บทที่ 4

ส่วนจัดการการวาดฟังก์ชั่นชาร์ต

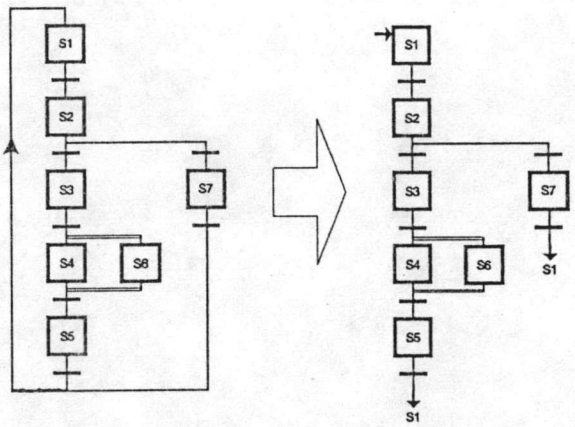
4.1 แนวความคิด

ภาษาฟังก์ชั่นชาร์ตเป็นภาษารูปภาพ การเขียนโปรแกรมจึงเป็นการวาดภาพขององค์ประกอบต่างๆ สิ่งที่ปรากฏต่อสายตาผู้ใช้ควรจะเป็นภาพของโปรแกรมที่ผู้ใช้ได้เขียนขึ้น แต่สำหรับระบบแล้วภาพที่ปรากฏบนจอภาพนั้นเป็นสิ่งที่ยากจะทำความเข้าใจได้ จึงต้องหาทางที่จะให้ระบบสามารถเข้าใจความหมายของภาพเหล่านั้นได้ ในการวาดภาพองค์ประกอบต่างๆ หากมองที่ผลลัพธ์เป็นหลักก็อาจกล่าวได้ว่าเป็นการจัดเรียงภาพองค์ประกอบเหล่านั้น โดยระบบได้จัดเตรียมภาพองค์ประกอบพื้นฐานต่างๆไว้ให้ และผู้ใช้เพียงแต่หยิบไปวางเรียงให้เป็นรูปร่างที่ต้องการ ตามแนวความคิดนี้มีข้อดีอยู่ 2 ประการ ได้แก่ ประการแรกช่วยให้เขียนโปรแกรมได้เร็วขึ้น ประการที่สองเป็นการปูทางสำหรับระบบในการทำความเข้าใจแผนภาพ กล่าวคือหากมีการออกแบบโครงสร้างข้อมูลที่ดีแล้วก็จะช่วยให้ระบบเข้าใจแผนภาพได้ ซึ่งจะกล่าวในหัวข้อถัดไป

งานหนึ่งที่จะต้องทำก็คือการตรวจสอบว่าชาร์ตที่ผู้ใช้เขียนขึ้นถูกต้องตามหลักภาษาหรือไม่ งานนี้อาจจะกระทำในขั้นตอนการแปลก็ได้ แต่เมื่อพิจารณาดูแล้ว เห็นสมควรให้ทำการตรวจสอบหลักภาษาทุกๆครั้งที่มีการวาดองค์ประกอบใดๆ มองโดยผิวเผินอาจเห็นว่าทำให้เสียเวลาในการวาดมากขึ้น และเสียเวลามากกว่าการตรวจสอบและรายงานผลในครั้งเดียว แต่หากวิเคราะห์ดูแล้วจะเห็นว่าในการป้อนโปรแกรมหรือวาดภาพองค์ประกอบลงไปนั้น มิได้เป็นการทำงานอย่างต่อเนื่องทุกเสี้ยววินาทีจะมีการหยุดพักหรือหยุดคิดอยู่บ้าง หรืออย่างน้อยช่วงเวลาระหว่างการป้อนแต่ละครั้งก็ห่างกันเป็นหน่วยหลายร้อยมิลลิวินาทีหรือวินาทีอยู่แล้ว ซึ่งจะเห็นว่าเป็นช่วงเวลาที่มากพอสมควรสำหรับการประมวลผลของเครื่อง ดังนั้นหากทำการตรวจสอบหลักภาษาในช่วงเวลาเหล่านี้ ก็อาจกล่าวได้ว่าเป็นการประหยัดเวลาในภายหลังเสียด้วยซ้ำ และยังเป็นการใช้เวลาอย่างคุ้มค่าเต็มเม็ดเต็มหน่วย อีกทั้งยังเป็นการดีต่อผู้ใช้ที่จะได้แก้ไขข้อผิดพลาดในทันทีที่เขียน

ในกราฟฟิกเอดิเตอร์นี้ได้ปรับปรุงสัญลักษณ์ของการเชื่อมต่อเส้นทางการควบคุมระยะไกล หรือการกระโดด ทั้งนี้เนื่องจากสัญลักษณ์เดิมซึ่งใช้เส้นลากเชื่อมต่อโดยตรงจากใต้ทรานสิชั่นที่จะกระโดด ไปยังด้านบนของสเต็ปที่จะกระโดดไป ใช้เนื้อที่บนจอภาพมาก และในบางครั้งดูระเกะระกะ จึงได้เปลี่ยนไปใช้ลูกศรแทน ดังรูปที่ 4.1 กล่าวคือ ใต้ ทรานสิชั่นตัวที่จะกระโดดจะเขียนลูกศรชี้ลง และใต้ลูกศรจะระบุชื่อของสเต็ปที่กระโดดไปหา และที่สเต็ปปลายทางด้านข้างซ้ายบนจะเขียนลูกศรชี้เข้าหาสเต็ปนั้น แต่จะไม่เขียนหมายเลขสเต็ประบุไว้ ทั้งนี้เพราะได้ออกแบบให้สเต็ปรับการ

กระโดดมาได้สูงสุดจาก 4 เส้นทาง แต่ทางด้านต้นทางการกระโดดนั้น จะกระโดดไปยัง สเต็ปได้เพียง 1 สเต็ปเท่านั้น ในการเขียนการกระโดดผู้ใช้เพียงแต่ระบุบอกไว้ที่ต้นทางว่าจะกระโดดไปยังสเต็ปใด ระบบก็จะค้นหาสเต็ปนั้นและเขียนลูกศรให้ทั้งต้นทางและปลายทาง

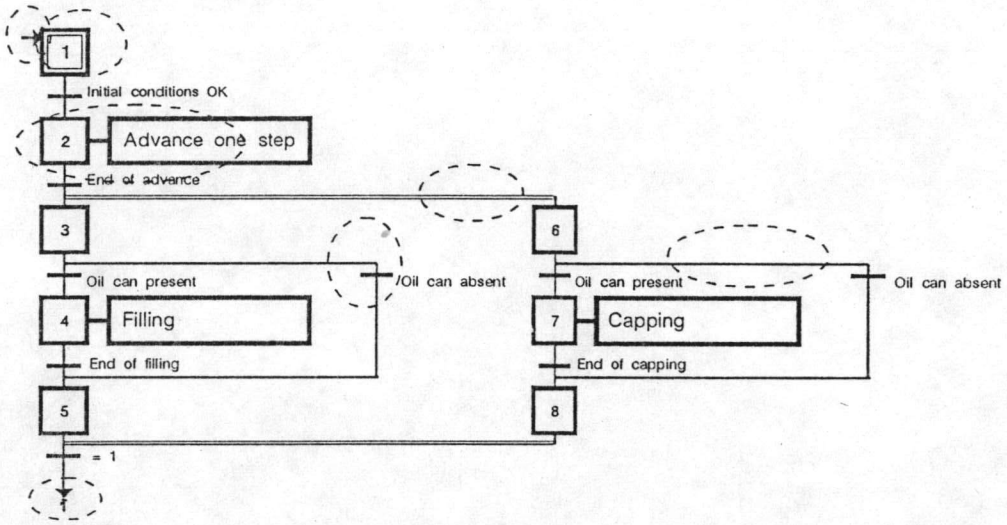


รูปที่ 4.1 : แก้ไขภาพการกระโดด

นอกจากนั้น ได้ออกแบบให้สเต็ป 1 สเต็ปควบคุมเอาต์พุต 1 เอาต์พุต กล่าวคือเมื่อสเต็ปใดแอ็กทีฟ ก็จะมีเอาต์พุต 1 เอาต์พุตที่แอ็กทีฟ นอกเสียจากว่าสเต็ปนั้นมิได้ควบคุมเอาต์พุตใด เอาต์พุตที่สเต็ปควบคุมก็คือหน้าสัมผัสของรีเลย์นั่นเอง ซึ่งได้ออกแบบให้สามารถควบคุมได้ใน 2 ลักษณะ คือเอาต์พุตแบบปกติ และเอาต์พุตแบบคงค่าไว้ (Hold) นอกจากนั้นเอาต์พุตยังสามารถเป็นอุปกรณ์ประเภทตัวตั้งเวลา (Timer) และวงจรนับ (Counter) ได้ โดยเมื่อผู้ใช้ระบุไปขณะวาดว่าเป็นตัวตั้งเวลา ระบบก็จะถามค่าเวลาที่ต้องการตั้ง หรือถ้าระบุไปว่าเป็นตัวนับ ระบบก็จะถามว่าจะนับเท่าไร ส่วนกรณีทรานสิชั่นก็จะรับอินพุตจาก 1 อินพุต ซึ่งก็เป็นหน้าสัมผัสของรีเลย์เช่นเดียวกัน อาจจะเป็นหน้าสัมผัสของรีเลย์ปกติ หรือเป็นหน้าสัมผัสของตัวตั้งเวลา หรือเป็นหน้าสัมผัสของวงจรนับก็ได้ และสามารถจะสั่งให้นำนิเสธ (Invert) ของหน้าสัมผัสเหล่านั้นมาใช้ได้

4.2 โครงสร้างข้อมูล

เนื่องจากการมองการวาดเป็นการจัดเรียงภาพ จึงออกแบบโครงสร้างข้อมูลให้เหมาะสมกับโครงสร้างรูปภาพ กล่าวคือภาพแต่ละภาพที่นำมาวางต่อกันนั้น จะมีที่เก็บข้อมูลเฉพาะของตัว เพื่อใช้เก็บลักษณะของตัวและข้อมูลที่จำเป็น ภาพที่นำมาวางเรียงต่อกันนั้นก็คือองค์ประกอบพื้นฐานนั่นเอง ในรูปที่ 4.2 แสดงตัวอย่างชาร์ตและตารางที่ 4.1 แสดงการนำองค์ประกอบพื้นฐานมาพิจารณาว่าต้องการข้อมูลอะไรบ้าง



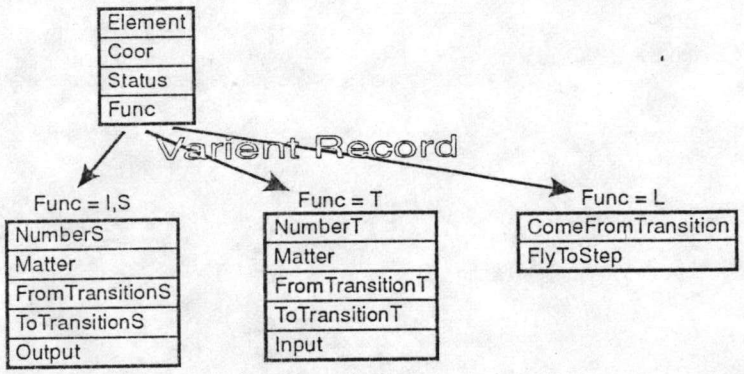
รูปที่ 4.2 : ตัวอย่างชาร์ต

Picture	Name	Calling	Information needed
	Initial step	Initial step	Number, Comment, Output
	Step	Step	Number, Comment, Output
	Transition	Transition	Number, Comment, Input
	Selection element	Branch	-
	Simultaneous element	Double	-
	Direct link	Wire	-
→ ↓ No.	Jumping element	Link	To step From step

ตารางที่ 4.1 : ข้อมูลที่ต้องการสำหรับแต่ละองค์ประกอบ

การออกแบบโครงสร้างข้อมูลนี้ จะต้องคิดเผื่อไปถึงงานในขั้นตอนการแปลด้วยเพื่อให้ข้อมูลที่ได้สามารถนำไปแปลได้โดยง่าย ในการแปลจะต้องวิเคราะห์ภาพต่างๆที่เรียงต่อกันอยู่ สร้างความสัมพันธ์ของภาพที่เชื่อมต่อกัน ขั้นต่อไปจึงเป็นการวิเคราะห์หาความสัมพันธ์โดยรวมของระบบเพื่อแปลเป็นสมการบูลลีน การวิเคราะห์หาความสัมพันธ์รอบข้างเป็นการสร้างโครงข่ายความสัมพันธ์เชื่อมโยงองค์ประกอบต่างๆ เข้าด้วยกัน เพื่อมิให้รูปภาพเป็นเพียงแค่รูปภาพ ผลที่ได้จะเป็นรูปภาพที่พร้อมจะแนะนำตัวเองให้ผู้อื่นรู้จักได้ ณ จุดนี้ได้พิจารณาที่จะนำงานขั้นตอนแรกของการแปลนี้มาบรรจุไว้ในโมดูลการวาด ด้วยเหตุผลเช่นเดียวกับการพิจารณานำการวิเคราะห์หลักภาษามาบรรจุไว้ในโมดูลการวาด ซึ่งจะช่วยให้ประหยัดเวลาในขั้นตอนการแปล อีกทั้งยังเป็นการใช้เวลาที่เปล่าประโยชน์ในช่วงการรอการวาดรูปต่างๆ ให้ได้ใช้ประโยชน์เต็มที่อีกด้วย การวิเคราะห์ความสัมพันธ์รอบข้างเป็นการสร้างความสัมพันธ์ระหว่างสเต็ปและทรานสิชั่นที่เชื่อมต่อกัน ในการที่จะหาความสัมพันธ์ได้นั้น จะต้องพิจารณาดูรอบข้าง เพื่อค้นหาสเต็ปหรือทรานสิชั่นที่เชื่อมโยงต่อกันอยู่และจดจำไว้ จึงได้ออกแบบโครงสร้างข้อมูลให้สามารถจดจำได้ว่ามีสเต็ปหรือทรานสิชั่นใดที่เชื่อมต่ออยู่รอบข้าง

จากที่กล่าวมาทั้งหมดจึงนำไปใช้ออกแบบโครงสร้างข้อมูลได้ดังรูปที่ 4.3 โครงสร้างข้อมูลแบ่งออกเป็น




```
Coordinate        = array[1..2]    of byte;
NameCoor          = Record
                        Name : String[3];
                        Coor : Coordinate;
                    end;
GStr              = string[8];
GStr3             = array[1..3] of string[4];
PointToCells      = array[1..4] of Coordinate;
Cell =  record                                        {55 byte/cell}
          Element  : String[3];
          Coor     : Coordinate;
          Status   : Boolean;
          case FUNC:char of
               I,S : (NumberS:string[3]; MatterS:string[11];
                      FromTransitionS,ToTransitionS :PointToCells;
                      Output:GStr3);
               M   : (NumberM:string[3]; MatterM:string[11];
                      FromTransitionM,ToTransitionM :PointToCells;
                      Chart:GStr);
               T   : (NumberT:string[3]; MatterT:string[11];
                      FromStep,ToStep              :PointToCells;
                      Input:GStr3);
               L   : (ComeFromTransition:array[1..4] of NameCoor;
                      FlyToStep:NameCoor);
        end;
```

รูปที่ 4.3 : โครงสร้างข้อมูล

2 ส่วน ส่วนแรกเป็นข้อมูลที่ใช้ร่วมกัน ในส่วนหลังเป็นข้อมูลเพิ่มเติมที่แต่ละองค์ประกอบต้องการ ซึ่งใช้โครงสร้างเป็น Variant record โดยใช้ Func เป็นตัวเลือกข้อมูลเพิ่มเติมซึ่งจะมีเฉพาะ Initial step, Step, Transition และ Link สามารถอธิบายโครงสร้างในส่วนต่างๆ ได้ดังนี้

Element : ชื่อของภาพแต่ละภาพที่นำมาเรียงต่อกัน

Coor : เก็บตำแหน่งของภาพบนชาร์ต

Status : ออกแบบเผื่อไว้ใช้งานในอนาคต

Func : ประเภทของภาพที่มีโครงสร้างข้อมูลเพิ่มเติม ได้แก่

I : Initial Step

S : Step

T : Transition

L : Link

ข้อมูลเพิ่มเติมสำหรับ I, S

NumberS : หมายเลขของสเต็ป

MatterS : หมายเหตุ

FromTransitionS : ตำแหน่ง (Coor) ของทรานสิชั่นที่เชื่อมต่อทางด้านบนมี 4 ฟิลด์

ToTransitionS : ตำแหน่งของทรานสิชั่นที่เชื่อมต่อทางด้านล่างมี 4 ฟิลด์

Output : หมายเลขของหน้าสัมผัสเอาต์พุต (Output contact) มี 3 ฟิลด์

ข้อมูลเพิ่มเติมสำหรับ T

NumberT : หมายเลขของทรานสิชั่น

MatterT : หมายเหตุ

FromStep : ตำแหน่งของสเต็ปที่เชื่อมต่อทางด้านบน

ToStep : ตำแหน่งของสเต็ปที่เชื่อมต่อทางด้านล่าง

Input : หมายเลขของหน้าสัมผัสอินพุต (Input contact)

