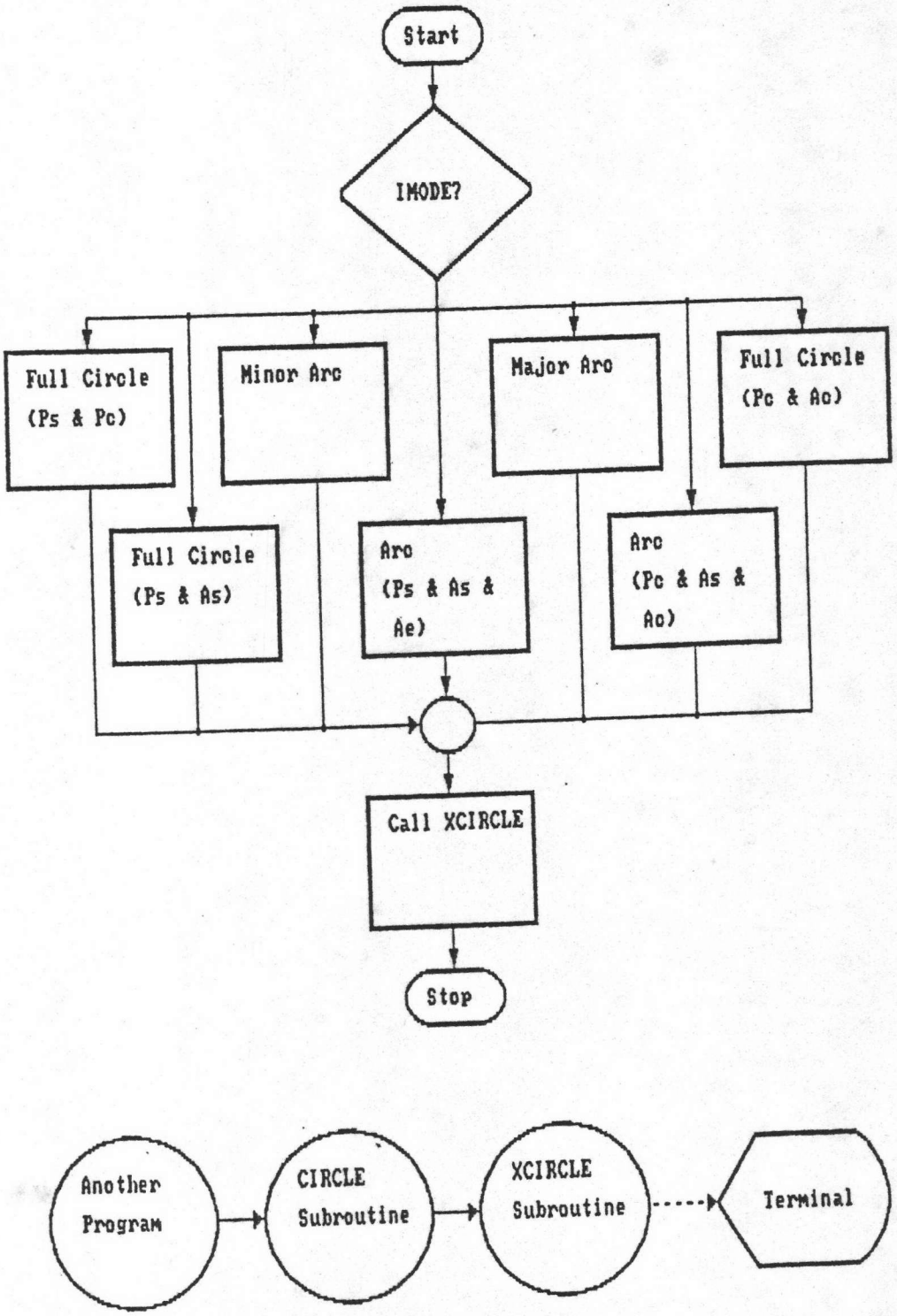
dX = -RsinAdA

dY = RcosAdA

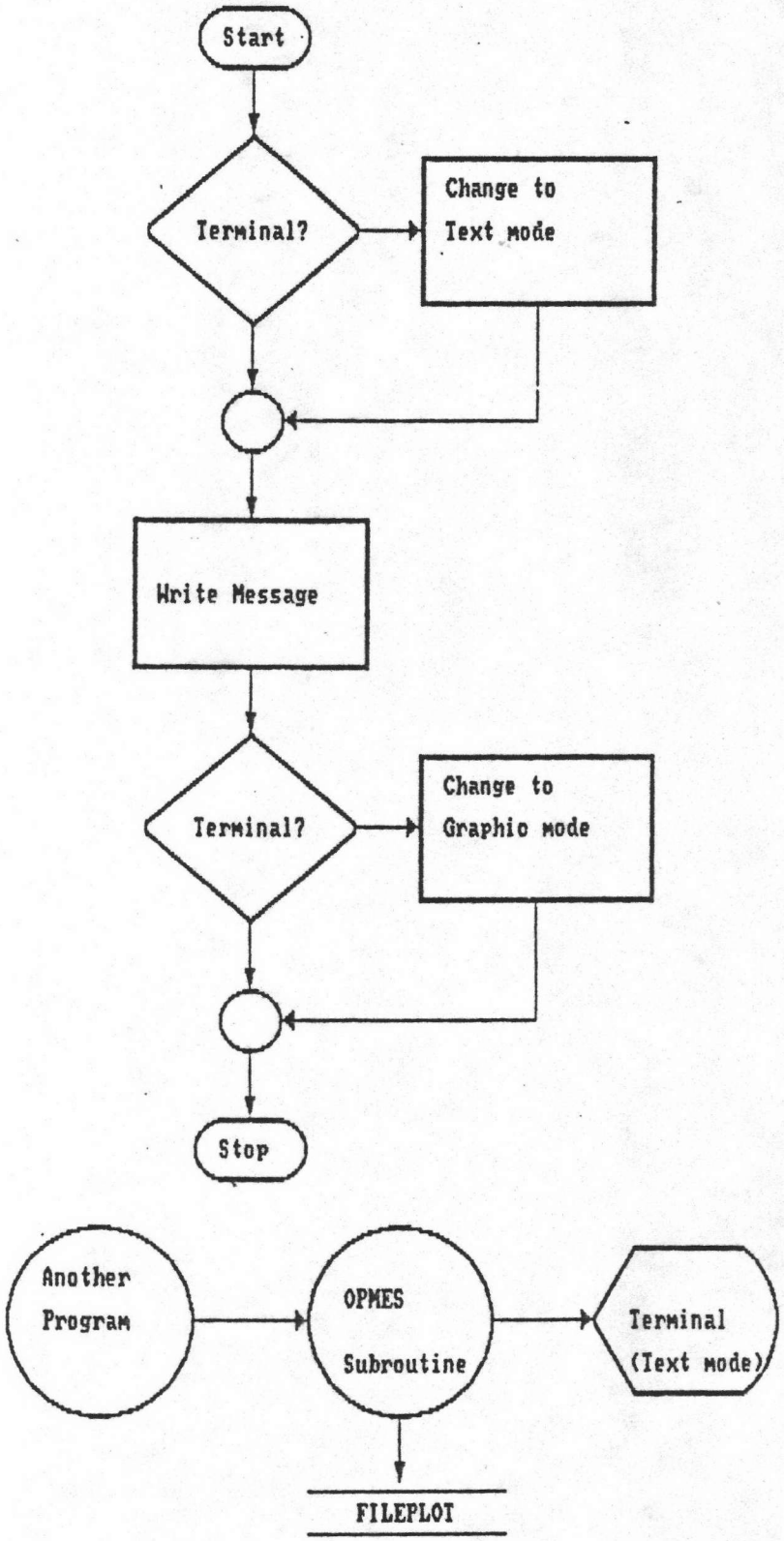
แล้วทำการพลอตใช้คำสั่ง PLOT (DX,DY,2) พลอตส่วนโค้งของวงกลมจนกระทั่ง A = Ae โดยเริ่มจาก A = As เพราะฉะนั้น ถ้าเปลี่ยนค่า A ทีละมากๆ ภาพที่ได้จะหยาบ ถ้าเปลี่ยนค่าทีละน้อยๆ ภาพที่ได้จะละเอียด โปรแกรมย่อยนี้จะแบ่งเป็นสองส่วน โปรแกรมย่อย CIRCLE ใช้เปลี่ยนรูปของพารามิเตอร์ และโปรแกรมย่อย XCIRCLE ใช้พลอตรูปวงกลม (โปรดดูรูปที่ 4-24 ประกอบ)

4.3.16 การสร้างโปรแกรมย่อย OPMES

จุดประสงค์เพื่อแสดงข้อความระหว่างมีการพลอตภาพ การทำงานของโปรแกรมจะเริ่มโดย ตรวจสอบว่าตอนนี้โปรแกรมแสดงผลลัพธ์ทางกราฟฟิกบนอุปกรณ์อะไร(ตรวจสอบจากตัวแปรร่วม FILEPLOT) ถ้าเป็นจอภาพจะได้มีการเปลี่ยนโหมด(mode) ของจอภาพมาเป็น โหมดตัวอักษร (text mode) แล้วแสดงข้อความ แต่ถ้าแสดงบนเครื่องพิมพ์ (ผ่านทางแฟ้มข้อมูล) ก็แสดงข้อความได้เลย(โปรดดูรูปที่ 4-25 ประกอบ)



รูปที่ 4 -24 แสดงการทำงานของโปรแกรมย่อย CIRCLE และ
