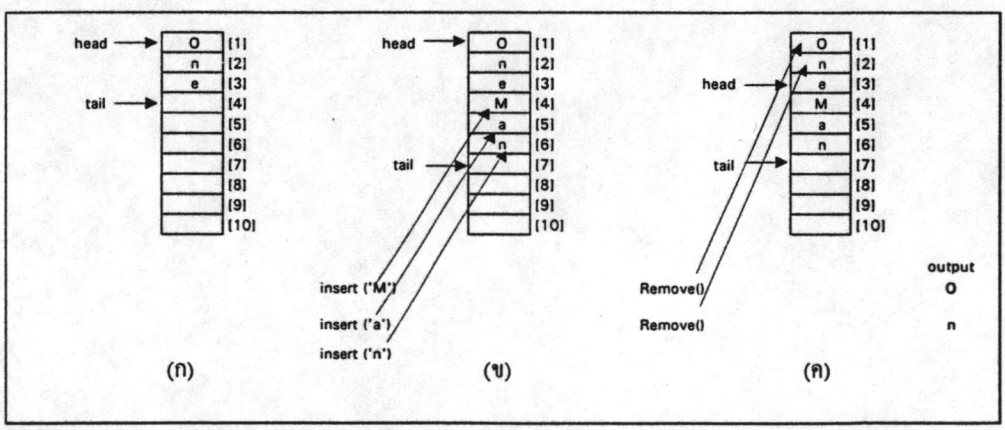
บทที่ 3

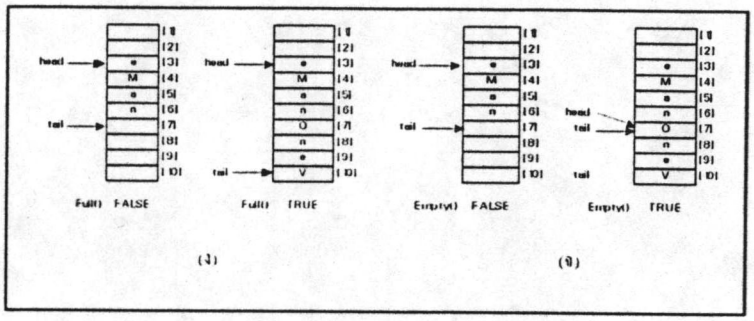
โปรแกรมจัดการข้อมูลแถวคอย และประโยชน์ของการใช้ข้อมูลแถวคอย

ข้อมูลแถวคอยและการใช้ประโยชน์

ข้อมูลแถวคอย (queue) เป็นโครงสร้างข้อมูลแบบลิสต์ ที่มีการใช้งานในลักษณะเข้าก่อนออกก่อน บางครั้งจะเรียกว่า First-In-First-Out list (FIFO) เหมือนการต่อแถวซื้อ บัตรชมภาพยนตร์ในชีวิตประจำวัน ข้อมูลที่เข้ามาใหม่จะถูกนำไปต่อท้ายแถวตามลำดับ และการดึงข้อมูลออกมาใช้ก็จะดึงข้อมูลที่อยู่หัวแถวก่อน (G.M.Schneider, S.C.Bruell, 1987) ฟังก์ชั่นของการจัดการข้อมูลแถวคอย ได้แก่ การนำข้อมูลเข้าไปยังแถวคอย (Insert) การนำข้อมูลออกจากแถวคอย (Remove) การตรวจสอบว่าแถวนั้นเต็มหรือไม่ (Queue Full) การตรวจสอบว่าแถวคอยนั้นว่างหรือไม่ (Queue Empty) ในรูปที่ 3.1 แสดงลักษณะของแถวคอยและฟังก์ชั่นต่าง ๆ ที่ใช้ในการจัดการแถวคอย โดยรูป 3.1 (ก) แสดงลักษณะของแถวคอย รูป 3.1 (ข) แสดงฟังก์ชั่นการนำข้อมูลต่อท้ายแถวคอย รูป 3.1 (ค) แสดงการนำข้อมูลออกจากแถวคอย รูป 3.1 (ง) แสดงการตรวจสอบแถวคอยเต็ม รูป 3.1 (จ) แสดงการตรวจสอบแถวคอยว่าง

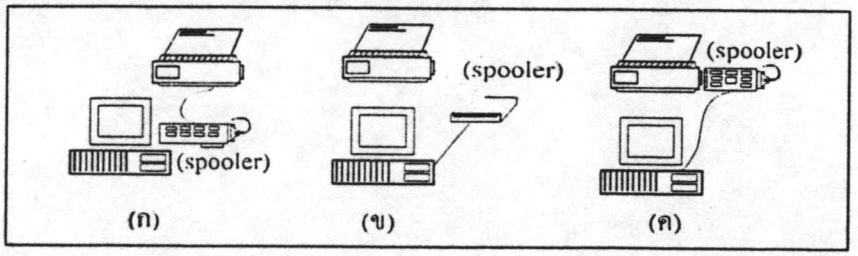


รูปที่ 3.1 ข้อมูลแถวคอยและตัวอย่างการทำงาน (ก) ลักษณะของข้อมูลแถวคอย (ข) การเพิ่มข้อมูลลงในแถวคอย (ค) การนำข้อมูลออกจากแถวคอย (ง) การตรวจสอบแถวคอยเต็ม (จ) การตรวจสอบแถวคอยว่าง