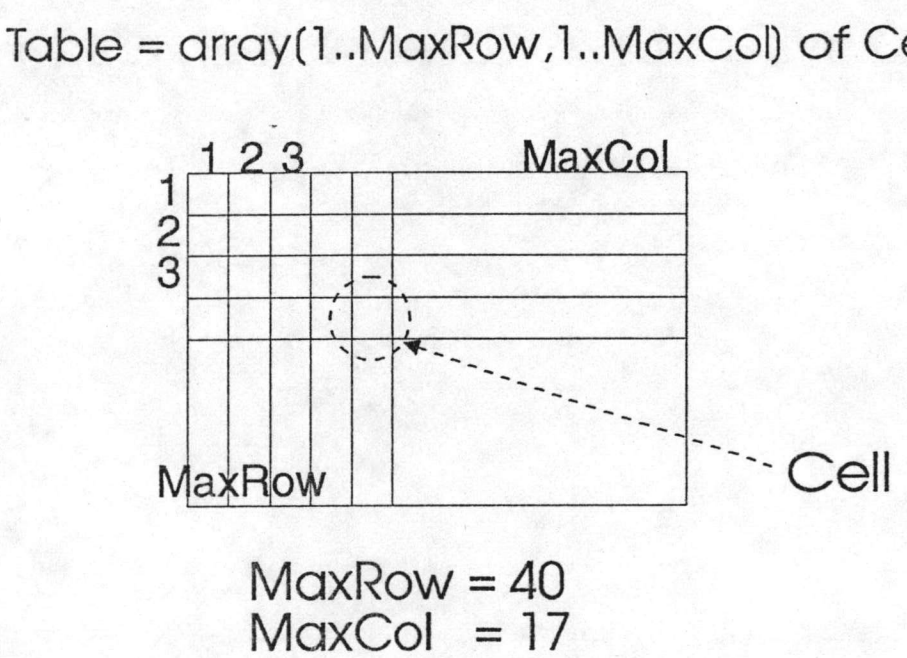
ข้อมูลเพิ่มเติมสำหรับ L

ComeFromTransition : หมายเลขและตำแหน่งของทรานสิชั่นที่กระโดดออกมามี 4 ฟิลด์

FlyToStep : หมายเลขและตำแหน่งของสเต็ปที่เป็นเป้าหมายการกระโดดมี 1 ฟิลด์

โครงสร้างที่สนับสนุนการสร้างความสัมพันธ์รอบข้างได้แก่ FromTransitionS, ToTransitionS, FromStep, ToStep ซึ่งได้ออกแบบให้เก็บได้ 4 ตำแหน่ง ดังนั้นในการวาดชาร์ตจะแตกกิ่งได้สูงสุด 4 กิ่ง ส่วนในกรณีของการกระโดดสามารถกระโดดไปยังสเต็ปเดียวกันได้สูงสุดจาก 4 ทรานสิชั่น และแต่ละทรานสิชั่นจะกระโดดไปหาสเต็ปได้เพียง 1 สเต็ป

จากโครงสร้างของแต่ละเซลล์ เมื่อนำมารวมกันก็จะได้โครงสร้างของชาร์ต รูปแบบของโครงสร้างที่จะนำมาใช้ พิจารณาจากโครงสร้าง 2 รูปแบบ ได้แก่ แบบแรกเป็นแถวลำดับ (Array) และแบบที่สองใช้ตัวชี้ (Pointer) ในแบบแรกเป็นแถวลำดับจะเก็บข้อมูลในลักษณะเป็นตาราง กล่าวคือมีลักษณะเหมือนเป็นเงาของภาพชาร์ต โดยทุกๆช่องจะมีที่เก็บข้อมูลที่แน่นอนสามารถอ้างอิงได้ทันที แบบที่สองใช้ตัวชี้ โดยจัดเก็บข้อมูลของชาร์ตเฉพาะตำแหน่งที่มีภาพและนำข้อมูลเหล่านั้นมาเชื่อมโยงกันให้มีลักษณะเหมือนการจัดเรียงภาพ โครงสร้างแบบนี้จะใช้เนื้อที่จัดเก็บข้อมูลน้อยกว่าแบบแรก แต่ขนาดของเซลล์จะใหญ่กว่า และในการเข้าถึงข้อมูลจะคล่องตัวน้อยกว่าแบบแรก และเสียเวลามากกว่า ในงานวิจัยนี้จึงเลือกใช้โครงสร้างแบบแถวลำดับโดยมองที่ความสะดวกรวดเร็วในการใช้งานเป็นหลัก

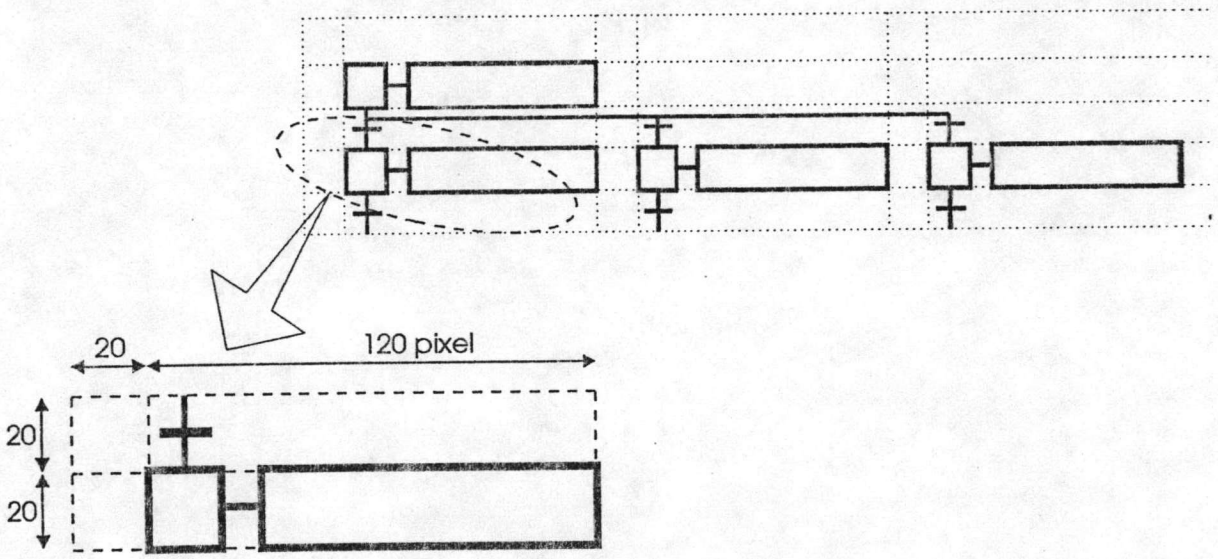


รูปที่ 4.4 : โครงสร้างของชาร์ต

ขนาดของชาร์ตจะขึ้นกับค่า MaxRow และ MaxCol ซึ่งเป็นค่าคงที่กำหนดไว้ในตอนต้นของโปรแกรม ซึ่งได้ออกแบบไว้ให้ชาร์ตมีขนาด 40 แถว x 17 คอลัมน์ หรือประมาณ 4 เท่าของจอภาพ (1 จอภาพจะเขียนได้ 20 แถว x 8 คอลัมน์) รายละเอียดในส่วนนี้จะกล่าวถึงในหัวข้อถัดไป

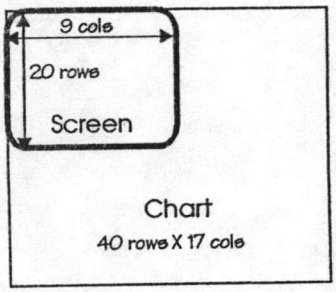
4.3 เคอร์เซอร์

เคอร์เซอร์คือตัวชี้ตำแหน่งปัจจุบันที่ผู้ใช้กำลังให้ความสนใจอยู่ เนื่องจากในการวาดภาพจะกระทำในลักษณะการจัดเรียงภาพ จึงออกแบบให้เคอร์เซอร์เคลื่อนไปตามช่องต่างๆที่จะนำภาพมาวางลง ขนาดและโครงสร้างของช่องพิจารณาได้ดังรูปที่ 4.5



รูปที่ 4.5 : การหาโครงสร้างและขนาดของภาพ

จะเห็นว่าช่องจะมีขนาดไม่คงที่ในแนวตั้ง โดยเป็นช่องเล็กสลับกับช่องใหญ่ เพราะสเต็ปจะต้องใช้พื้นที่ในการแสดงรายละเอียด จึงได้ออกแบบหน้าจอภาพให้คอลัมน์ที่ 1 มีขนาด 20 จุด (pixel) คอลัมน์ที่ 2 มีขนาด 120 จุด คอลัมน์ที่ 3 มีขนาด 20 จุด เช่นนี้สลับกันไป และแต่ละแถวมีขนาด 20 จุด จึงทำให้ 1 จอภาพมีขนาด 20 แถว x 9 คอลัมน์



รูปที่ 4.6 : ขนาดของชาร์ต

รูปร่างของเคอร์เซอร์ออกแบบให้เป็นสี่เหลี่ยมจัตุรัสโปร่งใสขนาด 20x20 จุด และมีเงาเพื่อให้เคอร์เซอร์ดูเด่นเป็นที่สังเกตได้ง่าย การเคลื่อนที่ของเคอร์เซอร์ได้ออกแบบให้ใช้ปุ่มลูกศรทั้ง 4 ปุ่ม ควบคุมการเคลื่อนที่ขึ้นบน, ลงล่าง, ไปทางซ้าย, ไปทางขวา ครั้งละ 1 ช่อง ซึ่งมีรายละเอียดดังแสดงในรูปที่ 4.7

```
Procedure CursorControl(Direction:Char);
begin
  case Direction of
   #77 : begin
              If Odd(Col) then inc(x,20)
                          else inc(x,120);
              inc(col);   CheckBorder; MoveRel(x,y) ; Cursor; { right }
         end;
   #75 : begin
              If Odd(Col) then Dec(x,120)
                          else Dec(x,20);
              dec(col);   CheckBorder; MoveRel(x,y) ; Cursor; { left  }
         end;
   #72 : begin
              Dec(y,20);
              dec(row);   CheckBorder; MoveRel(x,y) ; Cursor; { up    }
         end;
   #80 : begin
              inc(y,20);
              inc(row);   CheckBorder; MoveRel(x,y) ; Cursor; { down  }
         end;
  end;
end;
```

รูปที่ 4.7 : โปรแกรมย่อยควบคุมการเคลื่อนที่ของเคอร์เซอร์

โปรแกรมย่อยนี้ จะรับทิศทางการเคลื่อนที่ซึ่งผู้ใช้กดปุ่มมาจากโปรแกรมหลัก นำมาเลือกวิธีปฏิบัติ เช่น ในกรณีลูกศรขวาถูกกด (#77) ก็จะตรวจสอบว่าอยู่ในคอลัมน์ใด ถ้าอยู่ในคอลัมน์เลขคี่ ก็จะสั่งให้เพิ่มค่าตำแหน่งไปทางขวา 20 จุด แต่ถ้าอยู่ในคอลัมน์เลขคู่ก็จะสั่งให้เพิ่มค่าตำแหน่งไปทางขวา 120 จุด จากนั้นเพิ่มค่าคอลัมน์อีก 1 และตรวจสอบว่าจะทำให้เคอร์เซอร์ตกขอบจอหรือไม่ ถ้าตกขอบก็จะปรับแต่งค่าให้ใหม่ (CheckBorder) จากนั้นย้ายตำแหน่งปัจจุบันไปตามค่าที่ระบุ (MoveRel(x,y)) และเขียนภาพเคอร์เซอร์ ในการควบคุมตำแหน่งของเคอร์เซอร์และการอ้างอิงตำแหน่งของภาพมีตัวแปรหลายตัวที่เกี่ยวข้องได้แก่

x,y : เก็บค่าตำแหน่งปัจจุบันของเคอร์เซอร์ในแนวแกนนอน (X) และในแนวแกนตั้ง (Y) หน่วยเป็นจุด (pixel)

row : เก็บค่าแถวปัจจุบันของเคอร์เซอร์

col : เก็บค่าคอลัมน์ปัจจุบันของเคอร์เซอร์

การเคลื่อนที่ของเคอร์เซอร์ เมื่อไปถึงขอบภาพนั้นได้ออกแบบให้วนกลับไปยังอีกด้าน เช่น เมื่อเคอร์เซอร์เคลื่อนไปทางขวาจนตกขอบก็จะไปปรากฏทางขอบซ้ายของจอภาพ เมื่อเคอร์เซอร์เคลื่อนไปจนสุดด้านบน ก็จะไปปรากฏทางด้านล่างของจอภาพ เป็นต้น และเพื่อช่วยให้ผู้ใช้สามารถเคลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการได้เร็วขึ้น จึงได้ออกแบบให้ เมื่อผู้ใช้กดปุ่ม Home เคอร์เซอร์จะเคลื่อนไปอยู่ที่มุมซ้ายของจอภาพ และเมื่อกดปุ่ม End เคอร์เซอร์จะเคลื่อนที่ไปอยู่กลางจอภาพ (เนื่องจากเป็นตำแหน่งบนจอภาพที่ไกลจากตำแหน่ง Home มากที่สุด เพราะเคอร์เซอร์เคลื่อนเป็นวงรอบจอภาพ) การเคลื่อนย้ายเคอร์เซอร์แบบเร็วนี้ใช้การปรับค่าตัวแปร x,y,row,col ให้ทันกาล (update) และวาดเคอร์เซอร์ที่ตำแหน่งใหม่พร้อมกับลบภาพเคอร์เซอร์ที่ตำแหน่งเก่า เนื่องจากชาร์ตมีขนาดประมาณ 4 เท่าของจอภาพ จึงสร้างคำสั่งที่ช่วยย้ายมุมมอง (pan) ของจอภาพไปยังตำแหน่งใหม่บนชาร์ต ใน 4 ทิศทาง ได้แก่ ย้ายขึ้นบน กด PgUp, ย้ายลงล่าง PgDn, ย้ายไปทางซ้าย (Ctrl และลูกศรซ้าย), ย้ายไปทางขวา (Ctrl และลูกศรขวา) การย้ายมุมมองจะกระทำครั้งละครึ่งจอภาพ กล่าวคือในแนวตั้งครั้งละ 10 แถว และในแนวนอนครั้งละ 4 คอลัมน์ และจะมีการย้ายมุมมองพิเศษอีก 2 กรณี ได้แก่ การย้ายเคอร์เซอร์ไปที่ตำแหน่งมุมซ้ายบนสุดของชาร์ต (กด Ctrl Home) และการย้ายเคอร์เซอร์ไปที่ตำแหน่งมุมขวาล่างสุดของชาร์ต (กด Ctrl End)

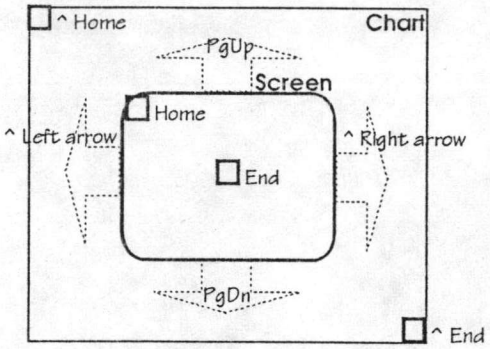
ในการอ้างอิงตำแหน่งบนชาร์ตนั้น นอกจากตัวแปรที่ได้กล่าวถึงไปแล้วยังมีตัวแปรที่สำคัญอีก ดังนี้

BeginRow : เก็บตำแหน่งแถวบนสุดที่แสดงอยู่บนจอภาพ

BeginCol : เก็บตำแหน่งคอลัมน์ซ้ายสุดที่แสดงอยู่บนจอภาพ

MaxScreenRow : ค่าคงที่ เก็บค่าจำนวนแถวสูงสุดที่แสดงได้ใน 1 จอภาพ (=20)

MaxScreenCol : ค่าคงที่ เก็บค่าจำนวนคอลัมน์สูงสุดที่แสดงได้ใน 1 จอภาพ (=9)



รูปที่ 4.8 : การควบคุมเคอร์เซอร์และการย้ายมุมมอง

โปรแกรมย่อยที่ควบคุมการย้ายมุมมองบนจอภาพแสดงไว้ในรูปที่ 4.9 ซึ่งหลังจากระบุทิศทางการย้ายมุมมองแล้ว ก็จะคำนวณหาตำแหน่งตั้งต้น และเรียกใช้โปรแกรมย่อย ShowScreen ซึ่งจะทำหน้าที่หยิบรูปภาพขึ้นไปแสดงบนจอ โดยมีขั้นตอนดังนี้

- หาขอบเขตของภาพที่จะนำขึ้นแสดงบนจอ
- หยิบภาพขึ้นแสดงทีละภาพ โดยคำนวณตำแหน่งที่จะแสดงบนจอภาพก่อน

```
Procedure MoveScreen(Direction:SearchDirection);
Begin
     Case Direction of
          Up    : begin
                       BeginRow:=BeginRow-10;
                       If (BeginRow<1) then BeginRow:=1;
                       ShowScreen;
                  end;
          Down  : begin
                       BeginRow:=BeginRow+10;
                       If (BeginRow+MaxScreenRow-1)>MaxRow then
                           BeginRow:=MaxRow-MaxScreenRow+1;
                       ShowScreen;
                  end;
          Left  : begin
                       BeginCol:=BeginCol-4;
                       If (BeginCol<1) then BeginCol:=1;
                       ShowScreen;
                  end;
          Right : begin
                       BeginCol:=BeginCol+4;
                       If (BeginCol+MaxScreenCol-1)>MaxCol then
                           BeginCol:=MaxCol-MaxScreenCol+1;
                       ShowScreen;
                  end;
          end;
End;
```

รูปที่ 4.9 : โปรแกรมย่อยควบคุมการย้ายมุมมองบนจอภาพ

4.4 การวาดรูป

ในการวาดรูปผู้ใช้จะต้องเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ จากนั้นเลือกภาพจากเมนู (Menu) ภาพก็จะปรากฏที่ตำแหน่งของเคอร์เซอร์ ก่อนที่ภาพจะปรากฏขึ้นมานั้นระบบจะทำงาน 3 ขั้นตอนดังนี้