แสดงความสัมพันธ์ระหว่างโปรแกรมย่อย CIRCLE กับส่วนอื่น



รูปที่ 4-25 แสดงการทำงานของโปรแกรมย่อย OPMES และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย OPMES กับส่วนอื่น

4.4 การสร้างโปรแกรมย่อยทางกราฟฟิกที่มีหน้าที่พิเศษเฉพาะอย่าง

3.4.1 การสร้างโปรแกรมย่อย RPLOT

จุดประลงค์ ใช้พลอตเส้นตรง โดยค่าพารามิเตอร์ที่ส่งเข้ามาจะอ้างอิงกับตำแหน่งล่าสุดที่มีการพลอต ไม่ได้อ้างอิงจุดกำเนิดเหมือนโปรแกรมย่อย PLOT โปรแกรมย่อยนี้จะเริ่มทำงานโดย อ่านตำแหน่งล่าลุดที่มีการพลอตผ่านจากโปรแกรมย่อย WHERE นำมาบวกกับตำแหน่งที่ส่งเข้ามาแล้วเรียกใช้โปรแกรมย่อย PLOT ต่อไป (โปรดดูรูปที่ 4-26 ประกอบ)

4.4.2 การสร้างโปรแกรมย่อย RELPLT[3]

จุดประลงค์ ใช้พลอตเส้นตรง โดยค่าพารามิเตอร์ ที่กำหนดตำแหน่งในการพลอตจะมีการเอียงตามมุมที่กำหนด โดยตัวแปรร่วม TCOS และ TSIN และอ้างอิงกับตำแหน่งล่าสุดที่มีการพลอตด้วย การหมุนภาพให้เอียงจะทำการหมุนไปรอบจุดกำเนิด โดยอาศัยหลักการดังต่อไปนี้