รูปที่ 3.1 ข้อมูลแถวคอยและตัวอย่างการทำงาน (ต่อ)

ประโยชน์ของการใช้ข้อมูลแถวคอย ได้ถูกนำมาใช้มากในโปรแกรมระบบเช่นใน Resource Management ทุกประเภทในระบบปฏิบัติการ (Operating System) เช่น โปรเซส(process), หน่วยความจำ (memory), อุปกรณ์บันทึกข้อมูล (storage), อุปกรณ์รับข้อมูล/แสดงผล (I/O) ฯลฯ บัฟเฟอร์ของแป้นพิมพ์ (Keyboard Buffer) ฯลฯ เป็นต้น ในโปรแกรมประยุกต์ก็มีการใช้งานข้อมูลแถวคอยเช่น โปรแกรมจำลองการทำงาน (Simulation) ที่มีการขอใช้บริการและการให้บริการ ซึ่งการใช้บริการมักช้ากว่าการขอบริการ เป็นต้น ประโยชน์ในการใช้ข้อมูลแถวคอยอย่างหนึ่งซึ่งจะยกเป็นตัวอย่างในวิทยานิพนธ์นี้ ได้แก่ สปูลเลอร์สำหรับเครื่องพิมพ์ (Printer Spooler) ซึ่งใช้ช่วยเก็บบันทึกข้อมูลจากระบบคอมพิวเตอร์ ก่อนที่จะส่งข้อมูลเหล่านั้นต่อไปยังเครื่องพิมพ์ เนื่องจากความเร็วในการทำงานของเครื่องพิมพ์แบบขนานนั้น จะใช้เวลาสำหรับการส่งแต่ละอักษรในระดับ 10-100 ไมโครวินาที ในขณะที่ความเร็วในการพิมพ์อยู่ประมาณ 1/300 วินาที (เครื่องพิมพ์ดอทแมททริกซ์) และเมื่อมีการขึ้นบรรทัดใหม่หรือการขึ้นแผ่นกระดาษใหม่ก็จะใช้เวลาในระดับไม่น้อยกว่า 1/10 วินาที จนถึงระดับวินาที หากคอมพิวเตอร์จะต้องรอให้เครื่องพิมพ์ พิมพ์อักษรเสร็จก่อนจึงจะส่งอักษรถัดไปให้แก่เครื่องพิมพ์ก็จะทำให้เสียเวลาโดยเปล่าประโยชน์ ดังนั้นจึงมีการจัดสร้างสปูลเลอร์สำหรับเครื่องพิมพ์เพื่อรับข้อมูลจากคอมพิวเตอร์เก็บบันทึกไว้ก่อนส่งให้แก่เครื่องพิมพ์ โดยอาจอยู่ในระบบคอมพิวเตอร์เองอยู่ในเครื่องพิมพ์ หรือจัดสร้างแยกต่างหากไว้ก็ได้ (รูปที่ 3.2 (ก) - (ค)) โดยหลักการของสปูลเลอร์สำหรับเครื่องพิมพ์นั้นก็คือการใช้ข้อมูลแถวคอยรับข้อมูลจากคอมพิวเตอร์มาเพิ่มในแถวคอย และนำข้อมูลที่เก็บในแถวคอยส่งไปยังเครื่องพิมพ์เมื่อเครื่องพิมพ์พร้อมโดยมีความอิสระในการจัดเก็บข้อมูลในระดับหนึ่ง



รูปที่ 3.2 การใช้งานสปูลเลอ (ก) อยู่รวมกับคอมพิวเตอร์ (ข) อยู่แยกต่างหาก (ค) อยู่ในเครื่องพิมพ์

