บทที่ 3

การออกแบบระบบข่ายงานไมโครคอมพิวเตอร์ผ่านพอร์ตอนุกรม

จากบทที่แล้วได้กล่าวถึงแนวความคิดและทฤษฎีต่าง ๆ เกี่ยวกับข่ายงานไมโครคอมพิวเตอร์ที่จะนำมาใช้เป็นแนวทางและอ้างอิงในการวิจัย ในบทนี้จะกล่าวถึงโครงสร้างของระบบข่ายงานไมโครคอมพิวเตอร์ว่าประกอบด้วยส่วนต่าง ๆ อะไรบ้าง ความสัมพันธ์ของส่วนประกอบเหล่านั้นเป็นอย่างไร และจะกล่าวถึงการออกแบบในรายละเอียดเกี่ยวกับโครงสร้างข้อมูล รูปแบบแฟ้มข้อมูล ขั้นตอนวิธีการทำงานของส่วนประกอบต่าง ๆ เหล่านั้น และรูปแบบการแสดงผล ซึ่งในการออกแบบและพัฒนาระบบจะดัดแปลง และเพิ่มเติมจากซอฟต์แวร์สำเร็จรูปต้นแบบคือ BP-LAN ของนาย Craig Chaiken

โครงสร้างของระบบ

ระบบข่ายงานไมโครคอมพิวเตอร์ผ่านพอร์ตอนุกรมประกอบด้วยส่วนหลัก ๆ 5 ส่วน คือ

ส่วนฮาร์ดแวร์ เป็นส่วนอุปกรณ์ต่าง ๆ ที่นำมาใช้ในข่ายงาน ได้แก่ เครื่องไมโครคอมพิวเตอร์ สายคู่บิดเกลียว และพอร์ตอนุกรมตามมาตรฐาน RS-232C ในส่วนนี้จะเป็นการกำหนดรูปร่างลักษณะการเชื่อมต่อข่ายงาน และวิธีการเชื่อมต่อพอร์ตอนุกรมระหว่างเครื่องผู้ใช้ และ เครื่องบริการ

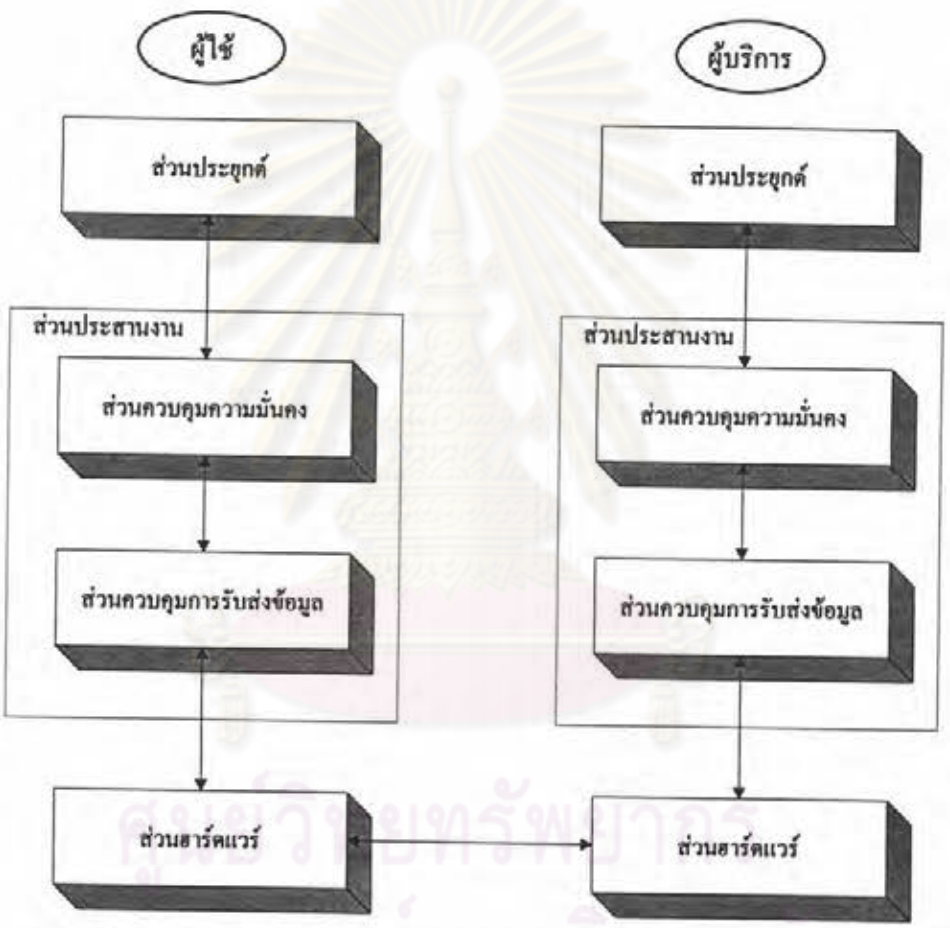
ส่วนประสานงาน เป็นส่วนประสานงานระหว่างส่วนประยุกต์ใช้งานกับส่วนอื่น ๆ ที่อยู่ระดับต่ำลงมา ส่วนนี้เปรียบเทียบได้กับส่วนบริการอุปกรณ์อินพุตเอาต์พุตของดอสคือ IO.SYS ซึ่งจะช่วยให้การพัฒนาส่วนประยุกต์ใช้งานทำได้ง่ายขึ้น และเป็นอิสระจากส่วนฮาร์ดแวร์

ส่วนควบคุมการรับส่งข้อมูล เป็นส่วนหนึ่งของส่วนประสานงาน ทำหน้าที่ควบคุมการรับส่งข้อมูลทั้งในระดับกายภาพและระดับเชื่อมโยงข้อมูลตามมาตรฐาน OSI

ส่วนควบคุมความมั่นคง เป็นส่วนที่ทำหน้าที่ควบคุมการเข้าถึงระบบของผู้ใช้ ได้แก่ การกำหนดให้มีรหัสผู้ใช้ รหัสผ่าน และสิทธิในการเข้าถึงข้อมูล

ส่วนประยุกต์ใช้งาน เป็นส่วนการเรียกใช้และให้บริการทรัพยากรข่ายงาน ได้แก่ จาน บันทึก แผ่นบันทึก และ เครื่องพิมพ์

โครงสร้างของระบบแสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 แสดงโครงสร้างของข่ายงานไมโครคอมพิวเตอร์ผ่านพอร์ตอนุกรม

จากรูปที่ 3.1 ส่วนประกอบทั้งหมดภายในระบบจะทำงานสัมพันธ์กัน ตัวอย่างเช่น เมื่อผู้ใช้เรียกใช้แฟ้มข้อมูล หรือ สารบบโดยผ่านบริการของดอส โปรแกรมในส่วนประยุกต์จะทำการตรวจสอบหน่วยบันทึก ถ้าไม่ใช่หน่วยบันทึกเสมือนของเครื่องบริการข่ายงานก็จะส่งการควบคุมกลับคืนไปยังดอส แต่ถ้าใช่ก็จะทำการตรวจสอบสิทธิการเข้าถึงระบบของผู้ใช้ โดยการเรียกใช้โปรแกรมในส่วนควบคุมความมั่นคงผ่านทางส่วนประสานงาน ถ้าผู้ใช้นั้นไม่มีสิทธิในการเข้าถึงระบบก็จะส่งข้อความกลับคืนไปให้ดอส ถ้ามีสิทธิเข้าถึงระบบก็จะจัดเตรียมข้อมูลตามรูปแบบที่กำหนดไว้สำหรับคำสั่งนั้น แล้วส่งต่อให้ส่วนควบคุมการรับส่งข้อมูล โดยผ่านทางส่วนประสานงานเช่นกัน หลังจากนั้นส่วนควบคุมการรับส่งข้อมูลก็จะจัดเตรียมส่วนตรวจสอบข้อมูลเพิ่มเติม แล้วทำการส่งข้อมูลออกทางพอร์ตอนุกรมในส่วนฮาร์ดแวร์ ในทำนองเดียวกันทางเครื่องบริการ โปรแกรมควบคุมการให้บริการในส่วนโปรแกรมประยุกต์จะคอยวนตรวจสอบขารับสัญญาณที่พอร์ตอนุกรมทุกพอร์ตที่เชื่อมต่อ โดยเรียกใช้โปรแกรมในส่วนควบคุมการรับส่งข้อมูลผ่านทางส่วนประสานงาน เมื่อได้รับข้อมูลก็จะตรวจสอบความถูกต้องตามกระบวนการการส่งโปรโตคอลที่กำหนด ถ้าข้อมูลไม่ถูกต้องก็จะส่งกลับคืนไปยังเครื่องผู้ใช้เพื่อให้ส่งข้อมูลใหม่ ถ้าถูกต้องโปรแกรมควบคุมการให้บริการก็จะตรวจสอบคำสั่งที่ได้รับ แล้วทำการเรียกใช้โปรแกรมประยุกต์ตามคำสั่งนั้น ๆ ต่อไป เช่นตามตัวอย่างนี้เป็นคำสั่งเรียกใช้แฟ้มข้อมูล หรือ สารบบก็จะเรียกใช้โปรแกรมให้บริการแฟ้ม เป็นต้น

ที่กล่าวมาข้างต้นจะเห็นได้ว่าโครงสร้างของระบบได้ถูกออกแบบให้มีสถาปัตยกรรมข่ายงานตามแบบมาตรฐาน OSI โดยแบ่งเป็นส่วนต่าง ๆ ตามระดับชั้นของมาตรฐาน OSI ยกเว้นในบางระดับจะไม่ได้ใช้ในการวิจัยนี้ ซึ่งการออกแบบในลักษณะนี้จะช่วยทำให้ง่ายต่อการศึกษาและทำความเข้าใจระบบ และการพัฒนาการเชื่อมต่อระหว่างข่ายงาน หรือ การปรับปรุงระบบจะทำได้สะดวกขึ้น

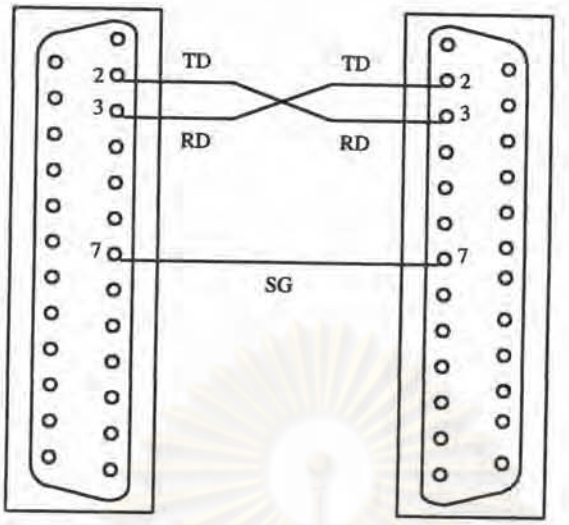
เมื่อได้ทราบถึงโครงสร้างของระบบว่าประกอบด้วยส่วนต่าง ๆ อะไรบ้าง และแต่ละส่วนประกอบมีความสัมพันธ์กันอย่างไรแล้ว ต่อไปผู้วิจัยใคร่ขอกล่าวถึงการออกแบบในรายละเอียดของแต่ละส่วนประกอบเกี่ยวกับโครงสร้างข้อมูล หรือ รูปแบบแฟ้มข้อมูลที่ใช้ รวมไปถึงขั้นตอนการทำงานของโปรแกรมต่าง ๆ เพื่อให้ผู้สนใจสามารถมองเห็นภาพ หรือ มีจินตนาการเกี่ยวกับการทำงานของระบบได้ชัดเจนขึ้น

การออกแบบส่วนฮาร์ดแวร์

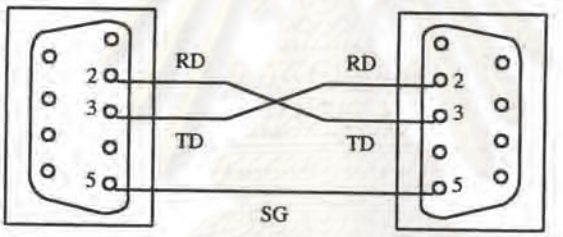
ในการสร้างระบบข่ายงานหนึ่ง ๆ นั้น สิ่งแรกที่จะต้องถูกกำหนดขึ้นก่อนคือ ลักษณะรูปร่างของข่ายงานที่มองเห็นภายนอกเป็นอย่างไร รูปร่างของข่ายงานนี้จะมีผลต่อประสิทธิภาพ ค่าใช้จ่าย การพัฒนา และการนำไปประยุกต์ใช้งาน ดังนั้นการเลือกรูปร่างข่ายงานให้เหมาะสมกับงานย่อมทำให้ได้ระบบที่ดีตามไปด้วย ในการวิจัยนี้จะเลือกใช้รูปร่างข่ายงานแบบดาว และประยุกต์ใช้แบบข่ายงานเดียว คือ Single Star ดังเหตุผลที่ได้กล่าวแล้วในบทที่ 1 ส่วนจำนวนเครื่องผู้ใช้ทั้งหมดที่สามารถต่อได้ในระบบจะเป็นเท่าใดนั้น ขึ้นอยู่กับจำนวนฮาร์ดแวร์คือพอร์ตอนุกรมที่มีอยู่ในเครื่อง ทั้งนี้ในส่วนของซอฟต์แวร์จะสนับสนุนให้เชื่อมต่อได้ถึง 256 พอร์ต

สำหรับการเชื่อมต่อระหว่างเครื่องผู้ใช้ และเครื่องบริการ จะใช้สายคู่บิดเกลียวเป็นตัวกลางสื่อสาร เนื่องจากหาได้ง่าย และราคาถูก โดยเชื่อมต่อผ่านทางพอร์ตอนุกรมมาตรฐาน RS-232C ที่มีอยู่ในเครื่อง ส่วนใหญ่ที่ใช้กันอยู่ในปัจจุบันจะมี 2 พอร์ต คือ COM1 และ COM2 การเชื่อมต่อจะเป็นแบบไม่ใช้โมเด็ม (Null-Modem Wiring) นั่นคือจะเชื่อมต่อทั้ง 2 เครื่องเข้าโดยตรง การต่อแบบนี้จะสามารถส่งข้อมูลได้เร็วกว่าต่อผ่านโมเด็ม โดยความเร็วสูงสุดตามมาตรฐานไม่เกิน 115 กิโลบิตต่อวินาที นอกจากนี้ยังช่วยลดปัญหาของสัญญาณรบกวนภายในสายสัญญาณ ซึ่งจะพบได้บ่อยกรณีต่อผ่านโมเด็มผ่านระบบโทรศัพท์ ชนิดของหัวต่อที่ใช้เป็นได้ทั้งชนิด 25 ขา และชนิด 9 ขา วิธีการเชื่อมต่อแสดงได้ดังรูปที่ 3.2 และ 3.3

จากรูปที่ 3.2 และ 3.3 วิธีการต่อจะใช้สายคู่บิดเกลียว 3 เส้น โดยใช้สาย 2 เส้นแรกต่อไขว้กันระหว่างขาส่ง (TD) และขารับ (RD) เช่นกรณีหัวต่อชนิด 25 ขา ให้ต่อขาที่ 2 ของเครื่องผู้ใช้เข้ากับขาที่ 3 ของเครื่องบริการ และทำนองเดียวกันต่อขาที่ 2 ของเครื่องบริการเข้ากับขาที่ 3 ของเครื่องผู้ใช้ ส่วนอีกเส้นหนึ่งต่อขากราวนด์ (SG) ของทั้ง 2 เครื่องเข้าด้วยกัน ส่วนกรณีหัวต่อชนิด 9 ขา ขาส่งและขารับจะสลับกับหัวต่อชนิด 25 ขา คือขาส่งจะเป็นขาที่ 3 และขารับจะเป็นขาที่ 2 ส่วนขากราวนด์จะเป็นขาที่ 5 ดังนั้นถ้าต้องการต่อหัวต่อชนิด 25 ขาเข้ากับหัวต่อชนิด 9 ขา ก็จะต้องต่อขาหมายเลขเดียวกันเข้าด้วยกันคือต่อขาที่ 2 เข้ากับขาที่ 2 และขาที่ 3 กับขาที่ 3 ส่วนขากราวนด์ต่อขาที่ 7 ของหัวต่อ 25 ขา กับขาที่ 5 ของหัวต่อ 9 ขา



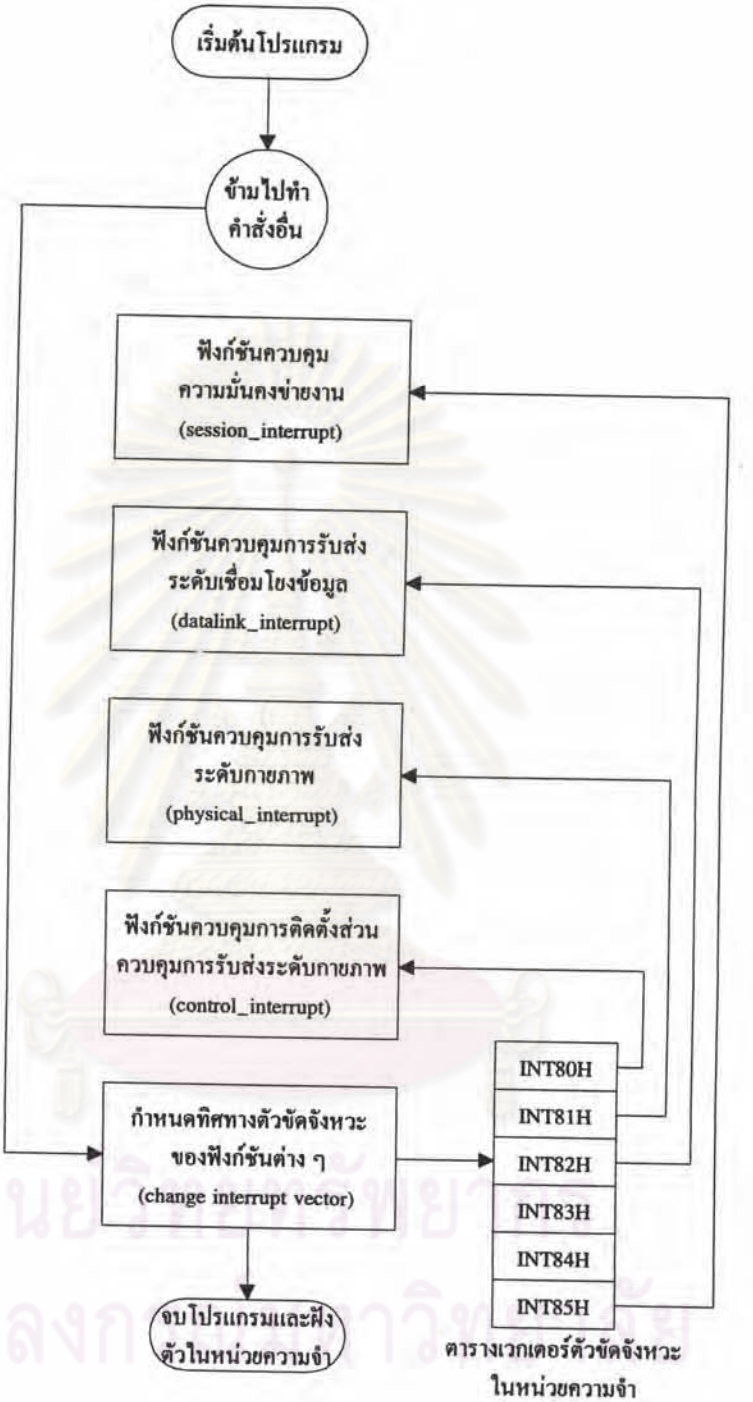
รูปที่ 3.2 แสดงการเชื่อมต่อแบบ null-modem โดยใช้ DB-25



รูปที่ 3.3 แสดงการเชื่อมต่อแบบ null-modem โดยใช้ DB-9

การออกแบบส่วนประสานงาน

ในส่วนนี้จะประกอบไปด้วยฟังก์ชันต่าง ๆ ที่ช่วยประสานการเรียกใช้ทรัพยากรข่ายงานให้แก่ส่วนประยุกต์ ได้แก่ ฟังก์ชันสำหรับควบคุมการติดตั้งส่วนควบคุมการรับส่งข้อมูลระดับกายภาพ (control_interrupt) ฟังก์ชันสำหรับควบคุมความมั่นคงข่ายงาน (session_interrupt) และ ฟังก์ชันสำหรับควบคุมการรับส่งข้อมูลทั้งระดับเชื่อมโยงข้อมูล (datalink_interrupt) และ ระดับกายภาพ (physical_interrupt) โปรแกรมส่วนประสานงานนี้จะถูกพัฒนาด้วยภาษาแอสเซมบลี และให้ชื่อว่า APBIOS.COM ซึ่งมีผังการทำงานดังรูปที่ 3.4



รูปที่ 3.4 แสดงผังงานโปรแกรมส่วนประสานงาน

จากรูปที่ 3.4 จะเห็นได้ว่าโปรแกรมส่วนประสานงานนี้ เป็นโปรแกรมประเภทฝังตัวในหน่วยความจำกล่าวคือ เมื่อเริ่มต้นโปรแกรมจะทำการติดตั้งฟังก์ชันต่าง ๆ ไว้ในหน่วยความจำ โดยวิธีการเปลี่ยนทิศทางของตัวขัดจังหวะในตารางเวกเตอร์ตัวขัดจังหวะ ให้ชี้ไปที่ตำแหน่งที่อยู่ในหน่วยความจำ (แอดเดรส) ของฟังก์ชันดังกล่าวข้างต้น เมื่อมีการเรียกใช้ฟังก์ชันเหล่านั้นผ่านตัวขัดจังหวะที่กำหนด ตัวควบคุมการขัดจังหวะก็จะแจ้งให้ซีพียูทำการปลุกฟังก์ชันที่ถูกเรียกขึ้นมาทำงาน และเมื่อทำงานเสร็จก็จะส่งการควบคุมกลับคืนไปยังโปรแกรมที่เรียกใช้ ซึ่งการออกแบบโปรแกรมลักษณะนี้แสดงให้เห็นถึงความสะดวกในการที่ผู้เขียนโปรแกรมประยุกต์ ไม่ต้องเสียเวลาพัฒนาโปรแกรมควบคุมการรับส่งข้อมูลเอง และทำให้โปรแกรมประยุกต์ที่พัฒนาขึ้นนั้นไม่ผูกติดกับส่วนฮาร์ดแวร์ดังที่กล่าวแล้วในเรื่องโครงสร้างระบบ

การใช้ตัวขัดจังหวะในการเรียกใช้ฟังก์ชันของระดับชั้นต่าง ๆ ของข่ายงานมาตรฐานนั้น ในการวิจัยนี้จะใช้เพียง 4 หมายเลขคือ