กำหนดให้จุด A คือ (X1, Y1) เป็นจุดที่ต้องหมุน

จุด A คือ (X2, Y2) เป็นจุดที่ถูกหมุนแล้ว

L คือ ระยะจากุดกำเนิดไปยังจุด a

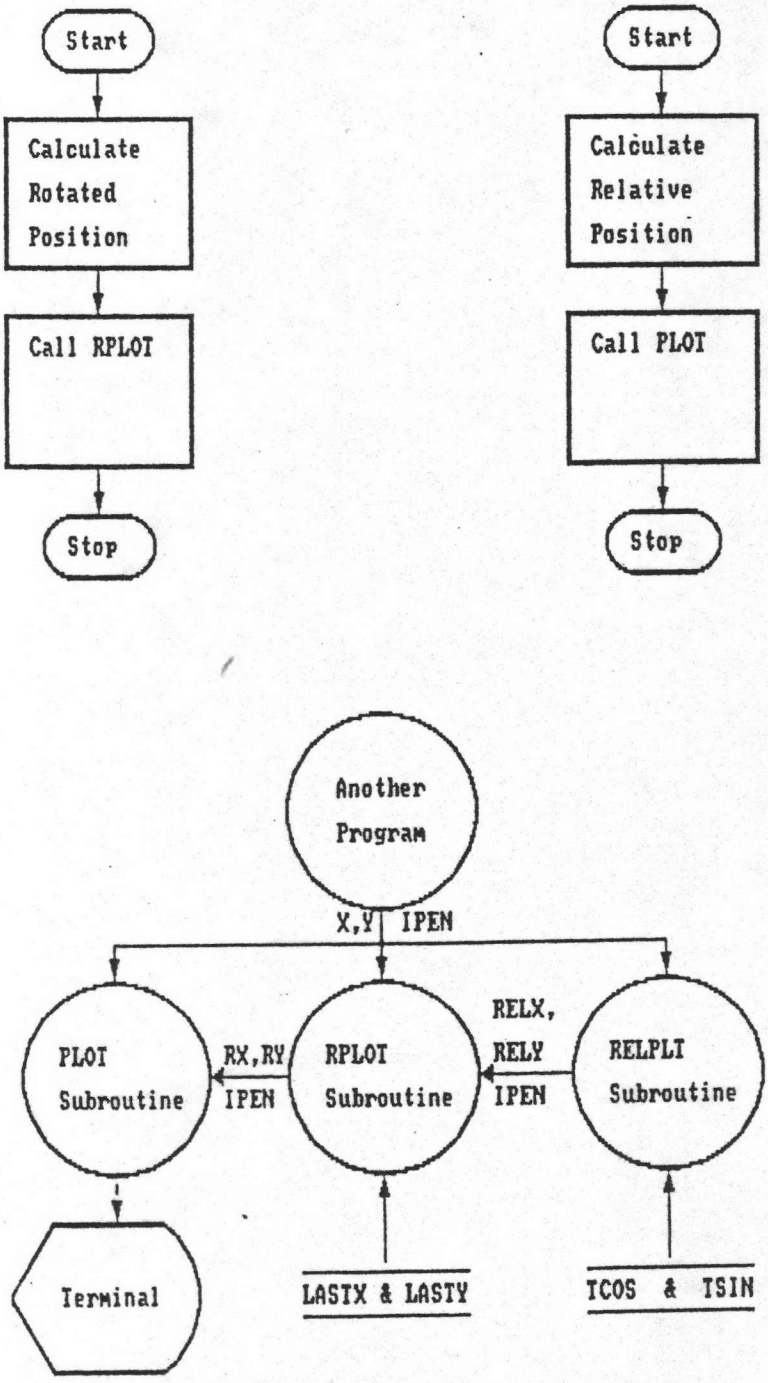
```
L คือ ((X1**2. + Y1**2.) **.5)
SIN (a) = Y1/L
COS (a) = X1/L
```

เมื่อมีการ หมุนไปเป็นมุม d โดย b = a +d

```
SIN (b) = SIN (a+d) =  Y2/L
Y2 = (COS (d) * SIN (a)*L) + (SIN(d) * COS (a)*L)
   = COS (d) * Y1 + SIN (d) * X1
```

ในทำนองเดียวกัน

```
X2 = COS (d) * X1 - SIN (d) * Y1
```



รูปที่ 4-26 แสดงการทำงานของโปรแกรมย่อย RPLOT และ RELPLT และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย RPLOT และ RELPLT กับส่วนอื่น

เพราะฉะนั้นเมื่อต้องหมุนรูปภาพ ก็ทำโดยการหาค่า COS และ SIN ของมุมที่ต้องการหมุน มาคูณกับตำแหน่งตามปกติ (เก็บไว้ในตัวแปรร่วม TCOS และ TSIN ซึ่งถูกกำหนดตอนเริ่มต้นโดยโปรแกรมย่อย INIT SYST หรือ ถูกกำหนดผ่านโปรแกรมย่อย ROTATE) แล้วเรียกใช้โปรแกรมย่อย RPLOT ต่อไป (โปรดดูรูปที่ 4-26 ประกอบ)

4.4.3 การสร้างโปรแกรมย่อย DASHLN

จุดประสงค์ใช้พลอตเส้นในลักษณะต่างๆ แบ่งได้เป็น 3 ชนิด คือ เส้นทึบ เส้นประ เส้นจุดไข่ปลา การทำงานของโปรแกรมนี้(โปรดดูรูปที่ 4-27 ประกอบ) เริ่มโดยการนำพารามิเตอร์ที่เข้ามา ซึ่งกำหนดจุดเริ่มต้นและจุดสิ้นสุดมาคำนวณหาความยาวของเส้นตรงที่จะทำการพลอตเมื่อได้ความยาวแล้ว ก็จะทำการตรวจลอบพารามิเตอร์ที่กำหนดลักษณะของเส้น เพื่อจะได้แยกไปทำการสร้างเส้นตรงตามต้องการ ในกรณีที่ต้องการเส้นทึบ ก็คือการเรียกใช้โปรแกรมย่อย PLOT เท่านั้น

```
CALL PLOT (XS,YS,3)
CALL PLOT (XE,YE,2)
```