1. <u>โครงสร้างและอัลกอริธึมของข้อมูลแถวคอย</u>

ดังที่กล่าวในตอนต้นมาแล้วว่าข้อมูลแถวคอยนั้นนับที่เป็นลิสต์ชนิดหนึ่ง ดังนั้นการจัดโครงสร้างของข้อมูลแถวคอยสามารถทำได้ทั้งการใช้ข้อมูลอาเรย์ (array) และข้อมูลลิงก์ลิสต์ (linked list) ในที่นี้จะกล่าวเฉพาะการใช้ข้อมูลอาเรย์ เท่านั้น โครงสร้างของข้อมูลแถวคอยนั้นนอกจากแถวคอยเองแล้วก็ยังมีตัวแปรสำหรับชี้ตำแหน่ง หัว (head) และท้าย (tail) ของแถวคอยเพื่อชี้ตำแหน่งที่จะนำข้อมูลออกไปใช้ (head) และตำแหน่งที่จะนำข้อมูลใหม่เพิ่มเข้าไป (tail) เนื่องจากการทำงานของโครงสร้างข้อมูลแถวคอยนั้นใช้ตัวชี้บอกตำแหน่งหัวและท้ายของแถวคอยไม่ได้ใช้วิธีขยับ ข้อมูลในแถวเหมือนแถวคอยในชีวิตประจำวัน ดังนั้นในโครงสร้างของข้อมูลแถวคอยจะประกอบด้วยข้อมูล 3 ตัวได้แก่ แถวคอย (Queue) ตัวชี้หัวแถว (Head) และตัวชี้ท้ายแถว (Tail) สามารถเขียนโครงสร้างในรูปแบบของภาษาปาสคาลได้เป็น

```
VAR          Q   :   ARRAY
TYPE   LIST    = ARRAY[1..<max_element>] OF <datatype> ;
       INDEX   = (1..<max_element>];
       QUEUE_TYPE    = RECORD
                       QUEUE : LIST;
                       HEAD,TAIL : INDEX;
                       END
```

1.1 <u>อัลกอริธึมการนำข้อมูลเข้าและการนำข้อมูลออกไปใช้</u> การนำข้อมูลไปในแถวคอยสามารถเขียนอย่างง่ายได้เป็น

```
IF  not  Q_Full (Q) THEN BEGIN
         Q. QUEUE [ Q.TAIL] := <data>;
         Q.TAIL             := Succ (QTAIL) ;
END;
```

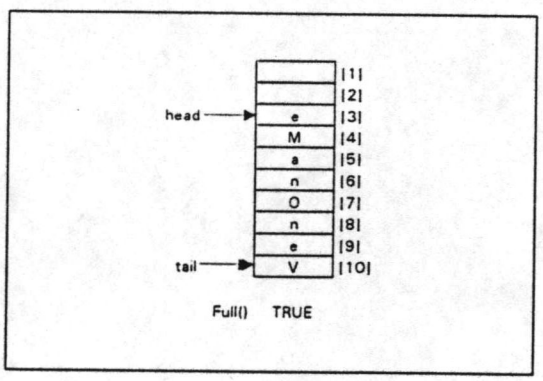
โดย Q_FULL เป็นฟังก์ชั่นที่ใช้ตรวจสอบว่ายังมีที่ว่างเพื่อจะนำข้อมูลเข้าเก็บในแถวคอยนั้นได้อีกหรือไม่

อัลกอร์ซึมการนำข้อมูลออกจากแถวคอยสามารถเขียนอย่างง่ายได้เป็น

```
IF not Q-Empty (Q) THEN BEGIN
        DATA : = Q.QUEUE [Q.HEAD]
        Q.HEAD := Succ (Q.HEAD);
END;
```

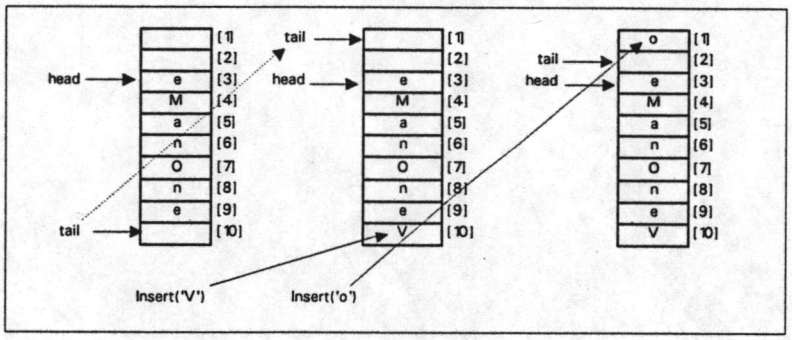
ปัญหาของการใช้อาเรย์และแถวคอยแบบวงแหวน

จากอัลกอริธึมแบบง่ายของการนำข้อมูลเข้าสู่และออกจากแถวคอยในหัวข้อ 3.1 สามารถยกตัวอย่างการนำข้อมูลเข้าและออกได้ดังรูป 3.1 ซึ่งอาจทำให้เกิดปัญหาเมื่อการเพิ่มค่าของ TAIL จนกระทั่งถึงค่า max-element แล้วจะไม่สามารถรับข้อมูลเพิ่มได้อีกเนื่องจากจะเกิดขอบเขตของตัวแปรอาเรย์ที่กำหนดไว้ถึงแม้ว่าในส่วนต้นของอาเรย์จะถูกดึงข้อมูลออกไปใช้แล้วก็ตาม ดังรูปที่ 3.3