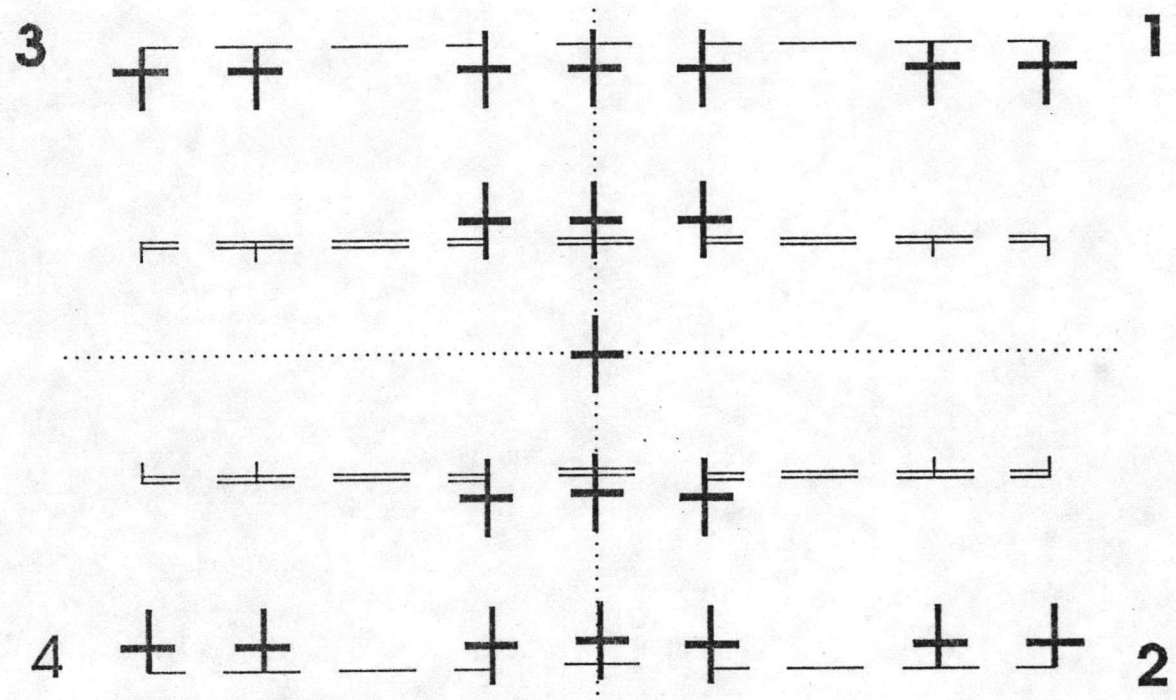
1. ตรวจสอบกฎเกณฑ์
2. วาดภาพ
3. ปรับข้อมูลในตารางให้ทันกาล

การกล่าวถึงรายละเอียดการวาดภาพองค์ประกอบต่างๆ จะขอนำไปกล่าวถึงในหัวข้อถัดไป ภาพองค์ประกอบทั้งหมด แสดงไว้ในตารางที่ 4.2

INITIAL	I						
STEP	S						
TRANSITION	T00	T01	T02	T03	T04	T05	T06
		T11	T12	T13	T14	T15	T16
		T21	T22	T23	T24	T25	T26
WIRE	W						
LINK	L1	L2 No.					
BRANCH	B1	B2 RIGHT	B3	B4 LEFT			
DOUBLE	D01	D02 RIGHT	D03	D04 LEFT			
	D11	D12	D13	D14	D21	D22	

ตารางที่ 4.2 : ภาพองค์ประกอบทั้งหมด

ภาพแต่ละภาพจะมีชื่อเรียก เพื่อให้สะดวกในการอ้างอิง หากสังเกตภาพของ Branch และ Double จะเห็นว่ามีภาพซ้ำกันแต่มีชื่อเรียกแตกต่างกัน ซึ่งเกิดจากการจัดแบ่งกลุ่มของภาพ เพื่อให้สะดวกในการแปลความหมาย



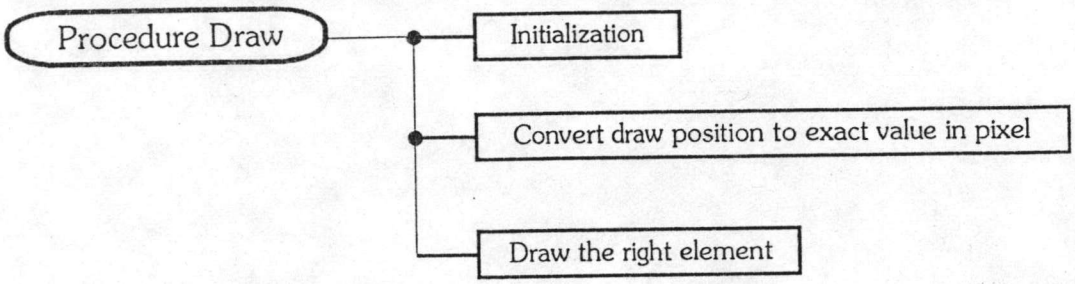
รูปที่ 4.10 : การจัดแบ่งกลุ่มของภาพ

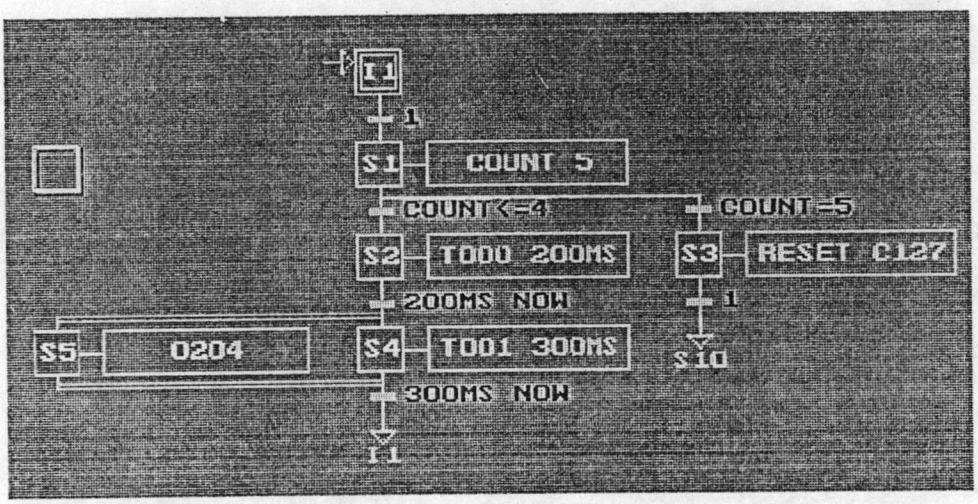
ในการจัดการวาดภาพนั้นหลังจากผู้ใช้เลือกภาพที่ต้องการแล้ว โปรแกรมย่อย Block จะทำหน้าที่ตระเตรียมข้อมูลต่างๆ และเรียกใช้โปรแกรมย่อยอื่นอีกทีเพื่อจัดการการวาดภาพ ซึ่งได้แก่ HandleStep, HandleTransition, HandleWire, HandleLink, HandleBranch, HandleDouble โปรแกรมย่อยเหล่านี้จะตระเตรียมข้อมูลเฉพาะของแต่ละองค์ประกอบ, ตรวจสอบกฎเกณฑ์และปรับข้อมูลในตารางให้ทันกาล แต่ในการวาดภาพแล้วโปรแกรมย่อยเหล่านี้จะเรียกใช้โปรแกรมย่อย Draw อีกทีหนึ่ง

โปรแกรมย่อย Draw จะรับพารามิเตอร์ต่างๆ ได้แก่ ตำแหน่งแถว และคอลัมน์ที่จะวาดภาพบนจอภาพ, ตำแหน่งแถว และคอลัมน์ของข้อมูลบนชาร์ต และชื่อของรูปที่จะให้วาดซึ่งมีการทำงานดังแผนภาพ Design Structure Diagram (DSD)[1] ในรูปที่ 4.11 (DSD เป็นแผนภาพที่ใช้เขียนอธิบายการทำงานของโปรแกรม รายละเอียดกล่าวถึงในภาคผนวก ก)

---

1. Charles Eastel and Gordon Davies. Software engineering. Analysis and design. Berkshire, England : McGRAW-HILL Book Company (UK) Ltd., 1989. pp. 102-118.





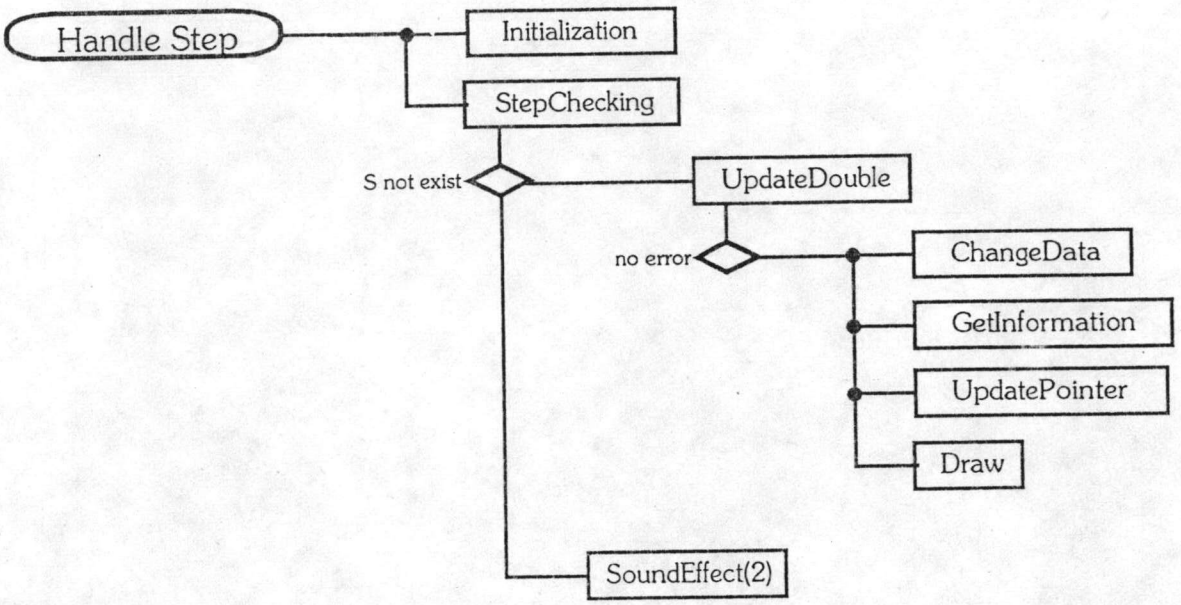
รูปที่ 4.11 : การทำงานของโปรแกรมย่อย Draw และภาพที่วาด

4.5 รายละเอียดการวาด

4.5.1 การวาดสเต็ป

หัวข้อนี้จะกล่าวถึงการวาดสเต็ปเริ่มต้นและสเต็ปพร้อมๆกัน เพราะองค์ประกอบทั้งสองประเภทนี้มีลักษณะและต้องการการจัดการที่คล้ายกันมาก ต่างกันเพียงรูปภาพที่ปรากฏเท่านั้น

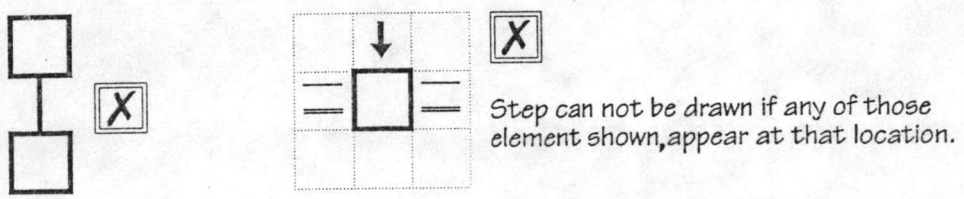
DSD



รูปที่ 4.12 : DSD ของการวาดสเต็ป

ตรวจสอบกฎเกณฑ์

จาก DSD ขั้นตอนที่ตรวจสอบกฎเกณฑ์ได้แก่ StepChecking ซึ่งเป็นโปรแกรมย่อยที่ทำหน้าที่ตรวจสอบว่าในการเขียนสเต็ปนี้นั้นจะทำให้เกิดมีสเต็ป 2 สเต็ป เชื่อมต่อกันโดยตรงหรือไม่ (กฎข้อที่ 1) และยังตรวจสอบบริเวณรอบข้างว่าพร้อมที่จะยอมรับสเต็ปนี้หรือไม่ ดังรูปที่ 4.13



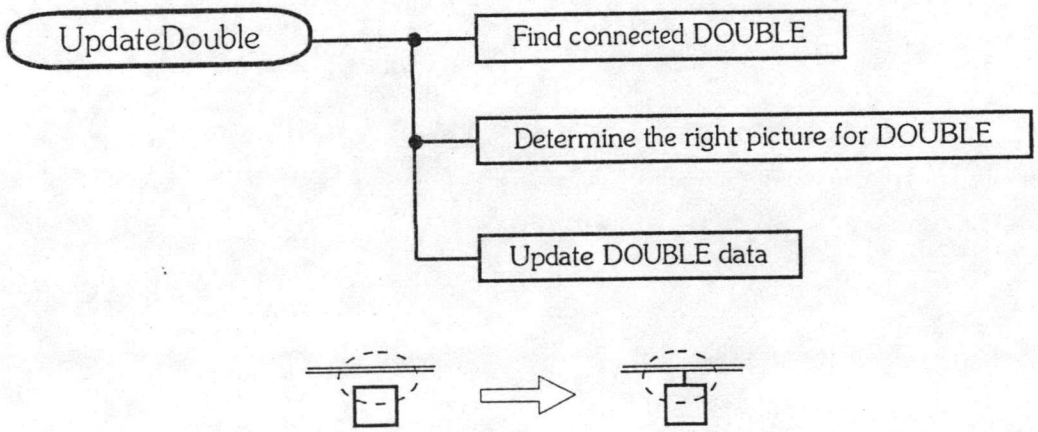
รูปที่ 4.13 : การตรวจสอบก่อนการวาดสเต็ป

การรับข้อมูล

เมื่อวาดภาพสเต็ปแล้วระบบจะสอบถามข้อมูลที่สเต็ปต้องการจากผู้ใช้ ซึ่งเป็นผลจากการเรียกใช้โปรแกรมย่อย GetInformation โปรแกรมย่อยนี้เป็นโปรแกรมส่วนกลางที่ทั้งสเต็ป และทรานสิชั่นเรียกใช้ รายละเอียดของโปรแกรมย่อย GetInformation นี้จะกล่าวถึงให้หัวข้อ "รายละเอียดภายในเซลล์" ข้อมูลที่จะถามจากผู้ใช้ได้แก่ หมายเลขของสเต็ปนี้, หมายเหตุ, หมายเลขของหน้าสัมผัสเอาต์พุต ซึ่งถ้าหากผู้ใช้ป้อนหมายเลขปกติของหน้าสัมผัส ระบบก็จะสอบถามประเภทของเอาต์พุต ผู้ใช้สามารถจะเลือกได้ว่าเป็นเอาต์พุตปกติ (N) ซึ่งแอ็กทีฟหรือไม่แอ็กทีฟตามสถานะของสเต็ป หรือเป็นการเซ็ทให้เอาต์พุตแอ็กทีฟไปตลอด (S) หรือเป็นการรีเซ็ทให้เอาต์พุตไม่แอ็กทีฟ (R) แต่ถ้าหากหมายเลขของเอาต์พุตที่ป้อนขึ้นต้นด้วยตัวอักษร "T" ก็จะหมายถึงการเรียกใช้งานตัวตั้งเวลา (Timer) ระบบก็จะสอบถามต่อว่าจะตั้งเวลานานเท่าไร แต่ถ้าหากตัวอักษรแรกขึ้นต้นด้วยตัว "C" ก็จะหมายถึงการเรียกใช้วงจรนับ (Counter) ระบบก็จะสอบถามต่อว่าจะนับเท่าไร แต่ถ้าหากต้องการจะรีเซ็ท ตัวนับให้ป้อนอักษร "R" แทน ในการใช้วงจรนับจะต้องมีการใช้อีกสเต็ปเพื่อรีเซ็ทวงจรนับเสมอ งานในขั้นตอนการวาดนี้เป็นเพียงการรับค่าไปเก็บไว้ และในขั้นตอนการแปลจึงจะทำการประมวลผลข้อมูลเหล่านี้

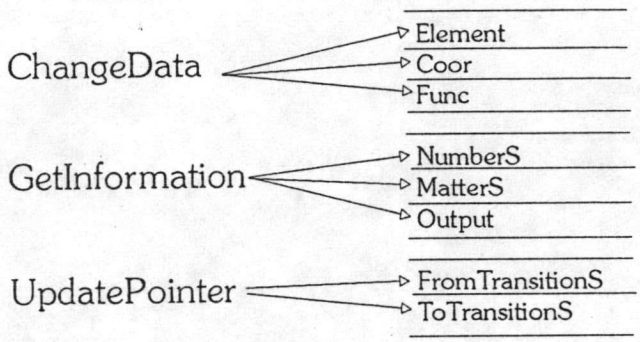
การปรับค่าข้อมูลของชาร์ต

จาก DSD จะมีงานใน 3 ขั้นตอนที่ทำหน้าที่ปรับค่าข้อมูลของชาร์ต ได้แก่ Update Double, ChangeData และ UpdatePointer โปรแกรมย่อย ChangeData จะทำหน้าที่ปรับค่าใน 3 ฟิลด์ (Field) ได้แก่ Element, Coor และ Func โปรแกรมย่อย UpdateDouble มีได้ปรับค่าข้อมูลของชาร์ตสำหรับสเต็ปนี้ แต่จะปรับแต่งภาพ Double ที่เชื่อมต่อและข้อมูลของชาร์ต สำหรับ Double นั้นให้ถูกต้อง ดังแสดงในรูปที่ 4.14 โปรแกรมย่อย UpdatePointer ทำหน้าที่ปรับค่าข้อมูลที่แสดงการเชื่อมต่อของสเต็ปนี้ และทรานสิชั่นรอบข้างให้ถูกต้อง



รูปที่ 4.14 การทำงานของโปรแกรมย่อย UpdateDouble

โดยค้นหาทรานสิชั่นที่เชื่อมต่อกับสเต็ปนี้ทุกๆตัว และปรับค่าข้อมูลที่แสดงการเชื่อมต่อขององค์ประกอบเหล่านี้ให้สอดคล้องกัน โดยการนำค่าในฟิลด์ Coor ไปแลกเปลี่ยนกัน ก็จะทำให้สามารถอ้างอิงถึงกันได้โดยง่าย