ตัวแปร XS,YS แทนจุดเริ่มต้น

ตัวแปร XE,YE แทนจุดสิ้นสุด

ในกรณีที่ต้องการเส้นจุดไข่ปลาต้องคำนวณหาจำนวนจุดที่จะมีการพลอตดังสมการต่อไปนี้

NPTS คือจำนวนจุดที่ต้องมีการพลอต

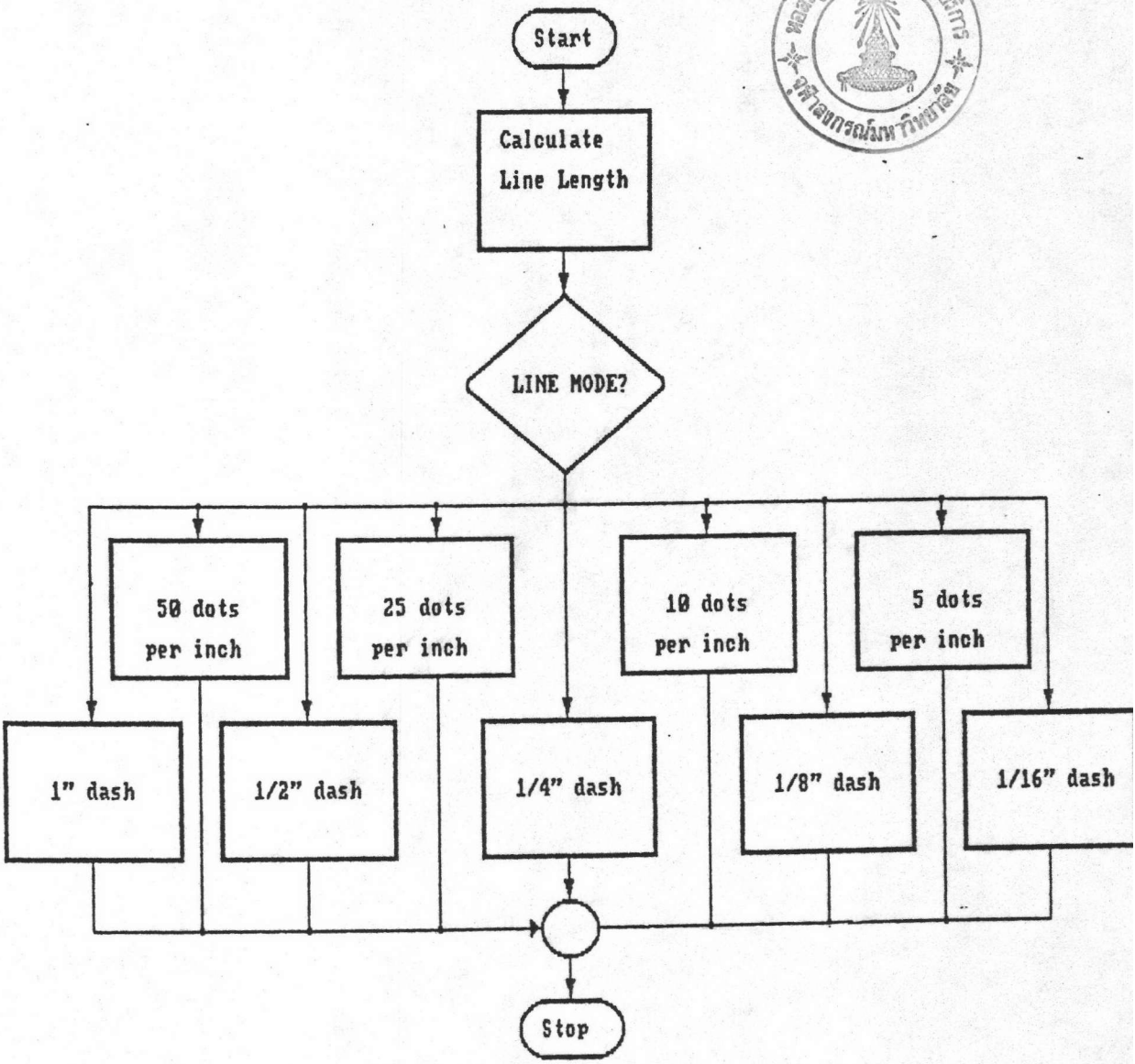
LENG คือความยาวของเส้นตรง

DPI คือจำนวนจุดต่อหนึ่งหน่วยความยาว

```
NPTS = LENG/DPI
```

แล้วทำการพลอตจุดทุกจุดจนครบ โดยระยะระหว่างจุดจะเท่ากับ 1/DPI

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 4-27 แสดงการทำงานของโปรแกรมย่อย DASHLN

ในกรณีที่ต้องการเส้นประ ต้องคำนวณหาว่าต้องใช้จำนวนรอยประเท่าไร ดังสมการต่อไปนี้

DASHLENG คือความยาวของรอยประ

SPACE คือความยาวของช่องไฟระหว่างรอยประ

SPACE = DASHLENG/2

เพราะฉะนั้นระยะห่างระหว่างรอยประหนึ่งไปยังอีกรอยประหนึ่งคือ DASHLENG + SPACE

NDASH คือจำนวนรอยประ

LENG คือความยาวเส้นตรง

NDASH = LENG/(DASHLENG+SPACE)

