บทที่ 2

แนวคิดและทฤษฎีที่เกี่ยวข้อง

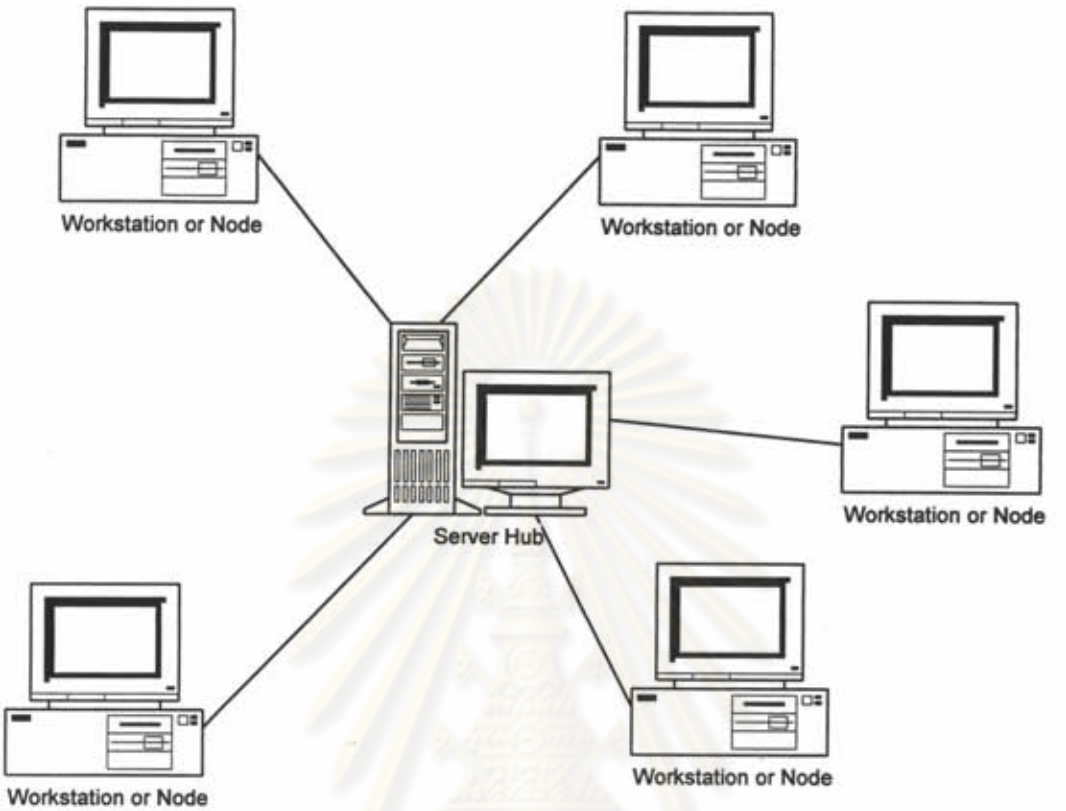
ในเรื่องเกี่ยวกับการสื่อสารข้อมูล หรือ ข่ายงานคอมพิวเตอร์ นั้น มีแนวคิด และ ทฤษฎีต่าง ๆ อยู่มากมาย ซึ่งในบทนี้จะขอกล่าวถึงเฉพาะที่นำมาประยุกต์ใช้ในงานวิจัย เช่น รูปร่างข่ายงานแบบดาว สถาปัตยกรรมข่ายงานตามมาตรฐาน OSI และ ส่วนประกอบของข่ายงานไมโครคอมพิวเตอร์ เพื่อเป็นแนวทางให้ผู้สนใจสามารถเข้าใจและติดตามผลงานวิจัยในบทต่อ ๆ ไปได้ดียิ่งขึ้น

รูปร่างของข่ายงาน (Network Topology)

ในการสร้างข่ายงานขึ้นมานั้น สิ่งแรกที่ต้องพิจารณาก็คือ รูปร่างของข่ายงาน ถ้าเปรียบเทียบกับบ้าน ก็จะหมายถึง รูปทรงภายนอกของบ้านนั่นเอง รูปร่างของข่ายงานมีอยู่หลายแบบ เช่น แบบดาว (Star) แบบวงแหวน (Ring) แบบบัส (Bus) และ แบบต้นไม้ (Tree or Hierachy) ในแต่ละแบบจะมีทั้งข้อดี และ ข้อเสีย การที่จะเลือกรูปร่างแบบใดนั้น ขึ้นอยู่กับขีดความสามารถของระบบหลาย ๆ ด้าน เช่น ความน่าเชื่อถือ (Reliability) ความยุ่งยากซับซ้อนในการติดตั้ง และ ขยายระบบ รวมทั้งค่าใช้จ่าย และ ประสิทธิภาพของระบบ สำหรับการวิจัยนี้จะขอกล่าวถึงเฉพาะรูปร่างแบบดาวเท่านั้น

ข่ายงานแบบดาว (Star Network)

ในการเชื่อมต่อข่ายงานเป็นรูปแบบดาวนั้น จะมีสถานีหนึ่งเป็นสถานีกลาง (Central Node) หรือ เครื่องบริการ (Server) คอยให้บริการสวิทช์ข้อความ (Message Switch) และ ข้อมูลข่าวสารแก่สถานีงาน หรือ ผู้ใช้ (Client) โดยจะรับข้อความ ข่าวสารจากสถานีต้นทาง แล้วจัดการส่งต่อไปที่สถานีปลายทาง เปรียบเสมือนกับชุมสายโทรศัพท์ ที่ทำหน้าที่สวิทช์สัญญาณจากผู้เรียกไปยังผู้รับเป็นต้น ดังรูปที่ 2.1



รูปที่ 2.1 แสดงรูปร่างข่ายงานแบบดาว

ลักษณะการเชื่อมต่อไมโครคอมพิวเตอร์เป็นรูปแบบดาวนี้ สามารถนำไปประยุกต์ใช้ได้ 3 รูปแบบ[1] คือ

1) Single Star การเชื่อมต่อรูปแบบนี้เป็นแบบที่ง่ายที่สุดคือ เป็นข่ายงานเดียว มีลักษณะเป็นแบบรวมศูนย์ โดยมีเครื่องผู้ใช้เชื่อมโยงเข้าหาเครื่องบริการที่จุดเดียว ดังรูปที่ 2.1ก การเชื่อมต่อแบบนี้จะง่ายต่อการพัฒนาและการติดตั้งใช้งาน แต่ถ้ามีจำนวนผู้ใช้มากการจราจรในข่ายงานจะหนาแน่น ทำให้ผู้ใช้ต้องใช้เวลาในการคอยการบริการนานขึ้น

2) Connected Star เป็นการนำรูปแบบ Single Star มาประยุกต์ใช้งาน โดยการแบ่งข่ายงาน Single Star ขนาดใหญ่ (10 เครื่องขึ้นไป) ออกเป็นข่ายงานขนาดเล็ก ๆ และเชื่อมต่อเครื่องบริการของข่ายงานเข้าด้วยกัน การเชื่อมต่อแบบนี้จะกำหนดให้เครื่องบริการของข่ายงานหนึ่งเป็นเครื่องผู้ใช้ของเครื่องบริการของอีกข่ายงานหนึ่งที่ต้องการเชื่อมต่อ ดังรูปที่ 2.1ข เมื่อผู้ใช้ของข่ายงาน A ต้องการขอใช้บริการทรัพยากรจากเครื่องบริการของข่ายงาน B ก็สามารถส่งคำขอผ่านทางเครื่องบริการ

---

[1] Chaiken, Blueprint of a LAN, (USA: M&T Plubishing, Inc., 1989), pp. 107-111

ของข่ายงาน A และเครื่องบริการของข่ายงาน A จะทำหน้าที่ส่งคำขอให้แก่เครื่องบริการ B อีกต่อหนึ่ง นอกจากนี้การเชื่อมต่อแบบ Connected Star ยังมีประโยชน์ในกรณีที่มีการแบ่งกลุ่มเครื่องผู้ใช้อยู่ต่างชั้นของอาคาร การจัดกลุ่มเครื่องผู้ใช้เป็นข่ายงานในแต่ละชั้น แล้วเชื่อมต่อเครื่องบริการเข้าด้วยกันจะช่วยให้การติดตั้งทำได้ง่าย และประหยัดค่าใช้จ่ายของสายสัญญาณ

3) Imploding Star เป็นการเชื่อมต่อโดยตรงระหว่างเครื่องผู้ใช้ 1 เครื่องกับเครื่องบริการหลาย ๆ เครื่อง ดังรูปที่ 2.1ค ซึ่งการเชื่อมต่อลักษณะนี้ผู้ใช้สามารถขอใช้ทรัพยากรได้จากหลาย ๆ ข่ายงาน ในการประยุกต์ใช้งานจะเหมาะสำหรับการประมวลผลแบบขนาน (Parallel Processing) โดยผู้ใช้สามารถควบคุมการส่งงานไปประมวลผลที่เครื่องบริการหลาย ๆ เครื่องพร้อมกันได้

ข่ายงานแบบดาวนี้มีข้อดีหลายประการ คือ

1) ในการติดต่อระหว่าง 2 สถานีจะใช้ระยะทางสั้น คือ ไม่เกิน 2 โยง (Link) ดังนั้นการส่งจะใช้เวลาน้อยด้วย

2) การเพิ่มจำนวนสถานีจะทำได้ง่าย เพราะว่าการเชื่อมต่อมีลักษณะเป็นคู่ ๆ อยู่แล้ว เพียงแค่เพิ่มสายเคเบิล ก็จะสามารถต่อเพิ่มได้ทันที

3) การควบคุมระบบทำได้ง่าย เพราะว่า การให้บริการจะอยู่ที่จุดเดียว จึงทราบถึงข้อความที่ผ่านเข้ามาในระบบ ทราบความผิดพลาดต่าง ๆ ที่เกิดขึ้น ทำให้สามารถวิเคราะห์ทางสถิติ ตลอดจนการกู้ระบบ (Recovery)ได้

4) ถ้าสถานีงานตัวใดตัวหนึ่งเสียหาย จะไม่ทำให้ทั้งระบบเสียหายไปด้วย

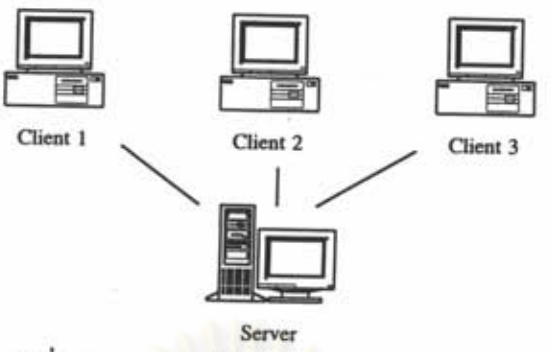
5) การติดตั้งและโยกย้ายสถานีทำได้ง่าย เพราะว่า การที่ทุกสถานีเชื่อมโยงผ่านสถานีกลางที่เดียว การเชื่อมต่อสายสัญญาณย่อมทำได้สดวก รวดเร็ว

อย่างไรก็ตามข่ายงานแบบดาวก็มีข้อเสียเหมือนกันกล่าวคือ

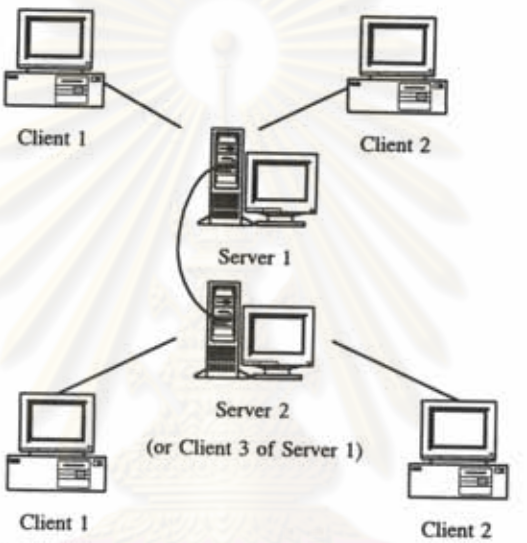
1) ถ้าต่อระยะทางไกล ๆ หรือมีจำนวนสถานีมากจะสิ้นเปลืองสายสัญญาณ นั่นคือค่าใช้จ่ายจะสูงตามไปด้วย

2) ถ้าสถานีกลางเกิดการเสียหาย ระบบจะใช้การไม่ได้ทันที จึงควรมีระบบทนต่อความผิดพร่อง (Fault Tolerant) เพื่อทำหน้าที่แทนสถานีกลาง ระบบก็จะดำเนินต่อไปได้

3) การที่ข้อความต่าง ๆ ต้องผ่านสถานีกลางที่เดียว ย่อมทำให้การรับส่งข้อความเกิดความล่าช้า เพราะว่าต้องคอยกัน และถ้าสถานีกลางต้องทำงานอื่นด้วยก็จะทำให้การรับส่งข้อความใช้เวลานานขึ้นอีก



รูปที่ 2.1ก แสดงรูปร่างข่ายงานแบบ Single Star



รูปที่ 2.1ข แสดงรูปร่างข่ายงานแบบ Connected Star

รูปที่ 2.1ค แสดงรูปร่างข่ายงานแบบ Imploding Star

เมื่อได้ทราบถึงลักษณะรูปร่างของข่ายงานแล้ว ต่อไปจะกล่าวถึงสถาปัตยกรรมข่ายงาน (Network Architecture) เพื่อใช้เป็นแนวทางในการออกแบบและพัฒนาข่ายงานให้มีมาตรฐานเดียวกัน

สถาปัตยกรรมข่ายงาน

ในระบบข่ายงานนั้น ส่วนที่เป็นจุดสำคัญก็คือ การสนับสนุนให้มีความเข้ากันได้ทั้งฮาร์ดแวร์ และ ซอฟต์แวร์ ที่มาจากหลาย ๆ บริษัท และ เพื่อให้สามารถออกแบบซอฟต์แวร์ ให้เป็นอิสระจากฮาร์ดแวร์ในบางระดับชั้น ดังนั้นทางองค์การมาตรฐานระหว่างประเทศ (International Standard Organization)จึงได้กำหนดมาตรฐานข่ายงานแบบเปิดขึ้นมาเรียกว่า OSI (Open System Interconnection) ซึ่งกำหนดมาตรฐานของฮาร์ดแวร์ เช่น สายสัญญาณ หรือ ตัวกลางสื่อสาร กำหนดมาตรฐานของโปรโตคอลสื่อสาร รูปแบบข้อความ เพื่อให้บริษัทผู้ผลิตยึดถือเป็นแบบแผนเดียวกัน

ตามมาตรฐาน OSI ได้แบ่งการสื่อสารออกเป็น 7 ระดับ คือ

1) ระดับกายภาพ (Physical Layer) กำหนดมาตรฐานการส่งข้อมูลในระดับบิตผ่านตัวกลางสื่อสาร เช่น กำหนดแรงดันไฟฟ้าที่ใช้แทนค่าบิต 0 และ 1 กำหนดความเร็วในการส่งแต่ละบิต กำหนดประเภทและคุณสมบัติของตัวเชื่อมต่อ (Connector) หรือ ตัวปรับ (Adaptor) และ กำหนดประเภทของตัวกลางสื่อสาร

2) ระดับเชื่อมโยงข้อมูล (Data Link Layer) กำหนดวิธีการส่งข้อมูลเป็นกรอบ (Frame) หรือ กลุ่ม (Packet) โดยมีตัวบอกจุดเริ่มต้น และ จุดสุดท้ายของกลุ่มข้อมูล นอกจากนี้ยังมีการตรวจสอบความผิดพลาดของกลุ่มข้อมูลด้วย ซึ่งรูปแบบหรือวิธีการเหล่านี้เรียกว่า โปรโตคอล

3) ระดับข่ายงาน (Network Layer) กำหนดทิศทางการส่งกลุ่มข้อมูลไปยังข่ายงานถัดไป เช่น ที่อยู่ และ ลำดับที่ของกลุ่มข้อมูล