ตัวขัดจังหวะ 80H จะชี้ไปที่แอดเดรสของฟังก์ชัน control_interrupt

ตัวขัดจังหวะ 81H จะชี้ไปที่แอดเดรสของฟังก์ชัน physical_interrupt

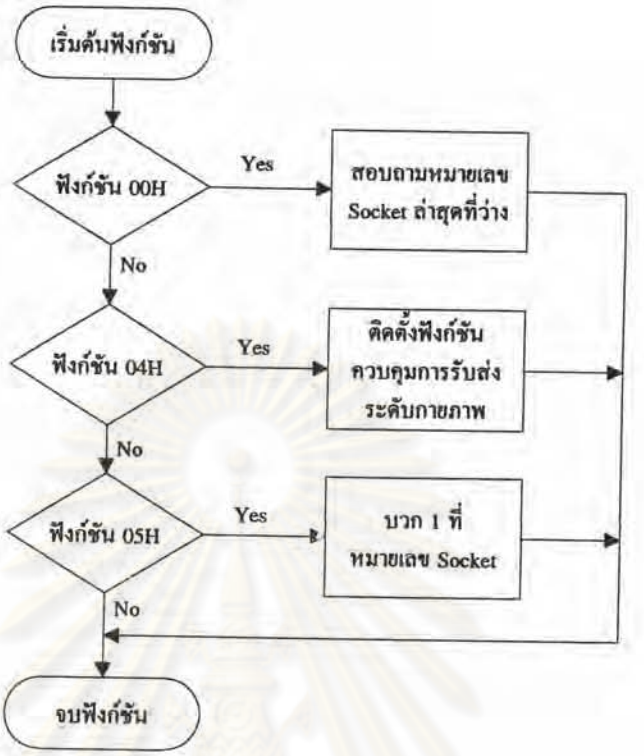
ตัวขัดจังหวะ 82H จะชี้ไปที่แอดเดรสของฟังก์ชัน datalink_interrupt

ตัวขัดจังหวะ 85H จะชี้ไปที่แอดเดรสของฟังก์ชัน session_interrupt

ในตัวขัดจังหวะแต่ละตัวจะมีฟังก์ชันต่าง ๆ ให้เรียกใช้ได้ ซึ่งรายละเอียดการทำงานของแต่ละฟังก์ชันจะกล่าวถึงอีกครั้งในส่วนของฟังก์ชันระดับนั้น ๆ

สำหรับหน้าที่และขั้นตอนการทำงานของแต่ละฟังก์ชันในโปรแกรมส่วนประสานงาน จะขอกล่าวถึงเฉพาะฟังก์ชันควบคุมการติดตั้งส่วนควบคุมการรับส่งระดับกายภาพ หรือ control_interrupt เท่านั้น ส่วนฟังก์ชันที่เหลือจะกล่าวถึงในส่วนต่อ ๆ ไป

ฟังก์ชัน control_interrupt มีหน้าที่หลักคือการติดตั้งฟังก์ชันควบคุมการรับส่งข้อมูลระดับกายภาพ (อยู่ในโปรแกรม APSERIAL.COM ซึ่งจะกล่าวถึงต่อไป) ลงใน Socket ซึ่งในการวิจัยนี้จะหมายถึงช่องทางตรรกะสำหรับการติดต่อระหว่างเครื่องผู้ใช้และเครื่องบริการ นั่นคือ Socket จะเป็นที่เก็บแอดเดรสของฟังก์ชันควบคุมการรับส่งระดับกายภาพของแต่ละพอร์ตอนุกรม Socket ของแต่ละเครื่องนี้มีได้หลาย Socket โดยกำหนดให้เป็นตัวแปรแถวลำดับขนาด 256 ไบต์ ดังนั้นการอ้างถึง Socket จะใช้ลำดับที่ของสมาชิกในแถวลำดับนั้น ในการติดตั้ง Socket จะต้องเริ่มตั้งแต่หมายเลข 0 เสมอ และหลังจากทำการติดตั้ง Socket เสร็จทุกครั้งจะเพิ่มหมายเลข Socket ให้เป็นหมายเลขที่ว่างถัดไป นอกจากหน้าที่ดังกล่าวแล้วฟังก์ชันนี้ยังให้บริการเกี่ยวกับการสอบถามหมายเลข Socket ล่าสุดที่ว่างอยู่ด้วย ผังงานของฟังก์ชันแสดงดังรูปที่ 3.5



รูปที่ 3.5 แสดงผังงานฟังก์ชัน control_interrupt

การเรียกใช้ฟังก์ชัน control_interrupt นี้ จะต้องเรียกผ่านตัวขัดจังหวะหมายเลข 80H โดยกำหนดรหัสของฟังก์ชันต่าง ๆ ในรีจิสเตอร์ AH ดังรายละเอียดต่อไปนี้

ฟังก์ชัน 00H - สอบถามหมายเลข Socket ล่าสุดที่ว่างอยู่ (หรือจำนวน Socket ที่ใช้แล้ว)

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 00H

ส่งค่ากลับ: หมายเลข Socket ในรีจิสเตอร์ AL

ฟังก์ชัน 04H - ติดตั้ง Socket ในตำแหน่งที่ว่างถัดไป

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 04H

ส่งค่ากลับ: ไม่มี

ฟังก์ชัน 05H - เพิ่มค่า Socket ให้เป็น Socket ที่ว่างถัดไป

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 05H

ส่งค่ากลับ: ไม่มี

วิธีการติดตั้งตัวขัดจังหวะกระทำได้โดยการเรียกใช้ตัวขัดจังหวะหมายเลข 21H ฟังก์ชันที่ 25H ของดอส โดยกำหนดให้รีจิสเตอร์ AH มีค่าเท่ากับ 25H รีจิสเตอร์ AL มีค่าเป็นหมายเลขตัวขัดจังหวะที่ต้องการติดตั้ง และรีจิสเตอร์ DX มีค่าเป็นค่าขจัด (Offset) ของส่วนโปรแกรมที่ต้องการเรียกผ่านตัวขัดจังหวะนั้น ตัวอย่างการติดตั้งตัวขัดจังหวะ แสดงด้วยภาษาแอสเซมบลีได้ดังนี้

```
control_interrupt   proc  far
                    .
                    .
                    iret
control_interrupt   endp
install     proc    near                        ;install interrupt service routine
            mov     ah,25h                      ;change interrupt vector
            mov     al,80h                      ;old interrupt service routine
            mov     dx,offset control_interrupt ;new interrupt service routine
            int     21h                         ;call DOS interrupt 21H
install     endp
```

ส่วนการเรียกใช้ฟังก์ชันของระบบ ได้ออกแบบเป็นแมคโครแอสเซมบลี ดังตัวอย่างต่อไปนี้

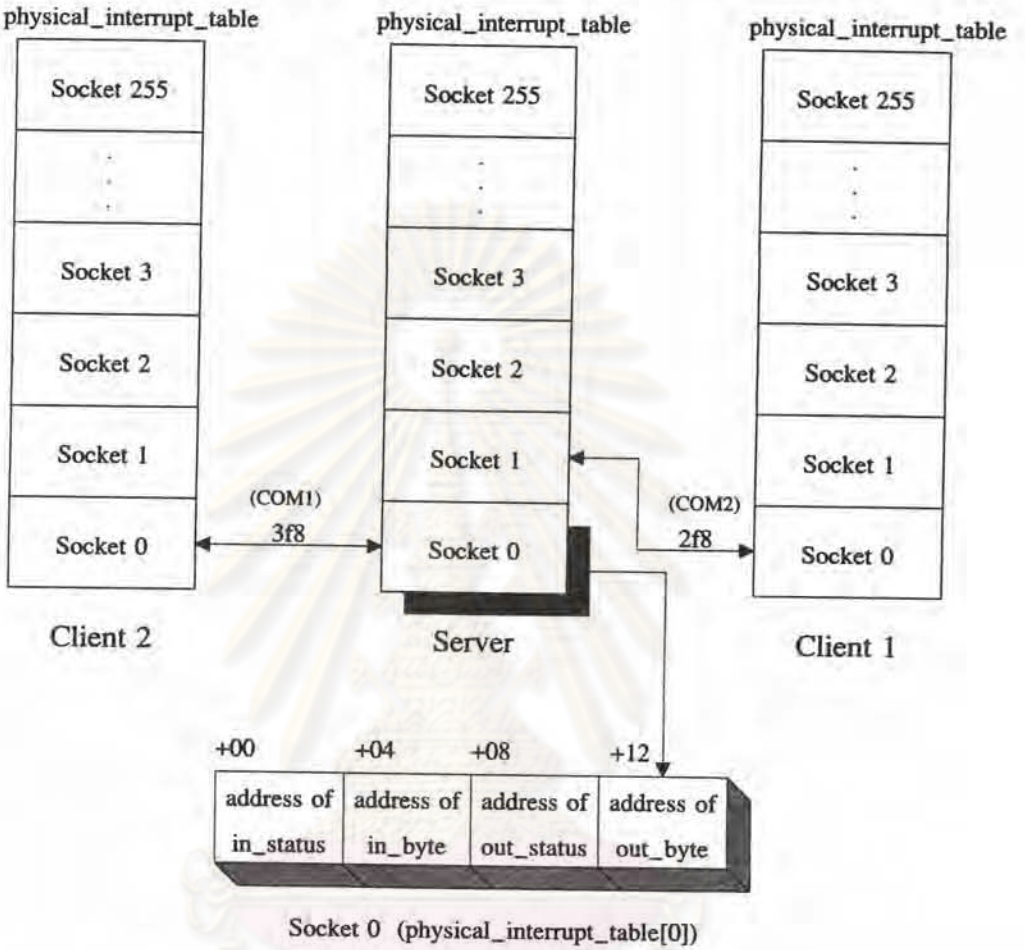
```
bpbios  macro  intnum, funcnum, portnum, count, buffer_addr, overflow
        mov    ah,funcnum                       ;function number
        IFNB  <portnum>
        mov    bl,portnum                       ;socket number
        ENDIF
        IFNB  <count>
        mov    cx,count
        ENDIF
        IFNB  <buffer_addr>
        mov    dx,offset buffer_addr overflow  ;ds:dx point to address of buffer
        ENDIF
        int    intnum
        endm
```

ตัวอย่างวิธีการติดตั้งฟังก์ชันควบคุมการรับส่งระดับกายภาพ แสดงด้วยภาษาแอสเซมบลี ได้ดังนี้

```
1)    mov   bl,cs:maxnode                    ;set number of sockets
2)    xor   bh,bh                            ;clear bh to zero
3)    push  ax
4)    add   bx,bx
5)    add   bx,bx
6)    add   bx,bx
7)    add   bx,bx                            ;multiply by 16 (for socket)
8)    add   al,al
9)    add   al,al                            ;multiply by 4 (for function)
10)   xor   ah,ah
11)   add   bx,ax                            ;point to address of the function
12)   pop   ax
13)   mov   word ptr cs:physical_interrupt_table[bx],dx
14)   mov   word ptr cs:physical_interrupt_table[bx+2],ds
```

ในบรรทัดที่ 1 เป็นการกำหนดหมายเลข Socket ล่าสุดที่ว่างลงในรีจิสเตอร์ BL ส่วนบรรทัดที่ 4 ถึง 7 เป็นการเลื่อนตำแหน่งใน physical_interrupt_table ตามหมายเลข Socket จากบรรทัดที่ 1 เพื่อเตรียมที่สำหรับติดตั้งแอดเดรสของฟังก์ชันควบคุมการรับส่งระดับกายภาพ เช่น ถ้าเป็น Socket 0 ค่า BX จะเท่ากับ 0 ถ้าเป็น Socket 1 ค่า BX จะเท่ากับ 4 และถ้าเป็น Socket 2 ค่า BX จะเท่ากับ 8 เป็นต้น จะสังเกตุได้ว่าค่าของ BX จะเพิ่มขึ้นครั้งละ 4 ทั้งนี้เพราะเหตุว่าการอ้างอิงแต่ละแอดเดรสต้องใช้เนื้อที่ 4 ไบต์ คือ เซกเมนต์ และ ออฟเซต (ค่าขจัด) อย่างละ 2 ไบต์ สำหรับบรรทัดที่ 8 ถึง 11 เป็นการเลื่อนตำแหน่งภายใน Socket ตามรหัสของฟังก์ชันที่ส่งมาจากโปรแกรมที่เรียกใช้ (เป็นค่าขจัด) ซึ่งจะถูกกำหนดเป็นหมายเลข 0, 1, 2 และ 3 จากนั้นจะทำการติดตั้งฟังก์ชันนั้นลงใน Socket ตามบรรทัดที่ 13 และ 14

ตัวอย่างการใช้ Socket เพื่อสนับสนุนระบบหลายผู้ใช้แสดงได้ดังรูปที่ 3.6



รูปที่ 3.6 แสดงตัวอย่างการใช้ Socket เพื่อสนับสนุนระบบหลายผู้ใช้

จากตัวอย่างนี้สมมุติว่าในข่ายงานประกอบด้วยเครื่องผู้ใช้ 2 เครื่องต่อเข้ากับเครื่องบริการ โดยเครื่องที่ 1 ต่อผ่าน COM2 และเครื่องที่ 2 ต่อผ่าน COM1 เมื่อติดตั้งโปรแกรมควบคุมการรับส่งระดับกายภาพที่แต่ละเครื่อง โปรแกรมจะเรียกใช้ฟังก์ชัน control_interrupt ผ่านตัวขัดจังหวะ 80H ฟังก์ชัน 04H ฟังก์ชัน control_interrupt จะทำการติดตั้งฟังก์ชันควบคุมการรับส่งระดับกายภาพ ซึ่งมีอยู่ 4 ฟังก์ชัน (จะขอกล่าวรายละเอียดภายหลัง) ลงใน Socket หรือ physical_interrupt_table ซึ่งเป็นตัวแปรแถวลำดับในฟังก์ชัน physical_interrupt โดยที่เครื่องบริการ Socket 0 จะเก็บฟังก์ชันที่ให้บริการทางพอร์ตหมายเลข 3f8 และ Socket 1 จะเก็บฟังก์ชันที่ให้บริการทางพอร์ตหมายเลข 2f8 นั่นแสดงว่าเครื่องบริการสามารถแยกแยะงานของผู้ใช้แต่ละคนได้จาก Socket เหล่านี้

การออกแบบส่วนควบคุมการรับส่งข้อมูล

ส่วนควบคุมการรับส่งข้อมูลนี้จะแบ่งการควบคุมออกเป็น 2 ระดับตามมาตรฐาน OSI คือ ระดับกายภาพ และระดับเชื่อมโยงข้อมูล

1) การออกแบบส่วนควบคุมการรับส่งข้อมูลระดับกายภาพ

จากที่กล่าวแล้วข้างต้นว่าส่วนนี้จะเป็นส่วนหนึ่งในส่วนประสานงาน และถูกเรียกใช้ผ่านทาง Socket ทั้งนี้เนื่องจากสิ่งสำคัญอย่างหนึ่งในระบบข่ายงานก็คือการสนับสนุนระบบหลายผู้ใช้ ซึ่งการวิจัยนี้ได้ออกแบบให้เก็บแอดเดรสของฟังก์ชันควบคุมการรับส่งระดับกายภาพของแต่ละพอร์ตไว้ใน Socket ในการนี้ต้องมีโปรแกรม 2 โปรแกรมสำหรับควบคุมการรับส่งคือ โปรแกรมแรกจะเก็บฟังก์ชันควบคุมการรับส่งและฝังตัวในหน่วยความจำ ส่วนโปรแกรมที่สองจะเก็บแอดเดรสของฟังก์ชันของโปรแกรมแรก ซึ่งส่วนโปรแกรมประยุกต์สามารถรับส่งข้อมูลผ่านพอร์ตอนุกรมได้โดยการเรียกใช้โปรแกรมที่สองผ่านตัวขัดจังหวะอีกต่อหนึ่ง โปรแกรมแรกมีชื่อว่า APSERIAL.COM และโปรแกรมที่สองเป็นฟังก์ชันมีชื่อว่า physical_interrupt ซึ่งจะถูกรวมอยู่ในส่วนประสานงานดังได้กล่าวแล้วการเรียกใช้ฟังก์ชัน physical_interrupt นี้ จะต้องเรียกผ่านตัวขัดจังหวะหมายเลข 81H โดยกำหนดรหัสของฟังก์ชันต่าง ๆ ในรีจิสเตอร์ AH ดังรายละเอียดต่อไปนี้

ฟังก์ชัน 00H - สอบถามสถานะภาพการได้รับข้อมูล

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 00H และ BL = หมายเลข Socket

ส่งค่ากลับ: แฟล็กไม่เป็น 0 หมายถึงได้รับข้อมูลแล้ว

แฟล็กเป็น 0 หมายถึงยังไม่ได้รับข้อมูล

ฟังก์ชัน 01H - อ่านข้อมูลจากบัฟเฟอร์

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 01H และ BL = หมายเลข Socket

ส่งค่ากลับ: ข้อมูล 1 ไบต์ในรีจิสเตอร์ AL

ฟังก์ชัน 02H - สอบถามสถานะภาพความพร้อมส่ง

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 02H และ BL = หมายเลข Socket

ส่งค่ากลับ: แฟล็กไม่เป็น 0 หมายถึงบัฟเฟอร์สำหรับส่งว่าง สามารถส่งข้อมูลได้

แฟล็กเป็น 0 หมายถึงบัฟเฟอร์สำหรับส่งเต็ม ไม่สามารถส่งได้

ฟังก์ชัน 03H - ส่งข้อมูล

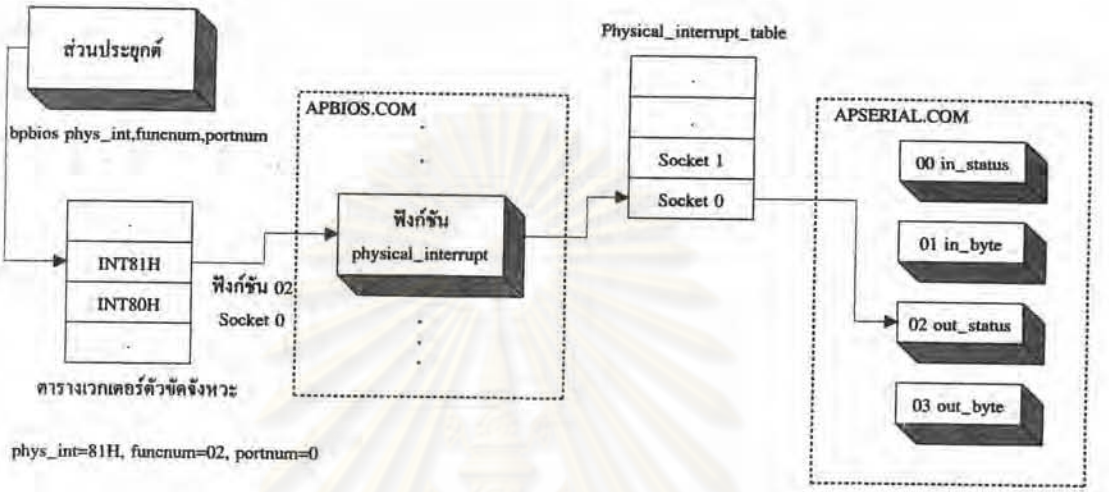
เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 03H

BL = หมายเลข Socket

AL = ข้อมูล 1 ไบต์ที่ต้องการส่ง

ส่งค่ากลับ: ไม่มี

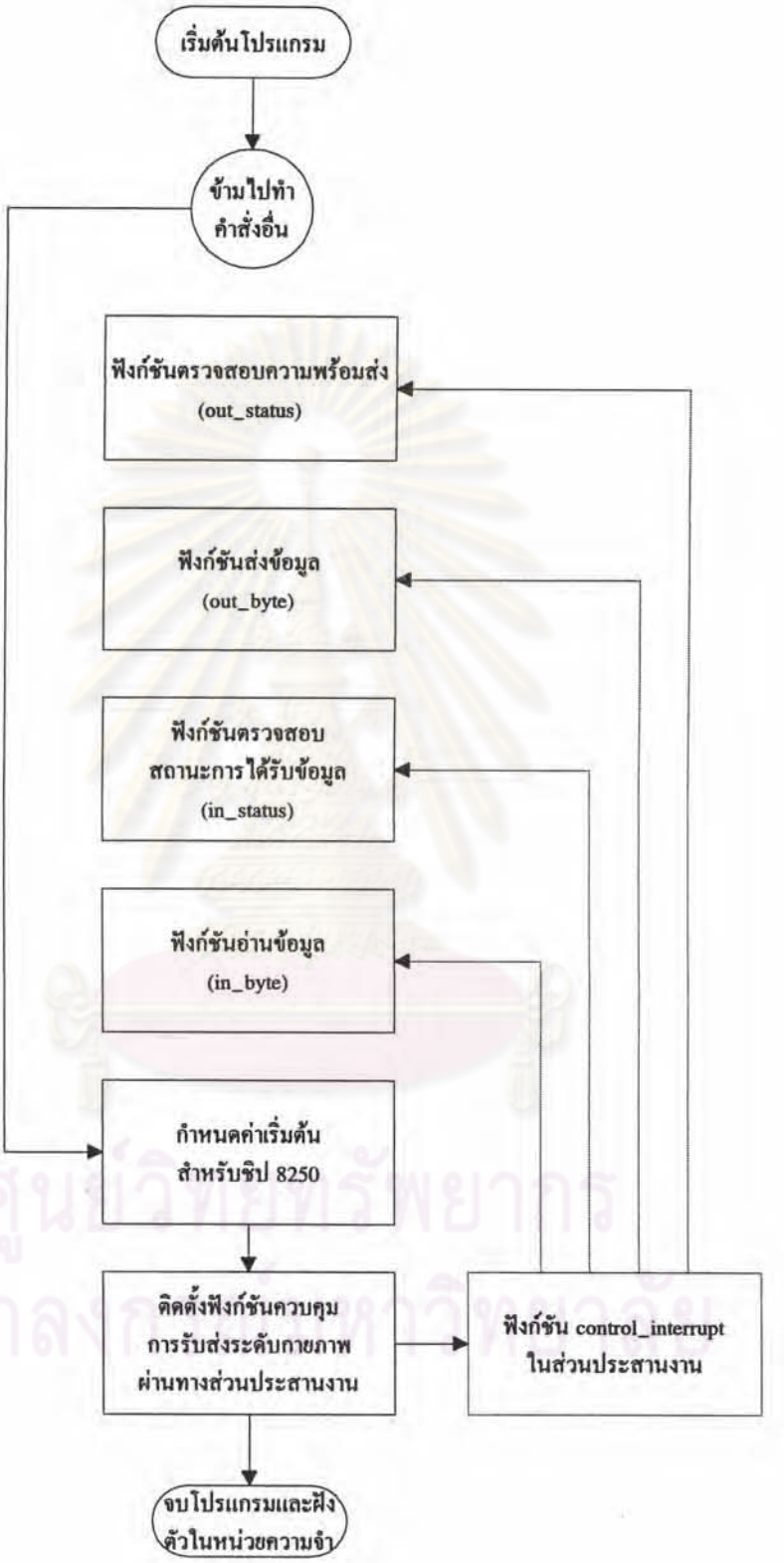
ตัวอย่างการเรียกใช้ส่วนควบคุมการรับส่งระดับกายภาพผ่าน Socket แสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 แสดงตัวอย่างการเรียกใช้ส่วนควบคุมการรับส่งระดับกายภาพผ่าน Socket