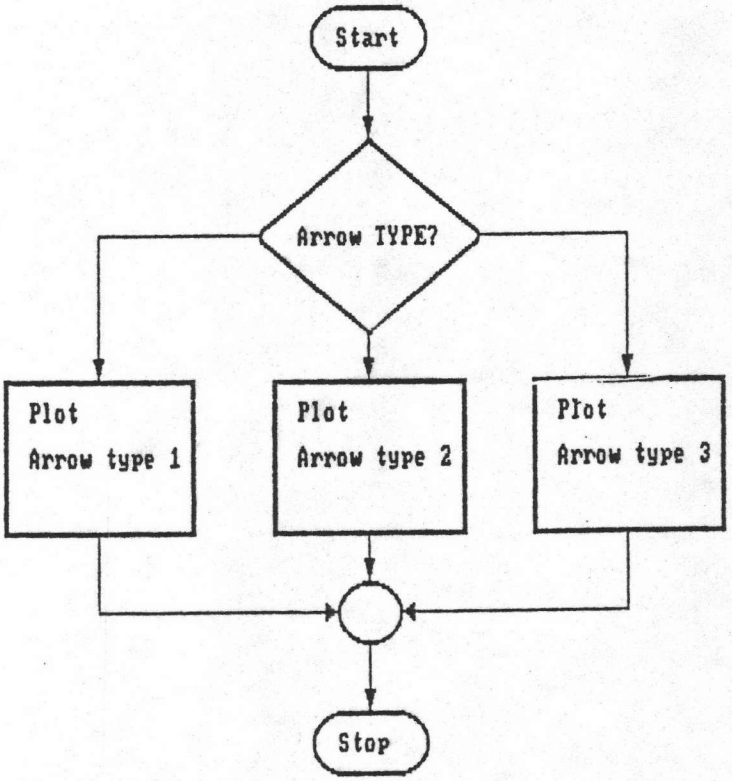
แล้วทำการพลอตรอยประจนครบความยาวของเส้นตรงที่กำหนดมา

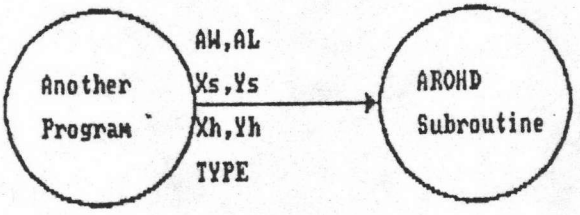
4.4.4 การสร้างโปรแกรมย่อย AROHD[๖]

จุดประสงค์ใช้พลอตรูปลูกศร การทำงานของโปรแกรมนี้ (โปรดดูรูปที่ 4-28 ประกอบ) เริ่มโดยการนำพารามิเตอร์ที่เข้ามา ซึ่งกำหนดจุดเริ่มต้นและจุดสิ้นสุดมาคำนวณหาความยาวของเส้นตรงที่จะทำการพลอต เมื่อได้ความยาวแล้วก็ทำการตรวจสอบว่าสั้นกว่าความยาวของหัวลูกครหรือไม่ ถ้าไม่ก็จะนำพารามิเตอร์ที่กำหนดรูปแบบของหัวลูกคร มาตรวจสอบเพื่อจะได้แยกไปทำการพลอตตามลักษณะที่ต้องการ

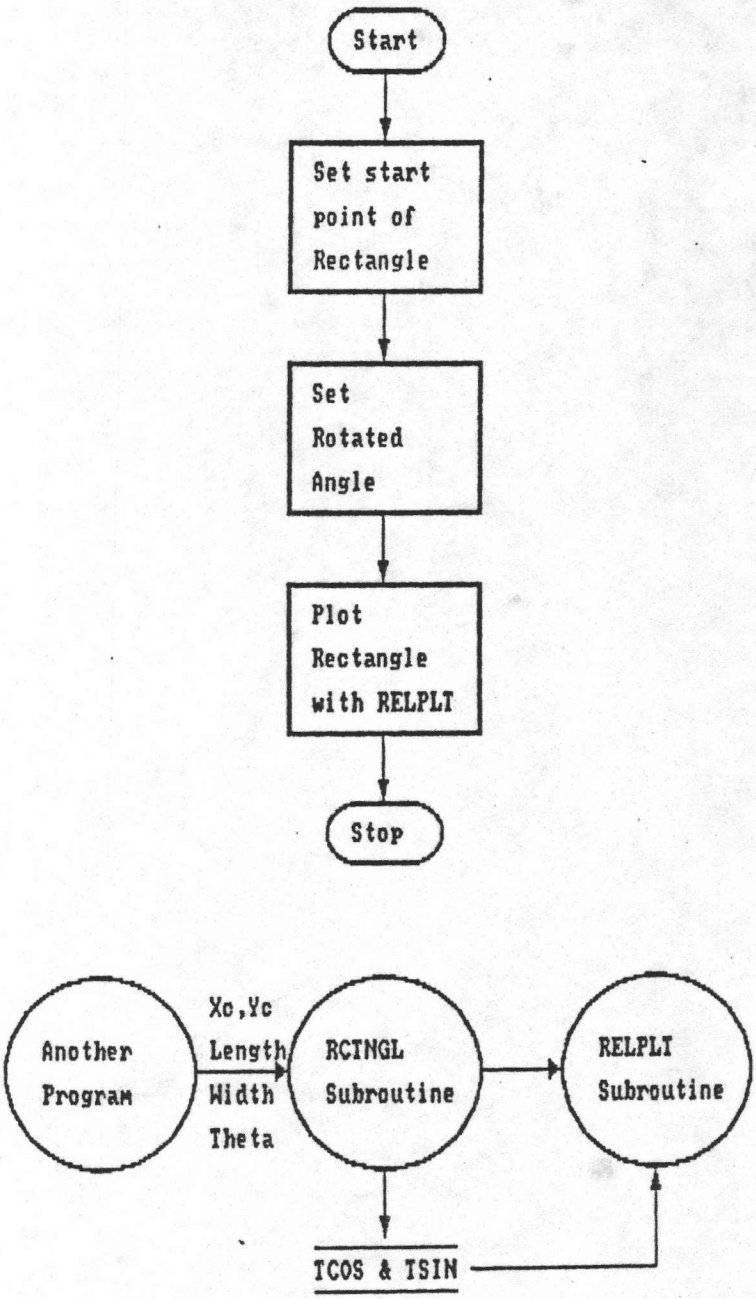
4.4.5 การสร้างโปรแกรมย่อย RCTNGL[๖]