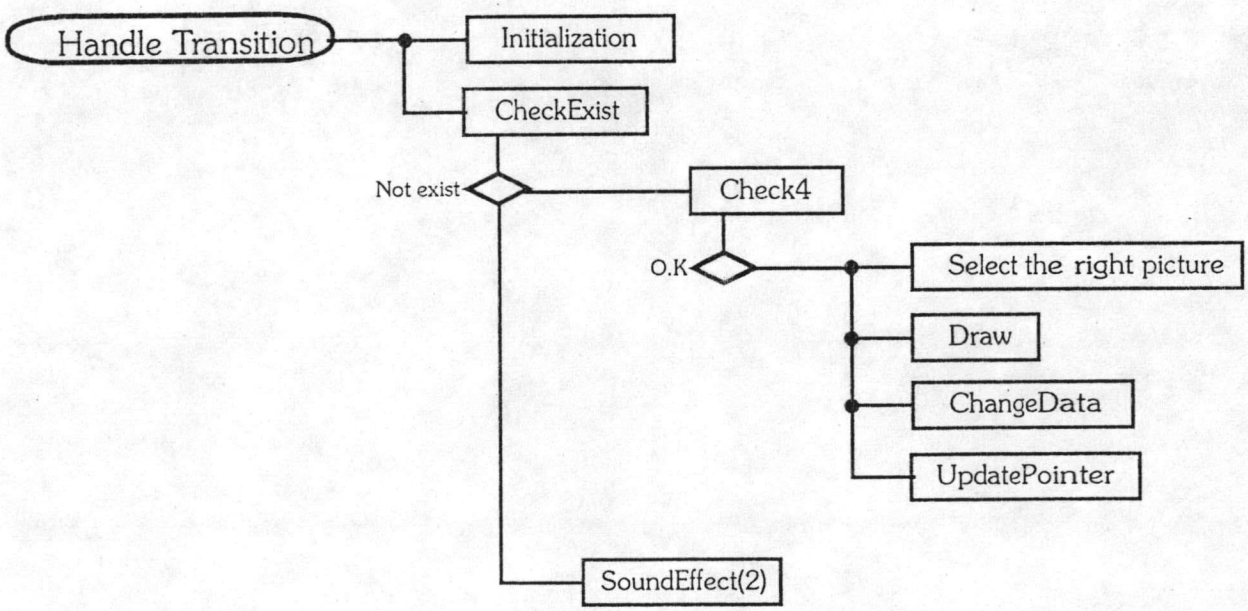


รูปที่ 4.15 : สรุปการปรับค่าข้อมูลในการวาดสเต็ป

4.5.2 การวาดทรานสิชั่น

ทรานสิชั่นเป็นองค์ประกอบที่มีภาพในหลายๆอิริยาบถ ในการวาดภาพทรานสิชั่น ระบบจึงต้องสามารถเลือกภาพทรานสิชั่นที่เหมาะสมไปแสดงได้

DSD



รูปที่ 4.16 : DSD ของการวาดทรานสิชั่น

ตรวจสอบกฎเกณฑ์

จาก DSD ขั้นตอนที่ตรวจสอบกฎเกณฑ์มี 2 ขั้นตอน ได้แก่ CheckExist และ Check4 CheckExist เป็นการตรวจสอบว่าหากเขียนทรานสิชั่นลงตรงนั้นแล้ว จะทำให้ผิดกฎหรือไม่ และพื้นที่ตรงนั้น สามารถเขียนทรานสิชั่นได้หรือไม่ ส่วน Check4 เป็นฟังก์ชั่นที่จะตรวจสอบว่าการเพิ่มทรานสิชั่นลง ณ จุดนั้น จะทำให้เกิดการแยกสาขา (Branch) มากกว่า 4 สาขาหรือไม่ เพราะโครงสร้างข้อมูลได้ถูกออกแบบให้สามารถแยกสาขาได้สูงสุด 4 สาขา Check4 ถูกออกแบบให้เรียกใช้ได้โดยทรานสิชั่น Branch และ Double เพราะเป็นองค์ประกอบที่มีผลต่อโครงสร้างข้อมูลในส่วนที่เก็บความสัมพันธ์

ก่อนที่จะวาดรูปทรานสิชั่น ระบบจะตรวจสอบบริเวณรอบข้างเพื่อนำมาประมวลหาภาพทรานสิชั่นที่เหมาะสม ซึ่งจะเป็นหนึ่งใน 19 ภาพอิริยาบถต่างๆของทรานสิชั่น จากนั้นจึงส่งชื่อภาพไปให้โปรแกรมย่อย Draw วาดภาพทรานสิชั่นนั้นอีกทีหนึ่ง

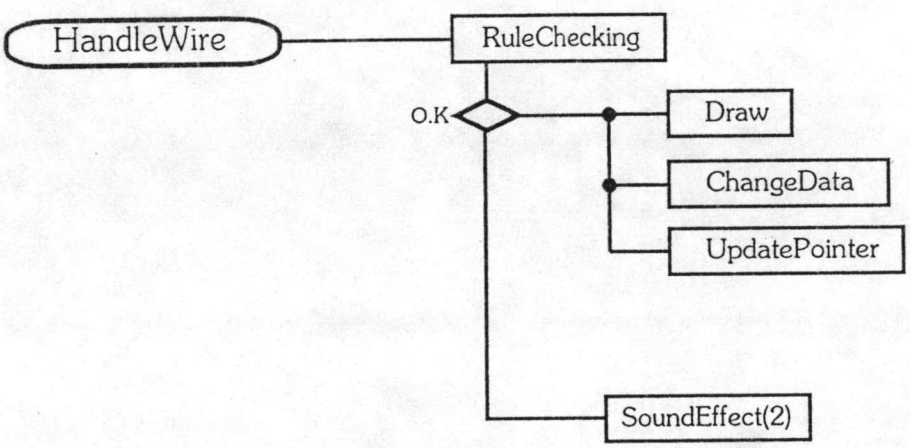
การรับข้อมูล

ระบบจะสอบถามข้อมูลจากผู้ใช้โดยการเรียกใช้โปรแกรมย่อย GetInformation เช่นเดียวกับสเต็ป ข้อมูลที่จะสอบถามได้แก่ หมายเลขของทรานสิชั่น หมายเหตุ หมายเลขหน้าสัมผัสของอินพุต และประเภทของอินพุต ซึ่งได้แก่อินพุตปกติหรือนิเสธของอินพุต ในกรณีที่ต้องการให้เงื่อนไขในทรานสิชั่นเป็นจริงตลอดให้ป้อนค่า "1" ในช่องหมายเลขหน้าสัมผัสของอินพุต จากนั้นระบบจะปรับค่าข้อมูลของชาร์ตตามข้อมูลที่ได้รับ และยังมีการปรับข้อมูลในอีก 2 ขั้นตอน ได้แก่ ChangeData, UpdatePointer ซึ่งทำการปรับข้อมูลในลักษณะเดียวกับที่ได้กล่าวถึงไปแล้วในการวาดสเต็ป ข้อมูลต่างๆที่ถูกปรับให้ทันกาล พิจารณาได้จากภาพที่ 4.15

4.5.3 การวาด Wire

Wire ก็คือ Directed Link ซึ่งเป็นเส้นตรงใช้ช่วยในการเชื่อมต่อ

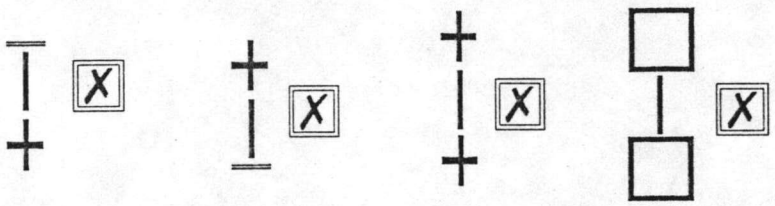
DSD



รูปที่ 4.17 : DSD ของการวาด Wire

การตรวจสอบกฎเกณฑ์

ขั้นตอน Rule Checking ทำหน้าที่ตรวจสอบกฎเกณฑ์ต่างๆ ได้แก่ ภาพ Wire ที่วาดขึ้นจะต้องไม่เชื่อมต่อสเต็ปกับสเต็ป หรือเชื่อมต่อทรานสิชั่นกับทรานสิชั่น และจะต้องไม่ทำให้เกิดการเชื่อมต่อระหว่างทรานสิชั่นกับ Double

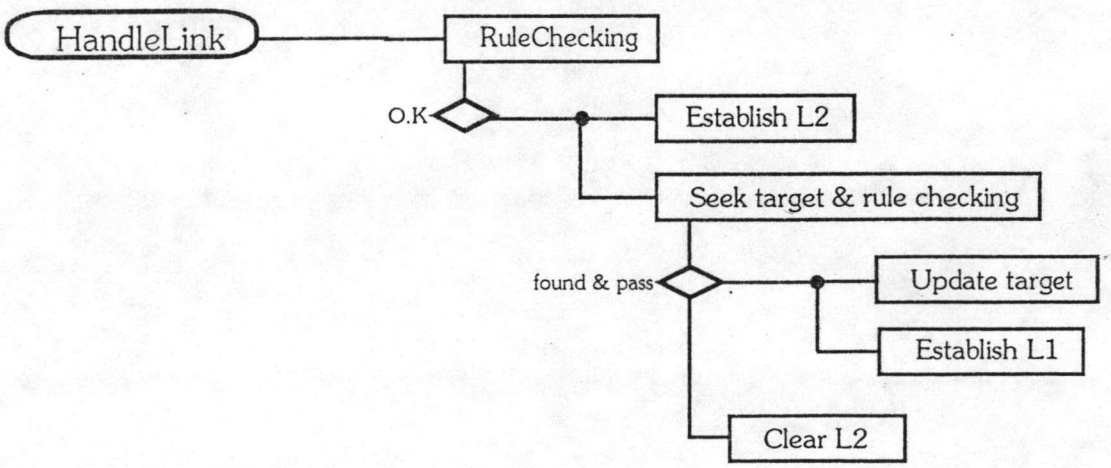


รูปที่ 4.18 : การตรวจสอบก่อนการวาด Wire

Wire เป็นเส้นเชื่อมต่อจึงไม่ต้องรับข้อมูลใดๆ แต่จะมีการปรับข้อมูลของชาร์ตโดยโปรแกรมย่อย ChangeData และ UpdatePointer เช่นเดียวกับสเต็ปและทรานสิชั่น

4.5.4 การวาด Link

Link เป็นการเชื่อมต่อเส้นทางการควบคุมระยะไกล โดยผู้ใช้เขียนภาพ Link ไว้ใต้ทรานสิชั่นพร้อมกับระบุหมายเลขของสเต็ปที่จะกระโดดไปหา ระบบจะค้นหาสเต็ปนั้นพร้อมกับสร้างภาพลูกศรชี้ที่สเต็ปนั้นไว้ให้

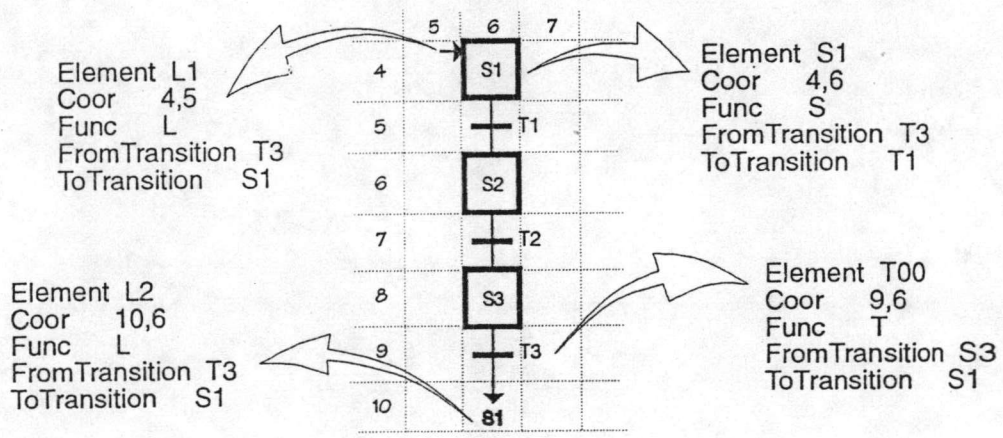


รูปที่ 4.19 : DSD ของการวาด Link

ในขั้นต้นระบบจะตรวจสอบบริเวณนั้นว่าสามารถวาดภาพ Link ณ จุดนั้นได้หรือไม่ ถ้าหากได้ก็จะวาดภาพ Link ณ จุดนั้น (Establish L2) ซึ่งเป็นภาพ L2 (ตามตารางที่ 2) พร้อมกับปรับข้อมูลบนชาร์ตของเซลล์นั้น จากนั้นจะค้นหาสเต็ปเป้าหมาย ถ้าเจอจะปรับค่าข้อมูลการเชื่อมต่อของสเต็ปนั้น พร้อมกับวาดภาพ L1 ที่ช่องซ้างซ้ายของสเต็ปและปรับค่าข้อมูลบนชาร์ต แต่ถ้าหากไม่เจอก็จะลบภาพ L2 และปรับค่าข้อมูลบนชาร์ตของเซลล์นั้นใหม่

ขั้นตอน Rule Checking จะตรวจสอบกฎเกณฑ์และความเป็นไปได้ที่จะวาดภาพ Link ณ จุดนั้น ซึ่งโดยหลักๆก็จะตรวจสอบว่าช่องข้างบนเป็นทรานสิชั่น และต้องเป็นทรานสิชั่นที่พร้อมจะกระโดด ซึ่งได้แก่ T00, T01, T03, T05, T11, T13, T15, T22, T24 และ T26 เท่านั้น เพราะทรานสิชั่นหมายเลขอื่นไม่สามารถกระโดดได้ หรือไม่ก็เป็นจุดเริ่มต้นของการแยกสาขา

ขั้นตอน Establish L2 เป็นการเรียกใช้โปรแกรมย่อย ChangeData, GetInforamtion และ Draw(L2) ChangeData จะปรับค่าข้อมูล 3 ฟิลด์ (Field) ได้แก่ Element, Coor และ Func ของเซลล์นั้น GetInformation จะถามหมายเลขสเต็ปที่กระโดดไป ซึ่งเป็นข้อมูลเดียวที่ต้องการ ในการค้นหาสเต็ปเป้าหมายระบบจะค้นไล่ไปตั้งแต่แถวที่ 1 และคอลัมน์ที่ 1 จนกว่าจะเจอหรือหมดชาร์ต ถ้าหากเจอก็จะตรวจสอบสเต็ปนั้นว่ามีการกระโดดมาครบ 4 แล้วหรือยัง เมื่อทุกอย่างเรียบร้อยก็จะปรับค่าข้อมูลการเชื่อมต่อ และสร้าง L1 ขึ้นมาในเซลล์ทางซ้ายของสเต็ปพร้อมกับปรับค่าข้อมูลบนชาร์ต

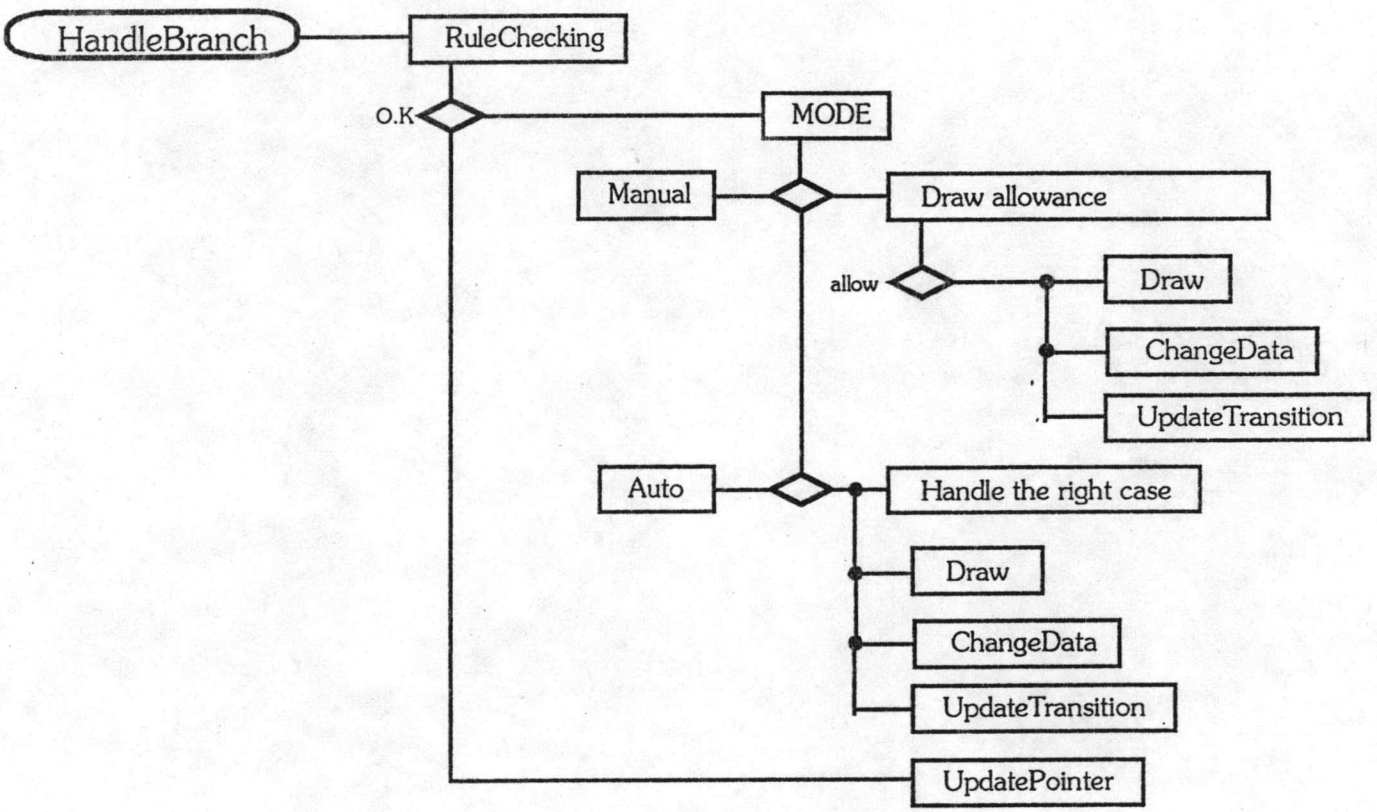


รูปที่ 4.20 : ข้อมูลที่ถูกปรับค่าจากการวาด L

4.5.5 <u>การวาด Branch</u>

ตามความเป็นจริง Branch มีเพียง 2 ลักษณะ ได้แก่ Branch ที่แยกออกจากด้านบนของทรานสิชั่นและ Branch ที่แยกออกจากด้านล่างของทรานสิชั่น แต่เพื่อช่วยอำนวยความสะดวกในการแปล จึงได้ออกแบบเพิ่มให้แปงภาพ Branch ทางด้านซ้ายและขวาด้วยภาพ Branch จึงมี 4 ภาพ เนื่องจาก Branch เป็นองค์ประกอบที่แยกออกจากทรานสิชั่น ดังนั้นในการเขียนภาพ Branch บ่อยครั้งจึงต้องปรับภาพทรานสิชั่นไปพร้อมๆกัน