รูปที่ 3.3 ปัญหาแถวคอยอย่างง่ายจะแสดงว่าแถวคอยเต็มเมื่อ TAIL ชี้ที่ตำแหน่งสูงสุด

ปัญหาดังกล่าวอาจมีวิธีแก้ไขได้หลายวิธี แต่วิธีที่นิยมใช้กันก็คือการสร้างแถวคอยแบบวงแหวนดังรูปที่ 3.4 หากเพิ่มค่าตัวชี้ถึงจุดสุดท้ายแล้วก็จะเริ่มย้อนไปตำแหน่งแรกใหม่ ในการจัดสร้างโครงสร้างข้อมูลแถวคอยแบบวงแหวนนี้ โครงสร้างข้อมูลก็ยังคงเป็นโครงสร้างอาเรย์เช่นเดิม แต่วิธีการที่จะทำให้เกิดวงแหวนคือการเพิ่มค่าของตัวชี้ต้นและท้าย จะต้องเปลี่ยนค่าไปเป็นค่าเริ่มต้นเมื่อชี้เกินตำแหน่งสุดท้ายแล้ว ซึ่งอาจเขียนอัลกอริธึมโดยใช้การคำนวณมอดูโล (modulo) ได้เป็นดังนี้



รูปที่ 3.4 แถวคอยวงแหวนเมื่อเพิ่มข้อมูลหลังจาก TAIL มีค่าสูงสุดแล้ว ค่า TAIL จะเริ่มที่ค่าต่ำสุดใหม่

```
FUNCTION NEXT_INDEX (IND : INDEX) : INDEX ;
BEGIN
        NEXT_INDEX := SUCC(IND) MOD (max_element);
END
```

ข้อจำกัดของการใช้โมดูโลนี้คือการกำหนด อาเรย์ที่ใช้เป็นลิสต์ต้องเริ่มจากตัวชี้ 0 ถึงตัวชี้เป็น max_element-1 (มีจำนวนข้อมูลในลิสต์เป็นจำนวน max_element ตัว)

2. <u>การตรวจสอบแถวคอยเต็มหรือว่างเปล่า</u>

การตรวจสอบแถวคอยเต็มหรือว่างเปล่าในกรณีของการใช้อาเรย์อย่างง่ายอาจทำได้โดย

```
Q_EMPTY  :=  Q.HEAD  = Q.TAIL
Q_FULL     := Q.TIAL = <max_element>
```

แต่หากเป็นแถวคอยแบบวงแหวนซึ่งตัวชี้ท้ายแถวและหัวแถวจะเปลี่ยนเป็นค่าเริ่มต้นได้ถ้าเกินตำแหน่งสูงสุดและในตอนต้นแถวนั้นว่างอยู่ จะต้องเปลี่ยนแปลงวิธีการตรวจสอบใหม่เป็น

```
Q_EMPTY        := Q.HEAD = Q.TAIL;
Q_FULL         :=  NEXT_INDEX (Q.TAIL) = Q.HEAD;
```

วิธีการตรวจสอบแบบนี้จะเหลือเนื้อที่ในลิสต์ 1 ช่อง เมื่อสถานะของแถวคอยจากฟังก์ชั่น Q_FULL เป็น TRUE หรือแสดงว่าแถวคอยนั้นเต็ม