จุดประสงค์ใช้พลอตรูปสี่เหลี่ยมผืนผ้า การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-29 ประกอบ) เริ่มโดยการนำพารามิเตอร์ที่เข้ามา ซึ่งกำหนดจุดล่างซ้ายของรูปสี่เหลี่ยมผืนผ้า ทำการย้ายตำแหน่งพลอตมายังจุดนี้แล้วเรียกใช้โปรแกรมย่อย ROTATE เพื่อกำหนดมุมที่จะหมุนรูปสี่เหลี่ยมผืนผ้า ตามพารามิเตอร์ที่กำหนดเข้ามา แล้วลากเส้นตรงผ่านโปรแกรมย่อย RELPLT จนครบสี่ด้านได้เป็นรูปสี่เหลี่ยมผืนผ้า





รูปที่ 4-28 แสดงการทำงานของโปรแกรมย่อย AROHD และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย AROHD กับส่วนอื่น



รูปที่ 4-29 แสดงการทำงานของโปรแกรมย่อย RCTNGL และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย RCTNGL กับส่วนอื่น

4.4.6 การสร้างโปรแกรมย่อย RCTBLK[6]

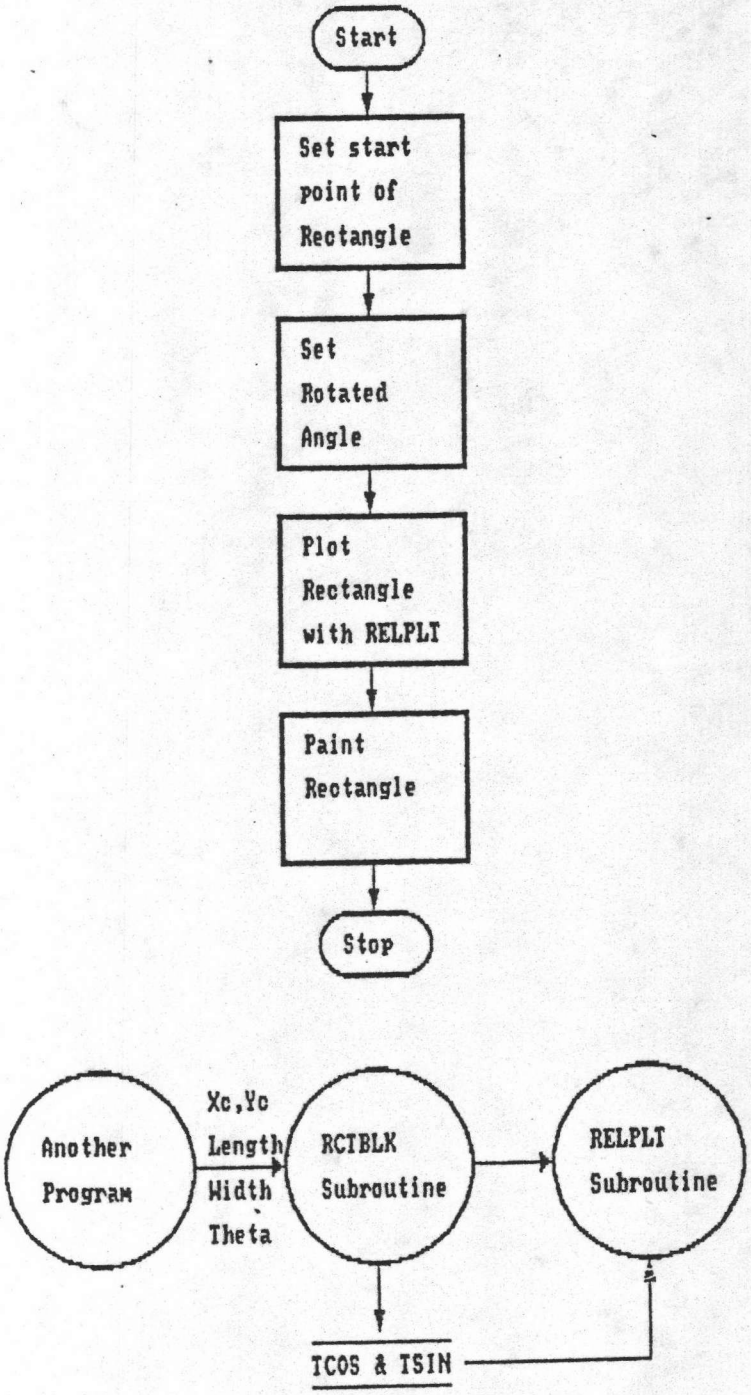
จุดประสงค์ใช้พลอตรูปสี่เหลี่ยมผืนผ้าทึบ การทำงานของโปรแกรมย่อยนี้ (โปรดดูรูปที่ 4-30 ประกอบ) เริ่มโดยการนำพารามิเตอร์ที่เข้ามา ซึ่งกำหนดจุดล่างซ้ายของรูปสี่เหลี่ยมผืนผ้า ทำการย้ายตำแหน่งพลอตมายังจุดนี้แล้วเรียกใช้โปรแกรมย่อย ROTATE เพื่อกำหนดมุมที่จะหมุนรูปสี่เหลี่ยมผืนผ้า ตามพารามิเตอร์ที่กำหนดเข้ามา แล้วลากเส้นตรงผ่านโปรแกรมย่อย RELPLT จนครบสี่ด้านได้เป็นรูปสี่เหลี่ยมผืนผ้า ทำการระบายภายในสี่เหลี่ยมจนทึบ