DSD

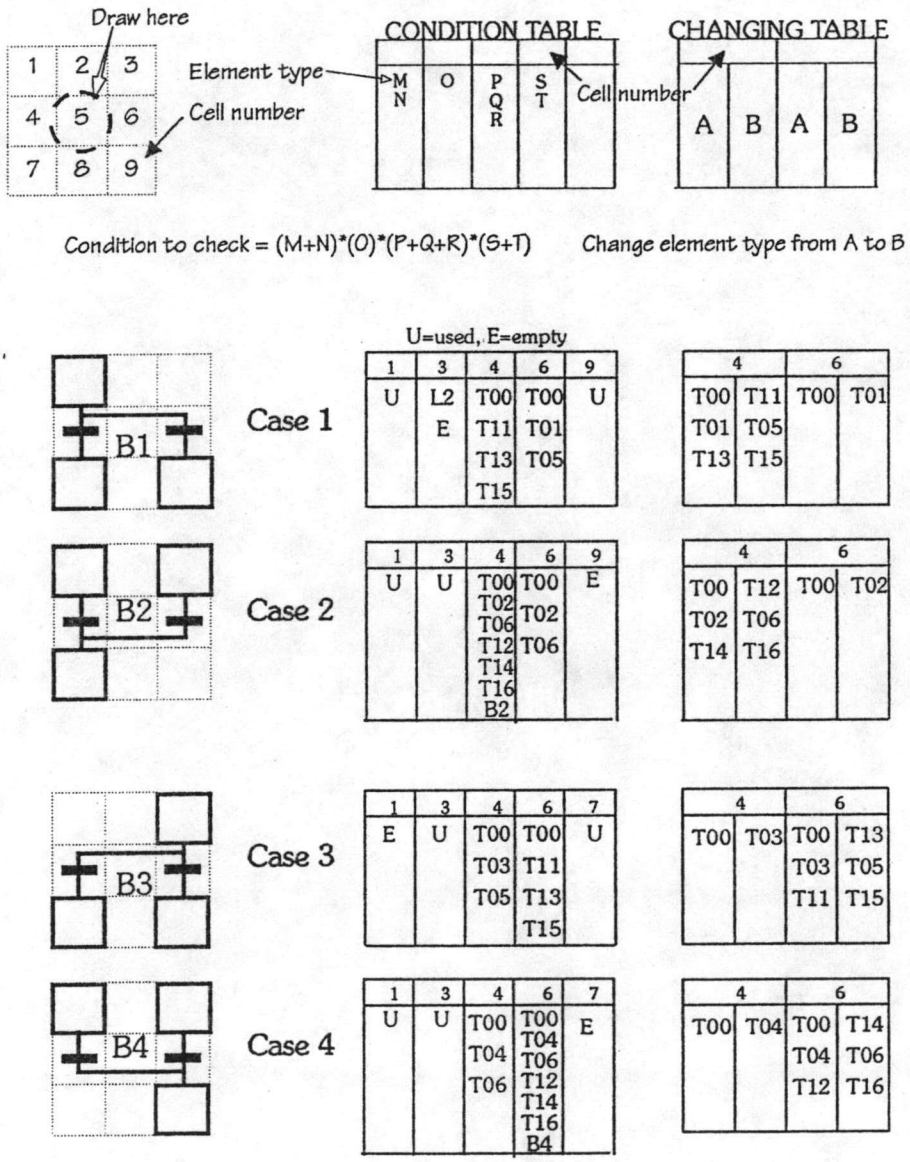


รูปที่ 4.21 : DSD ของการวาด Branch

การออกแบบการวาด Branch แบ่งเป็น 2 กรณีใหญ่ๆ ได้แก่ แบบ Manual และ แบบ Auto แบบ Manual จะเป็นการระบุโดยผู้ใช้โดยตรงว่าต้องการวาดภาพ Branch ภาพใด (BLB4) ส่วนแบบ Auto จะช่วยตัดสินใจเลือกภาพ Branch ที่เหมาะสมให้ โดยวิเคราะห์จากองค์ประกอบรอบข้าง

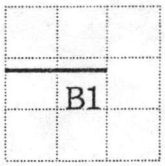
ตรวจสอบกฎเกณฑ์

การตรวจสอบกฎเกณฑ์และการเลือกภาพที่เหมาะสม (Auto) จะกระทำไปพร้อมๆกัน ในขั้นตอนแรกสุดจะเป็นการตรวจสอบโดยฟังก์ชั่น Check4 เพื่อวิเคราะห์ว่าการวาดภาพ Branch ลงตรงจุดนั้นจะทำให้เกิดการแยกสาขามากกว่า 4 หรือไม่ หากไม่มีปัญหาก็จะทำการตรวจสอบในขั้นต่อไป ซึ่งถ้าหากเป็นการทำงานแบบ Manual ก็จะตรวจสอบช้างซ้ายและช้างขวาว่าเป็นสเต็ปหรือไม่ และเป็นการสั่งให้เขียนภาพ Branch คนละประเภทปะปนกันหรือไม่ หากเป็นการทำงานแบบ Auto ก็จะตรวจสอบสภาพรอบข้างและวิเคราะห์วาดภาพ Branch ที่เหมาะสม ในบางครั้งก็จะปรับแต่งภาพทรานสิชั่นรอบข้างด้วย ดังแสดงในรูปที่ 4.22



รูปที่ 4.22(1) : กรณีต่างๆ ที่วิเคราะห์ในการวาด Branch แบบ Auto

U=used, E=empty



Case 5

3	4	6
L2	T00	E
E	T01	T01
	T05	T05
	T11	B1
	T13	
	T15	
	B1	

4	
T00	T11
T01	T05
T13	T15

Case 6

4	6	9
T00	E	E
T02	T02	
T06	T06	
T12	B2	
T14		
T16		
B2		

4	
T00	T12
T02	T06
T14	T16

Case 7

1	4	6
E	E	T00
L2	T03	T03
	T05	T05
	B3	T11
		T13
		T15
		B3

6	
T00	T13
T03	T05
T11	T15

Case 8

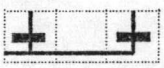
4	6	9
E	T00	E
T00	T04	
T04	T06	
T06	T12	
B4	T14	
	T16	
	B4	

6	
T00	T14
T04	T06
T12	T16

Case 9

3	4	6
E	T01	T00
L2	T05	
	T11	
	T13	
	T15	
	B1	

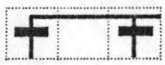
4	
T01	T05
T13	T15



Case 10

4	6	9
T02	T00	E
T06		
T12		
T14		
T16		
B2		

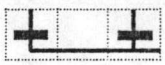
4	
T02	T06
T14	T16



Case 11

1	4	6
E	T00	T03
L2		T05
		T11
		T13
		T15
		B3

6	
T03	T05
T11	T15



Case 12

4	6	7
T00	T04	E
	T06	
	T12	
	T14	
	T16	

6	
T04	T06
T12	T16

รูปที่ 4.22(2) : กรณีต่างๆ ที่วิเคราะห์ในการวาด Branch แบบ Auto (ต่อ)

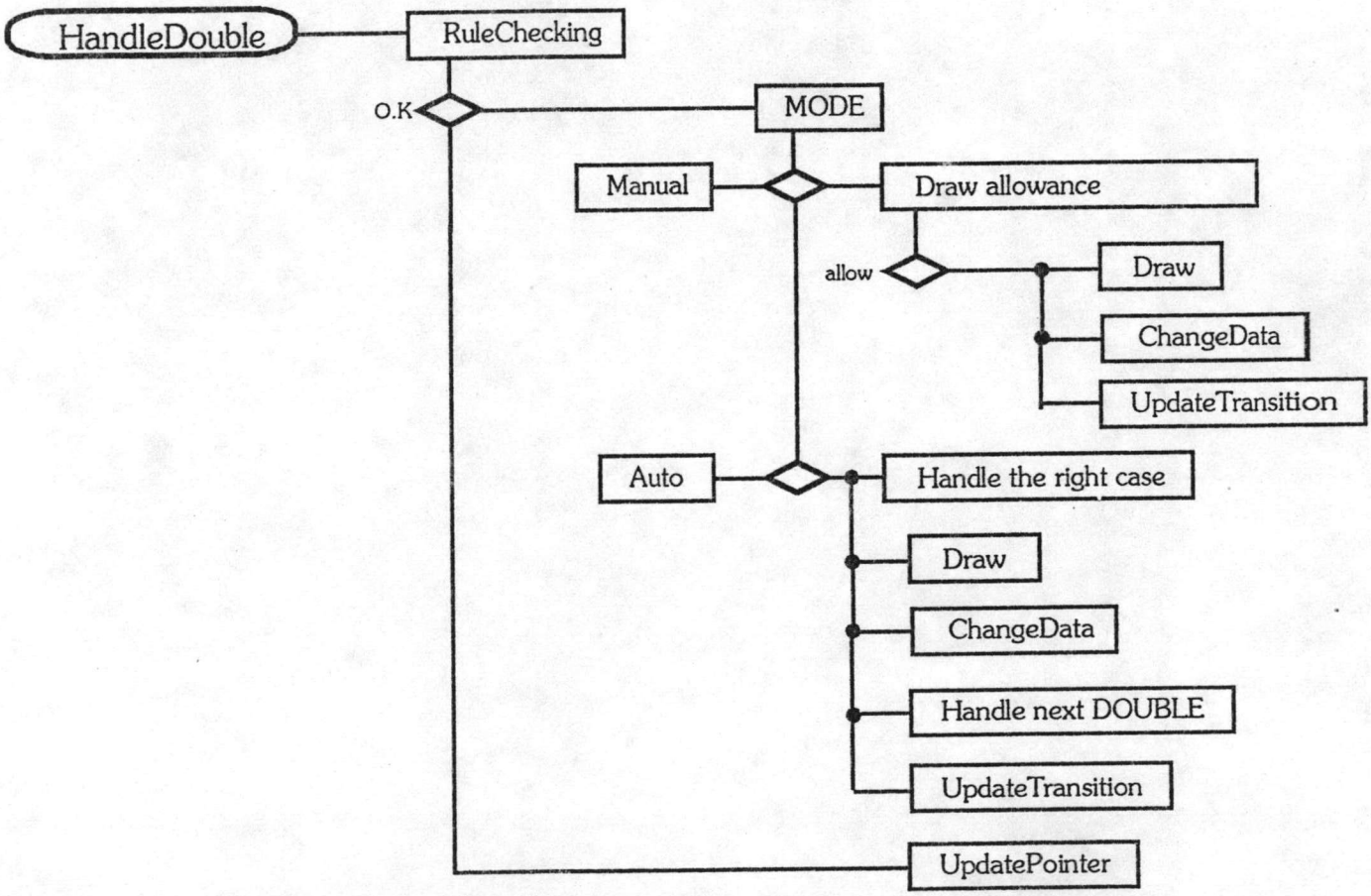
การปรับค่าข้อมูลบนชาร์ต

ขั้นแรกจะเป็นการปรับข้อมูลภายในเซลล์ของ Branch เอง โดยโปรแกรมย่อย ChangeData จากนั้นจะเป็นการปรับภาพทรานสิชั่นและข้อมูลบนชาร์ต ดังแสดงในรูปที่ 4.22 ในส่วน ChangeTable และส่วนสุดท้ายจะเป็นการปรับการเชื่อมต่อของสเต็ปและทรานสิชั่นที่เกี่ยวข้อง โดยโปรแกรมย่อย UpdatePointer

4.5.6 การวาด Double

Double ประกอบด้วยภาพต่างๆ 10 ภาพ ภาพ DO1 ถึง DO4 เป็นผลจากการออกแบบในลักษณะเดียวกับ B1 ถึง B4 การควบคุมการวาด Double มีลักษณะใกล้เคียงกับกรณีของ Branch

DSD

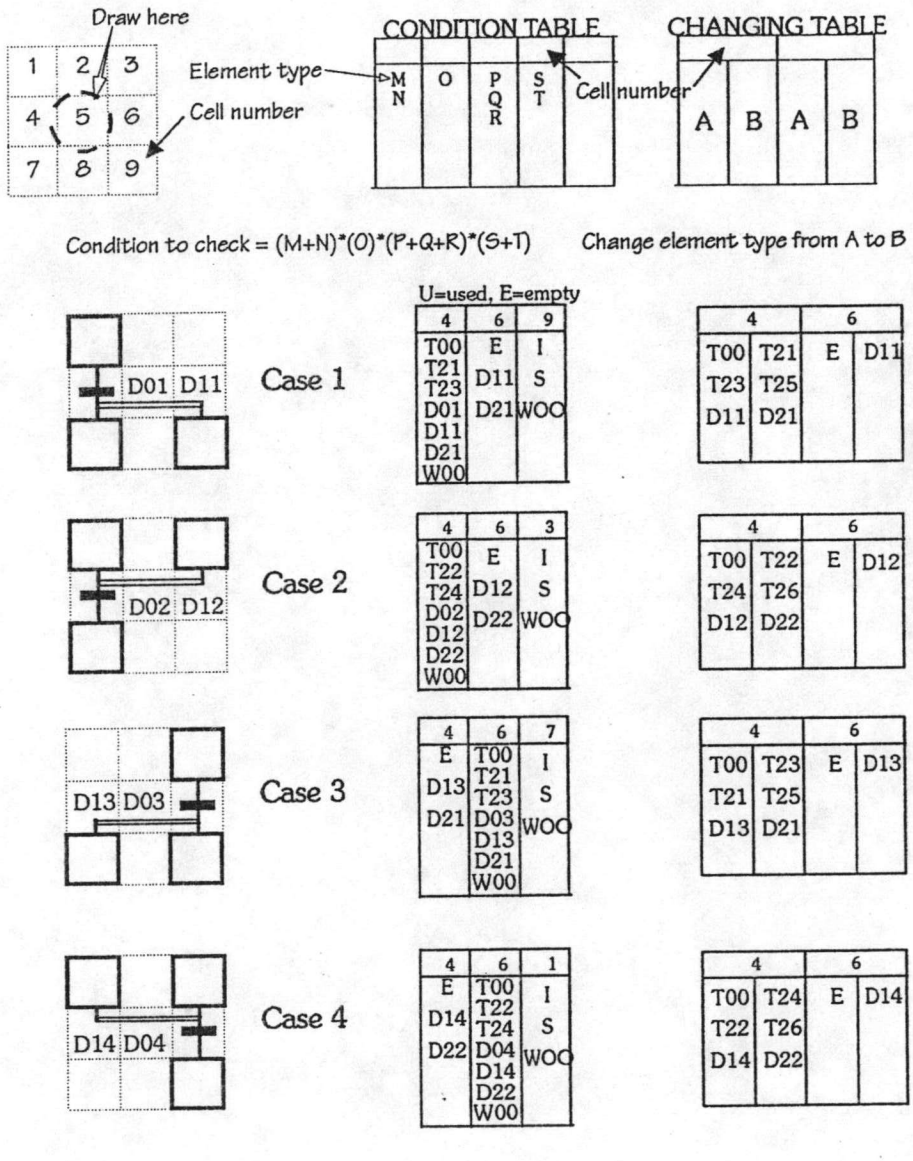


รูปที่ 4.23 : DSD ของการวาด Double

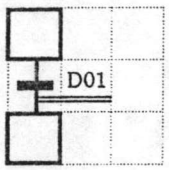
การควบคุมการวาดโดยผู้ใช้แบ่งเป็นแบบ Manual และ Auto เช่นเดียวกับ Branch

ตรวจสอบกฎเกณฑ์

หลังจาก Check4 แล้ว หากเป็นการวาดแบบ Manual ก็จะตรวจสอบดูข้างๆว่ามีสเต็ปอยู่หรือไม่ และตรวจสอบว่าจะทำให้วาด Double ต่างประเภทกันปะปนกันหรือไม่ ในบางกรณีก็จะวาดภาพ Double ตัวถัดไปให้ด้วย รูปที่ 4.24 แสดงรายละเอียด



รูปที่ 4.24(1) : กรณีต่างๆที่วิเคราะห์ในการวาด Double แบบ Auto

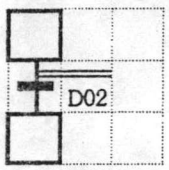


Case 5

<> = not equal to

4	6	8
T00 T21 T23 T25 D01 D11 D21 W00	E D01 D11 D21	<> I <> S <> W00

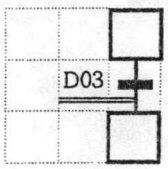
4	
T00	T21
T21	T25
T23	T25
D11	D21



Case 6

4	6	2
T00 T22 T24 T26 D02 D12 D22 W00	E D02 D12 D22	<> I <> S <> W00

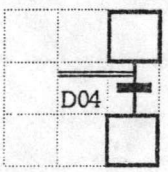
4	
T00	T22
T22	T26
T24	T26
D12	D22



Case 7

4	6	8
E D03 D13 D21	T00 T21 T23 T25 D03 D13 D21 W00	<> I <> S <> W00

4	
T00	T23
T21	T25
T23	T25
D13	D21



Case 8

4	6	2
E D04 D14 D22	T00 T22 T24 T26 D04 D14 D22 W00	<> I <> S <> W00

4	
T00	T24
T22	T26
T24	T26
D14	D22

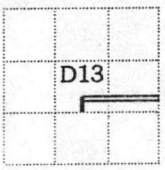
รูปที่ 4.24(2) : กรณีต่างๆที่วิเคราะห์ในการวาด Double แบบ Auto (ต่อ)

Case 9

4	6	8
D01	E	I S W00

Case 10

4	6	2
D02	E	I S W00



Case 11

4	6	8
E	D03	I S W00

Case 12

4	6	2
E	D04	I S W00

Case 13

4	6	8
D01 D03	D01 D03	I S W00

Case 14

4	6	2
D02 D04	D02 D04	I S W00

รูปที่ 4.24(3) : กรณีต่างๆที่วิเคราะห์ในการวาด Double แบบ Auto (ต่อ)