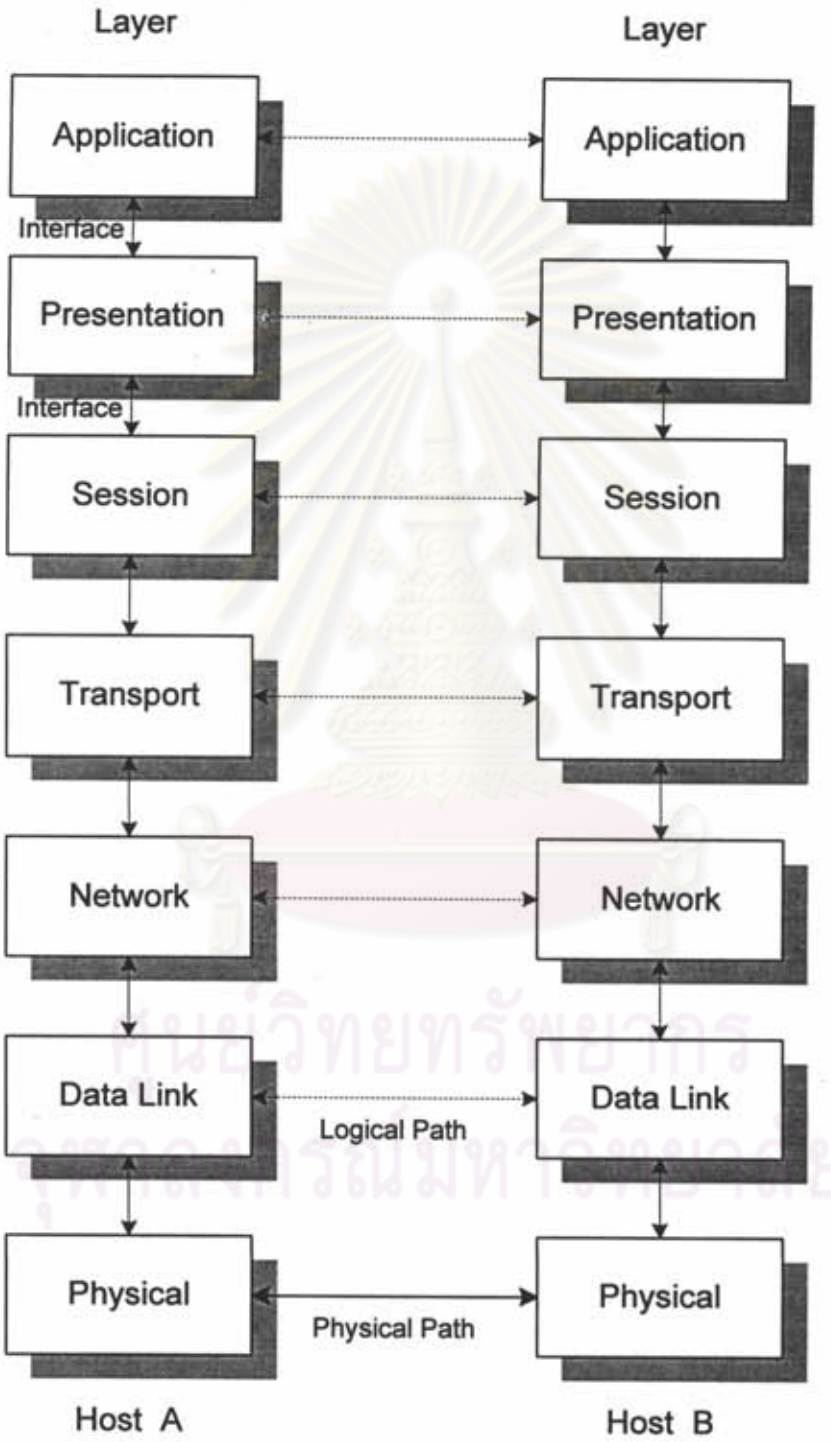
4) ระดับการขนส่ง (Transport Layer) กำหนดที่อยู่ของข่ายงานปลายทางที่ต้องการส่งถึง พร้อมทั้งจัดการตรวจสอบความครบถ้วนของกลุ่มข้อมูลที่ส่ง

5) ระดับการติดต่อ (Session Layer) ควบคุมการเข้าใช้ระบบ เช่น กำหนดให้มีการใช้รหัสประจำตัวผู้ใช้ (User ID) รหัสผ่าน (Password) การเก็บประวัติผู้ใช้ และ กำหนดวิธีการติดต่อระหว่างโปรแกรมประยุกต์ของ 2 ข่ายงาน

6) ระดับแสดงผล (Presentation Layer) กำหนดวิธีการแสดงผลข้อมูล เช่น ตัวอักษรกระพริบ กราฟฟิก กำหนดรูปแบบข้อมูล และ กำหนดการเข้ารหัส (Encryption)

7) ระดับประยุกต์ (Application Layer) กำหนดการประยุกต์ใช้งานของผู้ใช้ เช่น การโอนย้ายข้อมูล การส่งไปรษณีย์อิเล็กทรอนิกส์ เป็นต้น

แต่ละระดับที่กล่าวมาข้างต้นจะมีตัวประสาน (Interface) เพื่อส่งข้อมูลจากระดับหนึ่งไปยังระดับที่อยู่ติดกัน ทำให้สามารถส่งข้อมูลออกไปสู่ภายนอกได้ ดังรูปที่ 2.2



รูปที่ 2.2 แสดงสถาปัตยกรรมข่ายงานแบบมาตรฐาน OSI

จากรูปที่ 2.2 ข้อมูลจะถูกส่งจากระดับประยุกต์ผ่านลงมายังระดับแสดงผล และ ส่งต่อลงมาเรื่อย ๆ จนถึงระดับกายภาพด้วยโปรแกรมตัวประสาน จากนั้นโปรแกรมในระดับกายภาพจะทำการส่งข้อมูลผ่านตัวกลางสื่อสารออกไปยังข่ายงานอื่นที่ระดับกายภาพเช่นกัน (Physical Path) จากนั้นข้อมูลจะถูกส่งขึ้นไปในแต่ละระดับชั้นจนถึงระดับประยุกต์ ซึ่งในแต่ละระดับจะมีโปรโตคอลสื่อสารเพื่อให้สามารถติดต่อในระดับเดียวกันได้ (Logical Path)

สำหรับข่ายงานบริเวณเฉพาะที่ ตามมาตรฐานของสถาบันวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์ (IEEE) คือ IEEE Project 802 ซึ่งเป็นมาตรฐานของข่ายงานบริเวณเฉพาะที่ ที่ยอมรับกัน ได้ถูกจัดระดับการสื่อสารออกเป็น 3 ระดับ ซึ่งจะเทียบเท่ากับ 2 ระดับล่างของมาตรฐาน OSI คือ ระดับกายภาพ และระดับเชื่อมโยงข้อมูล

ต่อไปจะกล่าวถึงส่วนประกอบที่สำคัญภายในข่ายงานไมโครคอมพิวเตอร์ ว่ามีอะไรบ้างและแต่ละส่วนทำหน้าที่อย่างไร

<u>ส่วนประกอบของข่ายงานไมโครคอมพิวเตอร์</u>

โดยปกติภายในข่ายงานไมโครคอมพิวเตอร์หนึ่ง ๆ นั้นจะต้องประกอบด้วยส่วนหลัก ๆ 2 ส่วน คือ ส่วนฮาร์ดแวร์ และ ส่วนซอฟต์แวร์

ส่วนฮาร์ดแวร์

ประกอบด้วยอุปกรณ์ที่สำคัญคือ

1) เครื่องไมโครคอมพิวเตอร์ของผู้ใช้ และ ผู้บริการ เพื่อใช้สำหรับรับส่ง และ ประมวลผลข้อมูล

2) ตัวกลางสื่อสารข้อมูล ได้แก่ สายสัญญาณต่าง ๆ เช่น สายคู่บิดเกลียว สายโคแอคเชียล และ เส้นใยนำแสง ในที่นี้จะกล่าวถึงเฉพาะสายคู่บิดเกลียวพอสังเขป

สายคู่บิดเกลียว[2] เป็นสายตัวนำสัญญาณที่รู้จักกันดีในรูปของสายโทรศัพท์ สามารถรับส่งข้อมูลได้ทั้งแบบอะนาลอกและดิจิตอล แต่มีข้อจำกัดในเรื่องระยะทางของการส่งสัญญาณ มีลักษณะเป็นสายหุ้มฉนวนสองสายพันกันไปตลอดระยะความยาว ตัวนำที่ใช้เป็นลวดทองแดงหรือเหล็กฉาบทองแดง โดยที่ทองแดงจะให้คุณสมบัติในการนำไฟฟ้าได้ดี และ เหล็กจะให้คุณสมบัติในเรื่องแรงดึง และ ความเหนียว สายคู่นี้อาจอยู่รวมกันเป็นเคเบิลใหญ่ก็ได้ ปกติจะมีความหนาประมาณ 0.015 - 0.056 นิ้ว มีราคาถูก น้ำหนักเบา สามารถหาซื้อได้ง่าย การติดตั้งทำได้ง่ายกว่าสายสัญญาณชนิดอื่น

3) พอร์ตอนุกรม

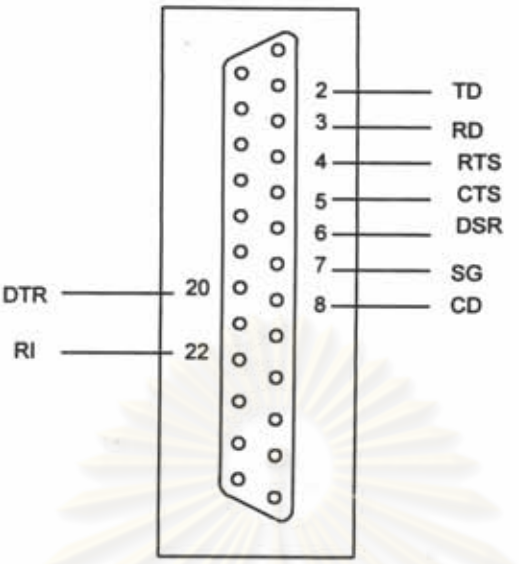
โดยทั่วไปแล้วภายในไมโครคอมพิวเตอร์แต่ละเครื่อง จะมีแผงวงจรสื่อสารอนุกรม สำหรับติดต่อกับอุปกรณ์ภายนอก เรียกว่า พอร์ตอนุกรม หรือ UART (Universal Asynchronous Receiver/ Transmitter) สำหรับมาตรฐานพอร์ตอนุกรมมีอยู่หลายมาตรฐาน แต่ที่นิยมใช้กันอยู่ในปัจจุบัน คือ มาตรฐาน RS-232C

มาตรฐาน RS-232C[3] ได้ถูกจัดพิมพ์ขึ้นเมื่อ ปี ค.ศ. 1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งสหรัฐอเมริกา หรือ EIA (Electronic Industries Assiciation) RS ย่อมาจากคำว่า Recommended Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับปรับปรุงล่าสุด จุดประสงค์ของมาตรฐานตัวนี้ ก็เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment - DTE) กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment - DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็หมายถึงตัวไมโครคอมพิวเตอร์ และ DCE ก็หมายถึง โมเด็ม อุปกรณ์อื่น ๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรมอาจจะเป็นได้ทั้ง DTE และ DCE ขึ้นอยู่กับผู้ผลิต ส่วนความเร็วและระยะทางของการเชื่อมต่อ RS-232C ปัจจุบันสามารถถ่ายโอนข้อมูลได้ถึง 115 บิตต่อวินาที ความยาวของสายสัญญาณไม่เกิน 50 ฟุต จุดข้อต่อของ RS-232C มีอยู่หลายแบบเช่น แบบ 25 ขา (DB-25) แบบ 15 ขา (DB-15) และแบบ 9 ขา (DB-9) แต่ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ 25 ขา และ 9 ขา ดังรูปที่ 2.3 ก และ 2.3 ข

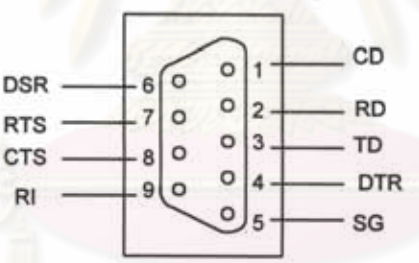
---

[2] ไพศาล สงวนหมู่ และชิน ภู่วรวรรณ, การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ก, (กทม.: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2528), หน้า 136-137

[3] เรื่องเดียวกัน, หน้า 61-62



รูปที่ 2.3 ก แสดงลักษณะของข้อต่อ 25 ขา



รูปที่ 2.3 ข แสดงลักษณะของข้อต่อ 9 ขา

จากรูปที่ 2.3 ก และ 2.3 ข จะแสดงเฉพาะขาสัญญาณที่ใช้กันส่วนใหญ่ ซึ่งแต่ละขามีหน้าที่ต่าง ๆ กันดังนี้

1) ขาที่ 2 (Transmit Data -TD) เป็นขาส่งสัญญาณออกจากไมโครคอมพิวเตอร์ไปยังโมเด็ม หรือ ต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือ เครื่องพิมพ์

2) ขาที่ 3 (Receive Data - RD) เป็นขารับสัญญาณเข้าไปยังไมโครคอมพิวเตอร์

3) ขาที่ 4 (Request to Send - RTS) ใช้สำหรับส่งสัญญาณไปยังโมเด็ม หรือ เครื่องพิมพ์ เป็นการเรียกร้องที่จะส่งสัญญาณออกทางขาที่ 2 สัญญาณนี้จะใช้คู่กับ ขาที่ 5 อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็ส่งสัญญาณออกมาเข้า ขาที่ 5

4) ขาที่ 5 (Clear to Send - CTS) ดังที่อธิบายไว้ในข้อ 3) เมื่อสัญญาณนี้อยู่ในสถานะออฟ หรือ ลอจิก "1" หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

5) ขาที่ 6 (Data Set Ready - DSR) เมื่อสัญญาณสายนี้อยู่ในสถานะออน หรือ ลอจิก "0" เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้ว และ พร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติ จะส่งสัญญาณนี้ไปบอกให้คอมพิวเตอร์รู้ว่า ต่อโทรศัพท์ได้สำเร็จแล้ว

6) ขาที่ 7 (Signal Ground - SG) ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุก ๆ สายของสัญญาณ จะมีแรงดันเป็น "0" เมื่อเทียบกับสัญญาณตัวอื่น

7) ขาที่ 8 (Carrier Detect - CD) โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน ไปบอก DTE เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง

8) ขาที่ 20 (Data Terminal Ready - DTR) คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน เมื่อพร้อมที่จะติดต่อกับโมเด็ม

9) ขาที่ 22 (Ring Indicator - RI) สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบได้อัตโนมัติ (Auto-answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และ ออฟระหว่างเสียงดังของกระดิ่ง

ผู้สนใจอาจสับสนระหว่างสถานภาพของลอจิกกับสถานภาพของสัญญาณ ซึ่งโดยปกติแล้วเรามักจะคุ้นเคยกับสถานะของลอจิกเป็น '1' เมื่อสถานะของสัญญาณเป็นบวก หรือออน แต่ที่กล่าวมาทั้งหมดข้างต้นนี้จะตรงข้ามกัน ทั้งนี้เนื่องมาจากเดิมการติดต่อกันทางโทรเลข การทำงานของสัญญาณต้องครบวงจรทั้งฝ่ายส่งและฝ่ายรับ เมื่อลอจิกเป็น '0' หรือขณะที่ไม่มีอะไรส่งควรจะมีสัญญาณทางไฟฟ้าครบวงจรอยู่ตลอดเวลา เพื่อจะได้ทราบว่ามีวงจรขาดระหว่างทางหรือไม่ ดังนั้นจึงต้องให้ค่าแรงดันที่ฝ่ายส่ง นั่นคือสัญญาณไฟบวกจะหมายถึงลอจิก '0'

การที่พอร์ตอนุกรมจะทำงานได้นั้น จำเป็นต้องอาศัยการควบคุมโดยชิป 8250[4] ซึ่งจะต้องได้รับการโปรแกรมก่อน หลังจากนั้นจะทำงานตามรูปแบบที่ได้โปรแกรมไว้จนกว่าจะมีการโปรแกรมใหม่ หน่วยประมวลผลกลางจะติดต่อกับชิป 8250 ในลักษณะพอร์ตที่เป็นอินพุตเอาต์พุต การจัดพอร์ตนี้กำหนดหมายเลขพอร์ตอย่างเจาะจง โดยทั่วไปจะมีพอร์ตสื่อสาร 2 พอร์ต คือ คอม 1 (COM1) และ คอม 2 (COM2) ซึ่งมีหมายเลขอินพุตเอาต์พุตพอร์ตดังตารางที่ 2.1