จากตัวอย่างจะแสดงให้เห็นถึงขั้นตอนการทำงานที่ประสานกันระหว่างส่วนประยุกต์ ส่วนประสานงาน และส่วนควบคุมการรับส่งระดับกายภาพ กล่าวคือ เมื่อส่วนประยุกต์ต้องการตรวจสอบสถานะภาพความพร้อมส่ง จะเรียกใช้ตัวขัดจังหวะหมายเลข 81H ฟังก์ชัน 02 ซีพียูจะตรวจสอบในตารางเวกเตอร์ตัวขัดจังหวะ แล้วส่งการควบคุมไปยังแอดเดรสที่พบ ณ ตำแหน่งของตัวขัดจังหวะนั้น ซึ่งตามตัวอย่างนี้จะชี้ไปที่ฟังก์ชัน physical_interrupt ของส่วนประสานงาน จากนั้นฟังก์ชันนี้ก็จะทำการเรียกใช้ฟังก์ชัน 02 ผ่านทางตัวแปรลำดับที่ 0 ของ physical_interrupt_table หรือ Socket 0 ซึ่งจะเก็บแอดเดรสของฟังก์ชันควบคุมการรับส่งระดับกายภาพทั้ง 4 ฟังก์ชันอยู่ โดยเลื่อนจากตำแหน่งเริ่มต้นของ Socket ไปอีก 8 ไบต์ก็จะเป็นแอดเดรสของฟังก์ชัน 02 ที่ต้องการ

โปรแกรม APSERIAL.COM จะถูกพัฒนาด้วยภาษาแอสเซมบลี ประกอบด้วยฟังก์ชันควบคุมการรับส่งระดับกายภาพ 4 ฟังก์ชัน และการกำหนดค่าให้แก่ชิป 8250 มีผังงานดังรูปที่ 3.8



รูปที่ 3.8 แสดงผังงานโปรแกรมควบคุมการรับส่งข้อมูลระดับกายภาพ

จากผังงานตามรูปที่ 3.8 ขั้นตอนการทำงานของโปรแกรมอธิบายได้ดังต่อไปนี้

ขั้นที่ 1 : กำหนดค่าพารามิเตอร์ที่ได้รับลงในตัวแปร ได้แก่ แอดเดรสของพอร์ตอนุกรม ค่ารีจิสเตอร์ควบคุมสายสื่อสาร และค่าตัวหาร

ขั้นที่ 2 : กำหนดค่าเริ่มต้นให้แก่ชิป 8250 ได้แก่

- กำหนดค่าให้รีจิสเตอร์ควบคุมสายสื่อสาร ได้แก่ ความยาวของข้อมูล จำนวนสต๊อปบิต การตรวจสอบบิตพาริตี้ และ แลตช์ตัวหาร (DLAB) ดังตัวอย่างเป็นภาษาแอสเซมบลีต่อไปนี้

```
01)     mov  dx,curport      ;address of serial port (ex. 2f8, 3f8,...)
02)     add  dx,3            ;address of Line Control Register
03)     mov  al,80h          ;DLAB=1
04)     out  dx,al
05)     mov  ax,conf         ;define configuration of serial port
06)     out  dx,al
```

จากตัวอย่างสมมุติว่า curport มีค่าเป็น 3f8 บรรทัดที่ 1 จะเป็นการกำหนดแอดเดรสเริ่มต้นไว้ในรีจิสเตอร์ dx เพื่อคำนวณหาแอดเดรสของรีจิสเตอร์ควบคุมสายสื่อสารในบรรทัดที่ 2 คือ 3fb ส่วนในบรรทัดที่ 3 เป็นการกำหนดค่า DLAB=1 และค่า conf ในบรรทัดที่ 4 จะได้จากพารามิเตอร์เป็นการกำหนดค่า 7 บิตที่เหลือของรีจิสเตอร์นี้ เช่น conf = 3 จะหมายถึง รับส่งข้อมูลขนาด 8 บิต มี 1 สต๊อปบิต และไม่มีการตรวจสอบบิตพาริตี้

- กำหนดค่าในรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์ ในการวิจัยนี้จะกำหนดค่าเป็น 0 หมายถึงไม่ใช้การอินเทอร์รัพท์ แต่จะใช้วิธีการ Polling แทน

- กำหนดค่าตัวหารลงในรีจิสเตอร์ LSB และ รีจิสเตอร์ MSB เพื่อคำนวณหาอัตราบอดในการรับส่ง เช่น ตัวหารเท่ากับ 1 จะได้อัตราบอดเป็น 115200 บอด ตัวหารเท่ากับ 2 จะได้อัตราบอดเท่ากับ 57600 บอด เป็นต้น ตัวอย่างการกำหนดค่าตัวหารแสดงด้วยภาษาแอสเซมบลีได้ดังนี้

```
01)     mov  cx,brw          ;define baud rate
02)     mov  dx,curport
03)     mov  al,cl           ;define LSB value
04)     out  dx,al
05)     mov  al,ch           ;define MSB value
06)     inc  dx              ;address of MSB
07)     out  dx,al
```

จากตัวอย่าง brw ในบรรทัดที่ 1 เป็นค่าของตัวหาร ส่วนการกำหนดค่า DLAB จะอยู่ในตัวอย่างที่แล้ว ซึ่งต้องมีค่าเป็น 1 บรรทัดที่ 3 และ 4 เป็นการกำหนดค่าให้แก่รีจิสเตอร์ LSB และบรรทัดที่ 5 ถึง 7 เป็นการกำหนดค่าให้แก่รีจิสเตอร์ MSB

ขั้นที่ 3 : ทำการติดตั้งฟังก์ชันควบคุมการรับส่งระดับกายภาพทั้ง 4 ฟังก์ชันลงใน Socket โดยเรียกใช้ฟังก์ชัน control_interrupt ของส่วนประสานงาน

ขั้นที่ 4 : กำหนดส่วนของโปรแกรมที่จะถูกฝังอยู่ในหน่วยความจำ และจบโปรแกรม

ต่อไปจะขอกล่าวถึงรายละเอียดเกี่ยวกับหน้าที่ และขั้นตอนการทำงานของฟังก์ชันควบคุมการรับส่งระดับกายภาพทั้ง 4 ฟังก์ชัน

ก) ฟังก์ชันตรวจสอบสถานะความพร้อมส่ง (out_status) จะทำหน้าที่คอยตรวจสอบค่าของรีจิสเตอร์แสดงสถานะสายสื่อสาร ณ ตำแหน่ง 3FD ถ้ามีค่าเป็น 20H แสดงว่าพร้อมที่จะส่งข้อมูลได้ ขั้นตอนการทำงานแสดงด้วยภาษาแอสเซมบลีดังต่อไปนี้

```
01)   out_status   proc   near
02)                push   ax
03)                push   dx
04)                mov    dx,cs:curport
05)                add    dx,5            ;point to address of LSR
06)                in     al,dx
07)                test   al,20h          ;test transmitter holding register empty
08)                pop    dx
09)                pop    ax
10)   out_status   endp
```

ในบรรทัดที่ 4 เป็นการกำหนดตำแหน่งเริ่มต้นของพอร์ต ถ้าเป็น COM1 จะเท่ากับ 3f8 และ COM2 เท่ากับ 2f8 ค่าของพอร์ตอนุกรมจะเป็นตัวแปร ซึ่งจะถูกกำหนดโดยผู้ใช้ขณะเรียกใช้โปรแกรมนี้ ส่วนบรรทัดที่ 5 จะเป็นการคำนวณหาตำแหน่งของรีจิสเตอร์สถานะสายสื่อสารเพื่อทำการอ่านข้อมูลเข้ามาเก็บที่รีจิสเตอร์นี้ในบรรทัดที่ 6 และตรวจสอบค่าในบรรทัดที่ 7 ถ้าค่าที่ได้รับเป็น 20H ก็จะส่งค่าแฟล็กเป็น 1 กลับคืนไปให้

ข) ฟังก์ชันส่งข้อมูล (out_byte) จะทำหน้าที่ส่งข้อมูลครั้งละ 1 ไบต์ โดยก่อนจะทำการส่งจะทำการคำนวณหา checksum แล้วเรียกฟังก์ชันตรวจสอบความพร้อมส่ง จากนั้นตรวจสอบค่าแฟล็กที่ได้รับ ถ้าค่าเป็น 1 จึงจะสามารถส่งข้อมูลได้ ขั้นตอนการทำงานแสดงด้วยภาษาแอสเซมบลีดังต่อไปนี้

```
01)  out_byte     proc    near                ;Transmit Byte to Current Socket
02)               bpbios  link_int,calc_checksum
03)  putby1:      call    cs:out_stat
04)               jz      putby1
05)               call    out_port
06)               retf
```

```
07) out_byte    endp
08) out_port    proc    near
09)             push    dx
10)             mov     dx,cs:curport
11)             out     dx,al
12)             pop     dx
13)             ret
14) out_port    endp
```

ในบรรทัดที่ 2 เป็นการเรียกส่วนควบคุมการรับส่งระดับเชื่อมโยงข้อมูล เพื่อคำนวณหา checksum ส่วนบรรทัดที่ 3 เป็นการวนเรียกฟังก์ชันตรวจสอบความพร้อมส่งจนกว่าจะได้รับค่าแฟล็กเป็น 1 จึงจะทำการส่งข้อมูลในบรรทัดที่ 9 และ 10

ค) ฟังก์ชันตรวจสอบสถานะการรับข้อมูล (in_status) จะทำหน้าที่ตรวจสอบค่าในรีจิสเตอร์แสดงสถานะสายสื่อสาร ถ้ามีค่าเป็น 01H แสดงว่าได้รับข้อมูลมาเก็บไว้ในบัฟเฟอร์ครบทุกบิตแล้ว ก็จะส่งค่าแฟล็กเป็น 1 กลับคืนไปให้ ขั้นตอนการทำงานแสดงด้วยภาษาแอสเซมบลีดังต่อไปนี้

```
01) in_status   proc    near
02)             push    ax
03)             push    dx
04)             mov     dx,cs:curport
05)             add     dx,5            ;point to address of LSR
06)             in      al,dx
07)             test    al,1            ;test data ready
08)             pop     dx
09)             pop     ax
10) in_status   endp
```

ง) ฟังก์ชันอ่านข้อมูลที่ได้รับ (in_byte) ทำหน้าที่อ่านข้อมูลจากบัฟเฟอร์ครั้งละ 1 ไบต์ หลังจากตรวจสอบสถานะการได้รับข้อมูล แล้วค่าแฟล็กเป็น 1 จากนั้นทำการคำนวณหาค่า checksum ขั้นตอนการทำงานแสดงด้วยภาษาแอสเซมบลีดังต่อไปนี้

```
01) in_byte     proc    near            ;Receive Byte from Current Socket
02) getby1:     call    cs:in_stat
03)             jz      getby1
04)             call    in_port
05)             bpbios  link_int,calc_checksum
```

```
06)                retf
07)  in_byte       endp
08)  in_port       proc    near
09)                push    dx
10)                mov     dx,cs:curport
11)                in      al,dx
12)                pop     dx
13)                ret
14)  in_port       endp
```

ในบรรทัดที่ 2 และ 3 เป็นการวนเรียกใช้ฟังก์ชันตรวจสอบสถานะการได้รับข้อมูลจนกว่าจะได้รับค่าแฟล็กเป็น 1 จึงจะทำการอ่านข้อมูลจากบัฟเฟอร์ในบรรทัดที่ 10 และ 11

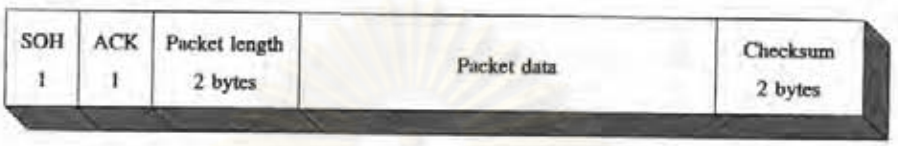
ฟังก์ชัน physical_interrupt ทำหน้าที่ให้บริการการเรียกใช้ฟังก์ชันควบคุมการรับส่งระดับกายภาพทั้ง 4 ฟังก์ชันของแต่ละ Socket ขั้นตอนการทำงานแสดงด้วยภาษาแอสเซมบลีดังต่อไปนี้

```
01)  physical_interrupt_table  dd  4*maxsockets dup(?)
02)  physical_interrupt        proc  far
03)                            push  bx
04)                            push  cx
05)                            xor   bh,bh
06)                            mov   cl,2
07)                            shl   bx,cl
08)                            add   bl,ah
09)                            shl   bx,cl
10)                            call  cs:physical_interrupt_table[bx]
11)                            pop   cx
12)                            pop   bx
13)                            retf  2
14)  physical_interrupt        endp
```

ในบรรทัดที่ 1 เป็นการกำหนดเนื้อที่สำหรับเก็บ Socket ส่วนบรรทัดที่ 6 และ 7 เป็นการเลื่อนตำแหน่งใน physical_interrupt_table ให้ชี้ไปที่ Socket ที่ได้รับในรีจิสเตอร์ BL บรรทัดที่ 8 และ 9 เป็นการเลื่อนตำแหน่งใน Socket ให้ชี้ไปที่ฟังก์ชันที่ได้รับในรีจิสเตอร์ AH จากนั้นจะทำการเรียกใช้ฟังก์ชันใน Socket นั้นในบรรทัดที่ 10

2) การออกแบบส่วนควบคุมการรับส่งระดับเชื่อมโยงข้อมูล

หลักการทำงานของส่วนนี้จะใช้โปรโตคอลในการควบคุมการรับส่งข้อมูลแบบกลุ่ม (Packet) ซึ่งในการวิจัยนี้จะออกแบบโปรโตคอลโดยดัดแปลงจากโปรโตคอล XMODEM เนื่องจากเป็นโปรโตคอลที่เข้าใจง่าย ไม่ซับซ้อน และการตรวจสอบความถูกต้องด้วยวิธี checksum สามารถให้ความถูกต้องได้ถึง 99.6%[1] รูปแบบโปรโตคอลรับส่งเป็นดังนี้



ไบต์แรกเป็นส่วนเริ่มต้นข้อมูล กำหนดค่าเป็น SOH ซึ่งมีค่าแอสกีเท่ากับ 01H
ไบต์ที่ 2 เป็นการแสดงการตอบรับ หรือ Acknowledgement กำหนดค่าเป็น ACK หรือ 06H
ไบต์ที่ 3-4 เป็นความยาวข้อมูล ตามด้วยข้อมูลขนาดไม่เกิน 64 กิโลไบต์
สองไบต์สุดท้ายเป็น checksum

การเรียกใช้ส่วนควบคุมการรับส่งระดับเชื่อมโยงข้อมูลนี้ จะผ่านทางตัวขัดจังหวะหมายเลข 82H โดยกำหนดรหัสของฟังก์ชันต่าง ๆ ในรีจิสเตอร์ AH ดังรายละเอียดต่อไปนี้

ฟังก์ชัน 00H - ล้างค่า checksum เป็น 0
เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 00H
ส่งค่ากลับ: ไม่มี

ฟังก์ชัน 01H - คำนวณหาค่า checksum
เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 01H และ
AL = ข้อมูล 1 ไบต์ที่จะบวกเข้าใน checksum
ส่งค่ากลับ: ไม่มี

ฟังก์ชัน 02H - สอบถามค่า checksum
เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 02H
ส่งค่ากลับ: ค่า checksum ลงในรีจิสเตอร์ AX

ฟังก์ชัน 03H - ส่ง packet
เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 03H
BL = หมายเลข Socket

---

[1] Jordan and Churchill, Communications and Networking for the IBM PC and Compatibles, 3rd ed. (New York: Brady Books, a division of Simon & Schuster, Inc., 1990), p.186

DS:DX = แอดเดรสของ packet ที่จะส่ง

CX = ความยาวของ packet ที่จะส่ง

ส่งค่ากลับ: ไม่มี

ฟังก์ชัน 04H - รับ packet

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 04H

BL = หมายเลข Socket

DS:DX = แอดเดรสของบัฟเฟอร์สำหรับเก็บข้อมูลที่ได้รับ

ส่งค่ากลับ: CX = ความยาวของกลุ่มข้อมูลที่ได้รับ

ฟังก์ชัน 05H - สอบถามจำนวนครั้งที่ส่งซ้ำ

เรียกใช้โดย: กำหนดให้รีจิสเตอร์ AH = 05H

ส่งค่ากลับ: AL = จำนวนครั้งที่ส่งซ้ำ

โปรแกรมควบคุมการรับส่งระดับเชื่อมโยงข้อมูลนี้ประกอบไปด้วยฟังก์ชันสำหรับตรวจสอบข้อมูล และฟังก์ชันรับส่งข้อมูล ซึ่งขั้นตอนการทำงานจะเริ่มจากการตรวจสอบค่าของฟังก์ชันในรีจิสเตอร์ AH ที่ได้รับ และกระโดดไปทำงานตามฟังก์ชันนั้น เสร็จแล้วจะส่งค่ากลับไปให้โปรแกรมที่เรียกใช้ (ถ้ามี) ผังงานโปรแกรมแสดงได้ดังรูปที่ 3.9

หน้าที่และขั้นตอนการทำงานของแต่ละฟังก์ชัน มีรายละเอียดดังต่อไปนี้

ก) ฟังก์ชันล้างค่า checksum จะทำการล้างค่าของ checksum ให้เป็น 0 ฟังก์ชันนี้จะถูกเรียกใช้ทุกครั้งเมื่อเริ่มส่งหรือรับ packet

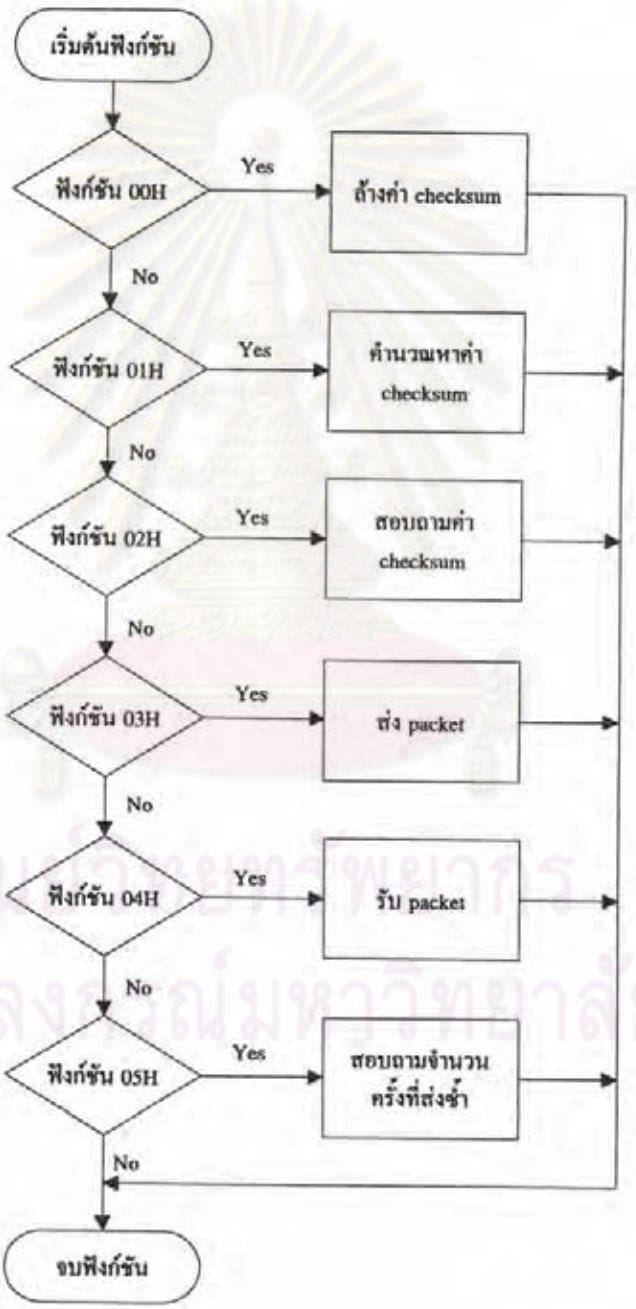
ข) ฟังก์ชันคำนวณหาค่า checksum ฟังก์ชันนี้จะคำนวณหา checksum โดยวิธีการบวกค่าแอสกีของข้อมูล 1 ไบต์เก็บไว้ใน checksum จะถูกเรียกใช้ทุกครั้งที่มีการส่งหรือรับข้อมูลแต่ละ packet