4.4.7 การสร้างโปรแกรมย่อย BAR

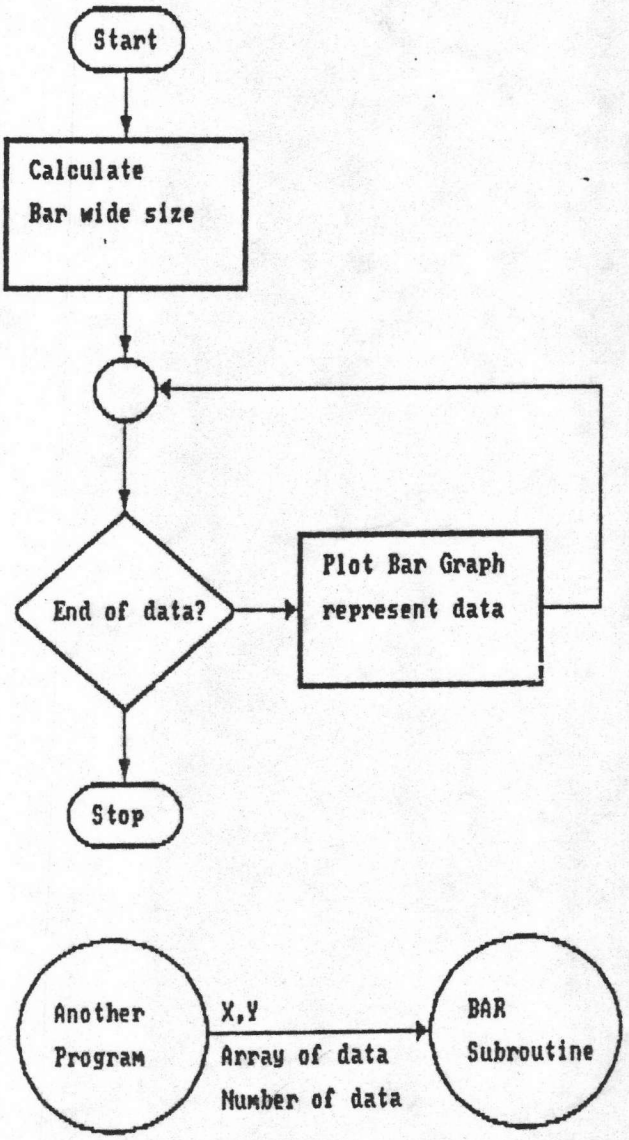
จุดประสงค์ใช้พลอตกราฟแท่ง โดยโปรแกรมนี้จะเริ่มต้นด้วยการคำนวณหาขนาดที่เหมาะสมของความกว้างของกราฟแท่งกราฟ แล้วเริ่มพลอตแท่งกราฟ โดยใช้ความสูงของแท่งกราฟแทนค่าของข้อมูล จนกระทั่งพลอตกราฟแท่งครบทุกค่าของข้อมูล ทั้งนี้จำนวนที่จะทำการพลอตจำกัดไปแค่ 20 ตัว ข้อมูลที่ใช้สำหรับโปรแกรมย่อยนี้จะมีชุดข้อมูล จำนวนข้อมูล และตำแหน่งเริ่มต้นที่มีการพลอตแท่งกราฟ (โปรดดูรูปที่ 4-31 ประกอบ)

4.4.8 การสร้างโปรแกรมย่อย BAR2

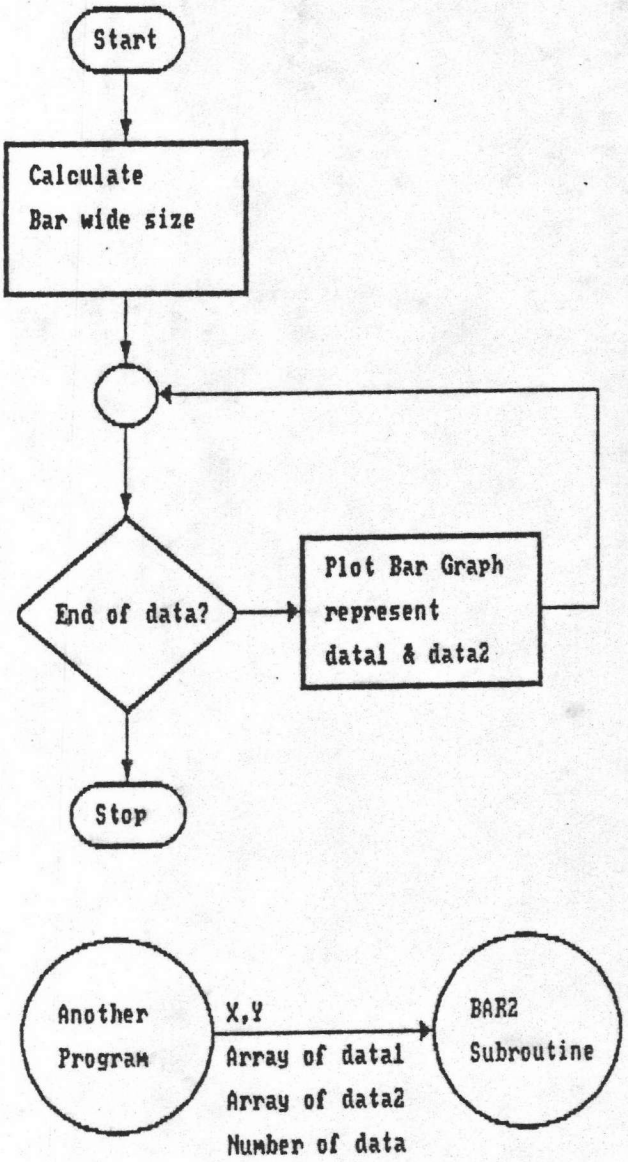
จุดประสงค์ใช้พลอตกราฟแท่ง โดยโปรแกรมนี้จะเริ่มต้นด้วยการคำนวณหาขนาดที่เหมาะสมของความกว้างของแท่งกราฟแล้วเริ่มพลอตกราฟ โดยความสูงของแท่งกราฟแทนค่าของข้อมูล จนกระทั่งพลอตกราฟแท่งครบทุกค่าของข้อมูล ทั้งนี้จำนวนที่จะทำการพลอตจะจำกัดไปแค่ 20 ตัว ข้อมูลที่ใช้สำหรับโปรแกรมย่อยนี้จะมีชุดข้อมูล และตำแหน่งเริ่มต้นที่มีการพลอตแท่งกราฟ (โปรดดูรูปที่ 4-32 ประกอบ)