---

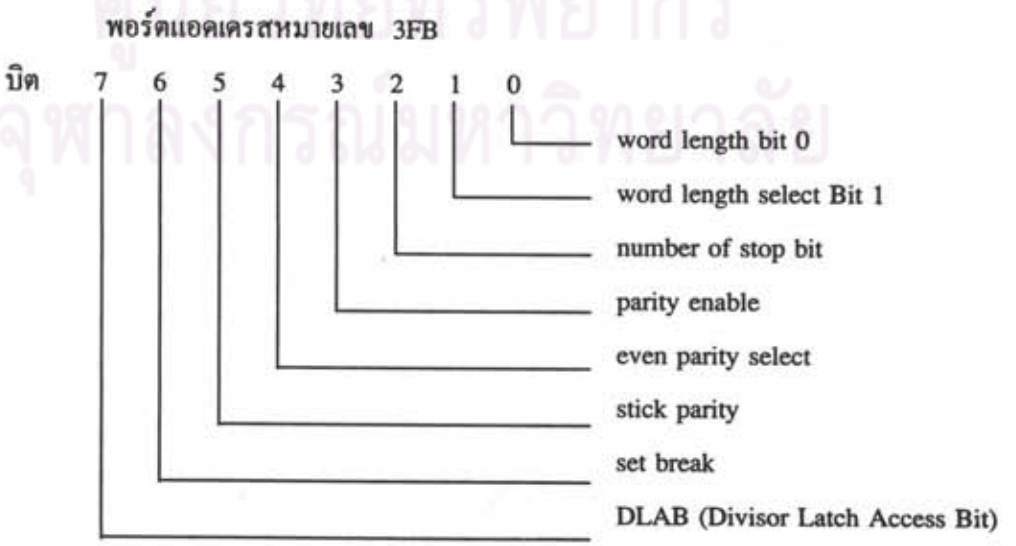
[4] ยืน ภู่วรวรรณ, เทคโนโลยีฮาร์ดแวร์ IBM PC, (กทม.: บริษัทซีเอ็ดยูเคชั่น จำกัด, 2521), หน้า 127-145

อินพุตเอาต์พุต		เลขอรีจิสเตอร์	สถานะ DLAB
พอร์ต COM1	พอร์ต COM2		
3F8	2F8	บัฟเฟอร์ IX	DLAB = 0 (เขียน)
3F8	2F8	บัฟเฟอร์ RX	DLAB = 0 (อ่าน)
3F8	2F8	แลตช์ตัวหาร (LSB)	DLAB = 1
3F9	2F9	แลตช์ตัวหาร (MSB)	DLAB = 1
3F9	2F9	อีนาเบิลอินเทอร์รัพท์	
3FA	2FA	กำหนดอินเทอร์รัพท์	
3FB	2FB	ควบคุมสายสื่อสาร	
3FC	2FC	ควบคุมโมเด็ม	
3FD	2FD	แสดงสถานะสายสื่อสาร	
3FE	2FE	แสดงสถานะโมเด็ม	

ตารางที่ 2.1 แสดงหมายเลขอินพุตเอาต์พุตพอร์ตของ COM1 และ COM2

ภายในชิป 8250 มีรีจิสเตอร์ควบคุมการสื่อสารอยู่หลายตัวด้วยกัน ซึ่งแต่ละตัวมีหน้าที่ต่าง ๆ กัน ในที่นี้จะขอกล่าวเฉพาะตัวที่นำมาใช้ ดังนี้

1) รีจิสเตอร์ควบคุมสายสื่อสาร (Line Control Register) มีขนาด 8 บิต แต่ละบิตมีความหมายดังนี้



บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	ความหมาย
0	0	ข้อมูลขนาด 5 บิต
0	1	ข้อมูลขนาด 6 บิต
1	0	ข้อมูลขนาด 7 บิต
1	1	ข้อมูลขนาด 8 บิต

บิต 2 เป็นบิตที่ใช้ในการกำหนดจำนวนสต็อปบิต

"0" หมายถึง ใช้สต็อปบิต 1 บิต

"1" กรณีส่งแบบ 5 บิต จะมีความยาวของสต็อปบิตเป็น 1.5 บิต นอกเหนือจากนั้น ความยาวของสต็อปบิตเป็น 2 บิต

บิต 3 เป็นบิตแสดงการอีนาเบิลให้มีการตรวจสอบพาริตี้

"0" ไม่ตรวจสอบพาริตี้

"1" ตรวจสอบพาริตี้โดยเพิ่มพาริตี้บิต

บิต 4 เป็นบิตเลือกชนิดของพาริตี้ โดยบิต 3 ต้องมีค่าเป็น "1" เสมอ

"0" เป็นพาริตี้คี่

"1" เป็นพาริตี้คู่

บิต 5 เป็นบิตกำหนดค่าพาริตี้ที่จะแทรก โดย

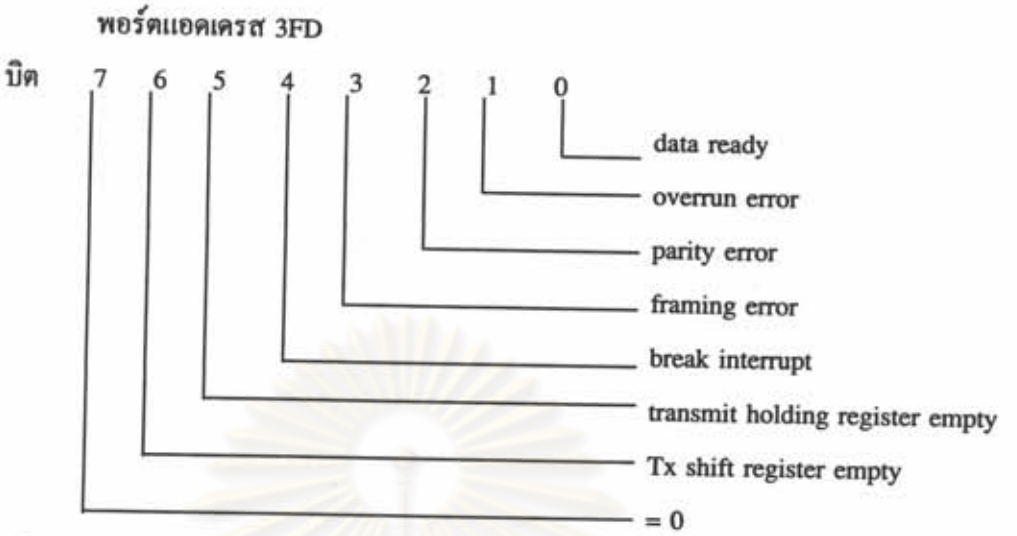
ถ้าบิต 3,4,5 เป็น "1" จะกำหนดพาริตี้เป็น "0"

ถ้าบิต 3,5 เป็น "1" และ บิต 4 เป็น "0" จะกำหนดพาริตี้เป็น "1"

บิต 6 เป็นบิตที่ควบคุมการเบรก

บิต 7 ทำหน้าที่เป็น DLAB บิต ที่จะมีผลต่อการแลตช์ตัวหาร

2) รีจิสเตอร์แสดงสถานะสายสื่อสาร (Line Status Register) รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่หน่วยประมวลผลกลางที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบิตต่าง ๆ เป็นดังนี้



บิต 0 เป็นบิตที่บอกสถานะการรับข้อมูล
ถ้าเป็น "1" แสดงว่ารับข้อมูลเข้ามาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น "0" เมื่อหน่วยประมวลผลกลางได้อ่านข้อมูลในบัฟเฟอร์ไปแล้ว

บิต 1 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิดข้อผิดพลาดในการเขียนทับ (overrun error) กล่าวคือ ขณะที่มีข้อมูลที่บัฟเฟอร์แต่หน่วยประมวลผลกลางยังไม่ได้อ่านไป ปรากฏว่ามีข้อมูลชุดใหม่มาเขียนทับบนบัฟเฟอร์นี้

บิต 2 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเกิดข้อผิดพลาดเกี่ยวกับบิตพาริตี้ (parity error) กล่าวคือถ้ามีการตรวจสอบบิตพาริตี้แล้วไม่เป็นไปตามที่กำหนดไว้

บิต 3 บิตนี้ถ้ามีค่าเป็น "1" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่น ตรวจสอบจำนวนบิตโดยดูที่พาริตี้และสต็อปบิตไม่เป็นไปตามที่กำหนด

บิต 4 บิตนี้จะได้รับการเซตให้เป็น "1" ถ้าหากว่ารับข้อมูลอินพุตเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิต 5 เป็นบิตที่บอกว่าชิป 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสารโดยถูกเซตค่าเป็น "1"

บิต 6 เป็นบิตที่จะบอกว่าชิปรีจิสเตอร์ว่างเปล่า บิตนี้จะได้รับการเซตค่าเป็น "1" เพื่อบอกว่าพร้อมส่งแล้ว

บิต 7 จะเป็น "0" ตลอด

3) รีจิสเตอร์บัฟเฟอร์สำหรับตัวรับข้อมูล (Receiver Buffer Register) เป็นรีจิสเตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนดคือ หมายเลขแอดเดรส 3F8 ขณะที่ DLAB=0 หากหน่วยประมวลผลกลางอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้อ่านข้อมูลที่มาจากสายสัญญาณนั่นเอง

4) รีจิสเตอร์โฮลดิงสำหรับตัวส่งข้อมูล (Transmitter Holding Register) เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจิสเตอร์นี้จะรับข้อมูลจากหน่วยประมวลผลกลาง โดยที่กำหนดพอร์ตเป็นหมายเลข 3F8 เมื่อ DLAB = 0 ข้อมูลของหน่วยประมวลผลกลางที่เอาต์พุตมาที่พอร์ตนี้ก็เพื่อที่จะส่งต่อออกไปยังสายสื่อสาร

5) รีจิสเตอร์แลตช์ตัวหาร (Divisor Latch Register) ทำหน้าที่กำหนดตัวหารในการคำนวณความเร็วการรับส่งข้อมูล หรือ อัตราบอด รีจิสเตอร์นี้มีอยู่ 2 ตัว คือ แลตช์ตัวหารบิตต่ำ (LSB - Least Significant Bit) และ แลตช์ตัวหารบิตสูง (MSB - Most Significant Bit) อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 เมกกะเฮิร์ซ และ สามารถโปรแกรมตัวหารได้ตั้งแต่ 1 ถึง (2^{16}-1) ค่าความถี่ เอาต์พุตของตัวกำหนดอัตราบอดมีค่าเท่ากับ 16 คูณอัตราบอด ดังนั้นตัวหาร เท่ากับ ความถี่สัญญาณนาฬิกา หารด้วย อัตราบอดคูณ16 ซึ่งตัวหารนี้จะต้องถูกกำหนดค่าก่อนแล้วโปรแกรมลงมาในรีจิสเตอร์นี้ การกำหนดต้องให้ DLAB = 1 แล้วให้ลดลงมาในรีจิสเตอร์ 3F8 ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน 3F9 เมื่อ DLAB = 1 จะเป็นค่าของตัวหาร MSB

สำหรับรีจิสเตอร์ที่เหลือจะเป็นรีจิสเตอร์เกี่ยวกับการใช้การขัดจังหวะ (Interrupt) และ การควบคุมโมเดมซึ่งจะไม่กล่าวถึงในที่นี้ ผู้สนใจจะศึกษาเพิ่มเติมได้จากรายการอ้างอิงที่กล่าวไว้

ส่วนซอฟต์แวร์

โดยทั่วไปแล้วซอฟต์แวร์สำหรับข่ายงานบริเวณเฉพาะที่ จะประกอบไปด้วย โปรแกรมประยุกต์ ระบบปฏิบัติการข่ายงาน (Network Operating System)[5] และระบบปฏิบัติการท้องถิ่น (Local Operating System) ซึ่งระบบปฏิบัติการข่ายงานเป็นส่วนเพิ่มเติมจากระบบปฏิบัติการท้องถิ่น ทำหน้าที่ควบคุมการใช้อุปกรณ์ของเครื่องอื่นในข่ายงาน ส่วนใหญ่แล้วระบบปฏิบัติการข่ายงานมักจะทำงานได้ภายใต้ระบบปฏิบัติการท้องถิ่นเท่านั้นเช่น Netware ทำงานภายใต้ดอส และ LAN Manager ทำงานภายใต้ OS/2 เป็นต้น แต่สำหรับการวิจัยนี้จะพัฒนาระบบตามแนวคิดของข่ายงานบริเวณเฉพาะที่แบบไม่ใช้แผงวงจรควบคุมข่ายงานพิเศษมาต่อพ่วงดังที่ได้กล่าวแล้วในบทที่ 1 ดังนั้นซอฟต์แวร์ดังกล่าวข้างต้นจะถูกนำมาพัฒนา และประยุกต์ใช้ในบางส่วนเท่านั้น โดยจะจัดระดับซอฟต์แวร์อ้างอิงตามมาตรฐานของ OSI ดังรูปที่ 2.4

---

[5] Jordan and Churchill, Communications and Networking for the IBM PC and Compatibles, pp. 251-252

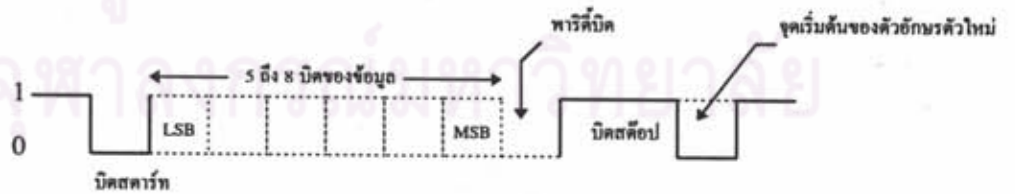
โปรแกรมประยุกต์ (Application)
โปรแกรมควบคุมความมั่นคง (Session)
ระบบปฏิบัติการ MS-DOS (Presentation)
โปรแกรมควบคุมการสื่อสารระดับข้อความ (Data Link)
โปรแกรมควบคุมการสื่อสารระดับไบต์ (Physical)

รูปที่ 2.4 แสดงส่วนซอฟต์แวร์ของข่ายงานไมโครคอมพิวเตอร์

ต่อไปจะกล่าวถึงรายละเอียดของซอฟต์แวร์ในแต่ละระดับว่า มีหน้าที่อย่างไร มีแนวคิดและวิธีการทำงานอย่างไร เป็นต้น

1) โปรแกรมควบคุมการสื่อสาร เป็นโปรแกรมที่ทำหน้าที่ควบคุมการรับส่งข้อมูล ข่าวสารระหว่างเครื่องผู้ใช้และเครื่องบริการในข่ายงาน แบ่งออกเป็น 2 ระดับ คือ

1.1) ระดับกายภาพ โปรแกรมส่วนนี้จะทำหน้าที่ควบคุมการรับส่งข้อมูลในระดับไบต์ผ่านทางอุปกรณ์สื่อสาร ซึ่งในที่นี้จะหมายถึงพอร์ตอนุกรม RS-232C โดยการรับส่งจะเป็นแบบอสมวาร (Asynchronous Transmission)[6] จะประกอบด้วยบิตเริ่มต้น (Start Bit) ตามด้วยข้อมูล 1 ไบต์ และบิตสิ้นสุด (Stop Bit) ดังรูปที่ 2.5



รูปที่ 2.5 แสดงการส่งข้อมูลแบบอสมวาร

[6] ไพศาล สงวนหมู่ และอื่น ภู่วรวรรณ, การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค, 2528 หน้า 29-30