ค) ฟังก์ชันสอบถามค่า checksum จะส่งค่าของ checksum กลับไปให้โปรแกรมที่เรียกใช้

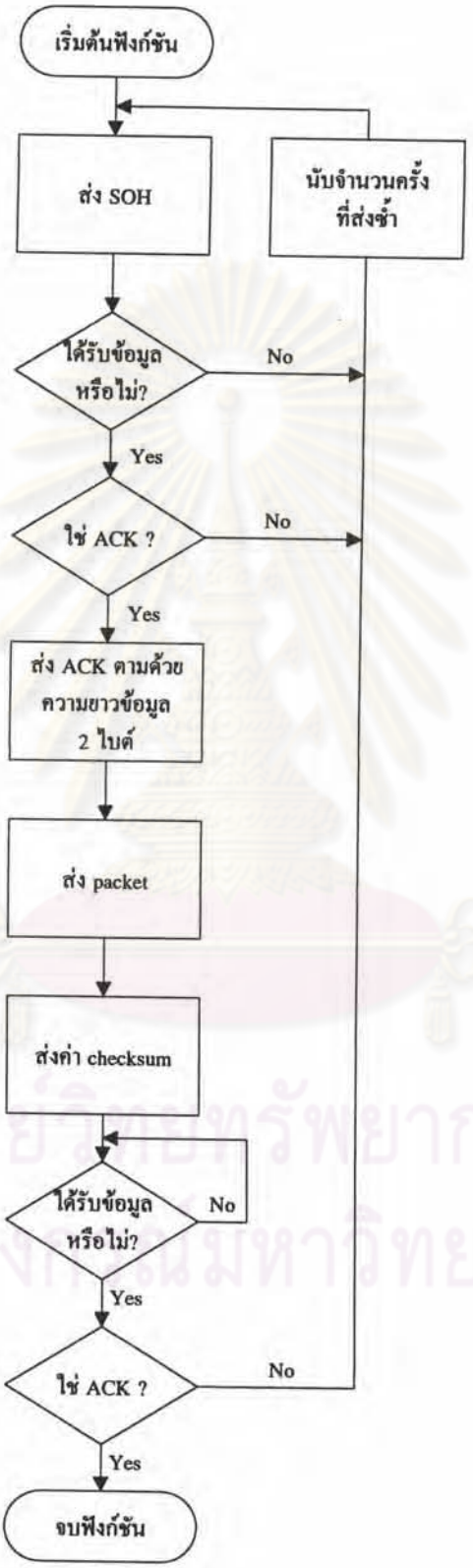
ง) ฟังก์ชันส่ง packet ในการส่งข้อมูลแต่ละ packet จะเริ่มด้วยการส่ง SOH ไปที่ฝ่ายรับก่อนเพื่อเป็นการตรวจสอบว่าฝ่ายรับพร้อมที่จะรับข้อมูลหรือไม่ และจะคอยวนตรวจสอบการตอบรับจากฝ่ายรับโดยเรียกใช้ส่วนควบคุมการรับส่งระดับกายภาพ เมื่อได้รับข้อมูลจะตรวจสอบว่าเป็น ACK หรือไม่ ถ้าไม่ใช่ก็จะเริ่มส่ง SOH ซ้ำใหม่ และทำการนับจำนวนครั้งที่ส่งซ้ำด้วย แต่ถ้าใช่ก็จะส่ง ACK ความยาวกลุ่มข้อมูล ตามด้วยกลุ่มข้อมูล และค่าของ checksum ที่คำนวณได้ แล้วคอยรับการตอบรับจากฝ่ายรับ ถ้าค่าที่ได้รับเป็น ACK แสดงว่าข้อมูลที่ส่งถูกต้อง แต่ถ้าไม่ใช่จะเริ่มส่งข้อมูลซ้ำใหม่ การส่งซ้ำจะดำเนินไปกี่ครั้งนั้นทางส่วนโปรแกรมที่เรียกใช้จะต้องควบคุมเอง โดยสามารถสอบถามจากฟังก์ชัน 05H ที่ได้กล่าวแล้วข้างต้น ผังงานของฟังก์ชันแสดงได้ดังรูปที่ 3.10

จ) ฟังก์ชันรับ packet ในการรับข้อมูลทางฝ่ายรับจะคอยวนตรวจสอบว่ามีข้อมูลถูกส่งเข้ามาหรือไม่ เมื่อได้รับข้อมูลจะตรวจสอบถ้าเป็น SOH ก็จะส่ง ACK แจ้งกลับไปให้ทางฝ่ายส่ง

ส่งข้อมูลต่อมาได้ เมื่อทางฝ่ายส่งส่งข้อมูลมาก็จะตรวจสอบ SOH อีกครั้ง เพื่อรับข้อมูลกรณีส่งซ้ำ ถ้าเป็น SOH ก็จะกลับไปคอยรับข้อมูลใหม่ ถ้าไม่เป็นก็จะตรวจสอบ ACK ถ้าเป็น ACK ก็จะคอยรับความยาวข้อมูล กลุ่มข้อมูล และค่า checksum จากนั้นจะทำการเปรียบเทียบค่า checksum ที่ได้รับกับค่าที่คำนวณได้ ถ้าเท่ากันแสดงว่าข้อมูลที่ได้รับนั้นถูกต้องก็จะส่ง ACK กลับไปบอกฝ่ายส่ง แต่ถ้าไม่เท่าก็จะส่ง NAK กลับไปแทน ผังงานของฟังก์ชันแสดงได้ดังรูปที่ 3.11



รูปที่ 3.9 แสดงผังงานส่วนควบคุมการรับส่งระดับเชื่อมโยงข้อมูล



รูปที่ 3.10 แสดงผังงานฟังก์ชันส่ง Packet

รูปที่ 3.11 แสดงผังงานฟังก์ชันรับ Packet

การออกแบบส่วนควบคุมความมั่นคงข่ายงาน

สำหรับการควบคุมความมั่นคงของข่ายงานในการวิจัยนี้จะกระทำใน 2 ระดับ คือ

1) ระดับการเข้าถึงข่ายงาน ในระดับนี้เป็นการควบคุมการเข้าถึงทั้งระบบข่ายงาน โดยจะกำหนดให้มีรหัสประจำตัวผู้ใช้และรหัสผ่าน ผู้ใช้จะต้องป้อนรหัสเหล่านี้ก่อนจึงจะสามารถเข้าถึงระบบได้ และเมื่อผู้ใช้สามารถเข้าถึงระบบได้แล้ว ก็จะมีสิทธิเข้าถึงทรัพยากรทั้งหมดของระบบ

2) ระดับผู้ใช้ เป็นการควบคุมสิทธิในการเข้าถึงทรัพยากรระบบของผู้ใช้แต่ละคนว่ามีสิทธิเข้าถึงแบบใด โดยจะมีการแบ่งกลุ่มผู้ใช้ตามระดับหน้าที่และความสำคัญออกเป็น 2 กลุ่ม ดังนี้

2.1) กลุ่มผู้บริหารระบบ เป็นกลุ่มที่ทำหน้าที่ติดตั้งและดูแลรักษาระบบให้สามารถปฏิบัติงานได้ กลุ่มนี้จะมีสิทธิในการเข้าถึงทรัพยากรระบบได้ทุกแบบ

2.2) กลุ่มผู้ใช้ทั่วไป เป็นกลุ่มผู้ใช้นอกเหนือจากกลุ่มผู้บริหารระบบที่ได้กล่าวมาแล้ว การที่ผู้ใช้แต่ละคนจะมีสิทธิในการเข้าถึงทรัพยากรแบบใดนั้น ขึ้นอยู่กับหน้าที่ความรับผิดชอบที่ได้รับมอบหมาย ซึ่งทางกลุ่มผู้บริหารระบบจะเป็นผู้สร้างรหัสประจำตัว รหัสผ่าน และสิทธิในการเข้าถึงตามหน้าที่นั้น ๆ นอกจากนี้ยังมีการควบคุมสิทธิในการใช้โปรแกรมในส่วนควบคุมความมั่นคงว่าโปรแกรมไหนจะให้กลุ่มผู้ใช้ทั่วไปใช้ได้บ้าง

ในการควบคุมความมั่นคงข่ายงานนี้ จำเป็นต้องมีการเก็บข้อมูลของผู้ใช้ลงในแฟ้มที่เครื่องบริการ เพื่อให้โปรแกรมควบคุมความมั่นคงสามารถตรวจสอบสิทธิการเข้าถึงระบบได้ แฟ้มข้อมูลนี้มีลักษณะเป็นแฟ้มลำดับ (Sequential File) ชนิดข้อความ (Text) เรียกว่า "แฟ้มผู้ใช้" ซึ่งให้ชื่อแฟ้มว่า "APATHUSR.SEQ" มีโครงสร้างข้อมูลผู้ใช้เป็นดังนี้

เขต (Field) แรก เป็นรหัสประจำตัวผู้ใช้ขนาดไม่เกิน 10 ไบต์
เขตที่สอง เป็นตัวคั่นระหว่างรหัสประจำตัวผู้ใช้และรหัสผ่าน
เขตที่สาม เป็นรหัสผ่านขนาดไม่เกิน 10 ไบต์
เขตที่สี่ เป็นสิทธิในการเข้าถึงแฟ้ม และกำหนดประเภทกลุ่มผู้ใช้ โดยกำหนดค่าดังต่อไปนี้
'0' หมายถึง กลุ่มผู้ใช้ทั่วไป และมีสิทธิเข้าถึงแฟ้มได้ทุกแบบคือ อ่าน บันทึก และลบได้
'1' หมายถึง กลุ่มผู้ใช้ทั่วไป และมีสิทธิอ่านแฟ้มได้เพียงอย่างเดียว

'9' หมายถึง กลุ่มผู้บริหารระบบ มีสิทธิในการเข้าถึงแฟ้ม และโปรแกรมควบคุมความมั่นคงได้ทั้งหมดทุกแบบ ส่วนกลุ่มผู้ใช้ทั่วไปจะมีสิทธิเข้าถึงเฉพาะโปรแกรมควบคุมความมั่นคงในส่วนผู้ใช้เท่านั้น

เขตสุดท้าย เป็นตัวคั่นระหว่างข้อมูลผู้ใช้แต่ละคน

ในการเก็บข้อมูลลงแฟ้มผู้ใช้นี้จะเก็บเป็นแบบความยาวแปรได้ (Variable Length) คือ เก็บตามจำนวนข้อมูลที่มีอยู่จริง (ไม่เก็บ space) และจะมีตัวคั่นระหว่างข้อมูลผู้ใช้และตัวคั่นระหว่างเขตข้อมูลผู้ใช้แต่ละคน ซึ่งในการวิจัยนี้จะใช้ค่า **0BH** เป็นตัวคั่นระหว่างรหัสประจำตัวผู้ใช้กับรหัสผ่าน และใช้ค่า **0CH** เป็นตัวคั่นระหว่างรหัสประจำตัวผู้ใช้แต่ละคน นอกจากนี้ในการเก็บข้อมูลจะทำการเข้ารหัส (Encode) ข้อมูลก่อน เพื่อป้องกันไม่ให้ผู้ใช้สามารถทราบรหัสต่าง ๆ ของผู้อื่นจากแฟ้มได้ง่าย วิธีการเข้ารหัสจะกระทำโดยการเปลี่ยนบิตที่ 8 ของแต่ละไบต์จาก 0 เป็น 1 และจาก 1 เป็น 0

สำหรับกลุ่มข้อมูล (packet data) ในโปรโตคอล ที่จะนำมาใช้ในส่วนควบคุมความมั่นคงข่ายงาน มีรูปแบบดังต่อไปนี้

เขตแรก เป็นรหัสประจำส่วน ขนาด 1 ไบต์ กำหนดค่าเป็น 'S' (Security Control)

เขตที่สอง เป็นรหัสฟังก์ชันที่จะเรียกใช้ ขนาด 1 ไบต์ กำหนดค่าเป็น

'0' หมายถึง ฟังก์ชันตรวจสอบสิทธิการเข้าถึงระบบ (chkauth)

'1' หมายถึง ฟังก์ชันเปลี่ยนรหัสผ่าน (chgusrpwd)

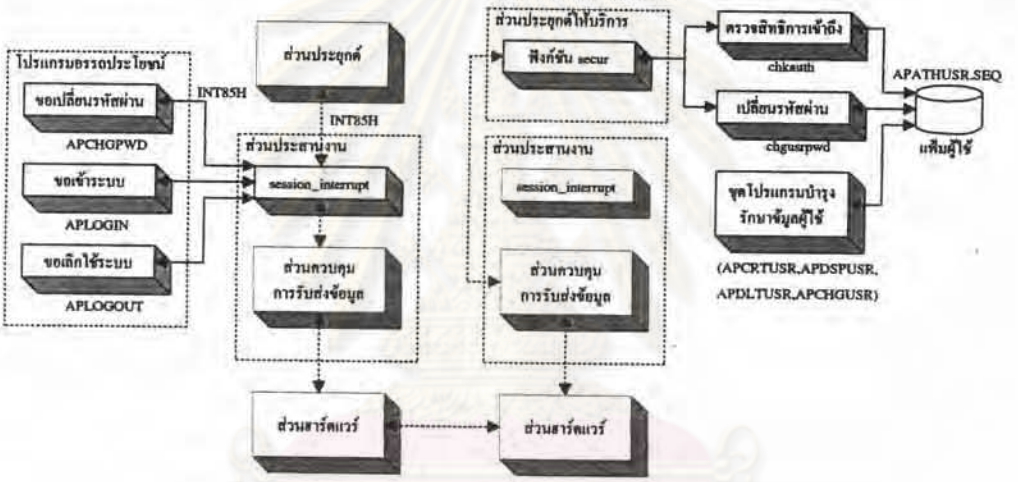
เขตที่สาม เป็นรหัสประจำตัวผู้ใช้ ขนาด 10 ไบต์

เขตสุดท้าย เป็นรหัสผ่าน ขนาด 10 ไบต์

เมื่อได้ทราบถึงรูปแบบ และวิธีการเก็บข้อมูลของผู้ใช้แล้ว ต่อไปจะกล่าวถึงการออกแบบหลักการทำงานของส่วนนี้

สำหรับส่วนควบคุมความมั่นคงข่ายงานที่ได้กล่าวถึงในเรื่องโครงสร้างระบบตอนต้นบทนั้น จะแบ่งออกเป็น 2 ส่วน คือ ส่วนควบคุมความมั่นคงในส่วนผู้ใช้ ซึ่งจะประกอบไปด้วยฟังก์ชันควบคุมความมั่นคงที่อยู่ในฟังก์ชัน session_interrupt ของส่วนประสานงาน และโปรแกรมอรรถประโยชน์ต่าง ๆ อีกส่วนหนึ่งคือ ส่วนควบคุมความมั่นคงในส่วนผู้ให้บริการ ซึ่งจะประกอบด้วยฟังก์ชันในส่วนประยุกต์ และโปรแกรมบำรุงรักษาความมั่นคงต่าง ๆ โปรแกรมทั้งสองส่วนนี้จะทำงานสัมพันธ์กัน ตัวอย่างเช่น เมื่อผู้ใช้เรียกโปรแกรมสำหรับเข้าระบบ โปรแกรมจะเรียกใช้ตัวขัดจังหวะหมายเลข 85H

ซึ่งจะโอนการควบคุมไปให้ฟังก์ชันควบคุมความมั่นคงในส่วนประสานงาน ฟังก์ชันนี้จะทำการตรวจสอบว่าขณะนี้กำลังอยู่ในระบบหรือไม่ ถ้าอยู่ก็จะแสดงข้อความเตือนบนจอภาพ แต่ถ้าไม่อยู่ก็จะแสดงข้อความให้ผู้ใช้ป้อนรหัสประจำตัว และ รหัสผ่าน จากนั้นก็จะนำรหัสที่ได้ไปจัดเตรียมรูปแบบกลุ่มข้อมูล แล้วส่งไปยังเครื่องบริการ เมื่อโปรแกรมตรวจสอบรหัสของส่วนผู้ให้บริการได้รับข้อมูลนั้น ก็จะทำการตรวจสอบรหัสประจำตัวผู้ใช้ และรหัสผ่านที่ได้รับกับที่มีอยู่ในแฟ้มผู้ใช้ ถ้าพบแสดงว่าผู้ใช้คนนั้นมีสิทธิเข้าถึงระบบก็จะส่งสิทธิในการเข้าถึงแฟ้มกลับไปยังเครื่องผู้ใช้ ฟังก์ชันควบคุมความมั่นคงส่วนผู้ใช้จะเก็บข้อมูลไว้ในหน่วยความจำเพื่อใช้ตรวจสอบครั้งต่อไปโดยไม่ต้องส่งไปตรวจสอบที่เครื่องบริการอีกจนกว่าจะมีการบูตเครื่องใหม่ แต่ถ้าไม่พบก็จะส่งรหัสข้อผิดพลาดกลับไปให้แทน โครงสร้างและความสัมพันธ์ภายในส่วนควบคุมความมั่นคงข่ายงานแสดงดังรูปที่ 3.12 (ที่แสดงด้วยเส้นทึบ)



รูปที่ 3.12 แสดงโครงสร้างและความสัมพันธ์ภายในส่วนควบคุมความมั่นคงข่ายงาน

ต่อไปจะกล่าวถึงการออกแบบในรายละเอียดเกี่ยวกับรูปแบบการรับ และ แสดงผลข้อมูลตลอดจนขั้นตอนการทำงานของโปรแกรมต่าง ๆ ในแต่ละส่วน ดังต่อไปนี้

1) การออกแบบโปรแกรมควบคุมความมั่นคงส่วนผู้ใช้ โปรแกรมส่วนนี้จะประกอบไปด้วยโปรแกรม และฟังก์ชันต่าง ๆ ได้แก่

ก) ฟังก์ชันควบคุมความมั่นคง ทำหน้าที่ให้บริการเรียกใช้ฟังก์ชันควบคุมการเข้าระบบและเลิกใช้ระบบ นอกจากนี้ยังให้บริการสอบถามสิทธิการเข้าถึงระบบด้วย การเรียกใช้ฟังก์ชันนี้จะเรียกผ่านฟังก์ชัน session_interrupt โดยใช้ตัวขัดจังหวะหมายเลข 85H ซึ่งประกอบไปด้วยฟังก์ชันต่าง ๆ ที่เรียกใช้ดังนี้

ฟังก์ชัน 05H - ควบคุมการเข้าระบบ (Log in)

เรียกใช้โดย : กำหนดให้รีจิสเตอร์ AH = 05H

BL = หมายเลข Socket

DS:DX = แอดเดรสของกลุ่มข้อมูลที่ส่ง

ส่งค่ากลับ : รหัสสถานะภาพต่าง ๆ ผ่านทางแอดเดรสของกลุ่มข้อมูลที่ส่ง

ฟังก์ชัน 06H - ควบคุมการเลิกใช้ระบบ (Log out)

เรียกใช้โดย : กำหนดให้รีจิสเตอร์ AH = 06H

BL = หมายเลข Socket

DS:DX = แอดเดรสของกลุ่มข้อมูลที่ส่ง

ส่งค่ากลับ : ไม่มี

ฟังก์ชัน 07H - สอบถามสิทธิการเข้าถึงระบบ

เรียกใช้โดย : กำหนดให้รีจิสเตอร์ AH = 07H

BL = หมายเลข Socket

DS:DX = แอดเดรสของกลุ่มข้อมูลที่ส่ง

ส่งค่ากลับ : รหัสสถานะภาพต่าง ๆ ผ่านทางแอดเดรสของกลุ่มข้อมูลที่ส่ง

ในฟังก์ชัน session_interrupt นี้จะเก็บค่าตัวแปรที่สำคัญไว้ เพื่อให้โปรแกรมอื่นเรียกใช้ได้ ซึ่งแสดงด้วยภาษาแอสเซมบลีได้ดังนี้

login_rdy db 'N' เป็นตัวแปรที่แสดงว่าขณะนี้กำลังอยู่ในระบบแล้วหรือไม่ มีไว้ให้ตรวจสอบเพื่อป้องกันการ login ซ้ำ ค่าเป็น 'Y' หมายถึง กำลังอยู่ในระบบ

wrtallow db 1 เป็นตัวแปรที่เก็บค่าสิทธิในการเข้าถึงแฟ้ม

ต่อไปจะกล่าวถึงรายละเอียดของแต่ละฟังก์ชันย่อยภายในฟังก์ชันควบคุมความมั่นคง

ฟังก์ชันเข้าระบบ (login) ทำหน้าที่ควบคุมการเข้าระบบ จะถูกเรียกใช้โดยโปรแกรม APLOGIN.COM มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบค่าในตัวแปร login_rdy ถ้าเป็น 'Y' แสดงว่าขณะนี้กำลังอยู่ในระบบแล้ว ไม่สามารถเข้าระบบได้ก็จะส่งค่า '$' กลับไปให้ แล้วจบการทำงาน แต่ถ้าเป็น 'N' จะทำขั้นตอนต่อไป

ขั้นที่ 2: แสดงข้อความบนจอภาพครั้งละ 1 บรรทัด เพื่อให้ผู้ใช้ป้อนรหัสประจำตัวและรหัสผ่าน ดังนี้

```
Enter user id   : xxxxxxxxxx
Enter password : xxxxxxxxxx
```

เนื่องจากรหัสผ่านเป็นข้อมูลที่ต้องเก็บรักษาเป็นความลับ ดังนั้นในการป้อนรหัสผ่านจะเรียกใช้ตัวขัดจังหวะ 21H ฟังก์ชัน 08H ของดอส ซึ่งจะไม่ปรากฏข้อความให้เห็นขณะที่ป้อนข้อมูล

ขั้นที่ 3: หลังจากผู้ใช้ป้อนรหัสเสร็จแล้ว ก็จะส่งไปตรวจสอบสิทธิการเข้าถึงระบบที่เครื่องบริการโดยเรียกใช้ฟังก์ชัน session_interrupt แล้วรอรับผลการตรวจสอบ

ขั้นที่ 4: เมื่อได้รับผลการตรวจสอบก็จะเก็บไว้ใน login_rdy และ wrtallow เพื่อให้โปรแกรมที่เรียกใช้นำไปตรวจสอบอีกครั้งหนึ่ง แล้วจบการทำงาน

ฟังก์ชันเลิกใช้ระบบ (logout) ทำหน้าที่ควบคุมการเลิกใช้ระบบ จะถูกเรียกใช้โดยโปรแกรม APLOGOUT.COM มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: เปลี่ยนค่าของ login_rdy ให้เป็น 'N' และค่าของ wrtallow ให้เป็น 1

ขั้นที่ 2: จบการทำงาน

ฟังก์ชันสอบถามสิทธิการเข้าระบบ ทำหน้าที่ให้ข้อมูลเกี่ยวกับการเข้าระบบว่าขณะนี้กำลังอยู่ในระบบหรือไม่ ให้รหัสประจำตัวผู้ใช้ และให้ค่าสิทธิการเข้าถึงแฟ้ม มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: ส่งค่า login_rdy, wrtallow และรหัสประจำตัวผู้ใช้กลับไปให้โปรแกรมที่เรียกใช้

ขั้นที่ 2: จบการทำงาน

ข) โปรแกรมขอเข้าระบบ (APLOGIN.COM) เป็นโปรแกรมอรรถประโยชน์ที่ทำหน้าที่ขอเข้าใช้ระบบ โดยการเรียกใช้ฟังก์ชันเข้าระบบใน session_interrupt แล้วตรวจสอบค่า login_rdy ที่ได้รับ ถ้าเป็น 'S' หมายถึงขณะนี้อยู่ในระบบแล้ว จะแสดงข้อความบนจอภาพว่า 'have already logged in' หรือ ถ้าค่าเป็น 'N' หมายถึงไม่มีสิทธิเข้าถึงระบบ ก็จะแสดงข้อความบนจอภาพว่า 'access denied' โปรแกรมนี้จะถูกพัฒนาด้วยภาษาแอสเซมบลี

ค) โปรแกรมขอเลิกใช้ระบบ (APLOGOUT.COM) เป็นโปรแกรมอรรถประโยชน์ที่ทำหน้าที่ขอเลิกใช้ระบบโดยการเรียกใช้ฟังก์ชันเลิกใช้ระบบใน session_interrupt หลังจากจบโปรแกรมนี้แล้วจะไม่สามารถเข้าถึงระบบได้ จนกว่าจะมีการเรียกใช้โปรแกรมขอเข้าระบบใหม่อีกครั้ง โปรแกรมนี้จะถูกพัฒนาด้วยภาษาแอสเซมบลี