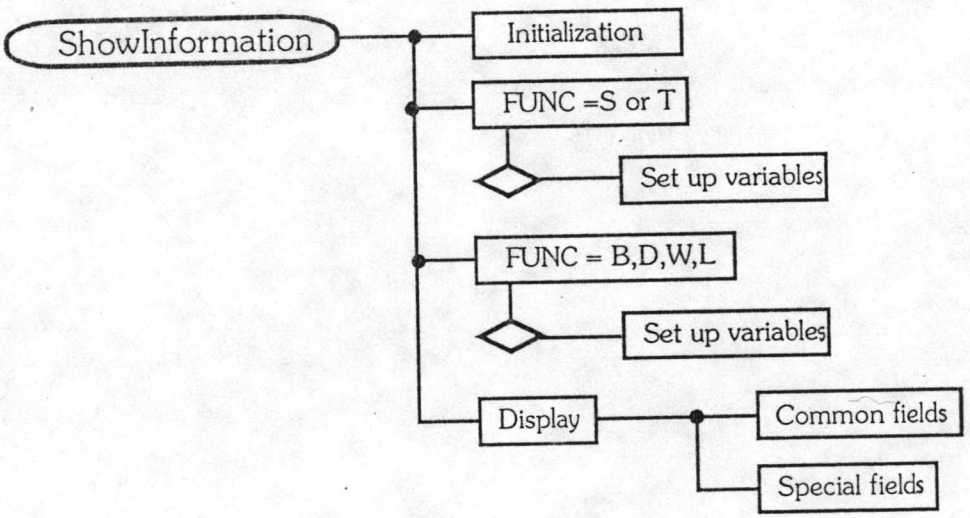
การปรับค่าข้อมูลบนชาร์ต

ขั้นต้นปรับค่าโดย ChangeData ขั้นต่อไปจะเป็นการปรับภาพทรานสิชั่นช้างๆให้ถูกต้อง นอกจากนั้นหากวิเคราะห์แล้วเห็นว่า ช่องช้างๆควรจะเป็นภาพ Double ระบบก็จะวาดภาพ Double ให้เลย และในขั้นตอนสุดท้ายก็จะเป็นการปรับเชื่อมต่อโดย UpdatePointer

4.6 รายละเอียดภายในเซลล์

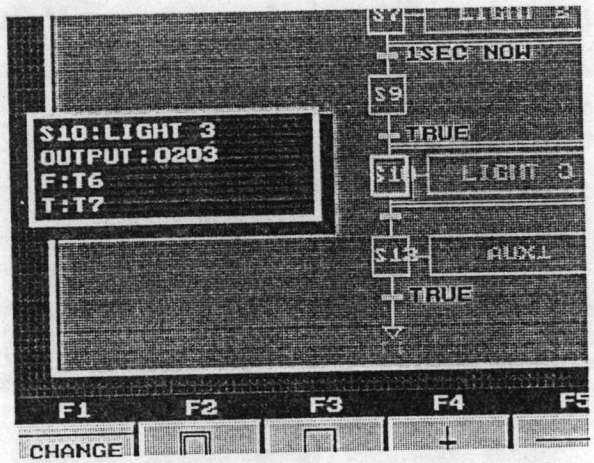
4.6.1 หน้าต่างแสดงรายละเอียดภายในเซลล์

เนื่องจากรายละเอียดภายในเซลล์มีมาก จึงสร้างหน้าต่างเล็กๆ ขึ้นมา เพื่อใช้แสดงรายละเอียดนี้โดยเฉพาะ เมื่อผู้ใช้เลื่อนเคอร์เซอร์ไปตามเซลล์ต่างๆ รายละเอียดที่แสดงภายในหน้าต่างก็จะเปลี่ยนไปเป็นรายละเอียดของเซลล์ที่เคอร์เซอร์กำลังชี้อยู่ ผู้ใช้สามารถจะซ่อนหน้าต่างนี้ไว้เมื่อไม่ต้องการดู หรืออาจจะเรียกขึ้นมาดูก็ได้ โปรแกรมย่อย Show Information เป็นตัวจัดการสำหรับหน้าต่างนี้ ซึ่งมีการทำงานดังรูปที่ 4.25



รูปที่ 4.25 : DSD ของ ShowInformation

ขั้นตอน Initialize เป็นการวาดภาพหน้าต่างและเคลียร์ค่าตัวแปรต่างๆ ในขั้นตอนถัดไปก็เป็นการตั้งค่าตัวแปรต่างๆ แล้วแต่กรณี ส่วนในการแสดงรายละเอียดต่างๆนั้นจะแบ่งเป็น 2 ขั้นตอน ขั้นตอนแรกจะแสดงฟิลด์ต่างๆที่มีร่วมกันในทุกๆองค์ประกอบ ขั้นที่สองจะจัดการแสดงข้อมูลพิเศษที่แตกต่างกันไปในแต่ละองค์ประกอบ ซึ่งได้แก่การแสดงเอาต์พุตประเภทคงค่าไว้, ตัวตั้งเวลา, ตัวนับ ตลอดจนอินพุตประเภทนิเสธ



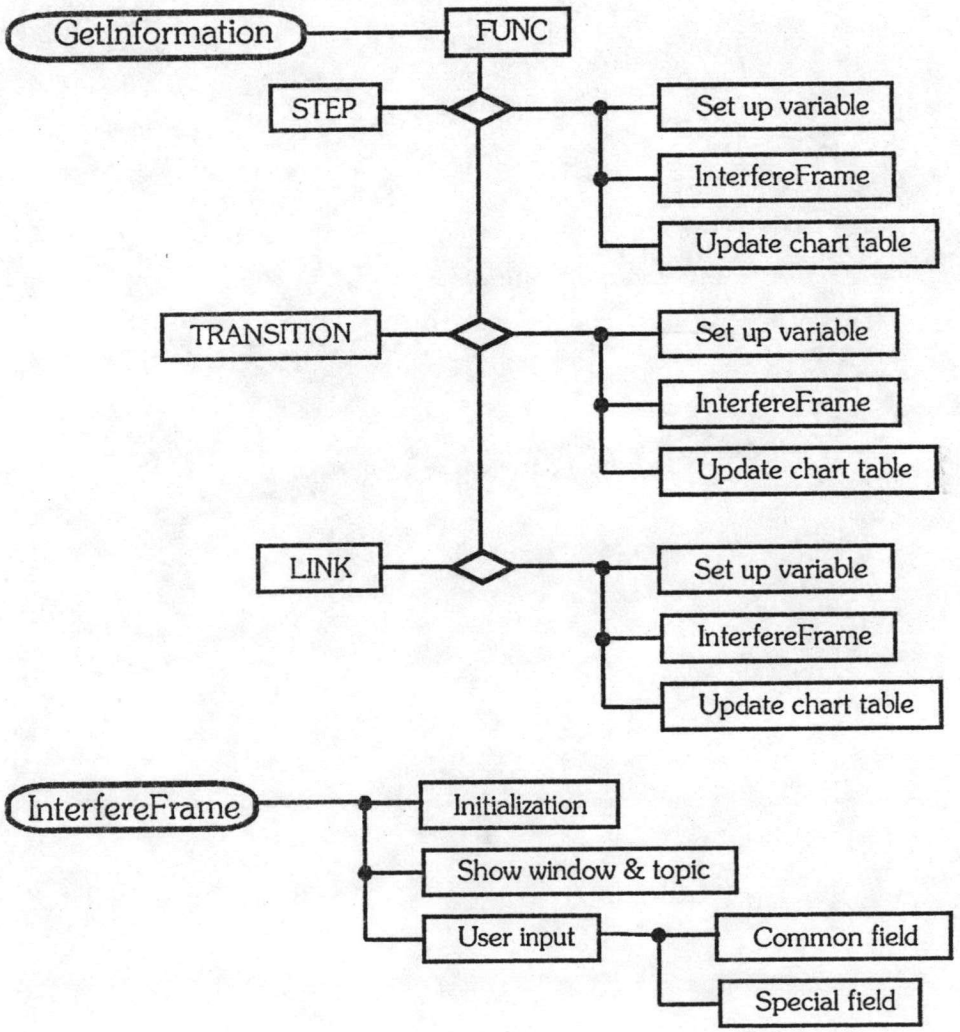
รูปที่ 4.26 : หน้าต่างแสดงรายละเอียดภายในเซลล์

4.6.2 <u>การรับรายละเอียดของเซลล์</u>

เป็นการรับรายละเอียดที่จำเป็นสำหรับองค์ประกอบนั้น หากเป็นสเต็ปก็จะรับหมายเลขสเต็ป, หมายเหตุ, หมายเลขหน้าสัมผัสเอาต์พุต และประเภทของเอาต์พุต (ได้แก่ แบบปกติ, แบบแอ็กทีฟตลอด (Set) หรือแบบสั่งไม่ให้แอ็กทีฟ (Reset)) แต่ถ้าหากหมายเลขหน้าสัมผัสเอาต์พุตขึ้นต้นด้วยตัวอักษร 'T' จะหมายถึงการเรียกใช้เอาต์พุตประเภทตัวตั้งเวลา (Timer) ซึ่งระบบจะถามว่ามีค่าคงที่ (เวลาที่ตั้ง) เป็นเท่าไร แต่ถ้าหากหมายเลขเอาต์พุตขึ้นต้นด้วยตัวอักษร 'C' จะหมายถึงการเรียกใช้ตัวนับ (Counter) ซึ่งระบบจะถามต่อไปว่า มีค่าคงที่เท่าไร หากเป็นทรานสิชั่นก็จะถามเช่นเดียวกัน แต่ฟิลด์ เอาต์พุตจะเปลี่ยนเป็นอินพุต และถามประเภทของอินพุตว่าเป็นปกติหรือนิเสธ องค์ประกอบประเภทสุดท้ายที่จะมีการรับ ข้อมูลได้แก่ Link ซึ่งจะถามว่ากระโดดไปสเต็ปใด

รูปที่ 4.27 : การถามข้อมูลขณะวาดสเต็ป

การควบคุมการรับรายละเอียดเป็นหน้าที่ของโปรแกรมย่อย GetInformation รูปที่ 4.28 แสดง DSD งานในขั้นตอน Initialize เป็นการตั้งค่าตัวแปร และเก็บภาพบริเวณที่จะวาดหน้าต่าง เพื่อจะได้นำมาแปะไว้ที่เดิมเมื่อสิ้นสุดการถามข้อมูล จากนั้นจะเป็นการวาดหน้าต่างสอบถามข้อมูล และรอรับการป้อนอินพุตจากผู้ใช้ ณ จุดนี้จะมีการเรียกใช้โปรแกรมย่อย GetGString ซึ่งจะช่วยควบคุมการรับข้อมูลจากผู้ใช้ ได้แก่จำนวนตัวอักษรสูงสุดที่ป้อนได้ในฟิลด์นั้น กลุ่มของอักษรที่จะรับในฟิลด์นั้น เช่นเฉพาะตัวเลข, ตัวเลขและตัวหนังสือ หรือรวมตัวอักษรพิเศษด้วย ซึ่งหากผู้ใช้ป้อนตัวอักษรที่มิได้อยู่ในกลุ่มที่ระบุไว้ ระบบก็จะไม่ยอมรับ นอกจากนั้น GetInformation ยังจัดการในการรับข้อมูลพิเศษสำหรับเอาต์พุตและอินพุตให้ได้ครบถ้วนอีกด้วย และเมื่อรับข้อมูลต่างๆจากผู้ใช้ครบแล้ว ก็จะนำข้อมูลที่ได้ไปปรับค่าข้อมูลบนชาร์ตให้ทันกาล



รูปที่ 4.28 : DSD ของการรับรายละเอียดของเซลล์

4.6.3 การแก้ไขรายละเอียดภายในเซลล์

การแก้ไขรายละเอียดต่างๆ ที่ได้เขียนไปแล้ว สามารถแก้ไขได้โดยการเรียกใช้คำสั่ง Edit จากเมนู โดยก่อนการเรียกใช้จะต้องเคลื่อนเคอร์เซอร์ไปไว้ยังเซลล์ที่ต้องการจะแก้ไขก่อน ทันทีที่เรียกใช้คำสั่ง Edit ระบบจะเรียกใช้โปรแกรมย่อย EditCell ซึ่งจะตรวจสอบประเภทของเซลล์นั้น จากนั้นจะเรียกใช้โปรแกรมย่อย GetInformation เพื่อรับข้อมูลจากผู้ใช้ เมื่อผู้ใช้แก้ไขเสร็จแล้ว GetInformation จะจัดการเก็บข้อมูลใหม่ให้ จากนั้น EditCell จะเรียกใช้โปรแกรมย่อย Redraw เพื่อปรับข้อมูลที่แสดงบนจอภาพให้ถูกต้อง

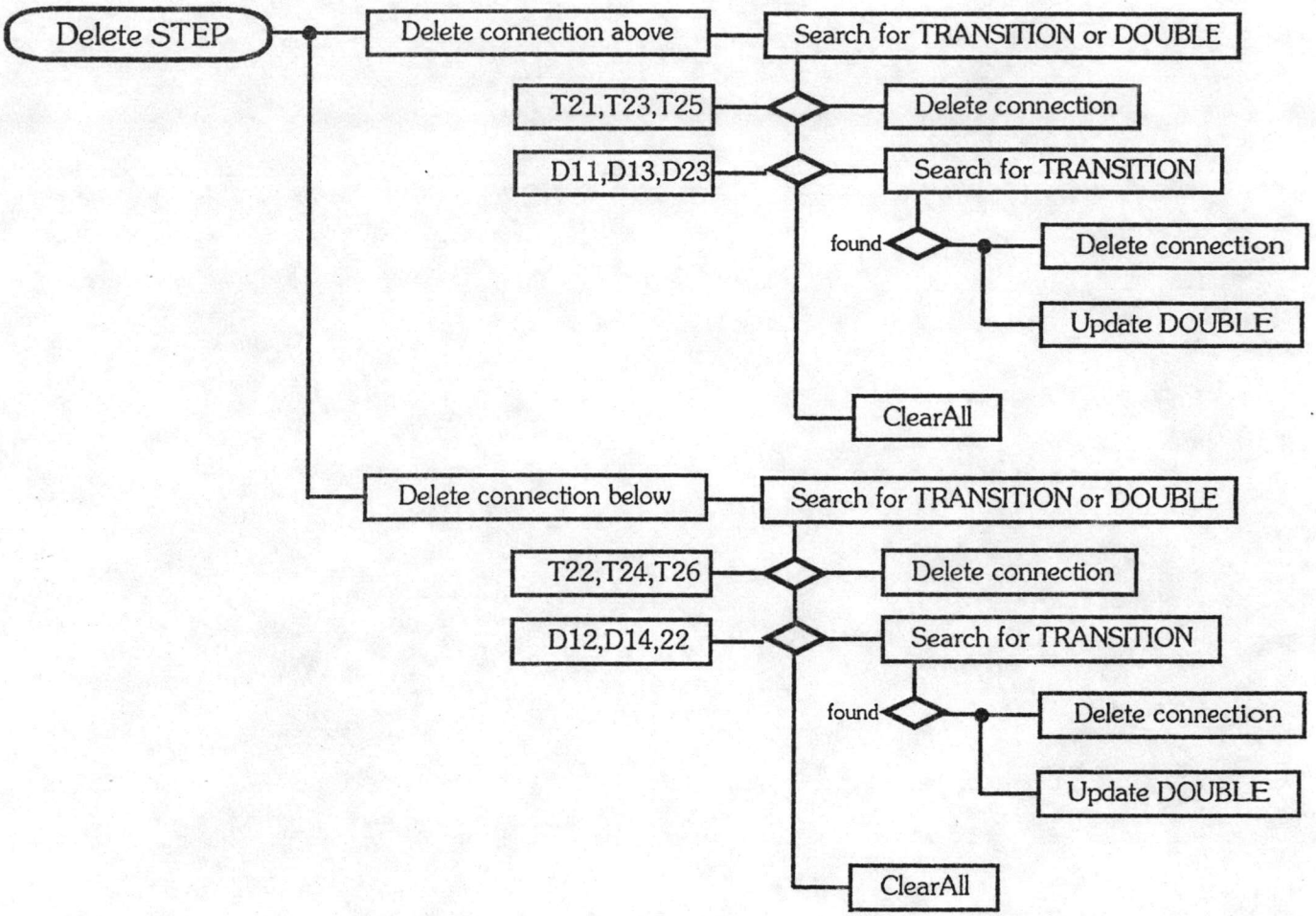
4.7 การลบ

เมื่อมีการสั่งให้ลบเซลล์ใดๆทิ้ง ระบบจะมาเรียกใช้โปรแกรมย่อย HandleDelete เนื่องจากข้อมูลในทุกๆเซลล์มีความสัมพันธ์ต่อกัน เกี่ยวพันกันเป็นโครงข่ายของสเต็ปและทรานสิชั่น การลบจึงต้องกระทำด้วยความระมัดระวังและรอบคอบ เพราะนอกจากจะลบเซลล์นั้นทิ้งไปแล้ว ยังต้องปรับโครงสร้างการเชื่อมต่อของสเต็ปและทรานสิชั่นให้ถูกต้องด้วย ในการลบจึงมีการทำงาน 2 ขั้นตอนหลักได้แก่ ลบเซลล์นั้นทิ้ง และปรับการเชื่อมต่อ การลบเซลล์นั้นทิ้งกระทำได้โดยง่าย กล่าวคือลบภาพเซลล์นั้นบนจอภาพทิ้ง และลบเซลล์นั้นจากข้อมูลของชาร์ต ส่วนการปรับการเชื่อมต่อก็จะปรับทั้งบนจอภาพและข้อมูลบนตารางชาร์ต ซึ่งจะกล่าวถึงต่อไปโดยแยกตามประเภทขององค์ประกอบ

4.7.1 การลบสเต็ป

จะต้องลบการเชื่อมต่อกับทรานสิชั่น ทางด้านบนและทางด้านล่าง ซึ่งหมายถึงทรานสิชั่นที่ถูกระบุไว้ในฟิลด์ FromTransitionS และ ToTransitionS จึงแบ่งการลบเป็น 2 ขั้นตอนใหญ่ๆ ขั้นแรกตัดความสัมพันธ์ทางด้านบน ขั้นต่อไปจึงตัดความสัมพันธ์ทางด้านล่าง ทั้งสองขั้นตอนมีการทำงานเหมือนกัน ต่างกันเพียงในรายละเอียด (ชื่อของภาพ)