3. <u>สรุปโครงสร้างและอัลกอริธึมจัดการแถวคอย</u>

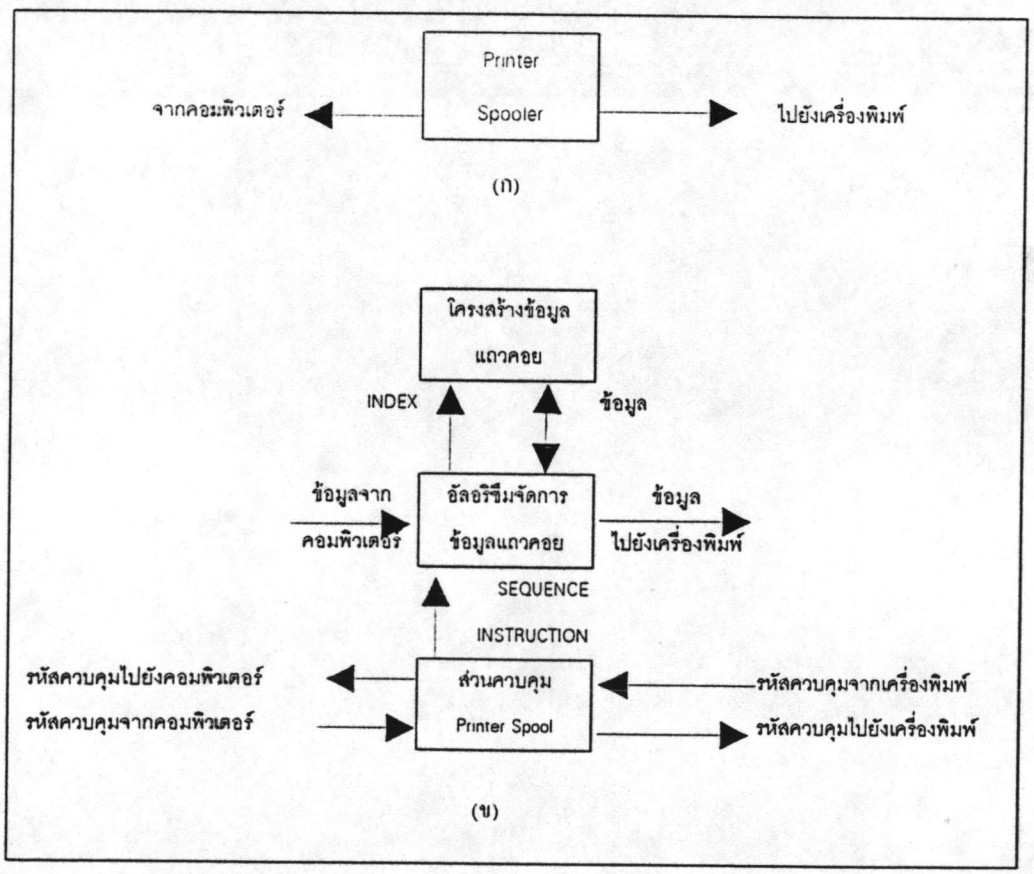
```
TYPE
        LIST     = ARRAY [0..<max_element -1>] OF <dbase datatype>;
        INDEX  = 0..<max_element -1> ;
        QUEUE_STRUCTURE     = RECORD
```

```
                    QUEUE       : LIST;
                    HEAD,TAIL: INDEX;
                END;
FUNCTION NEXT_INDEX (IND : INDEX) : INDEX ;
BEGIN
        NEXT_INDEX := SUCC(IND) MOD <max_element >;
END;
FUNCTION Q_EMPTY (Q:QUEUE_STRUCTURE):BOOLEAN;
BEGIN
        Q_EMPTY := Q.HEAD=Q.TAIL;
END;
FUNCTION Q_FULL (Q:QUEUE_STRUCTURE):BOOLEAN;
BEGIN
        Q_FULL := NEXT_INDEX (Q.TAIL) = Q.HEAD;
END;
PROCEDURE Q_INSERT (VAR Q: QUEUE_STRUCTURE; DAT :<base datatype>);
BEGIN
        IF NOT Q_FULL THEN BEGIN
                Q.QUEUE[Q.TAIL] := DAT;
                Q.TAIL      := NEXT_INDEX (Q.TAIL);
                END
        ELSE      { in case of QUEUE FULL}
                <some errors instruction>
END;
PROCEDURE Q_REMOVE (VAR Q: QUEUE_STRUCTURE;VAR DAT :<base
datatype>);
BEGIN
        IF NOT Q_EMPTY THEN BEGIN
          DAT := Q.QUEUE [Q.HEAD];
          Q.HEAD := NEXT_INDEX (Q.HEAD);
          END
        ELSE {in case of QUEUE EMPTY}
            <some errors instruction>
END;
```

```
PROCEDURE Q_INITIAL (VAR Q:QUEUE_STRUCTURE);
BEGIN
        Q.HEAD          := 0;
        Q.TAIL  := 0;
END;
```

4. <u>การใช้ข้อมูลแถวคอยในสปูลเลอร์สำหรับเครื่องพิมพ์</u>

หลักการสำคัญของสปูลเลอร์สำหรับเครื่องพิมพ์ดังที่กล่าวในตอนต้นมาแล้วว่า เป็นการใช้ข้อมูลแถวคอยเพื่อเก็บบันทึกข้อมูลจากคอมพิวเตอร์ลงในแถวคอย และตรวจสอบว่าเครื่องพิมพ์พร้อมรับข้อมูลหรือไม่หากพร้อมก็จะส่งข้อมูลจากแถวคอยไปยังเครื่องพิมพ์ตามลำดับที่ได้รับจากคอมพิวเตอร์ ในลักษณะที่กล่าวมาอาจแสดงในรูปผังภาพ (block diagram) ได้ดังรูป 3.5