ง) โปรแกรมขอเปลี่ยนรหัสผ่าน (APCHGPWD.EXE) เป็นโปรแกรมอรรถประโยชน์ที่ช่วยให้ผู้ใช้สามารถขอเปลี่ยนรหัสผ่านจากเครื่องบริการได้ เนื่องจากรหัสผ่านเมื่อถูกใช้ไปนาน ๆ อาจจะมีผู้อื่นทราบได้จึงควรมีการเปลี่ยนบ่อย ๆ โปรแกรมนี้จะถูกพัฒนาด้วยภาษาซี มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้ระบบโดยเรียกใช้ฟังก์ชันสอบถามสิทธิการเข้าระบบ (07H) ถ้าค่าที่ได้รับจาก login_rdy ไม่เป็น 'Y' จะแสดงข้อความบนจอภาพว่า 'access denied' แล้วจบการทำงาน

ขั้นที่ 2: แสดงข้อความบนจอภาพและรับค่ารหัสผ่านเดิม และรหัสผ่านใหม่ ดังนี้

```
Old password : xxxxxxxxxx
New password : xxxxxxxxxx
```

ขั้นที่ 3: เมื่อรับค่าแล้วทำการจัดเตรียมกลุ่มข้อมูลมีรูปแบบดังนี้

เขตแรก มีค่าเป็น 'S'

เขตที่สอง มีค่าเป็น 1

เขตที่สาม เป็นรหัสประจำตัวผู้ใช้ที่ได้จากขั้นตอนที่ 1

เขตที่สี่ เป็นรหัสผ่านเดิม

เขตสุดท้าย เป็นรหัสผ่านใหม่

ขั้นที่ 4: ส่งกลุ่มข้อมูลไปยังเครื่องบริการโดยเรียกใช้ฟังก์ชัน datalink_interrupt แล้วรอรับผลการเรียกใช้

ขั้นที่ 5: ตรวจสอบไบต์แรกของกลุ่มข้อมูลที่ได้รับกลับมา

ถ้าค่าเป็น '0' หมายความว่าไม่สามารถเปลี่ยนรหัสผ่านได้ จะแสดงข้อความให้ปรากฏบนจอภาพว่า 'cannot change your password'

ถ้าค่าเป็น '1' แสดงว่าไม่มีสิทธิเข้าถึงระบบ หรือ รหัสผ่านไม่ถูกต้อง จะแสดงข้อความว่า 'invalid user ID or incorrect password'

ค่านอกเหนือจากนั้นแสดงว่าเปลี่ยนรหัสผ่านได้สำเร็จ จะแสดงข้อความว่า 'Your password has already been changed'

ขั้นที่ 6: จบการทำงาน

2) การออกแบบโปรแกรมควบคุมความมั่นคงส่วนผู้ให้บริการ ในส่วนนี้จะประกอบไปด้วยฟังก์ชันให้บริการตรวจสอบสิทธิการเข้าถึงระบบ การเปลี่ยนรหัสผ่าน และชุดโปรแกรมบำรุงรักษาข้อมูลผู้ใช้ ได้แก่ โปรแกรมสร้าง แก้ไข ลบ และแสดงข้อมูลผู้ใช้ หน้าที่และขั้นตอนการทำงานของฟังก์ชันและโปรแกรมดังกล่าวเป็นดังนี้

ก) ฟังก์ชัน secur เป็นฟังก์ชันที่อยู่ในโปรแกรมควบคุมการให้บริการ (APSRV.COM) ในส่วนประยุกต์ ทำหน้าที่ให้บริการเรียกใช้ฟังก์ชันตรวจสอบสิทธิการเข้าถึงระบบ และฟังก์ชันเปลี่ยนรหัสผ่าน มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบไบต์แรกของกลุ่มข้อมูลที่ได้รับ ถ้าค่าไม่เป็น 'S' หมายความว่าเป็นคำสั่งของส่วนอื่นให้จบการทำงาน ถ้าเป็นให้ทำขั้นตอนต่อไป

ขั้นที่ 2: ตรวจสอบไบต์ที่สอง ถ้าค่าเป็น '0' จะเรียกใช้ฟังก์ชันตรวจสอบสิทธิการเข้าถึงระบบ ถ้าค่าเป็น '1' จะเรียกใช้ฟังก์ชันเปลี่ยนรหัสผ่าน

ขั้นที่ 3: จบการทำงาน

ข) ฟังก์ชันตรวจสอบสิทธิการเข้าถึงระบบ (chkauth) ทำหน้าที่ตรวจสอบรหัสประจำตัวผู้ใช้ว่ามีสิทธิเข้าถึงระบบหรือไม่ โดยตรวจสอบจากแฟ้มผู้ใช้ ฟังก์ชันนี้จะถูกเรียกใช้โดยฟังก์ชัน secur มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: เปิดแฟ้มผู้ใช้

ขั้นที่ 2: ทำการเข้ารหัสข้อมูลผู้ใช้ที่ได้รับ ได้แก่ รหัสประจำตัวผู้ใช้ และรหัสผ่าน

ขั้นที่ 3: อ่านข้อมูลจากแฟ้มผู้ใช้ทั้งแฟ้ม แล้วเก็บไว้ในบัฟเฟอร์

ขั้นที่ 4: ค้นหารหัสที่ได้จากขั้นที่ 2 โดยวิธีเปรียบเทียบสตริงกับในบัฟเฟอร์ ถ้าหาพบแสดงว่ามีสิทธิเข้าถึงระบบ ก็จะส่งค่า 'Y' และสิทธิการเข้าถึงแฟ้ม กลับไปให้ login_rdy และ wrtallow ในฟังก์ชัน session_interrupt ตามลำดับ แต่ถ้าหาไม่พบจะส่งค่า 'N' และ 1 กลับไปให้แทน

ขั้นที่ 5: ปิดแฟ้ม และจบการทำงาน

ค) ฟังก์ชันเปลี่ยนรหัสผ่าน (chgusrpwd) ทำหน้าที่เปลี่ยนรหัสผ่านเดิมเป็นรหัสผ่านใหม่ที่ได้รับ จะถูกเรียกใช้โดยฟังก์ชัน secur มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: เปิดแฟ้มผู้ใช้

ขั้นที่ 2: ทำการเข้ารหัสข้อมูลผู้ใช้ที่ได้รับ ได้แก่ รหัสประจำตัวผู้ใช้ และรหัสผ่าน

ขั้นที่ 3: อ่านข้อมูลจากแฟ้มผู้ใช้ทั้งแฟ้ม แล้วเก็บไว้ในบัฟเฟอร์

ขั้นที่ 4: ค้นหารหัสที่ได้จากขั้นที่ 2 โดยวิธีเปรียบเทียบสตริงกับในบัฟเฟอร์ ถ้าหาพบทำการเปลี่ยนรหัสเดิมในบัฟเฟอร์เป็นรหัสใหม่ นำข้อมูลจากบัฟเฟอร์ไปบันทึกลงแฟ้ม จากนั้นส่งค่า 'S' กลับไปให้โปรแกรม APCHGPWD.EXE แต่ถ้าหาไม่พบก็จะส่งค่า '1' กลับไปให้แทน

ขั้นที่ 5: ปิดแฟ้ม และจบการทำงาน

ง) โปรแกรมบำรุงรักษาข้อมูลผู้ใช้ เป็นชุดโปรแกรมที่ทำหน้าที่บำรุงรักษาข้อมูลผู้ใช้ในแฟ้มผู้ใช้ ประกอบด้วย โปรแกรมสร้างข้อมูลผู้ใช้ โปรแกรมแก้ไขข้อมูลผู้ใช้ โปรแกรมลบข้อมูลผู้ใช้ และโปรแกรมแสดงข้อมูลผู้ใช้ โปรแกรมชุดนี้จะอนุญาตให้เรียกใช้ได้เฉพาะกลุ่มผู้บริหารระบบเท่านั้น กลุ่มผู้ใช้ทั่วไปจะไม่มีสิทธิเรียกใช้ หน้าที่และขั้นตอนการทำงานของแต่ละโปรแกรมมีดังต่อไปนี้

โปรแกรมสร้างข้อมูลผู้ใช้ (APCRTUSR.EXE) ทำหน้าที่สร้างข้อมูลผู้ใช้ใหม่ ได้แก่ รหัสประจำตัวผู้ใช้ รหัสผ่าน และสิทธิการเข้าถึงแฟ้ม ลงในแฟ้มผู้ใช้ มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบแฟ้มผู้ใช้ ถ้าเป็นแฟ้มใหม่ ให้แสดงข้อความบนจอภาพว่า 'Password : ' จากนั้นตรวจสอบรหัสผ่านที่ผู้ใช้ป้อน ถ้าไม่ใช่ 'SYS1995' ให้จบการทำงาน ถ้าใช่ข้ามไปทำขั้นที่ 3

ขั้นที่ 2: ตรวจสอบสิทธิการใช้โปรแกรมนี้ โดยเรียกฟังก์ชันสอบถามสิทธิการเข้าระบบ ถ้าค่าแฟล็กที่ได้รับเป็น 'N' หรือ wrtallow ไม่เท่ากับ 9 หมายความว่าไม่มีสิทธิใช้ ก็จะแสดงข้อความบนจอภาพว่า 'access denied' แล้วจบการทำงาน ถ้ามีสิทธิใช้ก็จะทำขั้นตอนต่อไป

ขั้นที่ 3: เปิดแฟ้มผู้ใช้ แล้วอ่านข้อมูลทั้งแฟ้มเก็บไว้ในบัฟเฟอร์

ขั้นที่ 4: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ป้อนรหัสประจำตัวผู้ใช้

User id. : xxxxxxxxxx

ขั้นที่ 5: ตรวจสอบรหัสประจำตัวผู้ใช้ที่ได้รับโดยทำการเข้ารหัส แล้วค้นหาจากบัฟเฟอร์ ถ้าพบว่ามีอยู่แล้ว จะแสดงข้อความว่า 'This user has already existed in file' แล้วจบการทำงาน

ขั้นที่ 6: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ป้อนรหัสผ่าน และสิทธิการเข้าถึงแฟ้ม

Password : xxxxxxxxxx Access right : 9

ขั้นที่ 7: ตรวจสอบค่าของสิทธิการเข้าถึงแฟ้ม ถ้าเป็นค่านอกเหนือจาก 0, 1 และ 9 จะไม่ให้ผ่านไปทำขั้นตอนต่อไป ผู้ใช้ต้องป้อนข้อมูลเข้ามาใหม่

ขั้นที่ 8: ทำการเข้ารหัสข้อมูลทั้งสอง แล้วนำข้อมูลผู้ใช้ทั้งหมดไปเพิ่มต่อท้ายในบัฟเฟอร์

ขั้นที่ 9: นำข้อมูลในบัฟเฟอร์ไปบันทึกลงในแฟ้มผู้ใช้

ขั้นที่ 10: ปิดแฟ้ม แล้วจบการทำงาน

โปรแกรมแก้ไขข้อมูลผู้ใช้ (APCHGUSR.EXE) ทำหน้าที่แก้ไขเปลี่ยนแปลงรหัสผ่านและสิทธิในการเข้าถึงแฟ้ม มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้โปรแกรมนี้

ขั้นที่ 2: เปิดแฟ้มผู้ใช้ แล้วอ่านข้อมูลทั้งแฟ้มเก็บไว้ในบัฟเฟอร์

ขั้นที่ 3: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ป้อนรหัสประจำตัวผู้ใช้

User id. : xxxxxxxxxx

ขั้นที่ 4: ตรวจสอบรหัสประจำตัวผู้ใช้ที่ได้รับโดยทำการเข้ารหัส แล้วค้นหาจากบัฟเฟอร์ ถ้าหาไม่พบจะแสดงข้อความว่า 'invalid user ID' แล้วจบการทำงาน

ขั้นที่ 5: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ป้อนรหัสผ่าน และสิทธิการเข้าถึงแฟ้ม

New password : xxxxxxxxxx Access right : 9

ขั้นที่ 6: ตรวจสอบค่าของสิทธิการเข้าถึงแฟ้ม

ขั้นที่ 7: ทำการเข้ารหัสข้อมูลทั้งสอง แล้วนำข้อมูลใหม่ทั้งหมดไปเปลี่ยนทับข้อมูลเดิมในบัฟเฟอร์

ขั้นที่ 8: นำข้อมูลในบัฟเฟอร์ไปบันทึกลงในแฟ้มผู้ใช้

ขั้นที่ 9: ปิดแฟ้ม แล้วจบการทำงาน

โปรแกรมลบข้อมูลผู้ใช้ (APDELUSR.EXE) ทำหน้าที่ลบข้อมูลผู้ใช้ออกจากแฟ้มผู้ใช้ มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้โปรแกรมนี้

ขั้นที่ 2: เปิดแฟ้มผู้ใช้ แล้วอ่านข้อมูลทั้งแฟ้มเก็บไว้ในบัฟเฟอร์
ขั้นที่ 3: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ป้อนรหัสประจำตัวผู้ใช้

```
User id.    : xxxxxxxxxx
```

ขั้นที่ 4: ตรวจสอบรหัสประจำตัวผู้ใช้ที่ได้รับโดยทำการเข้ารหัส แล้วค้นหาจากบัฟเฟอร์ ถ้าหาไม่พบ จะแสดงข้อความว่า 'invalid user ID' แล้วจบการทำงาน
ขั้นที่ 5: แสดงข้อความบนจอภาพเพื่อให้ผู้ใช้ยืนยันการลบข้อมูลอีกครั้ง

```
Are you sure to delete this user? (y/n) :
```

ขั้นที่ 6: ถ้าผู้ใช้ตอบว่า 'n' หรือ 'N' หมายความว่ายังไม่ต้องการลบ ให้จบการทำงาน
ขั้นที่ 7: ลบข้อมูลนั้นออกจากบัฟเฟอร์ แล้วนำข้อมูลจากบัฟเฟอร์ไปบันทึกลงในแฟ้มผู้ใช้
ขั้นที่ 8: ปิดแฟ้ม แล้วจบการทำงาน

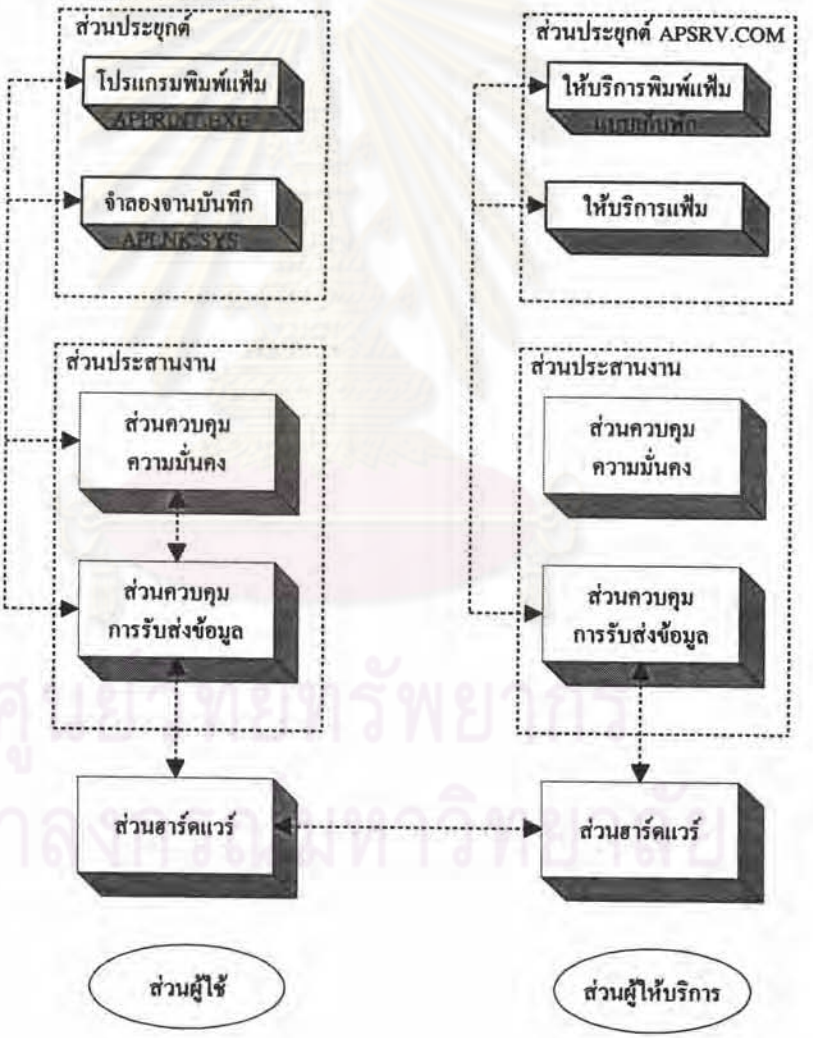
โปรแกรมแสดงข้อมูลผู้ใช้ (APDSPUSR.EXE) ทำหน้าที่แสดงข้อมูลผู้ใช้ทั้งแฟ้มให้ปรากฏบนจอภาพ มีขั้นตอนการทำงานดังต่อไปนี้
ขั้นที่ 1: ตรวจสอบสิทธิการใช้โปรแกรมนี้
ขั้นที่ 2: เปิดแฟ้มผู้ใช้ แล้วอ่านข้อมูลทั้งแฟ้มเก็บไว้ในบัฟเฟอร์
ขั้นที่ 3: ทำการถอดรหัส (Decode) ข้อมูลในบัฟเฟอร์ แล้วนำมาแสดงผลบนจอภาพ ดังนี้

User Id.	Access Right
xxxxxxxxxx	9
xxxxxxxxxx	9
xxxxxxxxxx	9
.	.
.	.
.	.

ขั้นที่ 4: ปิดแฟ้ม แล้วจบการทำงาน

การออกแบบส่วนประยุกต์

ในการประยุกต์ใช้งานข่ายงานสำหรับการวิจัยนี้จะเลือกประยุกต์ใช้ในงานพื้นฐานคือ การใช้จานบันทึก และเครื่องพิมพ์ของเครื่องบริการร่วมกัน ซึ่งในการออกแบบโปรแกรมส่วนประยุกต์ใช้งานดังกล่าว จะแบ่งออกเป็น 2 ส่วนเช่นเดียวกับการออกแบบส่วนควบคุมความมั่นคง คือ ส่วนผู้ใช้ และส่วนผู้ให้บริการ โครงสร้างของส่วนประยุกต์แสดงดังรูปที่ 3.13



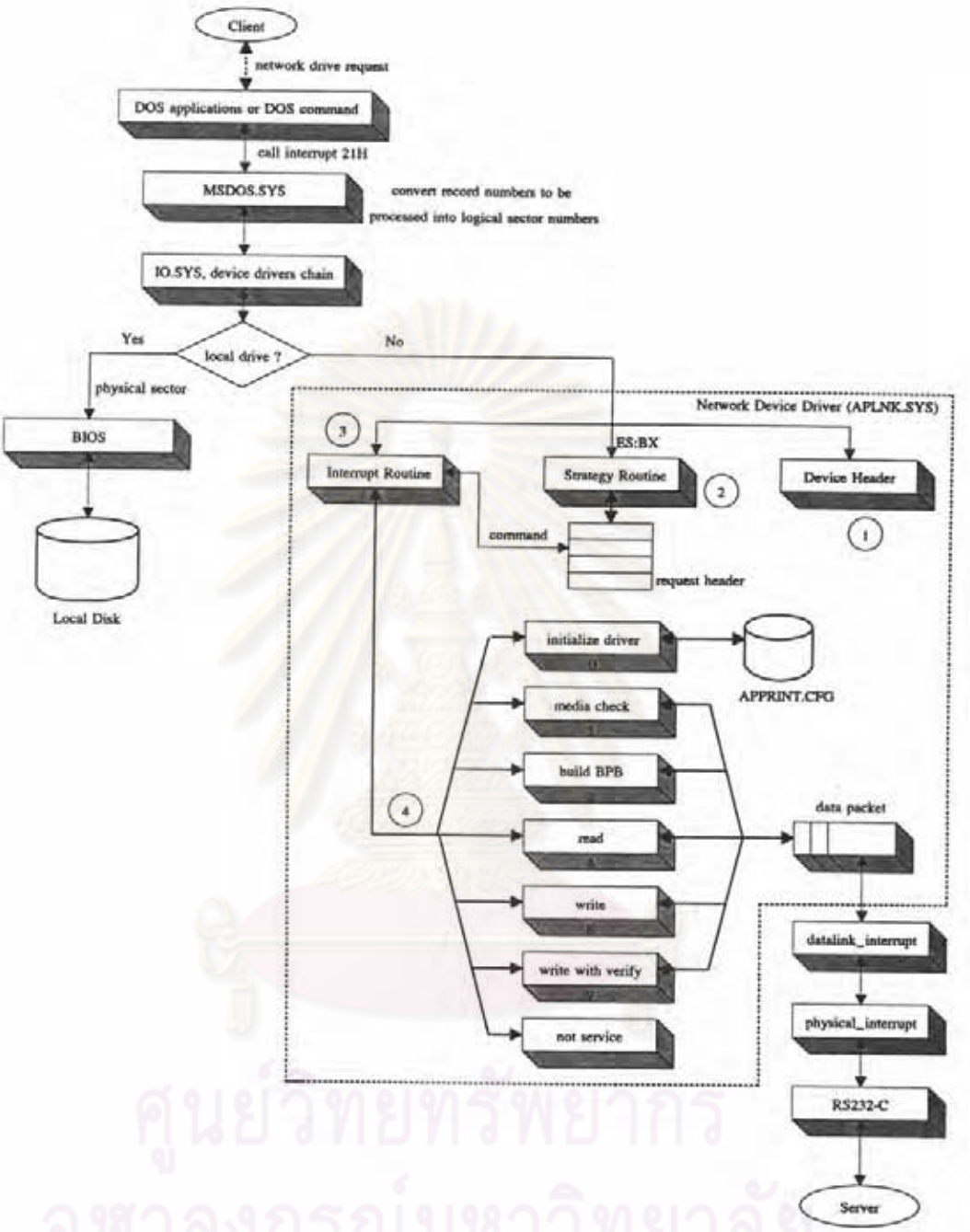
รูปที่ 3.13 แสดงโครงสร้างของส่วนประยุกต์