ขณะที่สถานะของการส่งเป็นแบบว่าง (Idle) คือ ยังไม่มีสัญญาณส่งออกมา จะมีสัญญาณหรือ มีแรงดัน (กระแส) ตลอดเวลา เพื่อความแน่ใจว่าฝ่ายรับยังติดต่ออยู่กับฝ่ายส่ง เมื่อเริ่มจะส่งข้อมูล สัญญาณจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา บิตนี้เรียกว่า บิตสตาร์ท ตามหลังของบิตสตาร์ทจะเป็นข้อมูลสำหรับ 1 ตัวอักษร ซึ่งอาจจะมีขนาดตั้งแต่ 5 บิต จนถึง 8 บิต โดยบิตต่ำ (LSB) จะถูกส่งออกมาก่อนไล่ไปจนถึงบิตสูง (MSB) ตามหลังข้อมูลก็จะเป็นบิตพาริตี้ ซึ่งอาจจะใช้หรือไม่ใช้ก็ได้ บิตพาริตี้ทำหน้าที่เป็นตัวตรวจสอบความถูกต้องของสัญญาณที่ได้รับ บิตพาริตี้อาจเป็นแบบคู่ (Even) หรือ แบบคี่ (Odd) หมายความว่าถ้าเป็นพาริตี้คู่ จำนวนบิตข้อมูลที่เป็น 1 รวมแล้วจะต้องเป็นจำนวนคู่ โดยผู้ส่งจะต้องทำหน้าที่ตรวจสอบข้อมูลแล้วใส่บิตพาริตี้เอง ฝ่ายรับเมื่อรับแล้วก็จะต้องตรวจสอบดูว่าเป็นจริงตามที่กำหนดไว้หรือไม่ หลังจากบิตพาริตี้แล้วก็จะต้องมีบิตสต๊อป ซึ่งอาจเป็น 1, 1.5 หรือ 2 ตัวก็ได้ จากที่กล่าวมาแล้วจะเห็นได้ว่าการส่งข้อมูลผ่านพอร์ตอนุกรมนั้น เมื่อเริ่มต้นจำเป็นจะต้องตั้งค่าต่าง ๆ สำหรับพอร์ตอนุกรมไว้ก่อน ได้แก่ ความเร็วในการส่ง จำนวนบิตของข้อมูล 1 ไบต์ บิตตรวจสอบ และ จำนวนบิตสต๊อป

1.2) ระดับเชื่อมโยงข้อมูล โปรแกรมจะทำหน้าที่รวบรวมข้อมูลจากระดับไบต์เป็นระดับกลุ่มข้อมูล (Packet) และ มีการตรวจสอบความถูกต้องของข้อมูลที่ฝ่ายรับ เนื่องจากอาจมีปัญหาในการรับส่งข้อมูลระหว่างไมโครคอมพิวเตอร์ ซึ่งอาจมาจากสาเหตุหลายประการ เช่น คุณภาพของสายสัญญาณไม่ดีพอ หรือ สัญญาณถูกรบกวน ซึ่งจะทำให้ข้อมูลที่ได้รับไม่ถูกต้อง จึงจำเป็นต้องมีการตรวจสอบการรับส่งสัญญาณในระดับซอฟต์แวร์ (Software Handshaking) โดยวิธีหนึ่งที่นิยมกันทั่วไปก็คือการส่งโปรโตคอล[7] (Protocol Transfer) วิธีนี้จำเป็นจะต้องมีโปรโตคอลเหมือนกันทั้งฝ่ายรับและฝ่ายส่ง โดยการใช้อักขระควบคุมในตารางแอสกี (ASCII) สำหรับควบคุมการส่งข้อมูลออกมาเป็นกลุ่มที่มีขนาดคงที่ เช่น

SOH (Start of Heading) มีค่าเท่ากับ 01 ในตารางแอสกี เป็นสัญญาณแจ้งฝ่ายรับว่าขณะนี้กำลังเริ่มต้นส่งข้อมูล

EOT (End of Transmission) มีค่าเท่ากับ 04 เป็นสัญญาณแจ้งฝ่ายรับว่าขณะนี้สิ้นสุดการส่งข้อมูลแล้ว

---

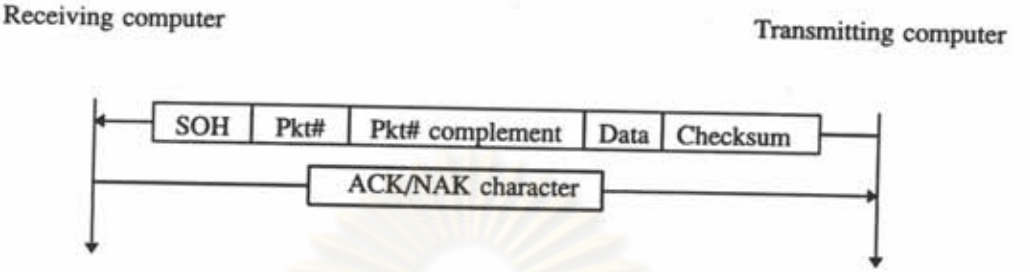
[7] ไพศาล สงวนหมู่ และอื่น ๆ, วรรณ, การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค, หน้า 34-37

NAK (Negative Acknowledge) มีค่าเท่ากับ 021 เป็นสัญญาณแจ้งฝ่ายส่งว่าข้อมูลที่ได้รับนั้นไม่ถูกต้อง หรือในโปรโตคอล XMODEM จะใช้เป็นสัญญาณแจ้งฝ่ายส่งให้เริ่มต้นส่งข้อมูลด้วยเหมือนกัน

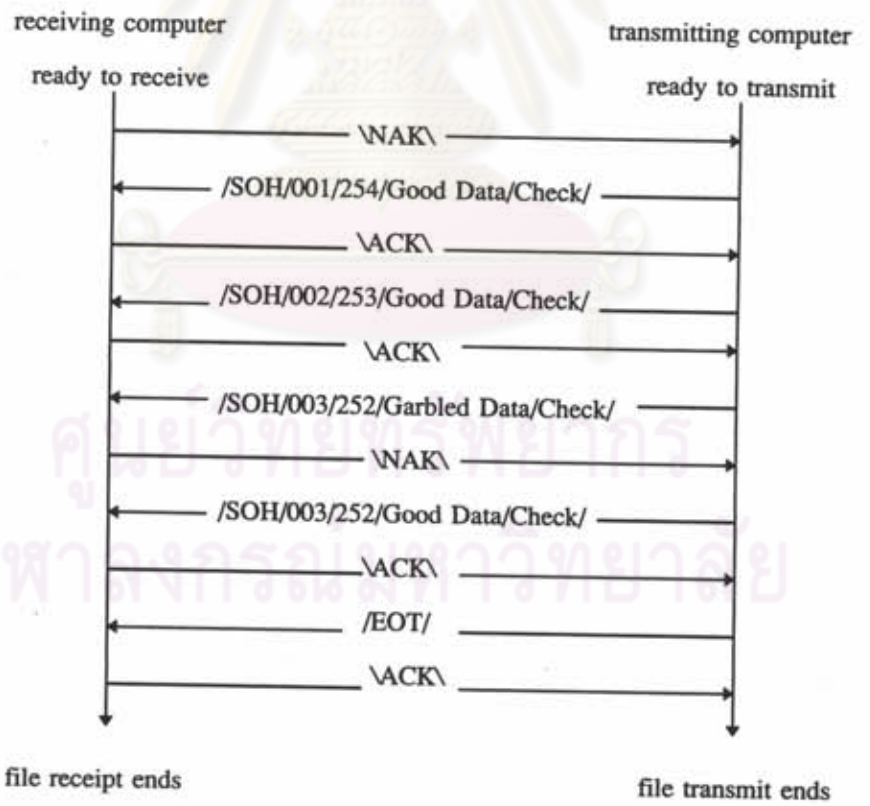
ACK (Acknowledge) มีค่าเท่ากับ 06 เป็นสัญญาณแจ้งฝ่ายส่งว่าข้อมูลที่ได้รับนั้นถูกต้องแล้ว

ตัวอย่างโปรโตคอลที่นิยมใช้ในเครื่องไมโครคอมพิวเตอร์ก็คือ โปรโตคอลโอนย้ายแฟ้มที่มีชื่อว่า เอ็กซ์โมเด็ม (XMODEM) ซึ่งมีรูปแบบและลักษณะการทำงานดังรูปที่ 2.6ก และ 2.6ข

จากรูปที่ 2.6ก และ 2.6ข ฝ่ายรับจะส่ง NAK เพื่อแจ้งฝ่ายส่งให้เริ่มต้นส่งข้อมูลได้ เมื่อฝ่ายส่งได้รับ NAK จากฝ่ายรับ ฝ่ายส่งจะส่งข้อมูลออกไปโดยมีรูปแบบเริ่มด้วย SOH ตามด้วยหมายเลขกลุ่มข้อมูล 2 ไบต์ และ ส่วนเติมเต็ม 1 (1's Complement) ของกลุ่มต่อไปที่จะส่ง ตัวอย่างเช่น /SOH/001/254/... 001 คือ ข้อมูลกลุ่มที่ 1 และ 254 คือ ส่วนเติมเต็ม 1 ของข้อมูลกลุ่มที่ 2 เป็นต้น ต่อจากนั้นก็จะเป็นข้อมูล 128 ไบต์ ตามหลังด้วยการตรวจสอบข้อผิดพลาดโดยวิธีตรวจสอบผลบวก (Checksum) ซึ่งคำนวณมาจากการบวกค่าแอสกีแต่ละไบต์ของข้อมูลที่ส่งออกไปทั้งหมด 128 ไบต์ แล้วหารด้วย 255 เศษที่เหลือก็คือค่า checksum ทางฝ่ายรับจะนำข้อมูลที่ได้ทั้ง 128 ไบต์มารวมกันเพื่อหาค่า checksum และนำไปเปรียบเทียบกับค่าที่ได้รับ หากตรงกันก็แสดงว่าข้อมูลที่ได้รับถูกต้องก็จะส่งสัญญาณ ACK ไปให้ฝ่ายส่งได้รู้ว่าข้อมูลที่ได้รับถูกต้องแล้ว และให้ส่งข้อมูลต่อไปมาได้ แต่ถ้าหากว่าค่า checksum ไม่ถูกต้อง ฝ่ายรับก็จะส่งสัญญาณ NAK ให้ฝ่ายส่ง เพื่อเป็นการบอกให้รู้ว่าข้อมูลที่ได้รับผิดพลาด ทางฝ่ายส่งเมื่อได้รับสัญญาณ NAK ก็จะทำการส่งข้อมูลเดิมออกไปอีกครั้ง การส่งใหม่จะดำเนินไป 9 ครั้ง หากยังคงได้รับแต่สัญญาณ NAK ฝ่ายส่งจะหยุดทำงาน การที่เอ็กซ์โมเด็มส่งเลขบอกกลุ่ม 2 ชุด ก็เพื่อความแน่นอนว่ากลุ่มเดียวกันจะไม่ถูกส่งออกไปสองครั้ง ถ้าหากว่าอักขระควบคุมการส่งเกิดสูญหายไประหว่างการส่ง ถ้าฝ่ายรับตรวจสอบพบว่าเป็นกลุ่มเก่าที่ถูกส่งมาใหม่อีกด้วยความผิดพลาดจาก ACK เป็น NAK ฝ่ายรับก็จะดึงข้อมูลนั้นทิ้งไป เมื่อสิ้นสุดแฟ้มที่จะส่ง ฝ่ายส่งก็จะส่ง EOT เป็นการบอกฝ่ายรับว่าหมดข้อมูลที่จะส่งแล้ว



รูปที่ 2.6 ก แสดงรูปแบบกลุ่มข้อมูลและการตอบรับของโปรโตคอลเอ็กซ์โมเด็ม



รูปที่ 2.6ข แสดงตัวอย่างการทำงานของโปรโตคอลเอ็กซ์โมเด็ม

2) โปรแกรมควบคุมความมั่นคงของข่ายงาน ดังที่ได้กล่าวแล้วในบทแรกว่าวัตถุประสงค์หลักของการนำคอมพิวเตอร์มาเชื่อมต่อเป็นข่ายงานคือ การแลกเปลี่ยนข้อมูลข่าวสารกัน และ การใช้อุปกรณ์ราคาแพงร่วมกัน จึงจำเป็นต้องมีโปรแกรมสนับสนุนให้มีผู้ใช้หลายคนในระบบได้ (Multiuser System) เมื่อมีผู้ใช้หลายคนในระบบโอกาสที่จะเกิดความเสียหายของข้อมูลในระบบย่อมเกิดขึ้นได้ ตัวอย่างเช่น ขณะที่คนหนึ่งกำลังบันทึกแฟ้มข้อมูลอยู่แล้วมีคนอื่นมาบันทึกแฟ้มข้อมูลนั้นด้วยถ้าไม่มีการปิดกั้น (Lock) แฟ้มข้อมูลนั้นก่อน ข้อมูลที่ต้องการก็จะไม่ถูกต้อง หรือกรณีการใช้จานบันทึกที่เครื่องบริการ ถ้าไม่มีการป้องกันส่วนที่สำคัญไว้ อาจมีผู้ใช้ที่ไม่ได้รับอนุญาตเข้าไปทำความเสียหายหรือคัดลอกข้อมูลได้เป็นต้น ดังนั้นจำเป็นต้องมีโปรแกรมอีกชุดหนึ่งสำหรับควบคุมการใช้ทรัพยากรในข่ายงาน โดยทั่วไปสามารถแบ่งการควบคุมออกเป็น 4 ระดับ (บางข่ายงานก็ไม่ได้ใช้ครบทั้งหมด) ดังนี้

2.1) การควบคุมการเข้าถึงข่ายงาน (Network Access) จะควบคุมโดยกำหนดให้มีรหัสประจำตัวผู้ใช้ (User ID) และ รหัสผ่าน (Password) เมื่อผู้ใช้ต้องการเข้าระบบ จะต้องป้อนรหัสประจำตัว และรหัสผ่าน โปรแกรมจะทำการส่งรหัสที่ได้ไปยังเครื่องบริการที่ต้องการ โปรแกรมควบคุมที่เครื่องบริการจะทำการตรวจสอบรหัสเหล่านั้นว่ามีสิทธิในการเข้าถึงระบบหรือไม่ ถ้าไม่มีก็จะส่งข้อความกลับไปบอกผู้ใช้ว่าไม่อนุญาตให้เข้าระบบ การป้องกันในลักษณะนี้จะได้ผลดีก็ต่อเมื่อผู้ใช้เองต้องเก็บรักษารหัสผ่านไว้เป็นอย่างดี และควรจะมีการเปลี่ยนรหัสผ่านบ่อย ๆ ปกติการควบคุมระดับนี้จะใช้ร่วมกับการควบคุมระดับผู้ใช้ ซึ่งจะกล่าวถึงต่อไป

2.2) การควบคุมระดับผู้ใช้ (User-Level Security) เป็นการควบคุมสิทธิในการเข้าถึงทรัพยากรระบบ ตามระดับความสำคัญ และ หน้าที่ของผู้ใช้แต่ละคน โดยอาจจะมีการแบ่งผู้ใช้ออกเป็นกลุ่ม ๆ แต่ละกลุ่มมีสิทธิในการเข้าถึงระบบไม่เท่าเทียมกัน ขึ้นอยู่กับระดับความสำคัญและหน้าที่ดังที่ได้กล่าวแล้ว เช่น

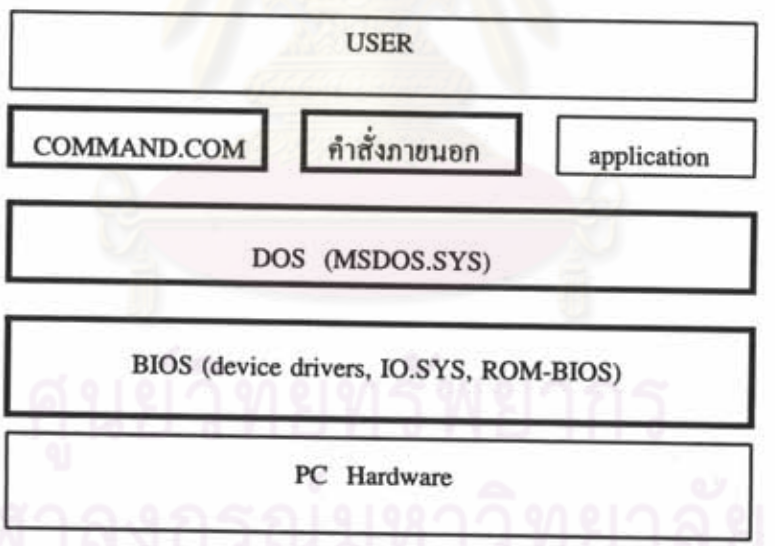
2.2.1) กลุ่มผู้บริหารระบบ (System Administrator or System Supervisor) เป็นกลุ่มผู้ใช้ที่ทำหน้าที่ติดตั้งและดูแลรักษาระบบ จึงมีสิทธิในการเข้าถึงทรัพยากรทั้งหมดในระบบ

2.2.2) กลุ่มผู้ใช้ทั่วไป (End User) จะมีการกำหนดว่าจะให้ใครเข้าถึงทรัพยากรใดได้บ้าง และมีสิทธิในการเข้าถึงระดับไหน ซึ่งอาจมีการแบ่งกลุ่มย่อยตามหน้าที่และความสำคัญอีกก็ได้