รูปที่ 3.5 ผังภาพวงจรปรินเตอสปูลเลอร์ (ก) แผนผังภายนอก (ข) ส่วนประกอบภาคต่างๆ

ในภาพ 3.5 (ก) คือแผนผังกล่อง แสดงระบบสปูลเลอร์สำหรับเครื่องพิมพ์ที่ใช้เชื่อมต่อระหว่างคอมพิวเตอร์กับเครื่องพิมพ์ ในภาพ 3.5 (ข) แสดงผังภาพของส่วนประกอบหลัก ๆ ของสปูลเลอร์สำหรับเครื่องพิมพ์ซึ่งแบ่งออกได้เป็น 3 ส่วนหลัก ๆ ได้แก่ โครงสร้างข้อมูลแถวคอย เป็นส่วนที่ใช้บันทึกข้อมูลหรือลิสต์ซึ่งจะถูกอ้างถึงด้วย INDEX และรับ-ส่ง ข้อมูลกับส่วนที่สองคือ อัลกอริธึม จัดการข้อมูลแถวคอย และส่วนสุดท้ายคือส่วนควบคุมที่สร้างให้เกิดการทำงานเป็นสปูลเลอร์สำหรับเครื่องพิมพ์อาจเรียกส่วนนี้ว่า Printer Spool Controller ซึ่งจะรับ-ส่งรหัสควบคุมกับทั้งคอมพิวเตอร์ และเครื่องพิมพ์ และสร้างลำดับของคำสั่งที่จะเรียกใช้อัลกอริธึมของแถวคอย รหัสควบคุมที่ใช้ติต่อกับคอมพิวเตอร์หรือเครื่องพิมพ์มีรายละเอียดอย่างไรนั้นจะขึ้นกับว่าสปูลเลอร์สำหรับเครื่องพิมพ์นี้อยู่ ณ ตำแหน่งใด ได้แก่อยู่ในคอมพิวเตอร์ที่ ซีพียู หลักของคอมพิวเตอร์ สามารถอ้างถึงเหมือนเป็นอุปกรณ์เอาต์พุท-อินพุท ได้โดยตรง รหัสควบคุมที่จะต่อเข้ากับคอมพิวเตอร์ก็จะเป็นลักษณะรหัสที่ใช้เชื่อมต่อกับระบบบัสของคอมพิวเตอร์นั้น ส่วนรหัสควบคุมที่ต่อกับเครื่องพิมพ์ก็จะเป็นรหัสที่ต่อเชื่อมช่องทางขนานสำหรับต่อเครื่องพิมพ์ (Parrallel Interface) หากอยู่ที่เครื่องพิมพ์ รหัสควบคุมติดต่อกับเครื่องคอมพิวเตอร์ก็จะเป็นรหัสต่อเชื่อมช่องทางขนานซึ่งส่วนใหญ่จะใช้ตามมาตรฐานเซนทรอนิก (Centronic) ส่วนรหัสควบคุมติดต่อเครื่องพิมพ์จะเป็นรหัสสัญญาณที่ติดต่อกับระบบบัสหลัก ภายในเครื่องพิมพ์แต่ละรุ่น หากสปูลเลอแยกอยู่ต่างหากจากทั้งคอมพิวเตอร์และเครื่องพิมพ์ รหัสควบคุมที่จะติดต่อก็จะเป็นรหัสควบคุม ที่เชื่อมต่อช่องทางขนานตามมาตรฐานเซนทรอนิก (ในการวิจัยนี้ใช้เฉพาะช่องทางขนาน)

การจัดสร้างสปูลเลอร์สำหรับเครื่องพิมพ์สามารถจัดทำได้ทั้งฮาร์ดแวร์, ซอฟต์แวร์และผสมกัน และสามารถเขียนอัลกอริธึมในส่วนของ Printer Spool Controller ได้ดังนี้

```
VAR SPOOL : QUEUE STRUCTURE;

BEGIN
    Q_INTTIAL (SPOOL);
    REPEAT
        IF HAVE DATA_IN AND NOT Q_FULL THEN BEGIN
            Q_INSERT (SPOOL, DATA_IN);
            CLEAR_DATA_IN;
        END
        ELSE IF NOT Q_EMPTY AND OUTPUT_READY THEN BEGIN
            Q_REMOVE (SPOOL, DATA_OUT);
```

```
                SEND_OUT_DATA;
                END
        UNTIL FALSE

    END
    FUNCTION HAVE_DATA_IN :BOOLEAN;
    BEGIN
        HAVE_DATA_IN := INPUT_DATA_ARRIVED;
    END;
    PRODUCRE CLEAR_DATA_IN;
    BEGIN
        INPUT_DATA_ARRIVED := FALSE;
        SEND_ACKNOWLEDBE; {implementation dependence}
    END;
    FUNCTION OUTPUT_READY : BOOLEAN;
    BEGIN
        OUTPUT_READY := ACKNOWLEDGE_ARRIVED AND NOT BUSY_OCCURED;
    END;
    PROCEDURE SEND_OUT_DATA;
    BEGIN
        ACKNOWLEDGE_ARRIVED := FALSE;
        OUT_DATA := DATA.OUT;          {implementation dependent}
        SEND_STROBE;                   {implementation dependent}
    END;
```

ในส่วนข้างบนนี้เป็นอัลกอริธึมโดยทั่วไปแต่ยังมีส่วนหนึ่งที่เกี่ยวข้องกับชนิดของสปูลเออร์สำหรับเครื่องพิมพ์ว่าจะอยู่ที่ใดในระบบในที่นี้จะยกกรณีเฉพาะเมื่อแยกอยู่ต่างหากจากคอมพิวเตอร์และเครื่องพิมพ์โดยติดต่อกับทั้งคอมพิวเตอร์และเครื่องพิมพ์ทางช่องทางติดต่อแบบขนานส่วนของอัลกอริธึมที่เกี่ยวข้องกับชนิดนั้นจะแสดงดังต่อไปนี้ซึ่งมีบางส่วนเป็นอัลกอริธึมที่ต้องตอบสนองแบบเรียลไทม์ (realtime) จึงใช้คำว่า REALTIME นำหน้า อัลกอริธึมนั้นเพิ่มเติมจากรูปแบบมาตรฐานของภาษาปาสคาล