จากโครงสร้างดังรูปที่ 3.13 จะเห็นได้ว่าในส่วนผู้ใช้จะประกอบไปด้วยโปรแกรมใช้เครื่องพิมพ์ และโปรแกรมจำลองหน่วยขับจานบันทึกของเครื่องบริการ ซึ่งจะตรวจสอบสิทธิการเข้าถึงระบบก่อนโดยเรียกใช้ส่วนควบคุมความมั่นคง จากนั้นจึงจะส่งงานผ่านทางส่วนควบคุมการรับส่งข้อมูล และส่วนฮาร์ดแวร์ไปที่เครื่องบริการ ทางส่วนผู้ให้บริการได้ออกแบบให้มีโปรแกรมหลักคอยให้บริการแก่ผู้ใช้ เรียกว่า APSRV.COM ซึ่งคอยวนตรวจสอบการรับข้อมูลผ่านทางส่วนควบคุมการรับส่งทั้งระดับกายภาพ และระดับเชื่อมโยงข้อมูล เมื่อได้รับข้อมูลเรียบร้อยแล้วจะตรวจสอบคำสั่ง แล้วเรียกฟังก์ชันให้บริการตามคำสั่งนั้น ซึ่งฟังก์ชันทางส่วนผู้ให้บริการจะประกอบไปด้วย ฟังก์ชันให้บริการพิมพ์แฟ้มแบบเก็บพัก และฟังก์ชันให้บริการแฟ้ม

ต่อไปใคร่ขอกล่าวถึงการออกแบบในรายละเอียดเกี่ยวกับรูปแบบข้อมูล การแสดงผล ขั้นตอนและหลักการทำงานของการประยุกต์ใช้งานแต่ละแบบ

1) การออกแบบวิธีการใช้จานบันทึกของเครื่องบริการร่วมกัน

1.1) ส่วนผู้ใช้ ในการใช้จานบันทึกของเครื่องบริการที่เครื่องผู้ใช้ จะใช้หลักการจำลองหน่วยขับจานบันทึก หรือหน่วยขับจานบันทึกเสมือนดังที่ได้กล่าวแล้วในบทที่ 2 ซึ่งผู้ใช้สามารถมองเห็นหน่วยขับจานบันทึกของเครื่องบริการเป็นเสมือนหน่วยขับจานบันทึกหนึ่งของตนเอง ในการจำลองหน่วยขับจานบันทึกสำหรับการวิจัยนี้จะใช้วิธีการของดอสคือ โปรแกรมขับอุปกรณ์ ซึ่งเป็นวิธีที่ดอสสนับสนุนให้สามารถติดต่อกับอุปกรณ์ได้โดยตรง โดยโปรแกรมขับจานบันทึกเสมือนนี้จะถูกเพิ่มเข้าไปในลูกโซ่โปรแกรมขับอุปกรณ์ เมื่อผู้ใช้เรียกใช้แฟ้ม หรือสารบบของหน่วยขับจานบันทึกเสมือนผ่านทางโปรแกรมประยุกต์ภายใต้ดอส เช่น CU Writer, Lotus 1-2-3 หรือใช้คำสั่งของดอส เช่น COPY, DIR เป็นต้น โปรแกรมเหล่านี้จะปฏิบัติงานตามคำสั่งโดยเรียกใช้ฟังก์ชันของตัวขัดจังหวะ 21H ที่อยู่ใน MSDOS.SYS อย่างต่อเนื่อง ในระหว่างการเรียกใช้นี้จะมีการค้นหาแฟ้มจากโครงสร้างสารบบ (Directory Structure) และตารางจัดสรรแฟ้ม (File Allocation Table - FAT) เพื่อเปลี่ยนจากระเบียนเป็นเซกเตอร์ตรรกะ และ มีการเลือกโปรแกรมขับอุปกรณ์สำหรับหน่วยขับจานนั้นจากในลูกโซ่ เพื่อเปลี่ยนจากเซกเตอร์ตรรกะให้เป็นเซกเตอร์กายภาพ ซึ่งถ้าไม่ใช่หน่วยขับจานบันทึกเสมือนก็จะเรียกใช้โปรแกรมรอมไบออสในการจัดการจานบันทึกตามปกติ แต่ถ้าใช่โปรแกรมขับจานบันทึกเสมือนก็จะทำการจำลองจานบันทึกของเครื่องบริการตามวิธีการของโปรแกรมขับอุปกรณ์คือ จัดเตรียมส่วนหัวอุปกรณ์ จัดเตรียมเนื้อที่สำหรับเก็บรีเควสเฮดเดอร์ในรูทีนสตราทีจี และจัดเตรียมการทำงานของแต่ละฟังก์ชันในรูทีนขัดจังหวะ ซึ่งจะกล่าวถึงการออกแบบในรายละเอียดต่อไป การจำลองหน่วยขับจานบันทึกโดยใช้โปรแกรมขับอุปกรณ์แสดงได้ดังรูปที่ 3.14



รูปที่ 3.14 แสดงการจำลองหน่วยขับจานบันทึกโดยใช้โปรแกรมขับอุปกรณ์

โปรแกรมจำลองจานบันทึกนี้มีชื่อว่า APLNK.SYS จะถูกพัฒนาด้วยภาษาแอสเซมบลี ซึ่งมีรายละเอียดการออกแบบในแต่ละส่วนดังต่อไปนี้

1.1.1) การออกแบบส่วนหัวอุปกรณ์ ใน 4 ไบต์แรกกำหนดให้มีค่าเป็น -1 หมายถึงเป็นอุปกรณ์สุดท้ายในลูกโซ่ ถ้ามีโปรแกรมขับอุปกรณ์ที่จะติดตั้งใหม่ดอสจะเปลี่ยนค่านี้เป็นแอดเดรสของโปรแกรมนั้น 2 ไบต์ถัดมากำหนดคุณลักษณะอุปกรณ์มีค่าเป็น 2002H หมายถึงเป็นโปรแกรมขับอุปกรณ์ประเภทบล็อก (บิต 15 เป็น 0) และสนับสนุนการอ้างอิงแอดเดรสของเซกเตอร์แบบ 32 บิต (บิต 2 เป็น 1) อีก 4 ไบต์ถัดมาเก็บค่าขจัดของรูทีนสตราทีจี และค่าขจัดของรูทีนขัดจังหวะอย่างละ 2 ไบต์ ส่วนจำนวนอุปกรณ์ที่สามารถใช้ได้กับโปรแกรมขับอุปกรณ์นี้กำหนดให้มีค่าเป็น 1 ตัวอย่างส่วนหัวอุปกรณ์แสดงด้วยภาษาแอสเซมบลีได้ดังนี้

```
header  dd  -1      ; link to next device (assume this to be the last)
        dw  2002h   ; device attribute = block device, 32-bit sector addresses
        dw  strat   ; strategy routine entry point
        dw  intr    ; interrupt routine entry point
blkdev  db  1       ; one block device
```

1.1.2) การออกแบบรูทีนสตราทีจี กำหนดตัวแปรสำหรับเก็บแอดเดรสของรีเควสเฮดเดอร์ และทำการเก็บค่าจากรีจิสเตอร์ ES:BX ไว้ในตัวแปรดังกล่าวทุกครั้งที่ดอสเรียกใช้รูทีนนี้ ตัวอย่างของรูทีนสตราทีจีแสดงด้วยภาษาแอสเซมบลีได้ดังนี้

```
rh_ptr  dd    ?                          ; pointer to request header
strat   proc  far                        ; entry point of strategy routine
        mov   word ptr cs:[rh_ptr],bx    ; keep offset of request header (from BX)
        mov   word ptr cs:[rh_ptr+2],es  ; keep segment of request header (from ES)
        ret
strat   endp
```

1.1.3) การออกแบบรูทีนขัดจังหวะ ในรูทีนขัดจังหวะมีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: เก็บค่ารีจิสเตอร์ทั้งหมดลงในกองซ้อน

ขั้นที่ 2: ตรวจสอบฟังก์ชันที่ได้รับจากไบต์ที่ 3 ของ rh_ptr จะต้องมีค่าไม่เกิน 9 ถ้าเกินจะส่งค่า 3 (unknown command) กลับไปให้ดอสในไบต์ต่ำของเซตสถานะภาพ และเซตบิต 15 ให้เป็น 1 แล้วไปทำขั้นตอนที่ 4

ขั้นที่3 : ทำงานตามฟังก์ชันที่ได้รับ

ขั้นที่ 4: เซตบิต 8 (DONE flag) ของเซตสถานะภาพให้เป็น 1 เพื่อระบุว่ามีการใช้ฟังก์ชันนั้น ๆ และส่งค่ากลับไปให้ดอส

ขั้นที่ 5: นำค่าเดิมของรีจิสเตอร์กลับคืนจากกองซ้อน แล้วจบการทำงาน

ก่อนที่จะกล่าวถึงการออกแบบการทำงานของแต่ละฟังก์ชัน ผู้วิจัยใคร่ขอกล่าวถึงรายละเอียดเพิ่มเติมเกี่ยวกับโครงสร้างข้อมูล หรือ พารามิเตอร์ของฟังก์ชันที่มีการเปลี่ยนแปลงในดอสรุ่น 4.0 ขึ้นไปที่ได้กล่าวค้างไว้ในท้ายบทที่ 2 เนื่องจากต้องนำมาใช้อ้างอิงในการวิจัยต่อ ๆ ไป ซึ่งมีรายละเอียดพอสรุปได้ดังต่อไปนี้

ก) เพื่อให้ดอสทราบว่าเป็นโปรแกรมขับอุปกรณ์ประเภท 32 บิต ต้องทำการเซตบิต 1 ในโครงสร้างคุณลักษณะอุปกรณ์ให้เป็น 1

ข) มีการขยายขนาดที่เก็บหมายเลขเซกเตอร์จาก 2 ไบต์ เป็น 4 ไบต์

ค) เพิ่มข้อมูลในโครงสร้างของรีเควสเฮดเดอร์ โดยดอสจะกำหนดค่าเดิมที่ตำแหน่ง +14H ให้เป็น -1 หรือ FFFFH ข้อมูลที่เพิ่มมีดังนี้

+ 16H	Volume Identifier	(4 bytes)
+ 1AH	32-bit starting-sector number	(4 bytes)

ง) เพิ่มข้อมูลในโครงสร้างไบออสพารามิเตอร์บล็อก โดยดอสจะกำหนดค่าเดิมที่ตำแหน่ง +08H ให้เป็น 0 ข้อมูลที่เพิ่มมีดังนี้

+ 11H	Number of hidden sectors	(4 bytes)
+ 15H	Number of huge sectors (> 32MB)	(4 bytes)

จ) เปลี่ยนวิธีการเรียกใช้ตัวขัดจังหวะ 25H และ 26H โดยต้องกำหนดค่าในรีจิสเตอร์ CX เป็น -1 เพื่อให้ดอสสามารถตรวจสอบได้ว่าเป็นแบบใหม่ และ เก็บค่าแอดเดรสของโครงสร้างพารามิเตอร์ใหม่ไว้ในรีจิสเตอร์ DS:BX โครงสร้างพารามิเตอร์ใหม่เป็นดังนี้

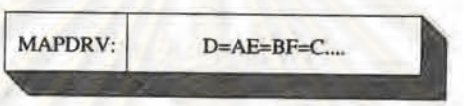
+ 00H	Number of first sector	(4 bytes)
+ 04H	Number of sectors	(1 word)
+ 06H	Pointer to buffer	(4 bytes)

สำหรับฟังก์ชันต่าง ๆ ในขั้นตอนที่ 3 ของรูทีนขัดจังหวะนี้ จะเลือกใช้เฉพาะบางฟังก์ชันเท่านั้น ได้แก่

1.1.3.1) ฟังก์ชัน 00H มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: จัดเตรียมตัวแปรสำหรับเก็บค่าหน่วยขับจานบันทึกเสมือน และหน่วยขับจานบันทึกของเครื่องบริการ โดยอ่านค่าจากพารามิเตอร์ ณ ตำแหน่ง +22H และ +18H (พารามิเตอร์ตัวแรกของโปรแกรม APLNK.SYS จากคำสั่ง device ในแฟ้ม CONFIG.SYS) ตามลำดับ รวมทั้งเก็บลำดับที่ของจานบันทึกเสมือนที่ติดตั้ง (พารามิเตอร์ตัวที่ 2) หมายเลข Socket (พารามิเตอร์ตัวที่ 3) และขนาดของบัฟเฟอร์ที่ใช้เก็บกลุ่มข้อมูลที่รับส่ง (พารามิเตอร์ตัวสุดท้าย)

ขั้นที่ 2: เก็บค่าหน่วยขับจานบันทึกทั้งสองจากขั้นที่ 1 ลงในแฟ้ม APPRINT.CFG เพื่อนำไปใช้ในการส่งแฟ้มไปพิมพ์ที่เครื่องบริการ โดยมีรูปแบบข้อมูลคือ



ส่วนแรกเป็นค่าคงที่มีค่าเป็น 'MAPDRV:' มีไว้เพื่อแสดงว่าเป็นการจำลองจานบันทึก

ส่วนที่เหลือเป็นหน่วยขับจานบันทึกเสมือนที่ติดตั้งแต่ละคู่ ตัวอย่างเช่น D=AE=BF=C โดยตัวแรกเป็นหน่วยขับจานบันทึกเสมือน และตัวที่สองเป็นหน่วยขับจานบันทึกของเครื่องบริการ ซึ่งต้องใช้ลำดับที่ของจานบันทึกในการติดตั้งคือ ในคู่แรกเป็นลำดับ 0 คู่ที่สองเป็นลำดับ 1 และคู่สุดท้ายเป็นลำดับ 2 เป็นต้น

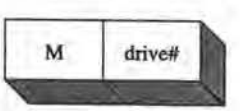
ขั้นที่ 3: จัดเตรียมค่าพารามิเตอร์ของฟังก์ชันที่จะส่งกลับให้ดอส ได้แก่

- ตำแหน่งที่ +13H กำหนดให้จำนวนจานบันทึกที่ควบคุมเท่ากับ 1
- ตำแหน่งที่ +14H กำหนดแอดเดรสแรกที่ว่างต่อจากโปรแกรมนี้โดยใช้ขนาดบัฟเฟอร์จากขั้นที่ 1
- ตำแหน่งที่ +18H กำหนดแอดเดรสของตัวแปร bpb_vec ที่เก็บแอดเดรสของโครงสร้างบีพีบี (bpb:)

ขั้นที่ 4: เซตบิต 8 ของเซตสถานะภาพให้เป็น 1 แล้วจบการทำงาน

1.1.3.2) ฟังก์ชัน 01H ทำหน้าที่ตรวจสอบจานบันทึกเสมือนว่ามีการเปลี่ยนหรือไม่ โดยส่งคำสั่งไปที่เครื่องบริการ แล้วรอรับผลกลับมา มีรูปแบบกลุ่มข้อมูลดังนี้

การส่ง :



ไบต์แรกเป็นรหัสคำสั่ง กำหนดค่าเป็น 'M'

ไบต์ที่สองเป็นหน่วยขับจานบันทึกของเครื่องบริการ ค่า 0 หมายถึง A, 1 หมายถึง B เป็นต้น

การรับ :

ไบต์แรกเป็นแฟล็กแสดงสิทธิการเข้าถึงหน่วยขับจานบันทึกของเครื่องบริการ

'1' หมายถึงไม่มีสิทธิเข้าถึง

ไบต์ที่สองเป็นค่าเดิม

ไบต์ที่สามเป็นแฟล็กแสดงว่ามีการเปลี่ยนจานบันทึกหรือไม่

ขั้นตอนการทำงานเป็นดังนี้

ขั้นที่ 1: ตรวจสอบสิทธิการเข้าถึงระบบโดยเรียกฟังก์ชัน session_interrupt ถ้าได้รับค่าแฟล็กเป็น 'N' หมายถึงไม่มีสิทธิเข้าถึงระบบ จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

ขั้นที่ 2: เตรียมกลุ่มข้อมูลตามรูปแบบข้างต้น แล้วส่งไปที่เครื่องบริการ จากนั้นรอรับผล

ขั้นที่ 3: ตรวจสอบไบต์แรกของกลุ่มข้อมูลที่ได้รับกลับมา ถ้าค่าเป็น '1' หมายความว่าที่เครื่องบริการไม่ได้ติดตั้งจานบันทึกเสมือนไว้ให้ จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

ขั้นที่ 4: นำค่าที่ได้รับในไบต์ที่ 3 ส่งกลับไปให้ดอสทางพารามิเตอร์ที่ตำแหน่ง +14H

ขั้นที่ 5: เซตบิต 8 ของเซตสถานะภาพให้เป็น 1 แล้วจบการทำงาน

1.1.3.3) ฟังก์ชัน 02H ทำหน้าที่สร้างไบออสพารามิเตอร์บล็อกสำหรับจานบันทึกเสมือนที่ติดตั้ง โดยอ่านค่าไบออสพารามิเตอร์บล็อกจากเซกเตอร์ 0 ของจานบันทึกที่เครื่องบริการ เพื่อเก็บไว้สำหรับคำนวณหาตำแหน่งข้อมูลที่แท้จริงต่อไป รูปแบบกลุ่มข้อมูลจะใช้ตามแบบฟังก์ชัน 04H ที่จะกล่าวต่อไป

ขั้นตอนการทำงานเป็นดังนี้

ขั้นที่ 1: ตรวจสอบสิทธิการเข้าถึงระบบโดยเรียกฟังก์ชัน session_interrupt ถ้าได้รับค่าแฟล็กเป็น 'N' หมายถึงไม่มีสิทธิเข้าถึงระบบ จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

ขั้นที่ 2: เก็บแอดเดรสของตัวแปร bpb ไว้ในพารามิเตอร์ตำแหน่งที่ +18H และเตรียมกลุ่มข้อมูลสำหรับอ่านจานบันทึกที่เซกเตอร์ 0 แล้วส่งไปที่เครื่องบริการ จากนั้นรอรับผล

ขั้นที่ 3: นำค่าไบออสพารามิเตอร์บล็อกที่ได้รับไปเก็บไว้ในตัวแปร bpb

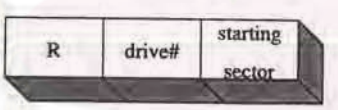
ขั้นที่ 4: เซตบิต 8 ของเซตสถานะภาพให้เป็น 1 แล้วจบการทำงาน

ตัวอย่างข้อมูลในตัวแปร bpb เป็นดังนี้

```
bpb_vec     dw    offset  bpb
bpb:                              ;Sample Values, Replaced by Getboot Procedure
seclen      dw    512             ;Bytes/Cluster
clulen      db    1               ;Sectors/Cluster
numres      dw    1               ;Reserved Sectors
numfat      db    2               ;Number of FATs
numdir      dw    0e0h            ;Number of Directory Entries
numsec      dw    960h            ;Number of Sectors
medtyp      db    0f9h            ;Media Type
fatlen      dw    7               ;Sectors/FAT
secspur     dw    ?               ;Sectors per Track
numrwhd     dw    ?               ;Number of Read/Write Heads
hidsec      dd    ?               ;Number of Hidden Sectors
hugesec     dd    ?               ;Number of Sectors in Volume  ( > 32MB)
```

1.1.3.4) ฟังก์ชัน 04H ทำหน้าที่อ่านข้อมูลจานบันทึกเสมือนจากเครื่องบริการตามเซกเตอร์ที่กำหนดครั้งละ 1 เซกเตอร์ มีรูปแบบกลุ่มข้อมูลดังนี้

การส่ง :

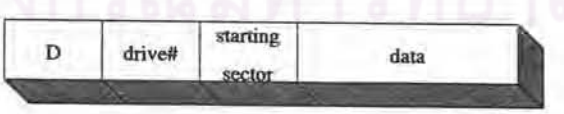


ไบต์แรกเป็นรหัสคำสั่ง กำหนดค่าเป็น 'R'

ไบต์ที่สองเป็นหน่วยขับจานบันทึกของเครื่องบริการ

ไบต์ที่ 3-6 เป็นหมายเลขเซกเตอร์เริ่มต้นที่ต้องการอ่าน

การรับ :



ไบต์แรกเป็นแฟล็กแสดงสิทธิการเข้าถึงหน่วยขับจานบันทึกของเครื่องบริการ หรือ เป็น 'D' หมายถึงเป็นข้อมูลที่อ่านได้

ไบต์ที่ 2-6 เป็นค่าเดิม

ตั้งแต่ไบต์ที่เจ็ดเป็นข้อมูลที่อ่านได้

ขั้นตอนการทำงานเป็นดังนี้

ขั้นที่ 1: ตรวจสอบสิทธิการเข้าถึงระบบโดยเรียกฟังก์ชัน session_interrupt ถ้าได้รับค่าแฟล็กเป็น 'N' หมายถึงไม่มีสิทธิเข้าถึงระบบ จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

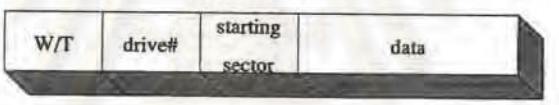
ขั้นที่ 2: จัดเก็บค่าต่าง ๆ จากพารามิเตอร์ลงในตัวแปร เช่น แอดเดรสของบัฟเฟอร์สำหรับเก็บข้อมูลที่อ่านได้ (+14H) จำนวนเซกเตอร์ที่อ่าน (+18H) และหมายเลขเซกเตอร์เริ่มต้น (+26H) เพื่อให้สามารถสนับสนุนจานบันทึกขนาดมากกว่า 32 เมกกะไบต์ ดังนั้นในการคำนวณเกี่ยวกับหมายเลขเซกเตอร์จะต้องใช้รีจิสเตอร์ขนาด 4 ไบต์แทนขนาด 2 ไบต์ เช่น EAX, ECX, EBX, EDX เป็นต้น

ขั้นที่ 3: จัดเตรียมกลุ่มข้อมูลตามรูปแบบข้างต้น แล้วส่งไปที่เครื่องบริการ โดยทยอยส่งครั้งละ 1 เซกเตอร์จนครบตามจำนวนเซกเตอร์ที่ต้องการอ่าน จากนั้นรอรับผล

ขั้นที่ 4: เก็บข้อมูลที่ได้รับไว้ในบัฟเฟอร์ ถ้าเป็นเซกเตอร์ 0 ต้องเก็บค่าบีพีบีลงในตัวแปร bpb ด้วย

ขั้นที่ 5: เซตบิต 8 ของเซตสถานะภาพให้เป็น 1 แล้วจบการทำงาน

1.1.3.5) ฟังก์ชัน 08H และ 09H ทำหน้าที่บันทึกข้อมูลลงในจานบันทึกของเครื่องบริการตามเซกเตอร์ที่กำหนดครั้งละ 1 เซกเตอร์ มีรูปแบบกลุ่มข้อมูลที่ส่งดังนี้



ไบต์แรกเป็นรหัสคำสั่ง กำหนดค่าเป็น 'W' เมื่อต้องการบันทึกข้อมูล และกำหนดค่าเป็น 'T' เมื่อต้องการแจ้งให้ส่วนบริการทราบว่าสิ้นสุดการบันทึกข้อมูลแล้ว ถ้าค่าเป็น 'E' แสดงว่ามีข้อผิดพลาดในการบันทึก