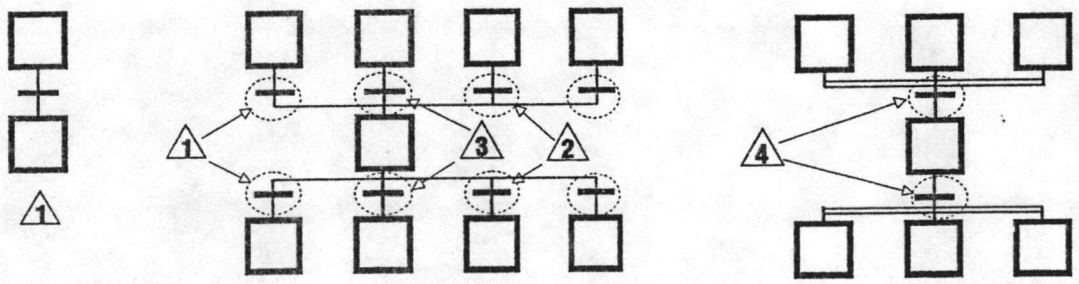
ไม่ว่าจะเป็นทางด้านบนหรือทางด้านล่าง เริ่มต้นก็จะค้นหาทรานสิชั่น หรือ Double โดยไต่ไปตาม Wire และเมื่อเจอทรานสิชั่น T21, T23, T25, T22, T24, T26 ก็จะตัดความสัมพันธ์ที่มีต่อกัน สาเหตุที่แยกทรานสิชั่นเหล่านี้ออกมาตรวจสอบเพราะสเต็ปที่กำลังจะลบ จะเป็นเพียง 1 ใน 4 ของสเต็ปที่เชื่อมต่อกับทรานสิชั่นนี้ (หากเป็นทรานสิชั่นแบบอื่นจะเชื่อมต่อกับสเต็ปเพียงสเต็ปเดียว) จึงต้องตัดความสัมพันธ์ด้วยความระมัดระวัง ถ้าหากค้นหาไปเจอ Double ก็จะต้องค้นหาทรานสิชั่นต่อไปอีกครั้งโดยไต่ไปตาม Double ซึ่งทิศทางทราบได้จากประเภทของ Double เมื่อเจอทรานสิชั่นก็จะตัดความสัมพันธ์ทิ้งไป พร้อมกับปรับภาพ และข้อมูลของ Double เสียใหม่



รูปที่ 4.29 : DSD ของการลบสเต็ป

4.7.2 <u>การลบทรานสิชั่น</u>

แบ่งเป็นหลายกรณีดังแสดงในรูปที่ 4.30



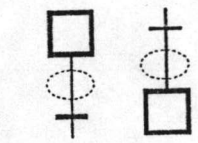
รูปที่ 4.30 : กรณีต่างๆ ในการลบทรานสิชั่น

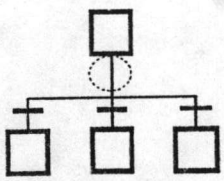
กรณีที่ 1 เป็นการลบความสัมพันธ์แบบหนึ่งต่อหนึ่ง โดยตัดความสัมพันธ์ทั้งทางด้านบนและทางด้านล่าง กรณีที่ 2 ทรานสิชั่นนี้มีความสัมพันธ์กับสเต็ปแบบหนึ่งต่อหนึ่งเช่นกัน แต่การลบทรานสิชั่นตัวนี้จะมีผลต่อทรานสิชั่นทางปลาย Branch คือจะทำให้ความสัมพันธ์ของทรานสิชั่นตัวปลาย Branch กับสเต็ปตัวเดียวกันถูกตัดลงด้วย กรณีที่ 3 การลบทรานสิชั่นตัวนี้มีผลต่อทรานสิชั่นร่วม Branch ทุกตัว จึงต้องตัดความ สัมพันธ์ให้ครบถ้วน และกรณีที่ 4 เป็นการเชื่อมโยงแบบหนึ่งต่อหลายๆ การลบทรานสิชั่นจึงต้องตัดความสัมพันธ์ กับสเต็ปร่วม Double ทุกตัว

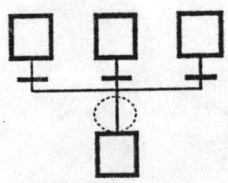
4.7.3 <u>การลบ Wire</u>

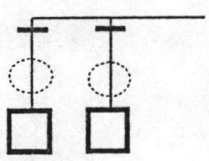
Wire จะเชื่อมต่ออยู่ระหว่างสเต็ปและทรานสิชั่นเสมอ ในบางครั้งอาจเชื่อมต่อผ่าน Double การลบ Wire แบ่งเป็น 2 กรณี ได้แก่ กรณีที่ 1 ลบทุกความสัมพันธ์ของทุกๆสเต็ปและทรานสิชั่น และกรณีที่ 2 ลบเฉพาะความสัมพันธ์ที่เกี่ยวข้องกัน เพราะสเต็ปจะเป็นเพียง 1 ในหลายๆสเต็ปที่เชื่อมต่อกับทรานสิชั่นนั้น

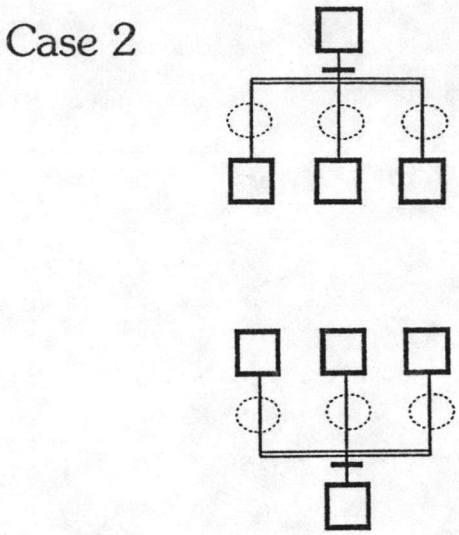
Case 1







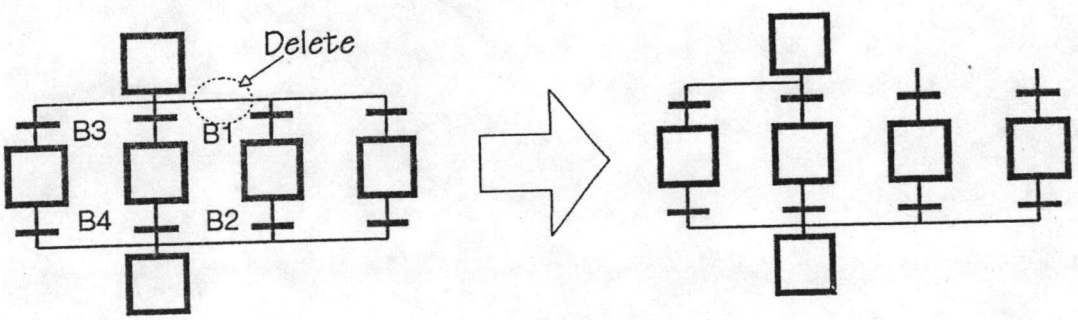




รูปที่ 4.31 : กรณีต่างๆ ในการลบ Wire

4.7.4 การลบ Branch

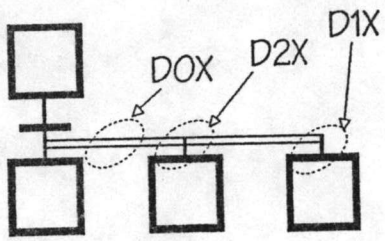
เนื่องจาก Branch จะมีลักษณะคล้ายคลองชลประทานที่ส่งน้ำไปยังผืนนาต่างๆ การตัดคลองชลประทาน ณ จุดใด ก็จะมีผลกระทบต่อผืนนาทางปลายน้ำ ดังนั้นการลบ Branch จึงต้องค้นหาทรานสิชั่นทางด้านปลายเพื่อตัดความสัมพันธ์ที่มีต่อสเต็ปร่วมและปรับภาพทรานสิชั่นเหล่านั้นใหม่ ทิศทางการค้นหาทรานสิชั่น พิจารณาได้จากชื่อ Branch ที่ลบ หากเป็น B1 ก็ค้นหาทรานสิชั่นไปทางขวาและค้นหาสเต็ปทางซ้ายบน หากเป็น B2 ก็ค้นหาทรานสิชั่นไปทางขวา และค้นหาสเต็ปไปทางซ้าย สเต็ปจะอยู่ด้านล่าง B3 และ B4 ก็เช่นเดียวกัน



รูปที่ 4.32 : การลบ Branch

4.7.5 การลบ Double

Double จะเป็นตัวเชื่อมทรานสิชั่น 1 ตัวกับสเต็ปหลายๆสเต็ป การลบ Double แบ่งเป็น 3 กรณี ได้แก่ การลบ D0X คือการลบ Double ที่มีชื่อขึ้นต้นด้วย D0 การลบ D1X และการลบ D2X



รูปที่ 4.33 : กรณีต่างๆในการลบ Double

การลบ D0X จะมีผลต่อสเต็ปทางปลาย Double ทุกๆสเต็ป เริ่มต้นจะค้นหาสเต็ปทางปลายโดยไต่ไปตาม Double เมื่อเจอแล้วจึงตัดความสัมพันธ์ของสเต็ปเหล่านั้นกับทรานสิชั่น และขั้นสุดท้ายปรับภาพ Double และทรานสิชั่นให้ถูกต้อง การลบ D1X จะง่ายกว่าเพราะเป็นสเต็ปตัวสุดท้าย โดยค้นหาสเต็ปข้างใต้และตัดความสัมพันธ์กับทรานสิชั่น การลบ D2X จะคล้ายกับการลบ D0X แต่ไม่ต้องปรับภาพทรานสิชั่นใหม่

4.7.6 การลบ Link

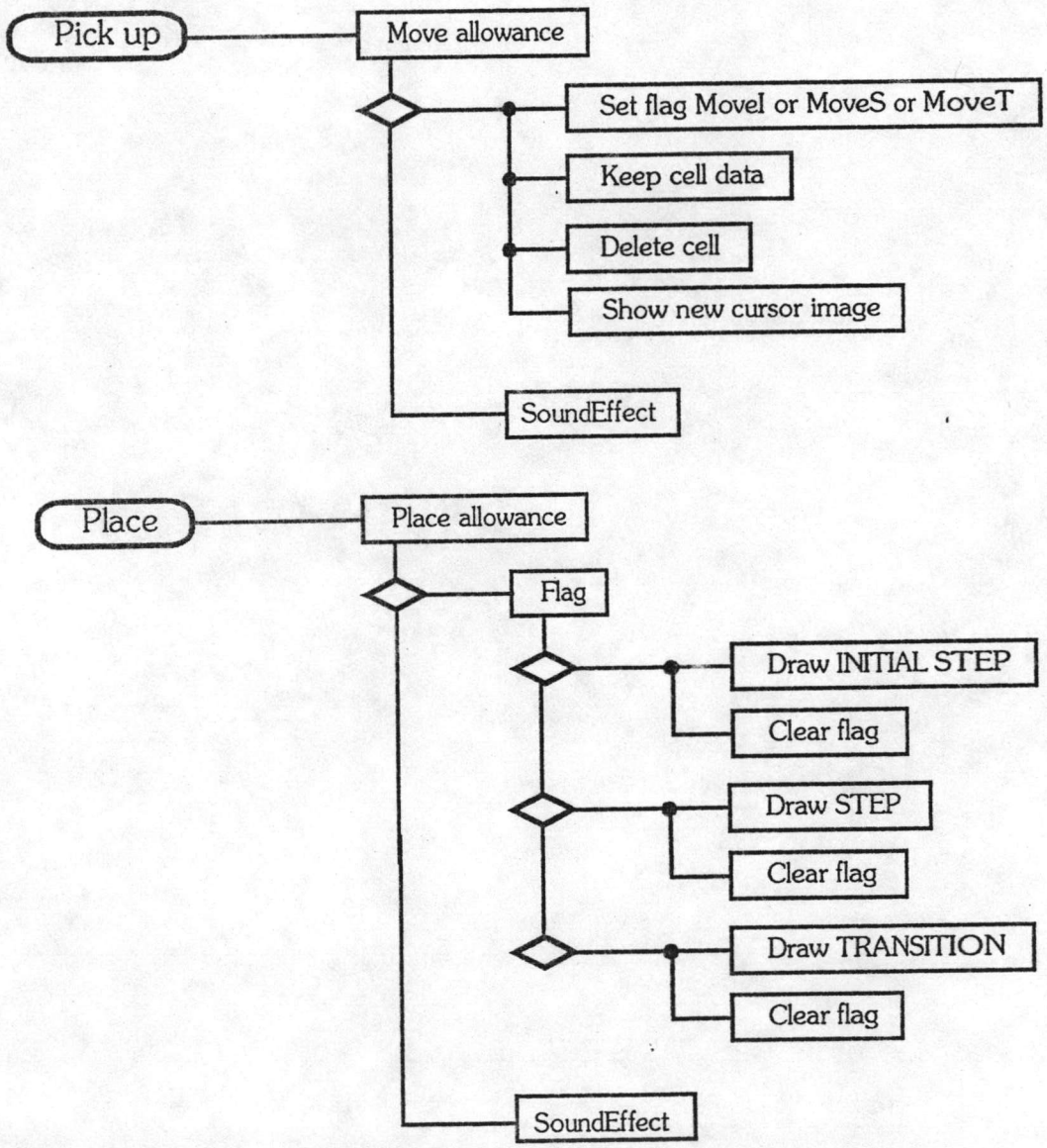
Link จะอยู่เป็นคู่ ดังนั้นการลบตัวใดตัวหนึ่ง ระบบจะต้องจัดการลบ Link อีกตัวให้ด้วย แบ่งเป็น 2 กรณี คือ การลบ L1 และการลบ L2 L1 คือ Link ที่ชี้สเต็ป อาจมาจากการกระโดดของทรานสิชั่นหลายๆตัวก็ได้ ดังนั้นการลบ L1 จะต้องลบ L2 ทุกตัวที่เกี่ยวข้อง ซึ่งจะลบทั้งภาพและข้อมูล ส่วนการลบ L2 จะต้องค้นหา L1 และวิเคราะห์ว่ามีการกระโดดมาจาก L2 ตัวนี้เพียงตัวเดียวหรือไม่ ถ้าใช่ลบทิ้งทั้งคู่ แต่ถ้าไม่ใช่ จะต้องตัดเฉพาะความสัมพันธ์ที่เกี่ยวข้องกันเท่านั้น ในการลบ Link ใดๆ จะต้องปรับความสัมพันธ์ของสเต็ปและทรานสิชั่นที่เกี่ยวข้องเสมอ

4.8 งานสนับสนุน

4.8.1 การเคลื่อนย้ายเซลล์

การเคลื่อนย้ายเซลล์นี้ออกแบบไว้สำหรับการเคลื่อนย้ายสเต็ปและทรานสิชั่นเท่านั้นเพราะองค์ประกอบแบบอื่น ดูเหมือนว่าการลบและเขียนใหม่จะเร็วกว่า การเคลื่อนย้ายเซลล์มีขั้นตอนดังนี้

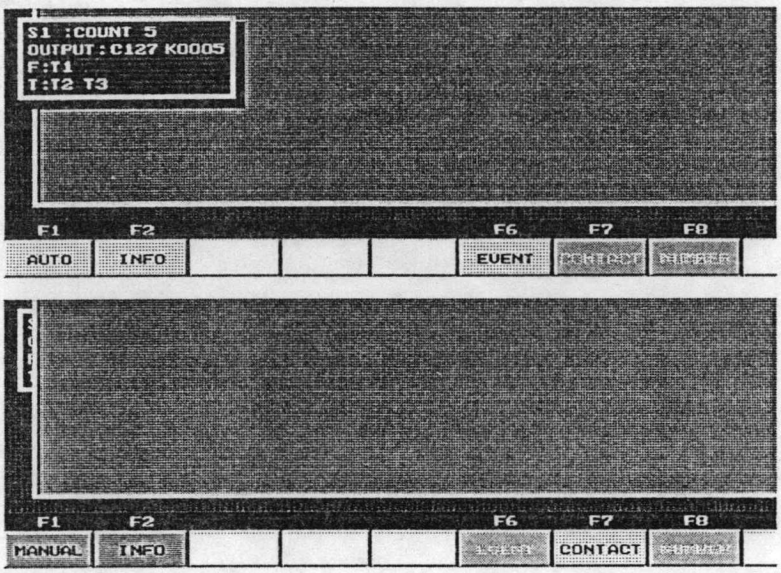
- เลื่อนเคอร์เซอร์ไปที่เซลล์นั้น
- ใช้คำสั่ง MOVE
- เลื่อนเคอร์เซอร์ไปที่ใหม่
- กด Enter



รูปที่ 4.34 : DSD ของการเคลื่อนย้ายเซลล์

4.8.2 <u>โหมด</u>

เป็นการรวบรวมโหมดการควบคุมและการแสดงภาพมารวมไว้ในเมนูเดียวกัน ได้แก่ การล็อคโหมดอัตโนมัติ(Auto) สำหรับ Branch และ Double การแสดงหรือไม่แสดงหน้าต่างรายละเอียดของเซลล์ และการแสดงชาร์ตในรูปแบบต่างๆ