การกำหนดข้อมูลและอัลกอริธึมด้านการรับข้อมูลเข้าเป็นดังนี้

```
    VAR    IND_STORBE, INP_4CK  : (0,1);
```

```
        INP_DATA  : CHAR;
        INP_BUSY  : REALTIME(0,1) = Q_FULL OR NOT INPUT_READY;
REALTIME PROCEDURE WHEN_DAT_ARRIVES;
BEGIN
        IF  INP_STROBE = ↓ THEN BEGIN
           INPUT_READY := FALSE;
           DATA_IN        := INP_DATA;
           INPUT_DATA_ARRIVED := TRUE;
           END;
END
PROCEDURE SEND_ACKNOWLEDGE;
BEGIN
        INPUT_READY := TRUE;
        INP_ACK        := 0;
        DELAY (1μS);
        INP_ACK        := 1;
END;
```

การกำหนดข้อมูลและอัลกอริธึมในการส่งข้อมูลไปยังเครื่องพิมพ์มีดังนี้

```
VAR  OUT_ACK, OUT_BUSY, OUT_STROBE : (0,1);
     OUT_DATA          : CHAR;
     BUSY_OCCURED   : REALTIME BOOLEAN = BOOLEAN (OUT_BUSY);
REALTIME PROCEDURE WHEN_ACK_ARRIVED;
BEGIN
        IF OUT_ACK = ↓ THEN BEGIN
           ACKNOWLEDGE_ARRIVED := TRUE;
           END;
END;
PROCEDURE SENT_STROBE;
BEGIN
        OUT_STROBE := 0;
        DELAY (1μS);
        OUT_STROBE := 1;
```

END;

ข้อมูลที่ถูกกำหนดนำหน้าด้วย INP ได้แก่ INP_ACK, INP_STROBE, INP_BUSY, INP_DATA เป็นข้อมูลที่เป็นอินพุทที่ติดต่อกับด้านคอมพิวเตอร์โดย INP_DATA เป็นข้อมูลที่ถูกส่งมาและที่เหลือเป็นข้อมูลรหัสควบคุมได้แก่

IND_ACK เป็นข้อมูลสัญญาณ ACKNOWLEDOE ซึ่งสปูลเลอร์สำหรับเครื่องพิมพ์จะส่งรหัสสัญญาณไปยังคอมพิวเตอร์เพื่อแสดงว่าได้รับข้อมูลจาก INP_DATA แล้ว

INP_STROBE เป็นข้อมูลสัญญาณ STROBE ที่คอมพิวเตอร์จะส่งมายังสปูลเลอร์สำหรับเครื่องพิมพ์เพื่อแสดงว่าได้ส่งข้อมูลใหม่มาแล้ว

INP_BUSY เป็นข้อมูลสัญญาณ BUSY ที่สปูลเลอร์สำหรับเครื่องพิมพ์จะใช้แสดงให้คอมพิวเตอร์ทราบว่าขณะนี้ยังไม่พร้อมที่จะรับข้อมูลใหม่ ซึ่งจะเกิดจาก ช่วงเวลาที่กำลังรับข้อมูลเดิมหรือเมื่อข้อมูลในถาวคอยเต็ม

อัลกอริธึมทางด้านการรับข้อมูลจากคอมพิวเตอร์ที่เกี่ยวข้องกับระบบการติดต่อแบบ Centronic ประกอบด้วย 2 โปรซิเยอร์ได้แก่ WHEN_DATA_ARRIVED ซึ่งต้องตอบสนองแบบ realtime และ SEND_ACKNOWLEDGE โดยมีรายละเอียดคือ

WHEN_DATA_ARRIVED จะทำงานเมื่อข้อมูลรหัสควบคุม INP_STROBE มีการเปลี่ยนสถานะจาก 1 ไปเป็น 0 หรือเรียกว่าที่ขอบขาลง (FALLING EDGE) ของ INP_STROBE ทำการรับข้อมูลและตั้งตัวแปรที่แสดงสถานด้านการรับข้อมูล