2.3) การควบคุมระดับทรัพยากร (Resource-Level Security) เป็นการควบคุมการเข้าถึงของทรัพยากรแต่ละประเภทว่าจะให้ใครเข้าถึงได้บ้าง และเข้าถึงได้ระดับไหน การควบคุมระดับนี้อาจเป็นอิสระจากการควบคุมระดับผู้ใช้ หรือไม่อิสระก็ได้ ถ้าไม่อิสระจากกันจะต้องมีการกำหนดว่าจะให้ใช้ตามระดับไหน อย่างไร

2.4) การควบคุมระดับแฟ้มข้อมูล (File-Level Security) เป็นการกำหนดสิทธิการเข้าถึงแฟ้มข้อมูลแต่ละแฟ้ม (Access Right) เช่น ให้อ่านข้อมูลได้อย่างเดียว หรือ ให้แก้ไขข้อมูลได้ เป็นต้น

3) ระบบปฏิบัติการ ในเครื่องไมโครคอมพิวเตอร์จำเป็นต้องมีโปรแกรมระบบปฏิบัติการเพื่อช่วยอำนวยความสะดวกแก่ผู้ใช้ในการเข้าถึงทรัพยากรระบบเช่น จานบันทึก แผ่นบันทึก เครื่องพิมพ์ และจัดสรรการใช้หน่วยประมวลผลกลาง หน่วยความจำ เป็นต้น ระบบปฏิบัติการที่นิยมใช้กันทั่วไปคือระบบปฏิบัติการเอ็มเอสดอส (MS-DOS) หรือดอส ซึ่งมีโครงสร้างและการทำงานดังรูปที่ 2.7



รูปที่ 2.7 แสดงโครงสร้างและการทำงานของดอส

โครงสร้างภายในของดอส[8] อาจแยกเป็นส่วนต่าง ๆ ได้ 6 ส่วน ซึ่งมีหน้าที่แตกต่างกันและถูกเก็บอยู่เป็นส่วน ๆ แยกจากกันดังนี้

ส่วนแรกคือ รอมไบออส (ROM-BIOS) ย่อมาจากคำว่า Read Only Memory-Basic Input Output System เป็นโปรแกรมที่ควบคุมอุปกรณ์พื้นฐานของไมโครคอมพิวเตอร์ทั้งหมด โปรแกรมไบออสนี้จะถูกเก็บอยู่ในหน่วยความจำแบบรอม ซึ่งยังคงเก็บข้อมูลได้แม้จะปิดเครื่อง จึงเป็นส่วนที่ขึ้นกับฮาร์ดแวร์

ส่วนที่สองเรียกว่า บูตเรคคอร์ด (Boot Record) เป็นโปรแกรมสั้น ๆ ที่ถูกเก็บอยู่ในเซคเตอร์แรกของแผ่นบันทึก หรือจานบันทึกที่ใช้บูตได้ ทำหน้าที่สั่งให้เครื่องไมโครคอมพิวเตอร์ทำการบรรจุโปรแกรมสองส่วนถัดไปของดอสเข้ามาในหน่วยความจำ

สองส่วนถัดไปของดอสคือ IO.SYS และ MSDOS.SYS ทั้งสองส่วนนี้จะถูกบรรจุเข้ามาโดยการทำงานของบูตเรคคอร์ด และจะคงอยู่ในหน่วยความจำตลอดเวลาที่ดอสทำงานอยู่ IO.SYS นั้นเป็นส่วนที่จะทำงานควบคู่กับรอมไบออส ในการควบคุมอุปกรณ์ต่าง ๆ แต่ละตัว นอกจากนี้ถ้ามีการติดตั้งอุปกรณ์ใด ๆ เพิ่มเข้าใหม่ในระบบ สำหรับดอสตั้งแต่รุ่น 2.0 ขึ้นไป ก็จะสามารถติดตั้งโปรแกรมย่อยที่ควบคุมอุปกรณ์ตัวนั้น ๆ ที่เรียกว่าโปรแกรมขับอุปกรณ์ (Device Driver) เข้าเป็นส่วนหนึ่งของ IO.SYS นี้ได้อีกด้วย ดังนั้นจึงกล่าวได้ว่าทั้งรอมไบออส หรือ IO.SYS และโปรแกรมขับอุปกรณ์ที่ติดตั้งเพิ่มนี้เป็นส่วนที่จะทำหน้าที่ร่วมกันในการควบคุมฮาร์ดแวร์โดยตรง คือรวมกันเป็นไบออสทั้งหมด ส่วน MSDOS.SYS นั้นจะเป็นตัวควบคุมในระดับสูงขึ้นไปคือ เป็นตัวประสานการทำงานของอุปกรณ์ต่าง ๆ เข้าด้วยกัน ถือว่าเป็นระดับแก่นของดอส เป็นที่รวมของโปรแกรมย่อยต่าง ๆ ในการจัดการระบบจานบันทึก และอุปกรณ์ต่าง ๆ ที่เรียกว่า DOS Service Function ซึ่งจะเป็นระดับที่ไม่ขึ้นอยู่กับฮาร์ดแวร์

ส่วนถัดไปของดอสคือ COMMAND.COM เป็นตัวแปลคำสั่งซึ่งจะรับคำสั่งที่ป้อนจากอุปกรณ์อินพุต แปลความหมายแล้วเรียกใช้ส่วนอื่น ๆ ของดอสอีกทีหนึ่ง คำสั่งที่ถูกเก็บอยู่ในส่วนนี้จะเรียกว่าเป็นคำสั่งภายใน สำหรับส่วนสุดท้ายจะเป็นคำสั่งภายนอก เป็นโปรแกรมย่อยที่ช่วยอำนวยความสะดวกต่าง ๆ แก่ผู้ใช้ จะถูกบรรจุเข้าไปในหน่วยความจำโดย COMMAND.COM เพื่อทำงานต่อไป มีฐานะเหมือนกับโปรแกรมประยุกต์อื่น ๆ ที่ทำงานภายใต้ดอส

---

[8] วศิน เพิ่มทรัพย์, ยินดีรู้ DOS (กรุงเทพ: บริษัท โปรวิชั่น จำกัด, 2536), หน้า 69-71

จากที่กล่าวมาแล้วจะเห็นได้ว่าในการทำงานของระบบข่ายงานนั้นจำเป็นต้องอาศัยระบบปฏิบัติการในการใช้ทรัพยากรของข่ายงานโดยต้องมีการเพิ่มเติมโปรแกรมบางส่วนเพื่อทำงานร่วมกับระบบปฏิบัติการในการควบคุมการใช้ทรัพยากรเหล่านี้

ต่อไปจะกล่าวถึงรายละเอียดของโปรแกรมขับอุปกรณ์ และ คำสั่ง PRINT ของดอสว่ามีลักษณะการทำงานเป็นอย่างไร เนื่องจากเป็นเครื่องมือสำคัญในการที่จะนำไปประยุกต์ใช้ในการจำลองงานบันทึกข้อมูล และการใช้เครื่องพิมพ์ร่วมกันแบบเก็บพัก (Spooling)

3.1) โปรแกรมขับอุปกรณ์[9] เป็นส่วนหนึ่งของระบบปฏิบัติการ ทำหน้าที่ควบคุมและติดต่อกับอุปกรณ์ต่าง ๆ เช่น จานหรือแผ่นบันทึก และ เครื่องพิมพ์ เป็นต้น โปรแกรมขับอุปกรณ์เป็นโปรแกรมในระดับต่ำสุดของระบบปฏิบัติการเอ็มเอสดอส ได้รับการออกแบบมาเพื่อให้โปรแกรมส่วนอื่นของระบบปฏิบัติการทำงานเป็นอิสระจากฮาร์ดแวร์นั่นคือถ้ามีการเปลี่ยนฮาร์ดแวร์ก็แก้ไขเฉพาะโปรแกรมส่วนนี้เท่านั้น ทำให้ประหยัดเวลาและค่าใช้จ่ายได้มาก โปรแกรมขับอุปกรณ์แบ่งออกเป็น 2 ประเภทคือ

ประเภทแรก เป็นประเภทตัวอักษร (Character Device Driver) ทำหน้าที่ติดต่อควบคุมอุปกรณ์ คีย์บอร์ด จอภาพ เครื่องพิมพ์ และโมเด็ม โดยให้บริการข้อมูลครั้งละ 1 ตัวอักษรในการเรียกใช้แต่ละครั้ง

ประเภทที่สอง เป็นประเภทบล็อก (Block Device Driver) ปกติแล้วจะใช้กับอุปกรณ์ที่เก็บข้อมูลจำนวนมาก เช่น จานบันทึก แผ่นบันทึก หรือ คาสเซทเทปความเร็วสูง โดยจะให้บริการข้อมูลเป็นชุดหรือบล็อกในการเรียกใช้แต่ละครั้ง ขนาดของบล็อกจะแตกต่างกันไปในแต่ละอุปกรณ์ สามารถควบคุมอุปกรณ์ได้มากกว่า 1 อุปกรณ์ในเวลาเดียวกัน นอกจากนี้ยังสามารถจัดแบ่งอุปกรณ์ออกเป็นหลาย ๆ หน่วยตรรกะ (Logical Unit) ได้ด้วย การอ้างถึงโปรแกรมขับอุปกรณ์แบบบล็อกนี้จะไม่อ้างถึงด้วยชื่อแต่จะอ้างถึงด้วยหมายเลขอุปกรณ์ตามลำดับในลูกโซ่ (Chain) เช่น A, B, C, D, ...

---

[9] Tischer, PC System Programming (Germany: Abacus, 1991), pp. 148-163

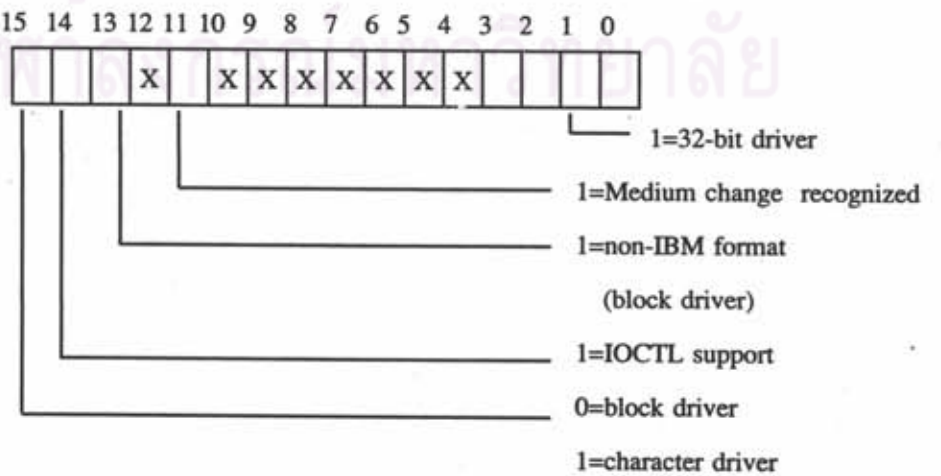
โครงสร้างของโปรแกรมขับอุปกรณ์ (Device Driver Structure) โครงสร้างหลักของโปรแกรมขับอุปกรณ์ประกอบด้วย 3 ส่วนคือ

3.1.1) ส่วนหัวอุปกรณ์ (Device Header) เป็นส่วนเริ่มต้นของโปรแกรมขับอุปกรณ์ ความยาว 18 ไบต์ จะเก็บข้อมูลที่จำเป็นสำหรับดอสในการใช้งานโปรแกรมขับอุปกรณ์นั้น โดย 4 ไบต์แรกเก็บตำแหน่งที่อยู่ของโปรแกรมขับอุปกรณ์ตัวถัดไปในลูกโซ่ ซึ่งผู้เขียนโปรแกรมต้องเตรียมเนื้อที่ในหน่วยความจำไว้ และดอสจะใส่ตำแหน่งให้เองตอนติดตั้งโปรแกรมขับอุปกรณ์นี้ในระบบ 2 ไบต์ถัดไปเก็บข้อมูลกำหนดคุณลักษณะของอุปกรณ์ อีก 2 เขตข้อมูลถัดไปเก็บตำแหน่งที่อยู่ของรูทีนสตราทีจี และ รูทีนขัดจังหวะ ส่วน 8 ไบต์สุดท้ายเก็บชื่อของโปรแกรมขับอุปกรณ์ถ้าเป็นประเภทตัวอักษร หรือ จำนวนอุปกรณ์ที่ควบคุมกรณีเป็นประเภทบล็อก พารามิเตอร์ของส่วนหัวอุปกรณ์เป็นดังตารางที่ 2.2

+ 00H	Offset address of next driver	(1 word)
+ 02H	Segment address of next driver	(1 word)
+ 04H	Device attribute	(1 word)
+ 06H	Offset address of strategy routine	(1 word)
+ 08H	Offset address of interrupt routine	(1 word)
+ 0AH	Driver name from character driver or number of devices used by block driver	(8 bytes)

ตารางที่ 2.2 แสดงพารามิเตอร์ส่วนหัวอุปกรณ์

สำหรับโครงสร้างของคุณลักษณะอุปกรณ์เป็นดังนี้



สำหรับโปรแกรมขับอุปกรณ์ประเภทบล็อกจะใช้เฉพาะบิตที่ 11 ถึง 15 เท่านั้น โดยบิตที่ 15 จะเป็นตัวบอกประเภทโปรแกรมขับอุปกรณ์ ส่วนบิตที่ 11 จะเป็นตัวบอกว่าสื่อบันทึกนั้นถูกเปลี่ยนแปลงได้หรือไม่ เช่น แผ่นบันทึกสามารถถูกเปลี่ยนแผ่นขณะใช้งานได้ เป็นต้น

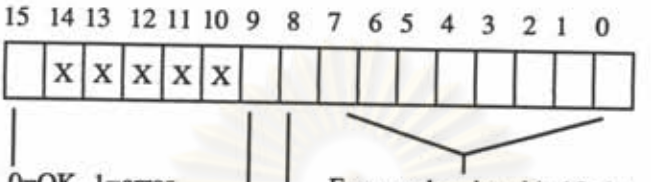
3.1.2) รูทีนสตราทีจี (Strategy Routine) ความยาว 2 ไบต์ เป็นส่วนที่เก็บตำแหน่งที่อยู่ของรีเควสเฮดเดอร์ (Request Header) ซึ่งเป็นโครงสร้างข้อมูลที่ส่งมาจากดอส ประกอบไปด้วย ความยาวของบล็อกข้อมูล หมายเลขอุปกรณ์ รหัสคำสั่ง และรหัสสถานะภาพ ตำแหน่งที่อยู่ของบัฟเฟอร์ที่ใช้เก็บผลลัพธ์ จำนวนเซคเตอร์ และ เซคเตอร์เริ่มต้น ดังตารางที่ 2.3

+ 00H	Data block length in bytes	(1 byte)
+ 01H	Device number in communication	(1 byte)
+ 02H	Command code	(1 byte)
+ 03H	Status field	(1 word)
+ 05H	Reserved	(8 bytes)
+ 0DH	Media descriptor	(1 byte)
+ 0EH	Buffer offset address	(1 word)
+ 10H	Buffer segment address	(1 word)
+ 12H	Number of sectors to process	(1 word)
+ 14H	Starting sector	(8 bytes)

ตารางที่ 2.3 แสดงโครงสร้างของรีเควสเฮดเดอร์

โดยปกติแล้วโครงสร้างของรีเควสเฮดเดอร์จะประกอบด้วยข้อมูลอย่างน้อยที่สุดคือ 13 ไบต์แรกเสมอ ซึ่งจะเก็บข้อมูลที่จะบอกโปรแกรมขับอุปกรณ์ว่าต้องทำอย่างไรกับงานที่จะตามมา และส่วนข้อมูลที่เพิ่มต่อท้ายจากนี้ก็ขึ้นอยู่กับงานที่ตามมานั้นต้องการข้อมูลอะไรบ้าง ซึ่งจะแตกต่างกันไปในแต่ละฟังก์ชัน

ส่วนเซตข้อมูลสถานะภาพเป็นดังนี้



รูทีนสตราทีจีจะถูกเรียกใช้ครั้งแรกจากดอส เพื่อจัดเตรียมค่าเริ่มต้นของโปรแกรมขับอุปกรณ์ และ จะถูกเรียกใช้ซ้ำ ๆ ทุกครั้งก่อนที่มีการขอใช้อินพุตเอาต์พุตจากรูทีนขัดจังหวะ โดยดอสจะส่งตำแหน่งที่อยู่ของรีเควสเฮดเดอร์ผ่านทางรีจิสเตอร์ ES:BX หลังจากนั้นการควบคุมจะถูกส่งกลับคืนไปยังดอส และดอสจะทำการเรียกรูทีนขัดจังหวะเพื่อทำงานต่อไป