ไบต์ที่สองเป็นหน่วยขับจานบันทึกของเครื่องบริการ

ไบต์ที่ 3-6 เป็นหมายเลขเซกเตอร์เริ่มต้นที่ต้องการบันทึก

ตั้งแต่ไบต์ที่ 7 เป็นข้อมูลที่ต้องการบันทึก

ขั้นตอนการทำงานเป็นดังนี้

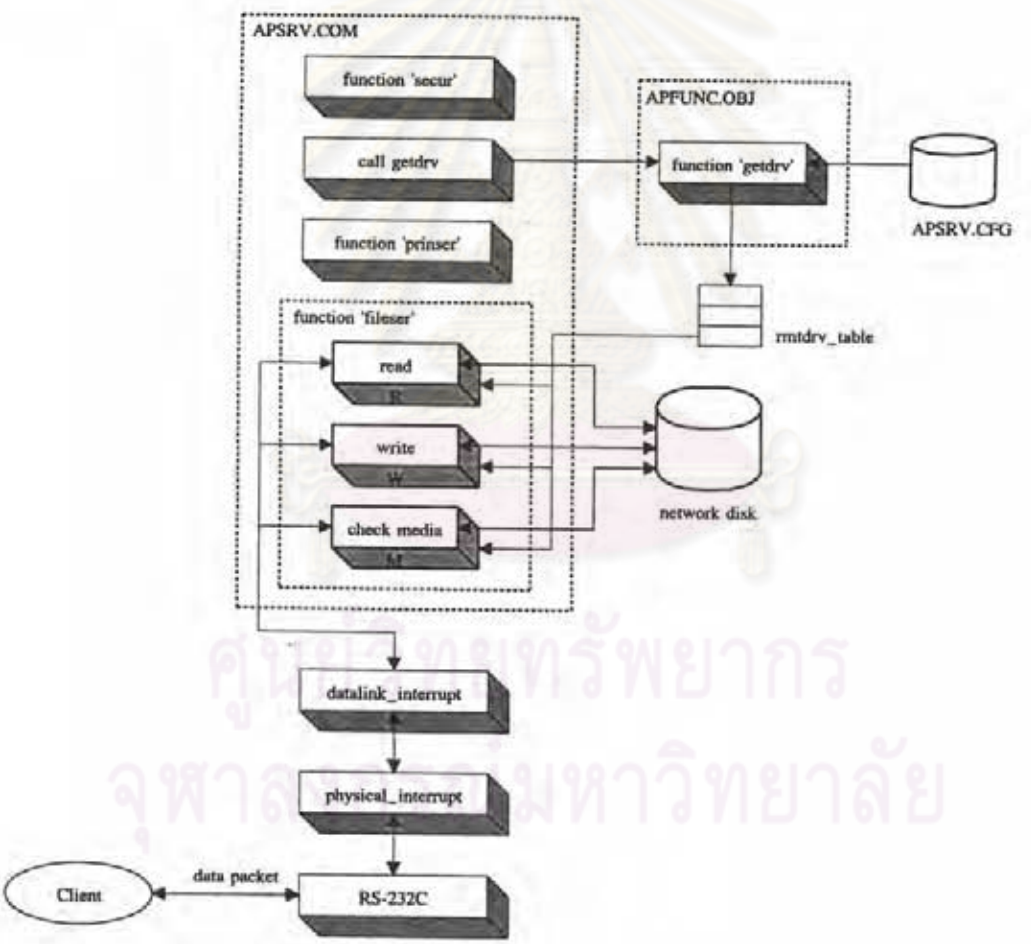
ขั้นที่ 1: ตรวจสอบสิทธิการเข้าถึงระบบโดยเรียกฟังก์ชัน session_interrupt ถ้าได้รับค่าแฟล็กเป็น 'N' หมายถึงไม่มีสิทธิเข้าถึงระบบ หรือ แฟล็กที่ได้รับจากค่า wrtallow เป็น 1 หมายถึงไม่มีสิทธิบันทึกแฟ้ม จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

ขั้นที่ 2: จัดเก็บค่าต่าง ๆ จากพารามิเตอร์ลงในตัวแปร เช่น แอดเดรสของบัฟเฟอร์สำหรับเก็บข้อมูลที่จะบันทึก (+14H) จำนวนเซกเตอร์ที่บันทึก (+18H) และ หมายเลขเซกเตอร์เริ่มต้น (+26H)

ขั้นที่ 3: โอนข้อมูลจากบัฟเฟอร์ไปไว้ในกลุ่มข้อมูลที่จะส่ง โดยจัดเตรียมกลุ่มข้อมูลตามรูปแบบข้างต้น แล้วทยอยส่งไปที่เครื่องบริการครั้งละ 1 เซกเตอร์จนครบตามจำนวนเซกเตอร์ที่ต้องการบันทึก ซึ่งในการ

ส่งแต่ละครั้งจะคอยรับ acknowledgement ด้วย ถ้าไบต์แรกเป็น 'E' แสดงว่ามีข้อผิดพลาดให้นำค่าใน ไบต์ที่ 3 และ 4 ส่งคืนให้คอส เมื่อสิ้นสุดการบันทึกหรือมีข้อผิดพลาดจะส่งคำสั่ง 'T' อีกครั้ง
ขั้นที่ 4: เซตบิต 8 ของเซตสถานะภาพให้เป็น 1 แล้วจบการทำงาน

1.2) ส่วนผู้ให้บริการ ในส่วนนี้จะมีฟังก์ชันคอยให้บริการแฟ้มข้อมูล เรียกว่า fileser ซึ่งเป็นฟังก์ชันหนึ่งในโปรแกรมควบคุมการให้บริการคือ APSRV.COM เมื่อผู้ใช้ส่งคำสั่งเรียกใช้แฟ้มมาที่เครื่องบริการ โปรแกรม APSRV.COM จะทำการตรวจสอบคำสั่งที่ได้รับ เมื่อพบว่าเป็นคำสั่งเรียกใช้แฟ้มก็จะเรียกฟังก์ชัน fileser ขึ้นมาทำงาน แล้วส่งข้อมูลกลับไปให้ผู้ใช้ตามที่ต้องการ โครงสร้างของฟังก์ชันให้บริการแฟ้มแสดงได้ดังรูปที่ 3.15



รูปที่ 3.15 แสดงโครงสร้างของฟังก์ชันให้บริการแฟ้มในส่วนผู้ให้บริการ

จากรูปที่ 3.15 โปรแกรม APSRV.COM จะประกอบไปด้วยฟังก์ชันที่ทำหน้าที่ให้บริการตรวจสอบสิทธิการเข้าถึงระบบ (ซึ่งได้กล่าวไปแล้วในส่วนควบคุมความมั่นคง) ฟังก์ชันให้บริการพิมพ์แฟ้มแบบเก็บพัก และ ฟังก์ชันให้บริการแฟ้มข้อมูล ลักษณะของโปรแกรมสามารถเป็นได้ทั้งแบบ dedicated server และ non-dedicated server แต่ถ้าเป็นแบบ non-dedicated จะไม่สามารถเรียกใช้ฟังก์ชันให้บริการพิมพ์แฟ้มได้ ในการให้บริการแฟ้มสามารถเลือกหน่วยขับจานบันทึกที่ต้องการจัดสรรให้แก่ผู้ใช้ในข่ายงานได้ โดยกำหนดไว้ในแฟ้มข้อความแบบเรียงลำดับชื่อ APSRV.CFG โปรแกรมจะเรียกฟังก์ชัน getdrv ในโปรแกรม APSRV.COM เพื่ออ่านหน่วยขับจานบันทึกที่ถูกจัดสรรจากแฟ้มดังกล่าว แล้วนำมาเก็บไว้ในแถวลำดับเรียกว่า rmtdrv_table จำนวน 26 สมาชิก (A-Z) โดยกำหนดค่าให้เป็น 1 สำหรับหน่วยขับจานบันทึกที่ถูกจัดสรร นอกเหนือจากนั้นกำหนดค่าเป็น 0 เพื่อให้ฟังก์ชันให้บริการแฟ้มนำไปใช้ตรวจสอบได้ต่อไปตัวอย่างเช่น ถ้าต้องการจัดสรรหน่วยขับจานบันทึก A, B และ C ในตัวแปร rmtdrv_table[0], rmtdrv_table[1] และ rmtdrv_table[2] จะมีค่าเป็น 1 ตามลำดับเป็นต้น ส่วนโครงสร้างของฟังก์ชันให้บริการแฟ้มจะประกอบไปด้วย 3 ฟังก์ชันหลัก คือ ฟังก์ชันตรวจสอบสื่อบันทึก ฟังก์ชันอ่านข้อมูล และ ฟังก์ชันบันทึกข้อมูล เมื่อถูกเรียกใช้โดยโปรแกรม APSRV.COM จะตรวจสอบคำสั่งที่ได้รับ ถ้าค่าเป็น 'R' ทำการเรียกใช้ฟังก์ชันอ่านข้อมูล ถ้าค่าเป็น 'W' ทำการเรียกใช้ฟังก์ชันบันทึกข้อมูล และ ถ้าค่าเป็น 'M' ทำการเรียกใช้ฟังก์ชันตรวจสอบสื่อบันทึก ซึ่งจะได้กล่าวถึงขั้นตอนการทำงานของแต่ละฟังก์ชันต่อไป

1.2.1) ฟังก์ชันอ่านข้อมูล ให้ชื่อว่า reasec ทำหน้าที่ให้บริการอ่านข้อมูลจากหน่วยขับจานบันทึกของเครื่องบริการครั้งละ 1 เซกเตอร์ มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้หน่วยขับจานบันทึกจาก rmtdrv_table ถ้าแฟล็กที่ได้รับมีค่าเป็น '1' หมายความว่าไม่มีสิทธิใช้หน่วยขับจานบันทึกนั้น ให้จบการทำงาน

ขั้นที่ 2: ตรวจสอบหมายเลขเซกเตอร์ที่จะอ่าน ถ้าเป็นเซกเตอร์ 0 ให้หาขนาดของเซกเตอร์โดยเรียกตัวขัดจังหวะ 21H ฟังก์ชัน 36H แล้วเก็บค่าไว้ในแถวลำดับ seclen_table (26 สมาชิก) ตามหน่วยขับจานบันทึกนั้น เพื่อใช้ในการคำนวณหาความยาวกลุ่มข้อมูลที่จะส่งกลับไปให้ผู้ใช้

ขั้นที่ 3: แสดงหน่วยขับจานบันทึก และหมายเลขเซกเตอร์ให้ปรากฏบนจอภาพ เพื่อให้ทราบว่ากำลังอ่านเซกเตอร์ที่เท่าไร จากหน่วยขับจานบันทึกไหน

ขั้นที 4: อ่านข้อมูลโดยเรียกตัวขัดจังหวะ 25H ดังตัวอย่างต่อไปนี้

```
new_struc:                  ;new data block structure for disk > 32MB
strsec   dd   ?             ;starting sector no.
numsec   dw   ?             ;number of sectors
dtaoff   dw   ?             ;offset of buffer to receive data
dtaseg   dw   ?             ;segment of buffer to receive data
```

```
mov   al,drive                        ;specify drive no.
mov   cx,-1                           ;tell DOS to use new data block structure
mov   edx,strsec                      ;ignored if cx = -1
mov   numsec,1                        ;specify number of sectors
mov   dtaoff,offset packet_buffer[6]  ;load offset of buffer to receive data
mov   dtaseg,ds                       ;load  segment of buffer to receive data
lea   bx,new_struc                    ;ds:bx point to address of the new data block
int   25h
```

ขั้นที่ 5: ส่งข้อมูลในบัฟเฟอร์ที่อ่านได้กลับไปให้ผู้ใช้ แล้วจบการทำงาน

1.2.2) ฟังก์ชันบันทึกข้อมูล ให้ชื่อว่า wrisec ทำหน้าที่ให้บริการบันทึกข้อมูลลงในหน่วยขับจานบันทึกของเครื่องบริการครั้งละ 1 เซกเตอร์ มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้หน่วยขับจานบันทึกจาก rmtdrv_table ถ้าแฟล็กที่ได้รับมีค่าเป็น '1' หมายความว่าไม่มีสิทธิใช้หน่วยขับจานบันทึกนั้น ให้จบการทำงาน

ขั้นที่ 2: กำหนดค่าแฟล็ก -1 ไว้ในแถวลำดับ drive_table ตามหมายเลขหน่วยขับจานบันทึก และหมายเลข Socket เพื่อแสดงว่าหน่วยขับจานบันทึกนั้นถูกเปลี่ยนแปลงข้อมูล ให้ทำการสร้างค่าต่าง ๆ เกี่ยวกับหน่วยขับจานบันทึกนั้นใหม่ เช่น ไบออสพารามิเตอร์บล็อก โครงสร้างสารบบ และตารางจัดสรรแฟ้ม เป็นต้น

ขั้นที่ 3: แสดงหน่วยขับจานบันทึก และหมายเลขเซกเตอร์ให้ปรากฏบนจอภาพ เพื่อให้ทราบว่ากำลังบันทึกเซกเตอร์ที่เท่าไร ลงในหน่วยขับจานบันทึกไหน

ขั้นที่ 4: บันทึกข้อมูลโดยเรียกตัวขัดจังหวะ 26H ซึ่งกำหนดพารามิเตอร์ทำนองเดียวกับตัวขัดจังหวะ 25H แล้วส่ง acknowledgement เป็นค่าเดิมกลับไปให้ผู้ใช้ แต่ถ้ามีข้อผิดพลาดส่งค่า 'E' ที่ไบต์แรก

ขั้นที่ 5: คอยรับคำสั่งบันทึกเซกเตอร์ต่อไปจากผู้ใช้ แล้วทำขั้นที่ 3-5 จนกว่าจะได้รับคำสั่ง 'T' จึงจะจบการทำงาน

1.2.3) ฟังก์ชันตรวจสอบสื่อบันทึก ให้ชื่อว่า ckmedia ทำหน้าที่ตรวจสอบว่ามีการเปลี่ยนจานบันทึกหรือไม่ เพื่อนำไปใช้ในการตัดสินใจเกี่ยวกับการสร้างค่าต่าง ๆ ในหน่วยความจำของหน่วยขับจานบันทึกนั้น มีขั้นตอนการทำงานดังต่อไปนี้

ขั้นที่ 1: ตรวจสอบสิทธิการใช้หน่วยขับจานบันทึกจาก rmtdrv_table ถ้าแฟล็กที่ได้รับมีค่าเป็น '1' หมายความว่าไม่มีสิทธิใช้หน่วยขับจานบันทึกนั้น ให้จบการทำงาน

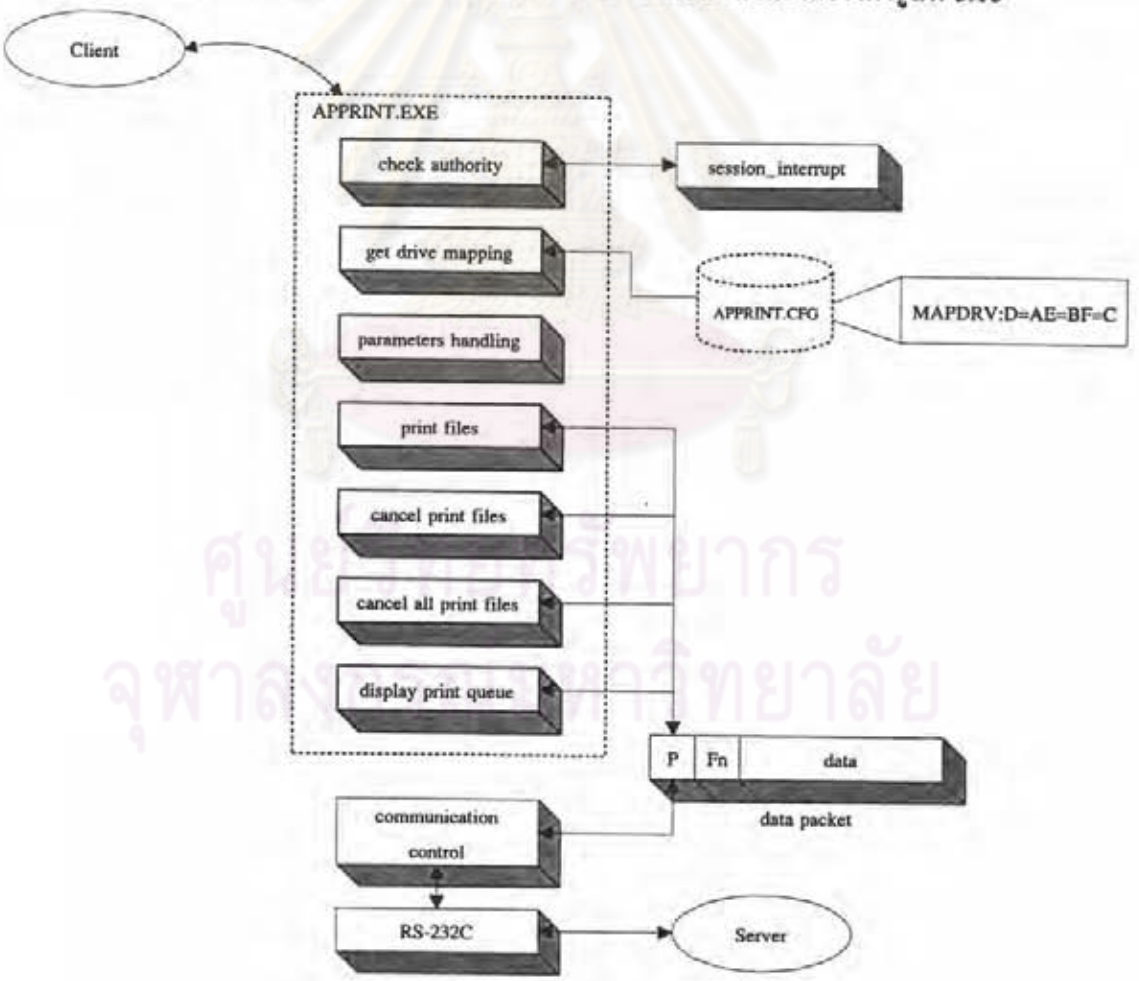
ขั้นที่ 2: อ่านค่าแฟล็กแสดงการเปลี่ยนจานบันทึกจากแถวลำดับ drive_table แล้วล้างค่าแฟล็กเดิมให้เป็น 1

ขั้นที่ 3: ส่งแฟล็กที่ได้กลับไปยังเครื่องผู้ใช้ แล้วจบการทำงาน

2) การออกแบบวิธีการใช้เครื่องพิมพ์ร่วมกันที่เครื่องบริการ

ดังที่ได้กล่าวไว้ท้ายบทที่ 2 แล้วว่า ในการใช้เครื่องพิมพ์ของเครื่องบริการร่วมกันนั้นสามารถทำได้ 2 วิธี คือ การเปลี่ยนทิศทางการพิมพ์ หรือ สั่งพิมพ์ด้วยโปรแกรมอรรถประโยชน์สำหรับการพิมพ์ที่ถูกสร้างขึ้นเป็นพิเศษ ซึ่งในการวิจัยนี้จะเลือกใช้วิธีแบบหลังเนื่องจากเป็นวิธีที่ง่ายต่อการพัฒนา ไม่ซับซ้อน ทำให้สามารถควบคุมความถูกต้องของโปรแกรมได้มากกว่าวิธีแรก ซึ่งต้องใช้วิธีฝังโปรแกรมในหน่วยความจำเพื่อคอยดักการเรียกใช้ตัวขัดจังหวะสำหรับการพิมพ์แล้วเปลี่ยนทิศทางการพิมพ์ นอกจากนั้นวิธีแรกจำเป็นต้องเพิ่มขั้นตอนการตรวจสอบการสิ้นสุดการส่งแฟ้มที่จะพิมพ์ ซึ่งถ้าใช้วิธีแรกสามารถโอนการควบคุมนี้ให้กับดอสโดยใช้คำสั่ง COPY ได้ การออกแบบโปรแกรมจะแบ่งออกเป็น 2 ส่วนเช่นเดียวกับการจำลองจานบันทึก ได้แก่

2.1) การออกแบบโปรแกรมส่วนผู้ใช้ โปรแกรมส่วนนี้เป็นโปรแกรมอรรถประโยชน์สำหรับสั่งพิมพ์แฟ้ม หรือยกเลิกการพิมพ์แฟ้มไปที่เครื่องบริการ ให้ชื่อว่า APPRINT.EXE จะถูกพัฒนาด้วยภาษาซี โดยเลียนแบบการทำงานคล้ายคำสั่ง PRINT ของดอส มีโครงสร้างดังรูปที่ 3.16



รูปที่ 3.16 แสดงโครงสร้างโปรแกรมอรรถประโยชน์การพิมพ์

จากรูปที่ 3.16 เมื่อผู้ใช้เรียกโปรแกรม APPRINT.EXE โปรแกรมจะตรวจสอบสิทธิการเข้าถึงระบบก่อนโดยเรียกใช้ฟังก์ชัน session_interrupt ถ้าไม่มีสิทธิเข้าถึงจะแสดงข้อความบนจอภาพแล้วจบการทำงาน แต่ถ้ามีสิทธิเข้าถึง ทำการจัดเตรียมหน่วยขับจานบันทึกเสมือน และ หน่วยขับจานบันทึกของเครื่องบริการ โดยอ่านจากแฟ้ม APPRINT.CFG เพื่อนำมาใช้สำหรับส่งแฟ้มไปพิมพ์ที่เครื่องบริการ จากนั้นจัดเตรียมค่าพารามิเตอร์ต่าง ๆ ที่ผู้ใช้ป้อนเข้ามาขณะเรียกโปรแกรมนี้ ให้เป็นไปตามรูปแบบที่ต้องการ หมายความว่าผู้ใช้สามารถป้อนพารามิเตอร์สลับที่กันได้ ใช้ตัวอักษรภาษาอังกฤษใหญ่หรือเล็กก็ได้ และอ้างชื่อแฟ้มแบบ wildcard ได้ นอกจากนี้โปรแกรมจะทำการเติมหน่วยขับจานบันทึกให้ใน pathname กรณีที่ผู้ใช้ไม่ได้ระบุมาด้วย เมื่อจัดเตรียมพารามิเตอร์เสร็จเรียบร้อยแล้วโปรแกรมจะทำการตรวจสอบรหัสฟังก์ชันจากพารามิเตอร์ แล้วเรียกฟังก์ชันนั้นขึ้นมาปฏิบัติงาน ในการส่งข้อมูลไปที่เครื่องบริการ กำหนดให้รูปแบบกลุ่มข้อมูลเป็นดังนี้

ไบต์แรกเป็นรหัสโปรแกรมกำหนดค่าเป็น 'P'

ไบต์ที่ 2 เป็นรหัสคำสั่งกำหนดค่าเป็น

'0' หมายถึง แจ้งไปที่เครื่องบริการว่าต้องการส่งแฟ้มไปพิมพ์ ทางเครื่องบริการจะจัดสรรเลขลำดับที่ว่างในตารางควบคุมแฟ้มพิมพ์กลับมาให้