SEND_ACKNOWLEDGE เมื่อการรับข้อมูลเข้าเสร็จสิ้นแล้วก็จะส่งข้อมูลรหัสสัญญาณ ACKNOWEDGE ผ่าน INP_ACK ไปยังคอมพิวเตอร์เป็นรหัสสัญญาณและพัลซ์ (Pulse) ที่ทำงานที่ระดับ 0 (ACTIVE LOW) เป็นระยะเวลาประมาณ 1 ไมโครวินาที เพื่อตอบรับว่าได้รับข้อมูลแล้วข้อมูลทางด้านการติดต่อกับเครื่องพิมพ์ที่ถูกกำหนดให้นำหน้าด้วย OUT ได้แก่ OUT_ACK, OUT_BUSY, OUT_STROBE และ OUT_DATA เป็นข้อมูลที่ติดต่อกับเครื่องพิมพ์โดยมี OUT_DATA เป็นข้อมูลที่ส่งไปพิมพ์และที่เหลือเป็นรหัสสัญญาณควบคุมได้แก่

OUT_ACK สัญญาณจากเครื่องพิมพ์ที่จะส่งมายังสปูลเลอร์สำหรับเครื่องพิมพ์เพื่อแจ้งว่าได้รับข้อมูลเรียบร้อยแล้ว

OUT_STROBE สัญญาณจากสปูลเลอร์สำหรับเครื่องพิมพ์ไปยังเครื่องพิมพ์เพื่อแสดงว่าได้ส่งข้อมูลใหม่ให้แล้ว

OUT_BUSY สัญญาณจากเครื่องพิมพ์ เพื่อแสดงว่าไม่พร้อมที่จะรับข้อมูลใหม่

อัลกอริธึม ด้านการส่งข้อมูลที่เกี่ยวข้องกับการส่งทางช่องทางขนานตามมาตรฐาน Centrinic ได้แก่ WHEN_ACKNOWLEDGE ARRIVED และ SEND_STROBE มีรายละเอียดดังนี้

WHEN_ACKNOLEDGE_ARRIVED เป็น realtime procedure ที่จะทำงานเมื่อได้รับสัญญาณ OUT_ACK เปลี่ยนแปลงจาก 1 เป็น 0 (FALLING EDGE) โดยจะทำการตั้งค่าตัวแปร ACKNOWLEDGE_ARRIVED ให้เป็น TRUE เพื่อใช้ตรวจสอบในฟังก์ชั่น OUTPUT_READY

SEND_STROBE จะถูกเรียกใช้เมื่อมีการส่งข้อมูลไปยังเครื่องพิมพ์โดยส่งข้อมูลสัญญาณ OUT_STROBE เป็นพัลซ์ที่ 0 ไปเป็นระยะเวลา 1 ไมโครวินาที

นอกเหนือจากข้อมูลรหัสสัญญาณควบคุมที่กล่าวมาแล้วในส่วนของการรับและส่งข้อมูลก็ยังมีรหัสสัญญาณควบคุมที่จะเกี่ยวข้องได้แก่

PAPER_EMPTY, SELECT, FAULT เป็นรหัสสัญญาณควบคุมจากเครื่องพิมพ์ที่จะแสดงเมื่อเกิดเหตุผิดพลาดขึ้นที่เครื่องพิมพ์โดย PAPER_EMPTY(PE) จะเป็น 1 เมื่อกระดาษที่เครื่องพิมพ์ไม่มี, SELECT (SLCT) จะเป็น 1 เมื่อเครื่องพิมพ์นั้นออนไลน์ และเป็น 0 เมื่อเครื่องพิมพ์ออฟไลน์ โดยการเลือกที่แผงหน้าปัทม์ของเครื่องพิมพ์ และ FAULT จะเป็น 0 เมื่อเครื่องพิมพ์ผิดปกติ (รวมทั้งกรณี PAPER_EMPTY และ SELECT เป็น 0 ด้วย)

AUTO_FEED, INPUT_PRIME,SLCT_IN เป็นสัญญาณควบคุมจากเครื่องคอมพิวเตอร์ที่ส่งไปยังเครื่องพิมพ์

สัญญาณ AUTO_FEED เมื่อเป็น 0 ให้เครื่องพิมพ์เลื่อนบรรทัดทันทีเมื่อได้รับคำสั่ง CR

INPUT_PRIME เป็นสัญญาณที่สั่งให้เครื่องพิมพ์เริ่มทำงานใหม่ (initialize), SLCT_IN ปกติจะเป็น 0 เพื่อแสดงว่าเลือกใช้เครื่องพิมพ์นี้

จากที่ได้กล่าวมาในบทนี้ซึ่งเป็นรายละเอียดของข้อมูลแถวคอย ทั้งโครงสร้างของข้อมูลอัลกอริธึมการนำข้อมูลแถวคอยไปใช้ประโยชน์ในด้านต่าง ๆ รายละเอียดของสปูลเลอร์สำหรับเครื่องพิมพ์ในรูปของโครงสร้างข้อมูลอัลกอริธึมและการใช้ประโยชน์จากข้อมูลแถวคอยโดยเน้นการติดต่อผ่านช่องทางติดต่อแบบขนานตามมาตรฐานเซนทรอนิก (ภาคผนวก ก) แล้วในบทต่อไปจะได้กล่าวถึงการออกแบบวงจรอิเล็กทรอนิกส์และวงจรรวมของแถวคอย