รูปที่ 4.35 : เมนูโหมด

ก. การล็อคโหมดอัตโนมัติสำหรับ Branch และ Double

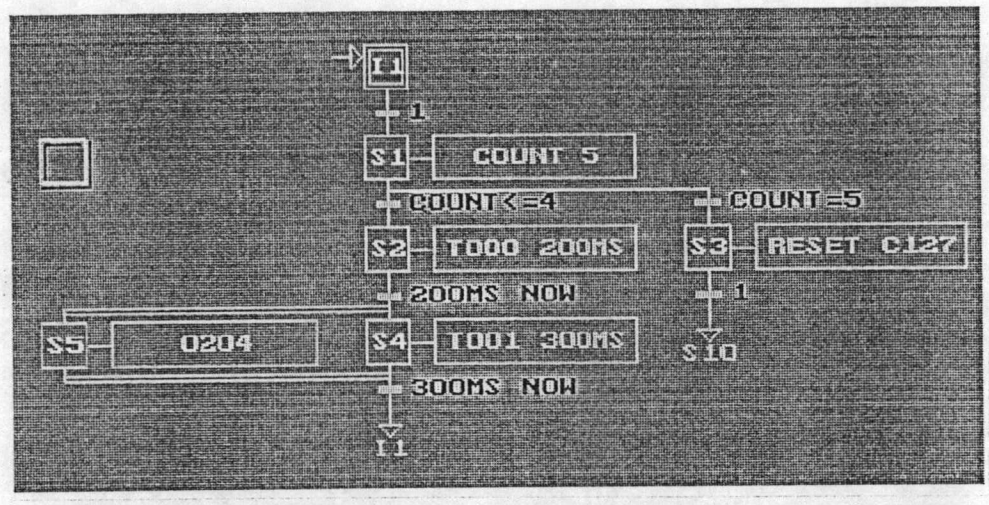
ตามที่กล่าวไปแล้วว่าในการวาดภาพ Branch และ Double จะแบ่งเป็น 2 กรณี คือแบบ Manual และ Auto โดยผู้ใช้จะต้องเลือกว่าจะวาดภาพแบบไหน ซึ่งอาจทำให้เสียเวลาไปบ้าง กอร์ปกับการวาด Branch ส่วนใหญ่ สามารถจัดการได้ในโหมด Auto จึงออกแบบให้สามารถล็อคโหมดอัตโนมัติไว้ได้โดยการกดปุ่ม F1 ซึ่งเป็นปุ่มกดแบบท็อกเกิล (Toggle) ดังแสดงในรูปที่ 4.35

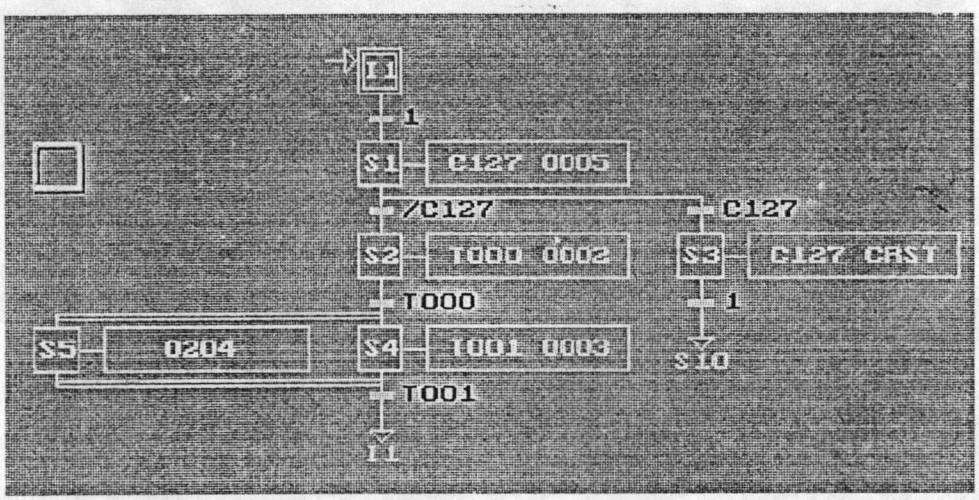
ข. การแสดงหน้าต่างรายละเอียดของเซลล์

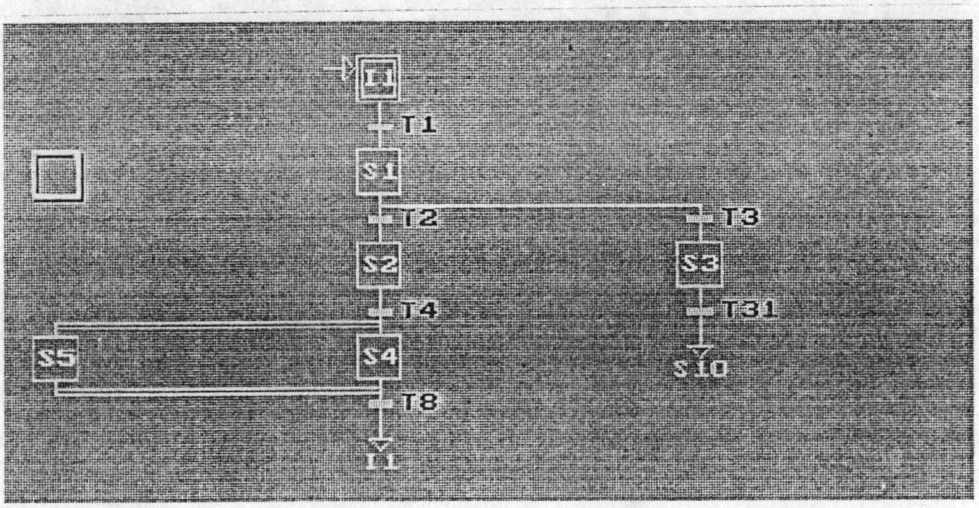
ควบคุมโดยการกด F2 ซึ่งทำงานแบบท็อกเกิล รูปที่ 4.35 บนและล่าง แสดงผลจากการควบคุม

ค. รูปแบบการแสดงภาพ

มี 3 รูปแบบได้แก่ การแสดงหมายเหตุของสเต็ปและทรานสิชั่น (EVENT) , การแสดงหน้าสัมผัสของรีเลย์ที่ถูกควบคุมโดยสเต็ปและทรานสิชั่น (CONTACT) และการแสดงเฉพาะหมายเลขของสเต็ปและทรานสิชั่น (NUMBER)







รูปที่ 4.36 : รูปแบบการแสดงภาพต่างๆ

4.8.3 แฟ้มข้อมูล

การจัดการแฟ้มข้อมูลแบ่งงานเป็น 4 รายการ ได้แก่ การเก็บชาร์ตลงแฟ้มข้อมูล (SAVE) การเก็บชาร์ตลงแฟ้มข้อมูลโดยระบุชื่อแฟ้ม (SAVE AS), การอ่านแฟ้มข้อมูลขึ้นมาใช้งาน (LOAD) และการเคลียร์ชาร์ตเพื่อเริ่มเขียนชาร์ตใหม่ (NEW)

ก. การเก็บชาร์ตลงแฟ้มข้อมูล

```
Procedure SaveOperation;
var FRow,FCol : byte;
Begin
        Rewrite(FCFile);
        For FRow:= 1 to MaxRow do
            For FCol:=1 to MaxCOl do
                If Chart[FRow,FCol].Func<>#00 then
                   Write(FCFile,Chart[FRow,FCol]);
        Write(FCFile,EOC);
        Close(FCFile);
        Message('Finish',a);
        SaveOK:=true;
End;

Procedure Save;
Var
    SaveFile  : string[12];
Begin
  If not SaveOk then
     begin
       If (CurrentFile<>'') then
          begin
             SaveFile:=CurrentFile+'.CHT';
             Assign(FCFile,SaveFile);
             {$I-} Reset(FCFile); {$I+}
             If IOResult=2
                then Message('File Not Found',a)
                else SaveOperation;
          end
          else begin
                Get('File',CurrentFile);
                If (CurrentFile<>'£šÛ') then
                 begin
                  SaveFile:=CurrentFile+'.CHT';
                  Assign(FCFile,SaveFile);
                  {$I-} Reset(FCFile); {$I+}
                  If IOResult=0
                    then begin
                          Message('File Exist, Do you want to overwrite? [Y/N]',a);
                          If a in ['Y','y']
                             then SaveOperation
                             else begin
                                   Message('SAVE cancel',a);
                                   CurrentFile:='';
                                  end;
                         end
                    else SaveOperation;
                 end
                 else CurrentFile:=OldFile;
               end;
     end
     else Message('Already',a);
End;
```

รูปที่ 4.37 : การจัดเก็บชาร์ต

เนื่องจากชาร์ตที่แสดงบนจอภาพ เก็บข้อมูลในลักษณะเป็นตารางจึงมีทั้งเซลล์ที่มีข้อมูลและไม่มี ข้อมูล หากจัดเก็บทั้งชาร์ตลงแฟ้มข้อมูล ก็จะทำให้ได้แฟ้มข้อมูลที่มีขนาดใหญ่โดยไม่จำเป็น จึงได้ออกแบบโครงสร้างข้อมูลเผื่อการจัดการในเรื่องนี้ไว้แล้ว กล่าวคือในโครงสร้างข้อมูลจะมีฟิลด์ Coor ซึ่งระบุตำแหน่งของเซลล์นั้นบนชาร์ต ดังนั้นในการจัดเก็บจึงสามารถออกแบบให้จัดเก็บชาร์ตเฉพาะเซลล์ที่มีข้อมูลเท่านั้น โดยค้นหาตั้งแต่มุมซ้ายบนชาร์ตไปจนถึงมุมขวาล่าง เซลล์ที่มีข้อมูล คือเซลล์ที่ฟิลด์ Func มีค่า แฟ้มข้อมูลที่จัดเก็บชาร์ต ได้ตั้งนามสกุลให้เป็น ".CHT"

ข. การอ่านแฟ้มข้อมูลขึ้นมาใช้งาน

เมื่อผู้ใช้เลือกใช้งานคำสั่งนี้ระบบจะถามชื่อแฟ้มข้อมูลที่ต้องการอ่าน จากนั้นจะไปค้นหา หากค้นหาเจอก็จะอ่านข้อมูลขึ้นมาทีละเซลล์ วาดภาพเซลล์นั้นบนจอภาพตามตำแหน่งที่ระบุ พร้อมกับเขียนข้อมูลลงบนตาราง

```
Procedure Load;
Var Buffer   : Cell;
    Thing    : ElementList;
    LoadFile : string[12];

    Procedure LoadOperation;
    Begin
     OldFile:=CurrentFile;
     Get('File',CurrentFile);
     If (CurrentFile<>'£šÛ') then
      begin
       LoadFile:=CurrentFile+'.CHT';
       Assign(FCFile,LoadFile);
       {$I-} Reset(FCFile); {$I+}
       If IOResult=2
          then begin
                 Message('File Not Found',a);
                 CurrentFile:=OldFile;
               end
          else begin
                 ClearChart;
                 Repeat
                    Read(FCFile,Buffer);
                    If (Buffer.Element<>'EOC') then
                    begin
                         Thing:=Str2Element(Buffer.Element);
                         Chart[Buffer.Coor[1],Buffer.Coor[2]]:=Buffer;
                         If (Buffer.Coor[1]<=MaxScreenRow) and
                            (Buffer.Coor[2]<=MaxScreenCol) then
                            Draw(Buffer.Coor[1],Buffer.Coor[2],Buffer.Coor[1],Buffer.Coor[2],Thing);
                    end;
                 Until (Buffer.Element='EOC');
                 Close(FCFile);
                 UpdateCursor;
                 SaveOK:=true;
               end;
      end
      else CurrentFile:=OldFile;
    End;

Begin
  If SaveOk
     then LoadOperation
     else begin
            Message('Not save yet, Do you want to overwrite? [Y/N]',a);
            If a in ['Y','y']
               then LoadOperation
               else Message('LOAD cancel',a);
          end;
End;
```

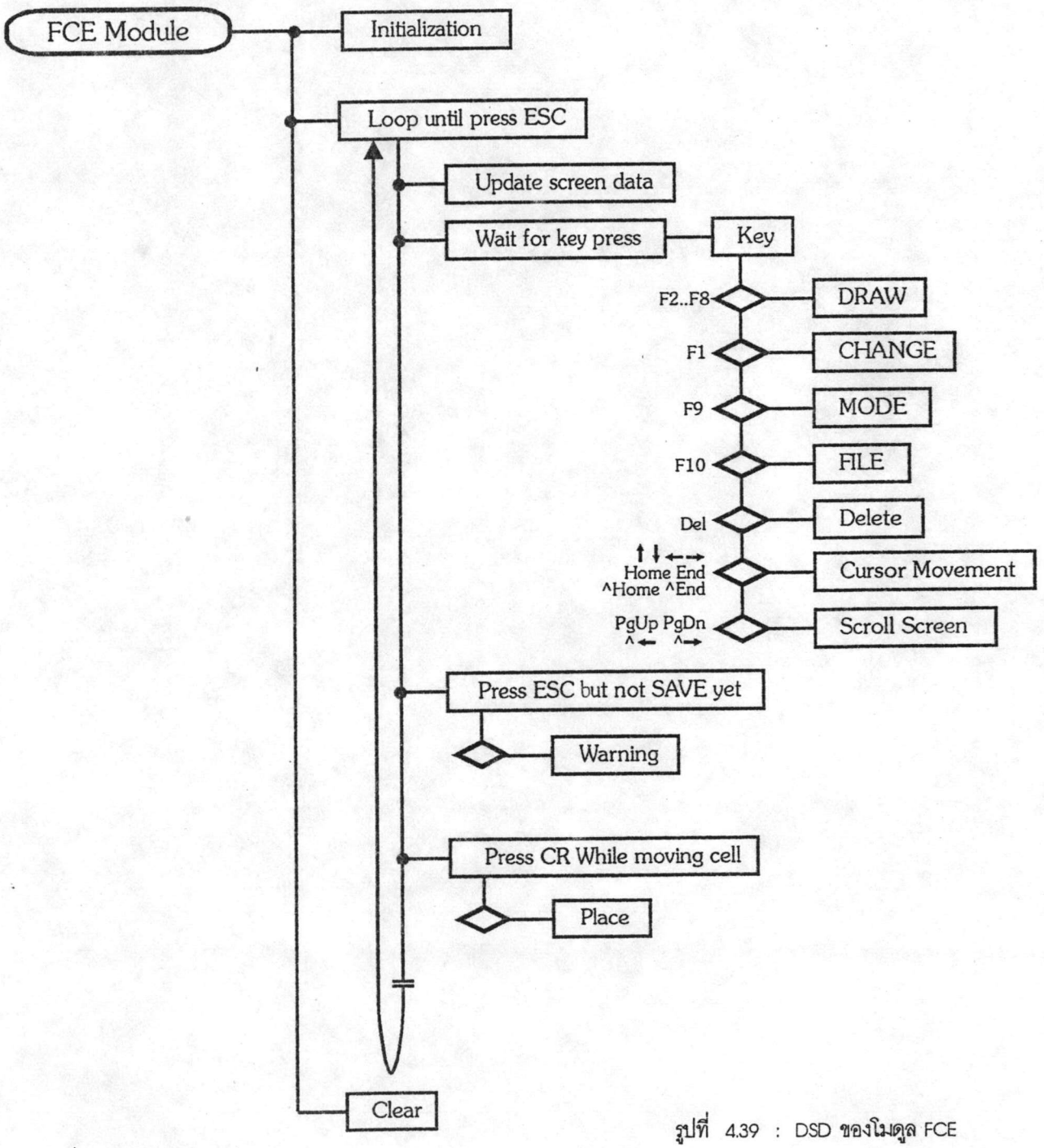
รูปที่ 4.38 : การอ่านแฟ้มข้อมูลขึ้นมาใช้งาน

ค. การเคลียร์ชาร์ต

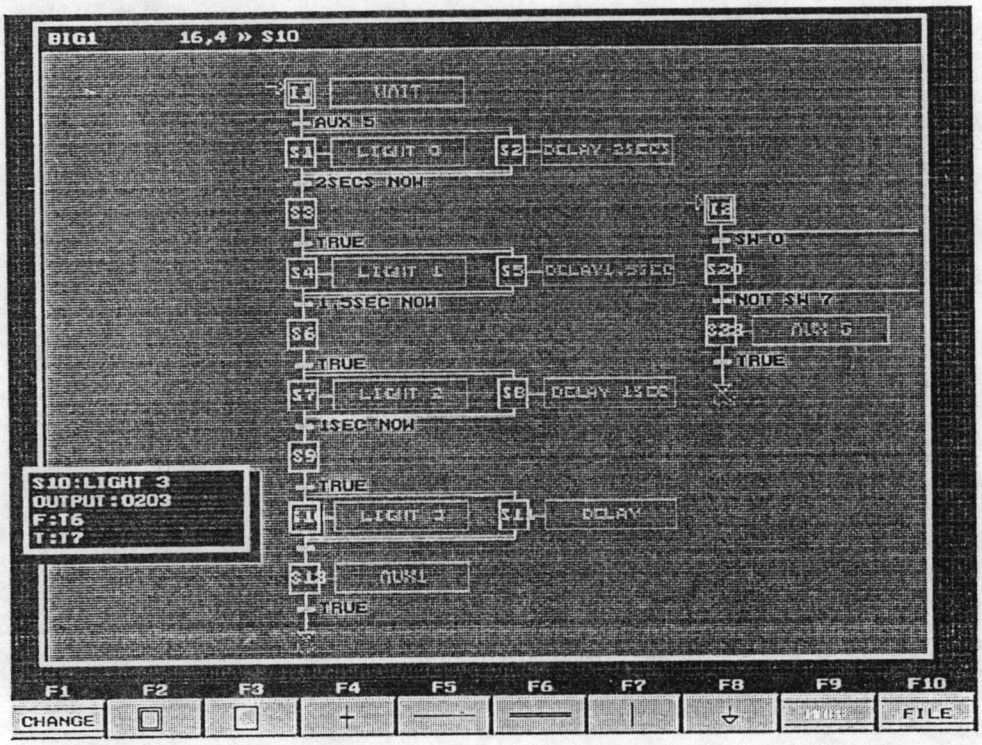
ใช้สำหรับการเริ่มเขียนชาร์ตใหม่ โดยจะลบภาพบนจอภาพ และข้อมูลบนตารางทิ้งไป และจะย้ายเคอร์เซอร์ไปไว้ที่แถวที่ 1 และคอลัมน์ที่ 1

4.9 สรุป

งานทั้งหมดที่กล่าวถึงในบทนี้เป็นงานในโมดูล Function Chart Editor (FCE) ทั้งสิ้น การจัดการภายใน โมดูลแสดงดังแผนภาพในรูปที่ 4.39 ซึ่งงานหลักก็จะเป็นการให้บริการต่อผู้ใช้ในเรื่องการวาดรูป



รูปที่ 4.39 : DSD ของโมดูล FCE



รูปที่ 4.40 : จอภาพขณะใช้งาน