รูปที่ 4 -30 แสดงการทำงานของโปรแกรมย่อย RCTBLK และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย RCTBLK กับส่วนอื่น



รูปที่ 4 -31 แสดงการทำงานของโปรแกรมย่อย BAR และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย BAR กับส่วนอื่น



รูปที่ 4-32 แสดงการทำงานของโปรแกรมย่อย BAR2 และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย BAR2 กับส่วนอื่น

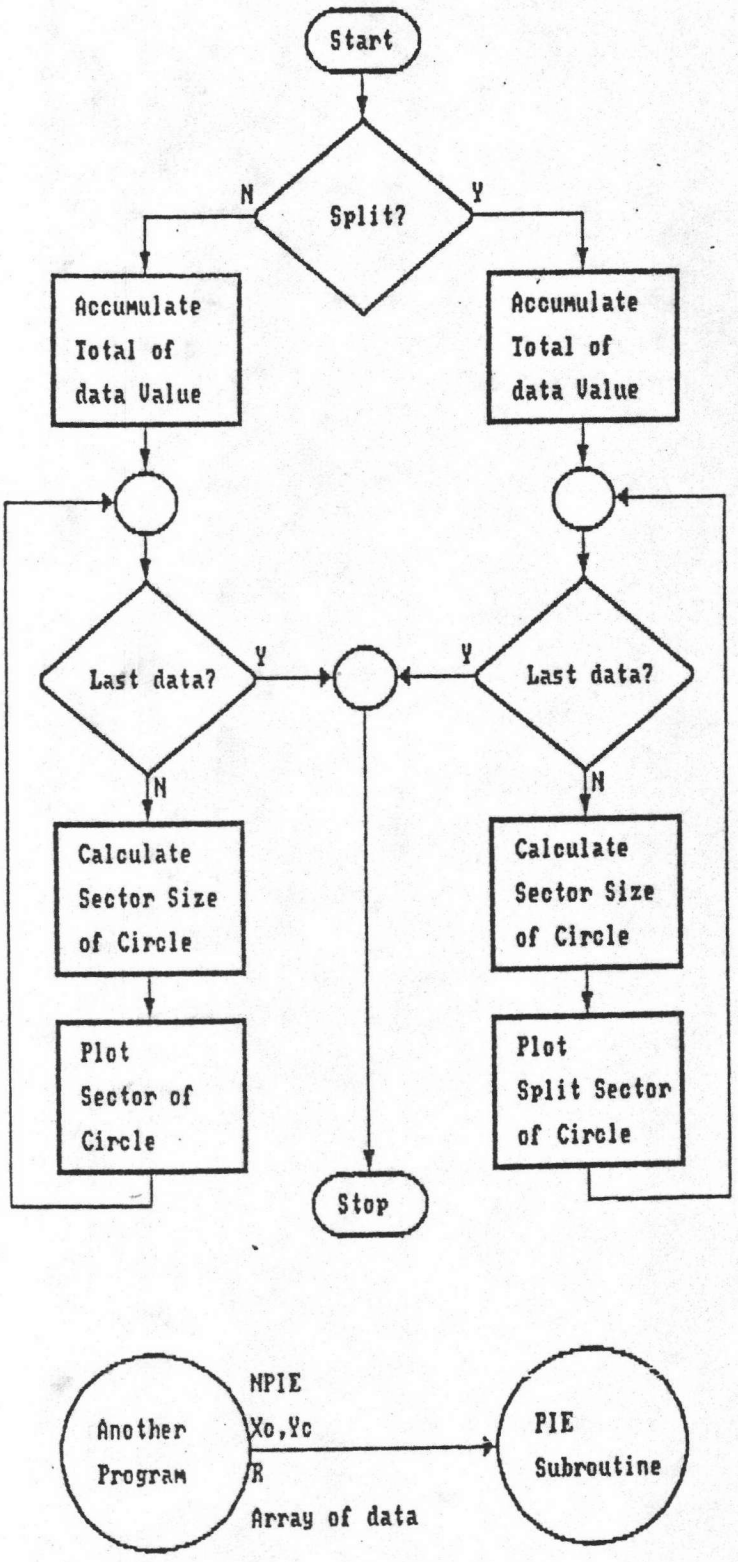
4.4.9 การสร้างโปรแกรมย่อย PIE

จุดประสงค์ใช้สร้างรูปกราฟวงกลม โปรแกรมย่อยนี้จะเริ่มทำงานโดยรวมค่าทั้งหมด ของทุกข้อมูลในชุดข้อมูลที่ส่งเข้ามา เพื่อใช้กำหนดขนาดของเลี้ยวของวงกลมใช้แทนข้อมูลแต่ละตัว แล้วตรวจสอบว่าต้องการรูปกราฟวงกลมโดยแต่ละเลี้ยวแยกกัน (split) หรือไม่ ถ้าใช่ เวลาพลอตแต่ละเลี้ยวของวงกลมจะพลอตแยกจากกัน การคำนวณขนาดของเลี้ยวของวงกลมคือ ขนาดของมุมนั่นเอง จะเป็นดังสมการข้างล่างนี้

$$Dangle = (360.*Di)/Dtotal$$

$$(i = 1,2,3,...,n \quad and \quad n <= 20)$$

เมื่อพลอตครบทุกข้อมูลแล้ว ก็จะพลอตค่าเปอร์เซนต์ของแต่ละข้อมูลกำกับไว้ให้ด้วย (โปรดดูรูปที่ 4-33 ประกอบ)



รูปที่ 4 -33 แสดงการทำงานของโปรแกรมย่อย PIE และแสดงความสัมพันธ์ระหว่างโปรแกรมย่อย PIE กับส่วนอื่น