3.1.3) รูทีนขัดจังหวะ (Interrupt Routine) เป็นส่วนที่จะทำงานตามคำสั่งของดอส โดยจะถูกเรียกใช้ทันทีหลังจากเสร็จจากรูทีนสตราทีจี ก่อนที่จะทำงานตามคำสั่งรูทีนขัดจังหวะจะทำการเก็บรีจิสเตอร์ต่าง ๆ ไว้ในกองซ้อน (Stack) จากนั้นจะทำงานตามคำสั่งหรือฟังก์ชันในรีเควสเฮดเดอร์ที่ได้รับจากรูทีนสตราทีจี เมื่อทำงานตามฟังก์ชันนั้นเสร็จก็จะทำการจัดเตรียมค่าสถานะภาพของรีเควสเฮดเดอร์ แล้วจึงทำการคืนค่ารีจิสเตอร์จากกองซ้อน

รหัสคำสั่งหรือฟังก์ชันของโปรแกรมขับอุปกรณ์มีอยู่ 13 ฟังก์ชัน สำหรับดอสตั้งแต่รุ่น 2.0 ลงมา แต่ถ้าเป็นดอสรุ่น 3.0 ขึ้นไป จะมีเพิ่มอีก 4 ฟังก์ชัน ในที่นี้จะขอกล่าวเฉพาะที่นำมาใช้งาน ซึ่งมีอยู่เพียง 6 ฟังก์ชัน คือ

3.1.3.1) ฟังก์ชัน 0 : ให้ค่าเริ่มต้นแก่โปรแกรมขับอุปกรณ์ (Driver Initialization) ดอสจะเรียกใช้ฟังก์ชันนี้ระหว่างที่กำลังบูตระบบอยู่เพื่อจัดเตรียมค่าเริ่มต้นต่าง ๆ ให้แก่โปรแกรมขับอุปกรณ์ เช่น ให้ค่าเริ่มต้นแก่ฮาร์ดแวร์ ตัวแปรที่ใช้ภายใน หรือ เปลี่ยนทิศทางของตัวขัดจังหวะต่าง ๆ ให้ชี้ไปยังรูทีนที่ต้องการ เป็นต้น พารามิเตอร์ของฟังก์ชันแสดงได้ดังตารางที่ 2.4

Calling parameters of function 0:	
Offset 2 (byte)	Function number (0)
Offset 18 (ptr)	Address of character that follows the equal sign after the DEVICE command in the CONFIG.SYS file
Offset 22 (byte)	Device number of the first device supported by the driver (0=A, 1=B) (applies to block device drivers from DOS version 3.0 up only)

Returned parameters of function 0:	
Offset 3 (word)	Status word
Offset 13 (byte)	Number of device supported (block device only)
Offset 14 (ptr)	Address of first available memory location following the driver
Offset 18 (ptr)	Address of array containing the addresses of BPB (block device only)

ตารางที่ 2.4 แสดงโครงสร้างพารามิเตอร์เมื่อเรียกใช้ และส่งค่ากลับคืนฟังก์ชัน 0

ฟังก์ชันนี้จะส่งค่ากลับแก่คอส ที่สำคัญคือตำแหน่งที่อยู่ของแถวลำดับ (Array) ซึ่งเก็บตำแหน่งที่อยู่ของไบออสพารามิเตอร์บล็อก (BIOS Parameter Blocks - BPBs) ประจำแต่ละอุปกรณ์ตรรกะ ตำแหน่งที่อยู่ของ BPB มีขนาด 4 ไบต์ โดย 2 ไบต์แรกเป็นค่าขจัด (Offset) และ 2 ไบต์หลังเป็นค่าเซกเมนต์ (Segment) BPB นี้จะเก็บรายละเอียดของอุปกรณ์ ดังตารางที่ 2.5

+ 00H	Bytes per sector	(1 word)
+ 02H	Sectors per cluster	(1 byte)
+ 03H	Reserved sectors (including boot sectors)	(1 word)
+ 05H	Nuber of FATs	(1 byte)
+ 06H	Maximum number of entries in root directory	(1 word)
+ 08H	Total number of sectors	(1 word)
+ 0AH	Media descriptor	(1 byte)
+ 0BH	Number of sectors per FAT	(1 word)

ตารางที่ 2.5 แสดงโครงสร้างของไบออสพารามิเตอร์บล็อก

สำหรับตัวอย่างรายละเอียดของสื่อบันทึก[10] เป็นดังนี้

F0H = 3.5", 2 sides, 18 sectors/track (1.44MB) 3.5", 2 sides, 36 sectors/track (2.88MB) 5.25", 2 sides, 15 sectors/track (1.2MB)
F8H = hard disk
F9H = 5.25", 2 sides, 15 sectors/track, 80 tracks/side (1.2MB) 3.25", 2 sides, 9 sectors/track, 80 tracks/side (720K)
FAH = 5.25", 1 side, 8 sectors/track (320K)
FBH = 3.5", 2 sides, 8 sectors/track (640K)
FCH = 5.25", 1 side, 9 sectors/track, 40 tracks/side (180K)
FDH = 5.25", 2 sides, 9 sectors/track, 40 tracks/side (360K)
FEH = 5.25", 1 side, 8 sectors/track, 40 tracks/side (160K)
FFH = 5.25", 2 sides, 8 sectors/track, 40 tracks/side (320K)

[10] Microsoft Corporation, Microsoft MS-DOS Programmer.s Reference. (Washington: Microsoft Press, 1993), p. 410

3.1.3.2) ฟังก์ชัน 1 : ตรวจสอบสื่อบันทึก (Media Check) ฟังก์ชันนี้ใช้สำหรับโปรแกรมขับอุปกรณ์ประเภทบล็อกเท่านั้น ดอสจะเรียกใช้ฟังก์ชันนี้เมื่อต้องการตรวจสอบว่ามีการเปลี่ยนแผ่นบันทึกหรือไม่ หรือตรวจสอบสารบบในสื่อบันทึกว่ามีการเปลี่ยนแปลงไปจากการเข้าถึงครั้งสุดท้ายหรือไม่ ถ้ามีการเปลี่ยนแปลง ดอสก็จะทำการอ่านสารบบจากสื่อบันทึกเข้าไปเก็บไว้ในหน่วยความจำอีกครั้ง และจะเปลี่ยนค่าบิต 11 ของคุณลักษณะอุปกรณ์เป็น 1 พารามิเตอร์ของฟังก์ชันนี้เป็นดังตารางที่ 2.6

Calling parameters of function 1:	
Offset 1 (byte)	Device number
Offset 2 (byte)	Function number (1)
Offset 13 (byte)	Media descriptor byte

Returned parameters of function 1:	
Offset 3 (word)	Status word
Offset 14 (byte)	Was media changed ? FFH = yes, 00H = don't know, 01H = no
Offset 15 (ptr)	Address of buffer containing the previous volume name (only if device indicates a media change)

ตารางที่ 2.6 แสดงโครงสร้างพารามิเตอร์เมื่อเรียกใช้ และส่งค่ากลับคืนฟังก์ชัน 1

3.1.3.3) ฟังก์ชัน 2 : สร้างไบออสพารามิเตอร์บล็อก (Build BPB) ใช้สำหรับโปรแกรมขับอุปกรณ์ประเภทบล็อก ดอสจะเรียกใช้ฟังก์ชันนี้ทุกครั้งที่ฟังก์ชัน 1 ตรวจสอบพบว่าสื่อบันทึกถูกเปลี่ยนแปลง และจะส่งตารางบีพีบีใหม่สำหรับสื่อบันทึกนั้นกลับคืนให้ดอส พารามิเตอร์เป็นดังตารางที่ 2.7

Calling parameters of function 2:	
Offset 1 (byte)	Device number
Offset 2 (byte)	Function number (2)
Offset 13 (byte)	Media descriptor byte
Offset 14 (ptr)	Address of a buffer containing the FAT

Returned parameters of function 2:	
Offset 3 (word)	Status word
Offset 18 (ptr)	Address of the BPB of addressed device

ตารางที่ 2.7 แสดงโครงสร้างพารามิเตอร์เมื่อเรียกใช้ และส่งค่ากลับคืนฟังก์ชัน 2

3.1.3.4) ฟังก์ชัน 4 : อ่านข้อมูล (Read)' ฟังก์ชันนี้จะอ่านข้อมูลจากสื่อบันทึกเข้าไปเก็บไว้ในบัฟเฟอร์ที่ถูกระบุในพารามิเตอร์ ดังตารางที่ 2.8

Calling parameters of function 4:	
Offset 1 (byte)	Device number (block device only)
Offset 2 (byte)	Function number (4)
Offset 13 (byte)	Media descriptor byte (block device only)
Offset 14 (ptr)	Address of buffer to which data should be read
Offset 18 (word)	Number of sectors to be read (block device) or Number of characters to be read (character device)
Offset 20 (word)	First sector to be read (block device only)

Returned parameters of function 4:	
Offset 3 (word)	Status word
Offset 18 (ptr)	Number of sectors read (block device) or Number of characters read (character device)
Offset 22 (ptr)	Pointer to volume ID on return of error 0FH (version 3.0 and higher)

ตารางที่ 2.8 แสดงโครงสร้างพารามิเตอร์เมื่อเรียกใช้ และส่งค่ากลับคืนฟังก์ชัน 4

3.1.3.5) ฟังก์ชัน 8 : บันทึกข้อมูล (Write) ฟังก์ชันนี้จะโอนย้ายข้อมูลจากบัฟเฟอร์ไปไว้ในสื่อบันทึก พารามิเตอร์เป็นดังตารางที่ 2.9

Calling parameters of function 8:	
Offset 1 (byte)	Device number (block device only)
Offset 2 (byte)	Function number (8)
Offset 13 (byte)	Media descriptor of device addressed (block device only)
Offset 14 (ptr)	Address of buffer containing data
Offset 18 (word)	Number of sectors to be written (block device) or Number of characters to be writted (character device)
Offset 20 (word)	First sector to be written (block device only)

Returned parameters of function 8:	
Offset 3 (word)	Status word
Offset 18 (ptr)	Number of sectors written (block device) or Number of characters written (character device)
Offset 22 (ptr)	Pointer to volume ID on return of error 0FH (version 3.0 and higher)

ตารางที่ 2.9 แสดงโครงสร้างพารามิเตอร์เมื่อเรียกใช้ และส่งค่ากลับคืนฟังก์ชัน 8

3.1.3.6) ฟังก์ชัน 9 : บันทึกและตรวจทาน (Write with Verify) จะทำหน้าที่เหมือนกับฟังก์ชัน 8 แต่ฟังก์ชันนี้จะทำการอ่านและตรวจทานข้อมูลอีกครั้งก่อนที่จะบันทึก พารามิเตอร์จะเหมือนกับของฟังก์ชัน 8

จากฟังก์ชันที่กล่าวมาทั้งหมดนี้ จะสามารถใช้ได้กับสื่อบันทึกที่มีขนาดความจุไม่เกิน 32 เมกกะไบต์ ทั้งนี้เพราะเหตุว่ามีการกำหนดขนาดของที่เก็บเซคเตอร์เริ่มต้นของสื่อบันทึกในรีเควสเฮดเดอร์ไว้เพียง 2 ไบต์ ดังนั้นตั้งแต่ดอสรุ่น 4.0 ขึ้นไปได้มีการสนับสนุนให้สามารถใช้ได้กับสื่อบันทึกที่มีขนาดความจุมากกว่า 32 เมกกะไบต์ โดยการเพิ่มขนาดที่เก็บของเซคเตอร์เริ่มต้นเป็น 4 ไบต์ ซึ่งจะสามารถใช้ได้กับสื่อบันทึกที่มีความจุถึง 4 กิกกะไบต์ การเปลี่ยนแปลงนี้จะมีผลกระทบต่อฟังก์ชัน 0,2,4,8 และ 9 สำหรับรายละเอียดการเปลี่ยนแปลงต่าง ๆ จะกล่าวถึงในบทต่อไป

เมื่อได้ทราบถึงคำจำกัดความ โครงสร้าง และ ฟังก์ชันต่าง ๆ ของโปรแกรมขับอุปกรณ์แล้ว ก็พอจะเป็นแนวทางในการนำไปใช้จำลองจานหรือแผ่นบันทึกข้อมูลของเครื่องบริการในข่ายงานได้เป็น

อย่างดี ส่วนเครื่องมือสำคัญอีกอย่างหนึ่งซึ่งจะช่วยในการใช้เครื่องพิมพ์ร่วมกัน ที่จะกล่าวถึงต่อไปก็คือ คำสั่ง PRINT ของดอส

3.2) คำสั่ง PRINT[11] เป็นคำสั่งภายนอกของดอส ทำหน้าที่จัดแถวคอย (Queue) ในการพิมพ์ข้อมูลจากแฟ้มหลาย ๆ แฟ้มออกทางเครื่องพิมพ์ โดยขณะเดียวกันก็สามารถทำงานอื่นต่อไปได้ด้วย จึงไม่ต้องเสียเวลารอจนเครื่องพิมพ์ทำงานเสร็จ โดยดอสจะแบ่งเวลาบางส่วนมาพิมพ์งาน วิธีการนี้เรียกว่าการเก็บพัก รูปแบบของคำสั่งเป็นดังนี้

PRINT [พารามิเตอร์] [ชื่อแฟ้มที่1 ชื่อแฟ้มที่2 ...]

คำสั่ง PRINT จะจัดการพิมพ์แฟ้มที่ระบุเหล่านี้ตามลำดับจนกว่าจะหมด โดยเมื่อพิมพ์จบแต่ละแฟ้มจะเลื่อนกระดาษขึ้นหน้าใหม่ให้ก่อนพิมพ์แฟ้มถัดไป ไม่พิมพ์ปนกัน ชื่อแฟ้มที่จะพิมพ์มีความยาวรวมทั้งไดรว์และพาท (Path) ด้วยไม่เกิน 64 ตัวอักษร สำหรับพารามิเตอร์อื่นมีดังนี้

/p จะเป็นการนำแฟ้มเข้าแถวคอยที่จะพิมพ์ตามลำดับในคำสั่ง ถ้าเป็นการเริ่มใช้คำสั่ง PRINT ครั้งแรก เครื่องจะมีคำถามขึ้นมาบนจอภาพ เพื่อให้เลือกอุปกรณ์ที่ต้องการพิมพ์ดังนี้

Name of list device [PRN]:

ถ้าไม่ใส่ค่าจะหมายถึง PRN อุปกรณ์ที่เลือกจะเป็นได้ทั้งพอร์ตขนาน และพอร์ตอนุกรม เช่น LPT1, LPT2, LPT3, AUX, COM1, COM2, COM3 และ COM4

/c ยกเลิกการพิมพ์บางแฟ้ม (cancel) โดยระบุชื่อแฟ้มที่จะยกเลิก /c นี้อาจจะใช้ร่วมกับ /p ได้ การใช้ /c และ /p นี้จะมีผลกับแฟ้มที่ /p หรือ /c นี้ต่อท้าย และมีผลกับแฟ้มที่ตามมาทั้งหมดจนกว่าจะจบบรรทัดหรือพบพารามิเตอร์อีกตัวหนึ่ง ถ้าแฟ้มที่ถูกยกเลิกนั้นกำลังถูกพิมพ์อยู่ จะพิมพ์ชื่อแฟ้มนั้นออกมาพร้อมกับข้อความว่า "File canceled by operator" แล้วเลื่อนกระดาษขึ้นหน้าใหม่ และเริ่มพิมพ์แฟ้มถัดไปในแถวคอย

/t เป็นการยกเลิกการพิมพ์ทั้งหมด (terminate) ดอสจะหยุดพิมพ์ ยกเลิกแถวคอยทั้งหมด และ พิมพ์ข้อความว่า "All files canceled by operator" ทางเครื่องพิมพ์ ส่วนบนจอจะมี

---

[11] วศิน เพิ่มทรัพย์, คัมภีร์ DOS, หน้า 255-258

ข้อความว่า "PRINT queue is empty"