'1' หมายถึง สั่งพิมพ์แฟ้มไปที่เครื่องบริการครั้งละ 1 แฟ้ม

'2' หมายถึง สั่งยกเลิกการพิมพ์แฟ้มที่ระบุครั้งละ 1 แฟ้ม

'3' หมายถึง สั่งยกเลิกการพิมพ์แฟ้มทั้งหมดของผู้ใช้คนนั้น

'4' หมายถึง ขอแสดงแฟ้มที่อยู่ในแถวคอยการพิมพ์ทั้งหมด ปกติแล้วถ้าผู้ใช้ไม่ป้อนพารามิเตอร์เลย โปรแกรมจะกำหนดให้เป็นฟังก์ชันนี้เสมอ

ตั้งแต่ไบต์ที่ 3 เป็นข้อมูลซึ่งขึ้นอยู่กับแต่ละฟังก์ชันว่าต้องการส่งอะไรบ้าง

เมื่อได้ทราบถึงรูปแบบกลุ่มข้อมูลแล้ว ต่อไปใคร่ขอกล่าวถึงรายละเอียดขั้นตอนการทำงานของแต่ละฟังก์ชันดังต่อไปนี้

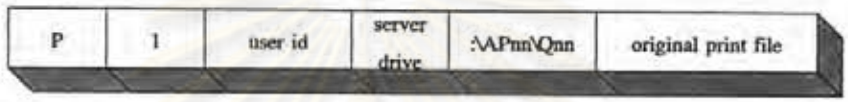
2.1.1) ฟังก์ชันพิมพ์แฟ้มข้อมูล (prt_fil) มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: ขอพิมพ์แฟ้มโดยส่งคำสั่ง '0' ไปที่เครื่องบริการ แล้วคอยรับผล

ขั้นที่ 2: ตรวจสอบไบต์ที่ 2 ที่ได้รับกลับมา ถ้าค่าเป็น '*' หมายความว่า แถวคอยการพิมพ์เต็มไม่สามารถส่งพิมพ์แฟ้มได้ จะแสดงข้อความบนจอภาพ แล้วจบการทำงาน

ขั้นที่ 3: นำเลขลำดับที่แฟ้มที่ได้รับไปเก็บไว้ในตัวแปร ซึ่งกำหนดรูปแบบเป็น 'Qnn' โดย nn เป็นเลขลำดับที่แฟ้มดังกล่าว จากนั้นส่งแฟ้มที่ต้องการพิมพ์ไปที่เครื่องบริการด้วยคำสั่ง COPY ของดอส โดยใช้คำสั่ง system ในภาษาซี และเก็บข้อความที่เกิดขึ้นจากคำสั่งไว้ในแฟ้มข้อความชื่อ 'APPRINT.MSG' ซึ่งในการคัดลอกแฟ้มต้องระบุแฟ้มปลายทาง (ต่อไปจะเรียกว่า 'แฟ้มสำเนา') ที่เครื่องบริการผ่านทางจานบันทึกเสมือน เพื่อไม่ต้องเขียนขั้นตอนการส่งแฟ้มขึ้นเอง โดยใช้คำสั่งดอสและโปรแกรมจำลองจานบันทึกที่มีอยู่แล้วเข้ามาช่วย ชื่อแฟ้มสำเนานี้จะกำหนดให้มีรูปแบบคือ 'drive:\APnn\Qnn' ตัวอย่างเช่น C:\AP02\Q02 หมายถึงเป็นแฟ้มลำดับที่ 2 ในตารางควบคุมแฟ้มพิมพ์ และอยู่ในหน่วยขับจาน C

ขั้นที่ 4: จัดเตรียมกลุ่มข้อมูลที่จะส่งดังนี้



server dive หมายถึง หน่วยขับจานบันทึกของเครื่องบริการ เป็นไบต์แรกของชื่อแฟ้มสำเนา

original print file หมายถึง แฟ้มที่ผู้ใช้สั่งพิมพ์ หรือยกเลิก

ขั้นที่ 5: ส่งข้อมูล แล้วรอรับผล

ขั้นที่ 6: แสดงรายการแฟ้มทั้งหมดในแถวคอยการพิมพ์ให้ปรากฏบนจอภาพ แล้วจบการทำงาน

2.1.2) ฟังก์ชันยกเลิกการพิมพ์แฟ้ม (cancel_prtf) มีขั้นตอนการทำงานดังนี้

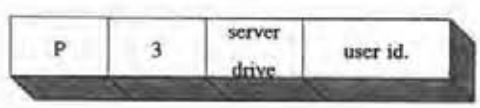
ขั้นที่ 1: จัดเตรียมกลุ่มข้อมูลที่จะส่งดังนี้

ขั้นที่ 2: ส่งข้อมูล แล้วรอรับผล

ขั้นที่ 3: แสดงผลการยกเลิกให้ปรากฏบนจอภาพ แล้วจบการทำงาน

2.1.3) ฟังก์ชันยกเลิกการพิมพ์แฟ้มทั้งหมด (cancel_all) มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: จัดเตรียมกลุ่มข้อมูลที่จะส่งดังนี้



ขั้นที่ 2: ส่งข้อมูล แล้วรอรับผล

ขั้นที่ 3: แสดงผลการยกเลิกให้ปรากฏบนจอภาพ แล้วจบการทำงาน

2.1.4) ฟังก์ชันแสดงรายการแฟ้มในแถวคอย มีขั้นตอนการทำงานดังนี้

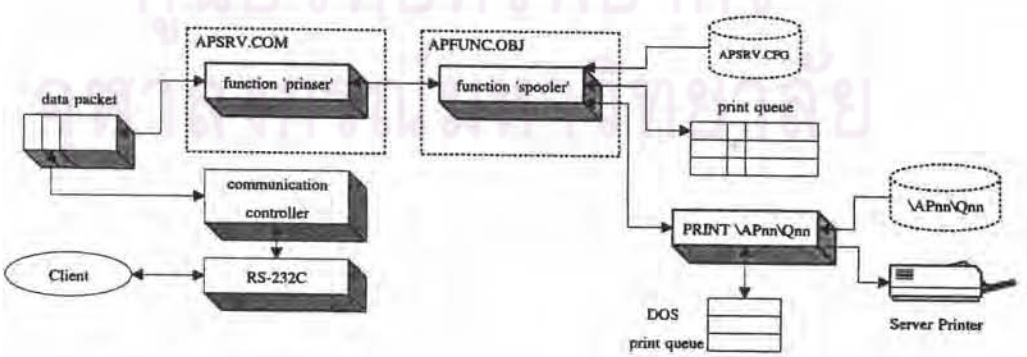
ขั้นที่ 1: จัดเตรียมกลุ่มข้อมูลที่จะส่งดังนี้

ขั้นที่ 2: ส่งข้อมูล แล้วรอรับผล

ขั้นที่ 3: แสดงรายการแฟ้มทั้งหมดในแถวคอยการพิมพ์ให้ปรากฏบนจอภาพ แล้วจบการทำงาน

2.1.5) ฟังก์ชันตรวจสอบ pathname (chk_path) ทำหน้าที่ตรวจสอบว่ามีการระบุ pathname ที่ถูกต้องหรือไม่ มีการสั่งพิมพ์ หรือยกเลิกการพิมพ์หลาย ๆ แฟ้มพร้อมกันหรือไม่ มีการระบุ wildcard หรือไม่ โดยค้นหาจากโครงสร้างสารบบของดอสด้วยคำสั่ง findfirst และ findnext ในภาษาซี จากนั้นทำการตรวจสอบรหัสฟังก์ชัน แล้วเรียกใช้ฟังก์ชันนั้น ถ้าเป็นกรณีส่งพิมพ์หลายแฟ้ม หรือส่งพิมพ์แบบ wildcard โปรแกรมจะทำการเรียกฟังก์ชันโดยส่งครั้งละ 1 แฟ้ม แล้วทยอยส่งจนครบตามต้องการ

2.2) การออกแบบส่วนผู้ให้บริการ ในส่วนนี้จะประกอบไปด้วยฟังก์ชัน prinser ทำหน้าที่เป็นตัวประสานระหว่างโปรแกรม APSRV.COM และ APFUNC.OBJ โดยเรียกใช้ฟังก์ชันพิมพ์แฟ้มแบบเก็บพัก (spooler) ในโปรแกรม APFUNC.OBJ มีโครงสร้างดังรูปที่ 3.17



รูปที่ 3.17 แสดงโครงสร้างส่วนให้บริการพิมพ์แฟ้มแบบเก็บพัก

จากรูปที่ 3.17 เมื่อผู้ใช้ส่งข้อมูลมาที่เครื่องบริการ โปรแกรม APSRV.COM จะตรวจสอบคำสั่งที่ไบต์แรก กรณีพบว่าเป็น 'P' จะเรียกใช้ฟังก์ชัน prinser ซึ่งเรียกฟังก์ชัน spooler อีกต่อหนึ่ง เพื่อให้จัดการทำตามคำสั่งจากกลุ่มข้อมูลที่ได้รับ

ฟังก์ชัน spooler เป็นฟังก์ชันที่ทำหน้าที่ควบคุมการให้บริการพิมพ์แฟ้มแบบเก็บพัก การยกเลิกการพิมพ์แฟ้มตามที่ระบุ หรือยกเลิกทั้งหมด และการแสดงรายการแฟ้มในแถวคอยการพิมพ์ โดยในการพิมพ์แฟ้มแบบเก็บพักนั้นจะอาศัยโปรแกรม PRINT ของดอสซึ่งมีคุณสมบัติในการพิมพ์แฟ้มแบบเก็บพักดังที่ได้กล่าวแล้วในบทที่ 2 การใช้วิธีนี้ทำให้เกิดปัญหาตามมาคือ เมื่อแฟ้มในแถวคอยของดอสถูกพิมพ์เสร็จแล้ว โปรแกรม PRINT จะทำการลบแฟ้มนั้นออกจากแถวคอยทันที ทำให้แฟ้มนั้นซึ่งถูกคัดลอกมาจากผู้ใช้ยังคงค้างอยู่ในจานบันทึก ดังนั้นผู้วิจัยจึงแก้ปัญหานี้โดยการสร้างตารางควบคุมแฟ้มพิมพ์ในหน่วยความจำเรียกว่า prt_que ซึ่งใช้ตัวแปรแบบแถวลำดับมีโครงสร้างข้อมูลดังนี้

	User Id.	Drive	Orignal Print File
0			
1			
.			
31			

เขตแรกเก็บรหัสประจำตัวผู้ใช้ ขนาด 10 ไบต์

เขตที่ 2 เก็บหน่วยขับจานบันทึกของเครื่องบริการที่ส่งมาจากผู้ใช้ ขนาด 1 ไบต์

เขตสุดท้ายเก็บชื่อแฟ้ม หรือ pathname ของผู้ใช้ (original print file)

แต่ละสมาชิกของแถวลำดับจะหมายถึงแฟ้ม 1 แฟ้มของผู้ใช้ และลำดับที่สมาชิกไม่จำเป็นต้องตรงกับลำดับที่ของแฟ้มในแถวคอยของดอส เนื่องจากอาจจะมีการนำลำดับที่ของแฟ้มที่ถูกยกเลิกแล้วมาใช้ใหม่ได้ ในการควบคุมลำดับที่ในตารางกำหนดให้มีตัวแปร 2 ตัว คือ qnum และ q_act ตัวแปร qnum ใช้ควบคุมลำดับที่สำหรับแฟ้มใหม่ที่เพิ่มเข้ามาในแถวคอย ส่วนตัวแปร q_act ใช้แสดงจำนวนแฟ้มที่มีอยู่จริงในตาราง (ไม่รวมรายการที่ยกเลิกแล้ว) ส่วนการกำหนดจำนวนแฟ้มสูงสุดในแถวลำดับนี้ จะอ่านตัวเลขหลังข้อความ 'maxq=' จากแฟ้ม APSRV.CFG ถ้าไม่ได้สร้างไว้จะกำหนดค่าให้เป็น 10 และค่าต้องไม่เกิน 32 ซึ่งกำหนดให้เท่ากับในโปรแกรม PRINT ตารางนี้นอกจากจะใช้สำหรับการลบแฟ้มที่พิมพ์แล้ว ยังใช้ในการอ่านชื่อแฟ้มของผู้ใช้ เพื่อแสดงผลบนจอภาพ และใช้ควบคุมการยกเลิกแฟ้มให้กระทำได้เฉพาะแฟ้มของตนเองเท่านั้น สำหรับขั้นตอนการทำงานของโปรแกรมมีดังนี้

ขั้นที่ 1: ตรวจสอบไบต์แรกของกลุ่มข้อมูล ถ้าค่าไม่เท่ากับ 'P' ให้จบการทำงาน

ขั้นที่ 2: ขั้นตอนนี้จะถูกกระทำกรณีเริ่มต้นทำงานครั้งแรกเท่านั้น ได้แก่

- กำหนดค่า '*' ที่ไบต์แรกของรหัสประจำตัวผู้ใช้ทุกรายการในตารางควบคุมแฟ้มพิมพ์ ซึ่งหมายความว่าเป็นรายการที่ยกเลิกแล้ว จะไม่ถูกนำไปพิมพ์

- สร้างตารางควบคุมแฟ้มพิมพ์ขึ้นใหม่ โดยอ่านรายการจากแฟ้ม APPRINT.MSG ซึ่งเก็บแฟ้มพิมพ์ในแถวคอยของดอส (แฟ้มนี้จะต้องถูกสร้างขึ้นก่อนด้วยคำสั่ง PRINT > APPRINT.MSG) แล้วใช้ลำดับที่ของแถวคอย (จาก Qnn) เป็นลำดับที่ในตาราง และกำหนดรหัสประจำตัวผู้ใช้ของแต่ละรายการให้เป็น 'N/A' หมายถึงหารหัสไม่ได้ (Not Available) กำหนดชื่อแฟ้มผู้ใช้ให้เป็นชื่อแฟ้มสำเนาที่จะถูกพิมพ์ เช่น C:\PRTF.Q01 ขั้นตอนนี้มีไว้สำหรับกรณีที่มีการออกจากโปรแกรม APSRV.COM ขณะที่แฟ้มในแถวคอยยังถูกพิมพ์ไม่หมด แล้วต้องการยกเลิกแฟ้ม ทั้งนี้เนื่องมาจากเมื่อเริ่มเข้าโปรแกรมใหม่อีกครั้ง แอดเดรสของตารางนี้อาจจะไม่ใช่ตำแหน่งเดิมดังนั้นรายการเดิมในตารางจะเปลี่ยนแปลงไป

ขั้นที่ 3: อ่านจำนวนแฟ้มสูงสุดในตารางควบคุมแฟ้มจาก APSRV.CFG

ขั้นที่ 4: ลบแฟ้มที่ถูกพิมพ์เสร็จแล้วออกจากตารางควบคุมแฟ้ม และจานบันทึก โดยค้นหารายการในแถวคอยจากแฟ้ม APPRINT.MSG ถ้าไม่พบแสดงว่าแฟ้มนั้นถูกพิมพ์เสร็จแล้ว ให้ใส่ค่า '*' ที่ไบต์แรกของรหัสประจำตัวผู้ใช้ของรายการนั้นในตาราง และใช้คำสั่ง delete ของดอสในการลบแฟ้มออกจากจานบันทึก โดยเรียกผ่านคำสั่ง system ในภาษาซี

ขั้นที่ 5: ตรวจสอบรหัสฟังก์ชัน

- ถ้ารหัสเป็น '0' ให้เรียกใช้ฟังก์ชันจัดสรรลำดับที่ในตารางควบคุมแฟ้มพิมพ์
- ถ้ารหัสเป็น '1' ให้เรียกใช้ฟังก์ชันพิมพ์แฟ้ม
- ถ้ารหัสเป็น '2' ให้เรียกใช้ฟังก์ชันยกเลิกแฟ้ม
- ถ้ารหัสเป็น '3' ให้เรียกใช้ฟังก์ชันยกเลิกแฟ้มทุกแฟ้ม
- ถ้ารหัสเป็น '4' ให้เรียกใช้ฟังก์ชันแสดงรายการแฟ้มในแถวคอย
- นอกเหนือจากนี้ ให้ส่งรหัสความผิดพลาดกลับไปให้ผู้ใช้

ขั้นที่ 6: จบการทำงาน

ฟังก์ชันจัดสรรลำดับที่ในตารางควบคุมแฟ้มพิมพ์ ทำหน้าที่กำหนดเลขลำดับที่แฟ้มในตารางควบคุมแฟ้มให้แก่ผู้ใช้ ฟังก์ชันนี้จะถูกเรียกใช้ก่อนฟังก์ชันพิมพ์แฟ้มเสมอ มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: ค้นหาลำดับที่ว่างลำดับแรกในตาราง โดยตรวจสอบแฟ้มที่ถูกยกเลิกแล้ว ถ้าพบก็ให้ใช้ลำดับที่ของแฟ้มนั้นเป็นลำดับที่ของแฟ้มใหม่ที่ต้องการสั่งพิมพ์ต่อไป แต่ถ้าไม่พบให้ใช้ลำดับที่ว่างถัดจากลำดับที่ของแฟ้มสุดท้ายแทน

ขั้นที่ 2: ตรวจสอบลำดับที่จัดสรรให้ต้องไม่เกินจำนวนสูงสุดที่กำหนด เฉพาะกรณีไม่มีรายการยกเลิก ถ้าพบว่าเกินจะส่งรหัสความผิดพลาด '*' กลับไปให้ผู้ใช้ แล้วจบการทำงาน

ขั้นที่ 3: ส่งลำดับที่แฟ้มกลับไปให้ผู้ใช้ แล้วจบการทำงาน

ฟังก์ชันพิมพ์แฟ้ม (prt_fil) ทำหน้าที่สั่งพิมพ์แฟ้มสำเนาออกไปที่เครื่องพิมพ์ มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: เพิ่มชื่อแฟ้มผู้ใช้ที่ถูกสั่งพิมพ์เข้าไปในตารางควบคุมแฟ้มพิมพ์ตามรูปแบบที่กำหนด โดยใช้ลำดับที่กำหนดให้จากฟังก์ชัน 0 ดังตัวอย่างภาษาซีต่อไปนี้

```
strcpy(prt_que[qnn_i-1].usr_id,usrid);        /* move user id. */
strcpy(prt_que[qnn_i-1].org_prtf,org_fn); /* move original print file */
prt_que[qnn_i-1].rmt_drv=fn[0];               /* move server drive or remote drive */
```

ขั้นที่ 2: สั่งพิมพ์แฟ้มด้วยคำสั่ง PRINT และให้เก็บข้อความที่เกิดขึ้นไว้ในแฟ้ม APPRINT.MSG

ขั้นที่ 3: ส่งข้อความกลับไปให้ผู้ใช้โดยเรียกฟังก์ชันแสดงรายการแฟ้มในแถวคอย แล้วจบการทำงาน

ฟังก์ชันยกเลิกแฟ้ม (cancel_q) ทำหน้าที่ยกเลิกการพิมพ์แฟ้มครั้งละ 1 แฟ้ม มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: จัดเตรียมหน่วยขับจานบันทึกของแฟ้มที่จะยกเลิก รหัสประจำตัวผู้ใช้ และชื่อแฟ้มของผู้ใช้ลงในตัวแปร แล้วนำไปค้นหาในตารางโดยเลือกเฉพาะรายการที่ไม่ถูกยกเลิก และเป็นรายการของตนเองเท่านั้น

ขั้นที่ 2: ถ้าพบให้สั่งยกเลิกแฟ้มที่ได้รับด้วยคำสั่ง PRINT โดยใช้ตัวเลือก (option) '/c' และให้เก็บข้อความที่เกิดขึ้นไว้ในแฟ้ม APPRINT.MSG แล้วลบแฟ้มนั้นออกจากจานบันทึก

ขั้นที่ 3: ส่งข้อความกลับไปให้ผู้ใช้โดยเรียกฟังก์ชันแสดงรายการแฟ้มในแถวคอย แล้วจบการทำงาน

ฟังก์ชันยกเลิกแฟ้มทั้งหมด (cancel_all) ทำหน้าที่ยกเลิกการพิมพ์แฟ้มทั้งหมดของผู้ใช้ในแถวคอย มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: จัดเตรียมรหัสประจำตัวผู้ใช้

ขั้นที่ 2: ค้นหาในตารางโดยเลือกเฉพาะรายการที่ไม่ถูกยกเลิก และเป็นรายการของตนเองเท่านั้น

ขั้นที่ 3: ถ้าพบให้สั่งยกเลิกแฟ้มด้วยคำสั่ง PRINT โดยใช้ตัวเลือก '/c' และให้เก็บข้อความที่เกิดขึ้นไว้ในแฟ้ม APPRINT.MSG แล้วลบแฟ้มออกจากจานบันทึก กลับไปทำขั้นที่ 2 ใหม่จนกว่าจะครบตามจำนวนแฟ้มสูงสุดที่กำหนด

ขั้นที่ 4: ส่งข้อความกลับไปให้ผู้ใช้โดยเรียกฟังก์ชันแสดงรายการแฟ้มในแถวคอย แล้วจบการทำงาน

ฟังก์ชันแสดงรายการแฟ้มในแถวคอย (dsp_q) ทำหน้าที่จัดเตรียมรูปแบบการแสดงผลรายการแฟ้มในแถวคอยให้แก่ผู้ใช้ มีขั้นตอนการทำงานดังนี้

ขั้นที่ 1: อ่านข้อมูลทั้งหมดจากแฟ้ม APPRINT.MSG ที่ได้รับ แล้วเก็บไว้ในบัฟเฟอร์

ขั้นที่ 2: จัดเตรียมรูปแบบการแสดงผลของแต่ละรายการ ปกติแล้วจะเป็นรูปแบบของคำสั่ง PRINT จึงต้องเพิ่มเติมรหัสประจำตัวผู้ใช้เข้าไปในแต่ละรายการด้วย และทำการเปลี่ยนชื่อแฟ้มในบัฟเฟอร์ให้เป็นชื่อแฟ้มของผู้ใช้ โดยค้นหาจากตาราง เช่น เปลี่ยนจาก C:\PRTF.Q03 เป็น C:\DIR1\FILE1 เป็นต้น รูปแบบการแสดงผลเป็นดังนี้

```
<user id.> drive:\path\file-name-1 is currently being printed
<user id.> drive:\path\file-name-2 is in queue
```

ขั้นที่ 3: ส่งข้อความกลับไปให้ผู้ใช้เพื่อแสดงผลบนจอภาพต่อไป แล้วจบการทำงาน

ตัวอย่างการแสดงผล เป็นดังนี้

```
<ANUCHIT> C:\UAN\APLNK.ASM is currently being printed
<SOMCHAI> C:\WINDOWS\SYSTEM.INI is in queue
<ANUCHIT> D:\CONFIG.SYS is in queue
```