/d:ชื่ออุปกรณ์ กำหนดชื่ออุปกรณ์ที่จะพิมพ์ เช่น PRN ถ้าจะใส่ต้องใส่เป็นพารามิเตอร์ตัวแรกเท่านั้น ค่าที่ใช้ได้คือ อุปกรณ์ตรรกะทั้งพอร์ตขนาน และพอร์ตอนุกรมดังกล่าวข้างต้น เนื่องจากโปรแกรม PRINT เป็นโปรแกรมประเภทฝังตัวในหน่วยความจำ (Terminate and Stay Resident) ดังนั้นเมื่อกำหนดชื่ออุปกรณ์แล้วจะเปลี่ยนแปลงแก้ไขไม่ได้ ถ้าต้องการเปลี่ยน ต้องทำการบูตเครื่องใหม่ก่อนแล้วจึงเรียกใช้คำสั่งอีกครั้งพร้อมกำหนดอุปกรณ์ที่ต้องการ

/b:ขนาดบัฟเฟอร์ กำหนดขนาดของบัฟเฟอร์สำหรับเก็บข้อมูลที่จะพิมพ์ (หน่วยเป็นไบต์) ถ้าไม่ใส่จะจัดให้ต่ำสุดเป็น 512 ไบต์ ค่าสูงสุดในดอส 4.0 ขึ้นไปเป็น 16 กิโลไบต์

/q:ขนาดของแถวคอย กำหนดว่าจะให้แถวคอยของแฟ้มที่จะพิมพ์มีขนาดยาวได้มากที่สุดกี่แฟ้ม ค่าที่ใช้ได้อยู่ระหว่าง 4 ถึง 32 ถ้าไม่ใส่จะจัดให้เท่ากับค่าที่ใช้ในดอสรุ่นต่ำกว่า 3.0 คือ 10 แฟ้ม

/s:ส่วนแบ่งเวลา กำหนดส่วนแบ่งเวลาการทำงานของเครื่องที่จะนำมาใช้ในการพิมพ์ มีค่าระหว่าง 1 ถึง 255 ค่าที่กำหนดให้ถ้าไม่ใส่คือ 8 สัญญาณนาฬิกา หรือ 8 tick (1 tick มีค่าประมาณ 1/18 วินาที)

/u:เวลาคอย กำหนดช่วงเวลาที่จะคอยถ้าเครื่องพิมพ์ไม่ว่าง มีค่าอยู่ระหว่าง 1 ถึง 255 ถ้าไม่กำหนดจะเป็น 1 tick ถ้าคอยเกินช่วงเวลานี้จะต้องข้ามไปพิมพ์ในรอบเวลาถัดไปแทน

/m:เวลาพิมพ์ กำหนดช่วงเวลาที่จะใช้พิมพ์ในแต่ละรอบ มีค่าระหว่าง 1 ถึง 255 ถ้าไม่กำหนดจะเป็น 2 tick

คำสั่ง PRINT นี้ถ้าไม่ใส่พารามิเตอร์เลย จะเป็นการแสดงรายการแฟ้มในแถวคอย

เมื่อได้ทราบถึงส่วนต่าง ๆ ของระบบปฏิบัติการที่จะอำนวยความสะดวกในการใช้อุปกรณ์ในระบบแล้ว โปรแกรมอีกส่วนหนึ่งที่สำคัญอย่างยิ่งในข่ายงาน นั่นคือโปรแกรมประยุกต์ใช้งาน ซึ่งจะกล่าวถึงแนวคิดหรือหลักการต่อไปพอสังเขป

4) โปรแกรมประยุกต์ใช้งาน การประยุกต์ใช้ข่ายงานนั้นมีอยู่หลายรูปแบบ เช่น Client-Server, Peer-to-Peer เป็นต้น ในที่นี้จะกล่าวถึงแบบ Client-Server เนื่องจากเป็นรูปแบบที่ง่ายต่อการพัฒนาและนำไปใช้งาน

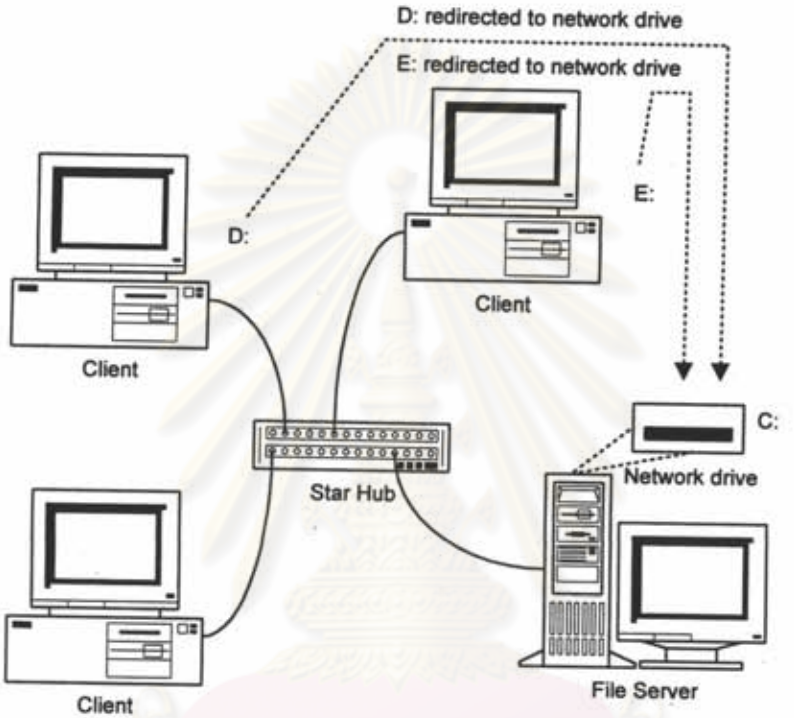
ระบบ Client-Server เป็นระบบที่มีเครื่องบริการ (Server) คอยให้บริการการขอใช้อุปกรณ์ และ การติดต่อสื่อสารกัน แก่ผู้ใช้หรือลูกค้า (Client) ซึ่งการติดต่อสื่อสารจะเป็นลักษณะทิศทางเดียวคือผู้ใช้จะขอใช้บริการได้ฝ่ายเดียว โดยเครื่องบริการไม่สามารถขอใช้บริการจากเครื่องผู้ใช้ได้ สำหรับการประยุกต์ใช้งาน โดยความต้องการพื้นฐานของข่ายงานจะนำไปใช้ในการใช้จานบันทึก และ เครื่องพิมพ์ร่วมกันที่เครื่องบริการของข่ายงาน ซึ่งจะกล่าวถึงหลักการกว้าง ๆ ที่ใช้กันทั่วไป เช่น ระบบปฏิบัติการเนตแวร์ เป็นต้น

4.1) การใช้จานบันทึกร่วมกัน เทคนิคหนึ่งที่นิยมใช้กันคือการจำลองหน่วยขับจานบันทึก หรือ หน่วยขับจานบันทึกเสมือน (Virtual Drive) โดยวิธีการนี้ผู้ใช้จะสามารถมองเห็นหน่วยขับจานบันทึกของเครื่องบริการเป็นเสมือนหน่วยขับจานบันทึกของตนเองอีกหนึ่งหน่วย เช่น หน่วยขับจานบันทึก C ของเครื่องบริการอาจจะถูกจำลองเป็นหน่วยขับจานบันทึก D ของเครื่องผู้ใช้ เป็นต้น จากนั้นผู้ใช้ก็จะสามารถใช้คำสั่งเกี่ยวกับแฟ้มและสารบบของดอสได้ตามปกติ (อาจมีข้อจำกัดบ้าง)

ลักษณะการทำงานแบบนี้จะต้องมีโปรแกรมจำลองจานบันทึกทางด้านผู้ใช้สำหรับคอยดักการเรียกใช้หน่วยขับจานบันทึก แล้วตรวจสอบว่าเป็นหน่วยขับจานบันทึกเสมือนของข่ายงานหรือไม่ ถ้าไม่ใช่ก็คืนการควบคุมกลับไปให้ดอสเพื่อทำงานตามปกติ แต่ถ้าใช่ก็จะทำการส่งคำสั่งการเรียกใช้แฟ้มบนจานบันทึกเสมือนนั้น ออกทางตัวกลางสื่อสารไปที่เครื่องบริการ แล้วคอยรับผลการเรียกใช้ ส่วนทางเครื่องบริการจะมีโปรแกรมบริการแฟ้ม (File Server) คอยตรวจสอบว่ามีคำสั่งเรียกใช้หน่วยขับจานบันทึกจากเครื่องผู้ใช้หรือไม่ ถ้ามีก็จะทำงานตามคำสั่งที่ได้รับ แล้วส่งผลกลับไปที่เครื่องผู้ใช้ เมื่อโปรแกรมทางด้านผู้ใช้ได้รับผลก็จะจัดเตรียมผลนั้นตามความต้องการของคำสั่ง แล้วส่งคืนกลับไปยังดอสเพื่อทำการแสดงผลต่อไป

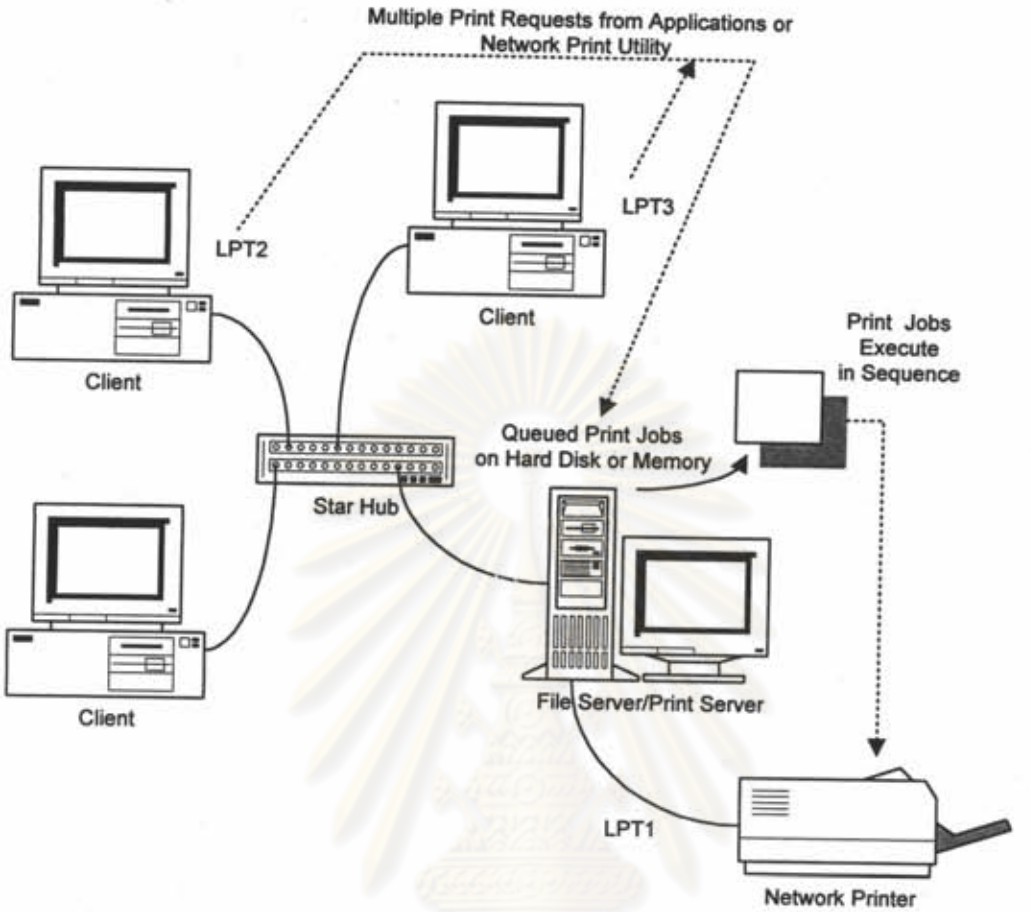
โปรแกรมบริการแฟ้มของเครื่องบริการนี้อาจเป็นแบบ dedicated หรือ non-dedicated ก็ได้ กรณีเป็นแบบ dedicated เมื่อเรียกใช้โปรแกรมแล้วจะไม่สามารถใช้เครื่องทำงานอื่นได้ แต่ก็มีข้อดีคือความน่าเชื่อถือของระบบสูง เนื่องจากไม่มีการรบกวนจากโปรแกรมอื่น และการพัฒนาโปรแกรมทำได้ง่ายส่วนกรณีแบบ non-dedicated นั้น เมื่อเรียกใช้โปรแกรมแล้ว โปรแกรมจะทำงานแบบฝังตัวในหน่วยความจำ หรือ ทำงานเบื้องหลัง (Background) จึงสามารถใช้เครื่องทำงานอื่น ๆ ได้ แต่อาจทำให้

ความน่าเชื่อถือของระบบลดลง เพราะว่าการทำงานพร้อม ๆ กันหลาย ๆ งานย่อมทำให้มีโอกาสเกิดความผิดพลาดได้ง่ายกว่าทำงานเดียว ลักษณะการจำลองจานบันทึกแสดงได้ดังรูปที่ 2.8



รูปที่ 2.8 แสดงการจำลองจานบันทึกที่เครื่องบริการ

4.2) การใช้เครื่องพิมพ์ร่วมกัน ในการใช้เครื่องพิมพ์ร่วมกันที่เครื่องบริการนั้นอาจเป็นแบบส่งพิมพ์ได้ครั้งละงานคือต้องรอให้พิมพ์งานแรกจนเสร็จแล้วจึงส่งพิมพ์งานที่สองต่อได้ หรือ เป็นแบบส่งพิมพ์ได้หลาย ๆ งานพร้อมกันโดยไม่ต้องรอให้งานแรกเสร็จ ที่เรียกว่าการเก็บพัก ซึ่งแบบหลังจะนิยมใช้มากกว่า ดังรูปที่ 2.9



รูปที่ 2.9 แสดงการใช้เครื่องพิมพ์ร่วมกันแบบเก็บพัก

โปรแกรมที่ใช้จะประกอบด้วยโปรแกรม 2 ส่วน คือ โปรแกรมเปลี่ยนทิศทางการพิมพ์ (Print Redirector) ทางเครื่องผู้ใช้ และ โปรแกรมบริการการพิมพ์ (Print Server) ทางเครื่องบริการ เมื่อผู้ใช้สั่งพิมพ์แฟ้มออกทางพอร์ตขนาน หรือ พอร์ตอนุกรมที่ถูกกำหนดให้เป็นเสมือนเครื่องพิมพ์ของเครื่องบริการ เช่น LPT2, LPT3 หรือ COM1, COM2 เป็นต้น โปรแกรมเปลี่ยนทิศทางที่คอยดักตรวจสอบพอร์ตก็จะทำการส่งแฟ้มและคำสั่งไปที่เครื่องบริการ โปรแกรมบริการการพิมพ์เมื่อได้รับคำสั่งก็จะทำการจัดแฟ้มนั้นเข้าไปไว้ในแถวคอยการพิมพ์ (Print Queues) ตามที่ระบุในคำสั่ง แล้วจึงดึงแฟ้มในแถวคอยออกไปพิมพ์ตามลำดับ สำหรับโปรแกรมทางเครื่องผู้ใช้นอกจากจะเป็นแบบเปลี่ยนทิศทางการพิมพ์แล้ว ยังมีอีกวิธีหนึ่งคือ เขียนโปรแกรมสำหรับควบคุมการพิมพ์โดยตรง โดยผู้ใช้ต้องสั่งพิมพ์แฟ้มผ่านทางโปรแกรมนี้เท่านั้น โปรแกรมก็จะทำการส่งแฟ้มและคำสั่งไปพิมพ์ที่เครื่องบริการ วิธีนี้เป็นวิธีที่ง่ายต่อการพัฒนา แต่การใช้งานจะไม่สะดวกเท่ากับวิธีแรกที่ผู้ใช้สามารถสั่งพิมพ์แฟ้มจากโปรแกรมอื่น ๆ ได้ เช่น Lotus 1-2-3, CU writer และ MS-Word เป็นต้น

เทคนิคการเขียนโปรแกรมประเภทฝังตัวในหน่วยความจำ

โปรแกรมประเภทฝังตัวในหน่วยความจำ (Terminate and Stay Resident Program - TSR)[12] เป็นโปรแกรมที่หลังจากคืนการควบคุมกลับไปให้โปรแกรมที่เรียกใช้แล้ว ยังคงอยู่ในหน่วยความจำ เพื่อให้โปรแกรมอื่นสามารถเรียกใช้ได้ในภายหลัง โปรแกรมประเภทนี้แบ่งออกเป็น 3 ชนิด คือ

1) Service Programs เป็นโปรแกรมที่ประกอบด้วยฟังก์ชันต่าง ๆ สำหรับให้โปรแกรมอื่นเรียกใช้ จะถูกเรียกขึ้นมาทำงานด้วยคำสั่ง INT ในภาษาแอสเซมบลี ซึ่งเป็นลักษณะของการขัดจังหวะด้วยซอฟต์แวร์ ตัวอย่างเช่น คำสั่ง PRINT ของดอส

2) Pop-up Programs เป็นโปรแกรมที่ถูกเรียกใช้โดยการกดอักขระพิเศษบนแผงแป้นอักขระ (Keyboard) ตัวอย่างเช่น SHIFT+PRINT SCREEN หรือ CTRL+BREAK ซึ่งเป็นลักษณะของการขัดจังหวะด้วยฮาร์ดแวร์ ตัวอย่างโปรแกรมที่ใช้วิธีนี้ได้แก่ Sidekick เป็นต้น

3) Hardware-Support Programs เป็นโปรแกรมควบคุมการทำงานของอุปกรณ์ในระดับต่ำ ซึ่งจะมีฟังก์ชันให้โปรแกรมอื่นเรียกใช้ผ่านตัวขัดจังหวะ เช่น โปรแกรมขับอุปกรณ์

โปรแกรม TSR นี้ ต้องประกอบด้วยอย่างน้อย 2 ส่วน คือ

1) Initialization Routine เป็นส่วนที่ทำหน้าที่

- ตรวจสอบว่าโปรแกรม TSR นั้นถูกบรรจุไว้ในหน่วยความจำแล้วหรือยัง เพื่อป้องกันไม่ให้เกิดการบรรจุซ้ำ โดยอาจใช้ตัวขัดจังหวะ 2FH เช่นเดียวกับคำสั่ง PRINT หรือ เลือกตัวขัดจังหวะที่ว่างอยู่ในการเซตแฟล็กก็ได้

- ติดตั้งรูทีนขัดจังหวะ (Interrupt Handler) โดยเปลี่ยนแอดเดรสของรูทีนขัดจังหวะเดิมในตารางเวกเตอร์ตัวขัดจังหวะ (Interrupt Vector Table) ให้ชี้ไปที่รูทีนขัดจังหวะของโปรแกรม TSR นั้น ซึ่งภายในดอสจะมีฟังก์ชันของตัวขัดจังหวะ 21H สำหรับเปลี่ยนแอดเดรสดังกล่าว คือ ฟังก์ชัน 25H แต่ก่อนที่จะเปลี่ยนต้องเก็บแอดเดรสเดิมไว้ก่อนโดยเรียกฟังก์ชัน 35H เพื่อนำมาเรียกใช้ภายหลัง หรือให้โปรแกรมอื่นเรียกใช้ในลักษณะของลูกโซ่การขัดจังหวะ (Interrupt Chain) และ คืนค่ากลับไปยังตำแหน่งเดิมเมื่อจะออกจากโปรแกรม สำหรับตารางเวกเตอร์ตัวขัดจังหวะนั้นจะเริ่มต้นที่ตำแหน่งแรก คือ 0000:0000 ของหน่วยความจำ มีขนาด 1024 ไบต์ ซึ่งแต่ละรายการจะเก็บแอดเดรสของรูทีนขัดจังหวะ

- ปลดปล่อยแอดเดรสของส่วนโปรแกรมที่ไม่ได้ใช้กลับคืนไปให้ดอส

- ฝังตัวในหน่วยความจำโดยใช้ตัวขัดจังหวะ 27H หรือตัวขัดจังหวะ 21H ฟังก์ชัน 31H

---

[12] Microsoft Corporation, Microsoft MS-DOS Programmer's Reference, (Washington: Microsoft Press, 1993), pp. 125-130

2) รูทีนขัดจังหวะ เป็นส่วนปฏิบัติการสำหรับให้โปรแกรมอื่นเรียกใช้ อาจมีมากกว่า 1 รูทีนก็ได้ ในรูทีนนี้โดยทั่วไปแล้วจะต้องทำหน้าที่ดังต่อไปนี้

- เก็บค่ารีจิสเตอร์ทั้งหมดไว้ในกองซ้อน และคืนค่าก่อนที่จะออกจากรูทีน
- เปลี่ยนกองซ้อนมาเป็นของรูทีนตนเอง เพื่อป้องกันไม่ให้เกิดกองซ้อนล้น และเปลี่ยนกลับคืนก่อนออกจากรูทีน
- ปิดทางการขัดจังหวะ (Disable Interrupt) เมื่อมีการกระทำกระบวนการวิกฤติ เช่น การเปลี่ยนกองซ้อน และเปิดทางการขัดจังหวะนั้นทันทีหลังจากเสร็จสิ้นกระบวนการ
- ใช้คำสั่ง IRET เพื่อออกจากรูทีนและโอนการควบคุมกลับไปยังโปรแกรมที่เรียกใช้

ในการเขียนโปรแกรมแบบ TSR นี้ ต้องระมัดระวังเกี่ยวกับเรื่องการกลับเข้าใหม่ (Reentry)[13] ในการเรียกฟังก์ชันของดอส ทั้งนี้เนื่องจากดอสได้ใช้กองซ้อนภายในดอสเองในการทำงานของฟังก์ชัน และ ดอสไม่มีกลไกสำหรับจัดเก็บค่าเดิมของกองซ้อนเหล่านี้ ดังนั้นเมื่อมีการเรียกใช้ฟังก์ชันใหม่ขณะที่ฟังก์ชันเดิมเดียวกันนั้นยังไม่เสร็จ จะเกิดการทับกันของกองซ้อน อาจทำให้ระบบหยุดชะงักได้ ซึ่งวิธีการแก้ปัญหาที่ดอสไม่ได้ประกาศเป็นทางการ มีอยู่ 2 วิธี คือ วิธีแรกเป็นการตรวจสอบค่าตัวนับจำนวนครั้งที่ดอสเรียกฟังก์ชันนั้น ดอสได้กำหนดให้มีตัวนับจำนวนเรียกว่า INDOS Flag ซึ่งจะถูกเพิ่มค่า และลดค่าทุกครั้งที่มีการเรียกใช้และเลิกใช้ฟังก์ชันตามลำดับ ซึ่งสามารถอ่านค่านี้ได้จากฟังก์ชัน 34H ถ้าค่าเป็น 0 แสดงว่าดอสอยู่ในสถานะว่างสามารถเรียกใช้ฟังก์ชันได้ทุกฟังก์ชัน การใช้วิธีนี้ยังมีข้อจำกัดอยู่บ้างกล่าวคือ เมื่อไรก็ตามที่ดอสอยู่ในสถานะรอรับคำสั่ง (มีตัวพร้อมปรากฏอยู่บนจอภาพ) ค่าของ INDOS Flag จะไม่เป็น 0 นั่นคือ ถ้าโปรแกรม TSR จะถูกเรียกขึ้นมาทำงานได้เมื่อ INDOS Flag ต้องมีค่าเป็น 0 เท่านั้น โปรแกรม TSR นี้ก็จะไม่ถูกเรียกขึ้นมาทำงาน ดังนั้นต้องมีวิธีการเพิ่มเติมอีกวิธีหนึ่งก็คือการใช้ตัวขัดจังหวะ 28H เนื่องจากดอสจะเรียกตัวขัดจังหวะ 28H ทุก ๆ ครั้งที่มีการเรียกใช้ฟังก์ชัน 01H ถึง 0CH และ INDOS Flag ไม่เป็น 0 นั่นคือ สามารถเรียกโปรแกรม TSR ขึ้นมาทำงานได้โดยดักการทำงานของตัวขัดจังหวะ 28H นี้ และโปรแกรมสามารถเรียกใช้ฟังก์ชันนอกเหนือจากนี้ได้อย่างปลอดภัย เนื่องจากใช้กองซ้อนต่างกองกัน แต่ไม่สามารถเรียกใช้ฟังก์ชัน 01H ถึง 0CH ได้ อย่างไรก็ตามการเรียกโปรแกรม TSR ผ่านทางตัวขัดจังหวะ 28H วิธีเดียวอาจเกิดปัญหาได้ กรณีถ้าไม่มีการเรียกใช้ฟังก์ชัน 01H ถึง 0CH โปรแกรม TSR นั้นก็จะไม่ถูกเรียกขึ้นมาทำงาน ดังนั้นต้องใช้ตัวขัดจังหวะอื่นในการเรียกแทน

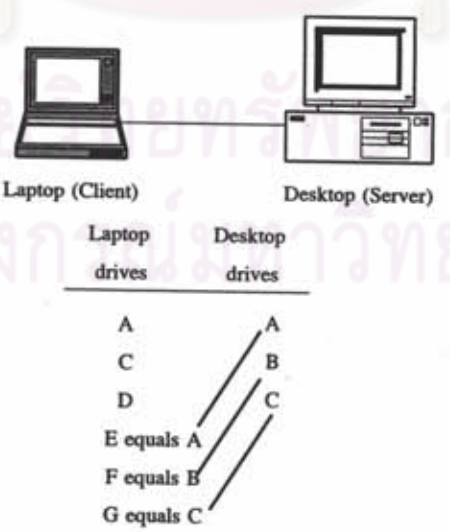
---

[13] Simrin, The Waite Group's MS-DOS Bible, 3rd ed. (USA: The Waite Group, Inc., 1989), pp. 274-276

ซอฟต์แวร์อื่นที่พบ

ในตอนท้ายบทที่ 1 ได้กล่าวถึงซอฟต์แวร์อื่น ๆ ที่พบในปัจจุบันไปบ้างแล้ว ซึ่งซอฟต์แวร์ที่รู้จัก และหาได้ง่ายที่สุด ได้แก่ โปรแกรม Interlink[14] ของดอส ดังนั้นในการวิจัยนี้จะกล่าวถึงรายละเอียดเฉพาะโปรแกรมนี้เท่านั้น เพื่อนำมาใช้ในการทดสอบเปรียบเทียบกับโปรแกรมที่ได้พัฒนาขึ้นต่อไป

โปรแกรม Interlink เป็นโปรแกรมสำหรับควบคุมการเชื่อมต่อไมโครคอมพิวเตอร์ 2 เครื่องเข้าด้วยกันผ่านพอร์ตอนุกรม หรือพอร์ตขนานโดยใช้สายเคเบิล เพื่อให้เครื่องหนึ่งสามารถใช้ข้อมูลหรือโปรแกรม และเครื่องพิมพ์ของอีกเครื่องหนึ่งได้ เครื่องที่ใช้ป้อนคำสั่งจะเรียกว่าผู้ใช้ และเครื่องที่เชื่อมต่อกับผู้ใช้จะเรียกว่า ผู้บริการ เครื่องผู้ใช้จะใช้หน่วยขับจานบันทึก และเครื่องพิมพ์ของเครื่องผู้บริการ ส่วนเครื่องผู้บริการจะแสดงสถานะของการเชื่อมต่อระหว่างเครื่องทั้งสอง ตัวอย่างเช่น สมมุติว่าทำการเชื่อมต่อเครื่องแลปทอปซึ่งติดตั้งให้เป็นเครื่องผู้ใช้ กับเครื่องเดสก์ทอปซึ่งติดตั้งให้เป็นเครื่องผู้บริการเข้าด้วยกัน ในเครื่องแลปทอปมีหน่วยขับจานบันทึก 3 หน่วย คือ A และหน่วยขับจานบันทึกชนิดแข็ง C, D ส่วนเครื่องเดสก์ทอปก็มีหน่วยขับจาน 3 หน่วยเช่นกัน คือ A, B และหน่วยขับจานบันทึกชนิดแข็ง C เมื่อวิ่งโปรแกรม Interlink หน่วยขับจานบันทึกต่าง ๆ บนเครื่องเดสก์ทอป จะกลายเป็นหน่วยขับจานบันทึกที่เพิ่มขึ้นมาในเครื่องแลปทอป กล่าวคือ ที่เครื่องแลปทอปนอกจากจะมีหน่วยขับจาน A, C และ D แล้ว ก็จะมีหน่วยขับจาน E, F และ G เพิ่มขึ้นมาดังรูปที่ 2.10



[14] Microsoft Corporation, คู่มือการใช้ Microsoft MS-DOS Thai Edition, (Thailand: Microsoft Press, 1993), p. 169-176

จากรูปที่ 2.10 หน่วยขับจาน E ของเครื่องแลปทอปจะหมายถึงหน่วยขับจาน A ของเครื่องเดสก์ทอป หน่วยขับจาน F จะหมายถึงหน่วยขับจาน B และหน่วยขับจาน G จะหมายถึงหน่วยขับจาน C ซึ่งถ้าใช้คำสั่งกับหน่วยขับจาน E, F, G ก็จะส่งผลต่อหน่วยขับจาน A, B, C ตามลำดับด้วย เช่น ถ้าพิมพ์คำสั่ง dir g:\ บนเครื่องแลปทอป คอสก็จะแสดงรายการแฟ้มทั้งหมดที่อยู่ในสารบบบนหน่วยขับจาน C ของเครื่องเดสก์ทอป เป็นต้น

ก่อนที่จะเรียกใช้โปรแกรม Interlnk ต้องจัดเตรียมฮาร์ดแวร์ ซอฟต์แวร์ และหน่วยความจำให้ตรงตามรายการต่อไปนี้

- พอร์ตอนุกรมหรือพอร์ตขนานที่ยังว่างอยู่ของเครื่องไมโครคอมพิวเตอร์ทั้ง 2 เครื่อง
- สายเคเบิลอนุกรมแบบ 3 เส้น หรือสายเคเบิลแบบ null-modem ชนิด 7 เส้น หรือสายเคเบิลขนานแบบ 2 ทาง
- เอ็มเอสดอสรุ่น 6.0 ขึ้นไป
- หน่วยความจำของเครื่องผู้ใช้ 16 กิโลไบต์ และของเครื่องบริการ 130 กิโลไบต์

ขั้นตอนการติดตั้งโปรแกรม INTERLNK.EXE ที่เครื่องผู้ใช้ มีดังนี้

1) ตรวจสอบดูว่ามีโปรแกรม INTERLNK.EXE อยู่บนจานบันทึกหรือไม่ ถ้าไม่มีอาจคัดลอกแฟ้ม INTERLNK.EXE โดยใช้กระบวนการคัดลอกแฟ้มแบบรีโมต

2) เพิ่มคำสั่ง device ในแฟ้ม CONFIG.SYS เพื่อระบุตำแหน่งของแฟ้ม INTERLNK.EXE และสามารถระบุตัวเลือกสำหรับกำหนดทิศทางของหน่วยขับจาน และเครื่องพิมพ์ได้ ตัวอย่างเช่น คำสั่ง device=c:\dos\interlnk.exe /drives:5 เป็นการระบุตำแหน่งของแฟ้ม INTERLNK.EXE ในสารบบ DOS และกำหนดทิศทางให้หน่วยขับจานทั้งหมด 5 หน่วย แทนที่จะเป็น 3 หน่วยตามปกติ

3) บูตเครื่องใหม่อีกครั้ง

จากนั้นถ้าต้องการทราบสถานะของหน่วยขับจาน และพอร์ตต่าง ๆ สามารถกระทำได้โดยป้อนคำสั่ง interlnk โปรแกรมจะแสดงข้อความบนจอภาพดังตัวอย่างต่อไปนี้

```
Microsoft Interlnk version 1.00

Port=LPT1
Drive letters redirected  5  (D:  through  H:)
Printer ports redirected  2  (LPT1:  through  LPT2:)

This  Computer              Other  Computer
     (Client)                      (Server)
-----------------          ---------------------
        D:      equals          A:
        E:      equals          B:
        F:      equals          C: (85Mb) MS-DOS_6
        G:      equals          D:
        H:      equals          E:
      LPT1:   equals          LPT2:
      LPT2:   equals          LPT3:
```

จากตัวอย่างข้างต้นเป็นการแสดงสถานะของเครื่องผู้ใช้ ซึ่งมีหน่วยขับจานบันทึกที่ได้จากการกำหนดทิศทางใหม่ 5 หน่วย และพอร์ตเครื่องพิมพ์ที่ได้จากการกำหนดทิศทางใหม่ 2 พอร์ต ในบรรทัดที่ระบุพอร์ตต่าง ๆ จะแสดงให้ทราบว่า Interlnk ใช้พอร์ตใดในการเชื่อมต่อกับเครื่องบริการ และจะแสดงขนาดและป้ายหมวด (Volume Label) ของจานบันทึกชนิดแข็งบนเครื่องบริการ เช่น หน่วยขับจาน C ของเครื่องบริการมีขนาด 85 เมกกะไบต์ และมีป้ายหมวดคือ "MS-DOS_6"

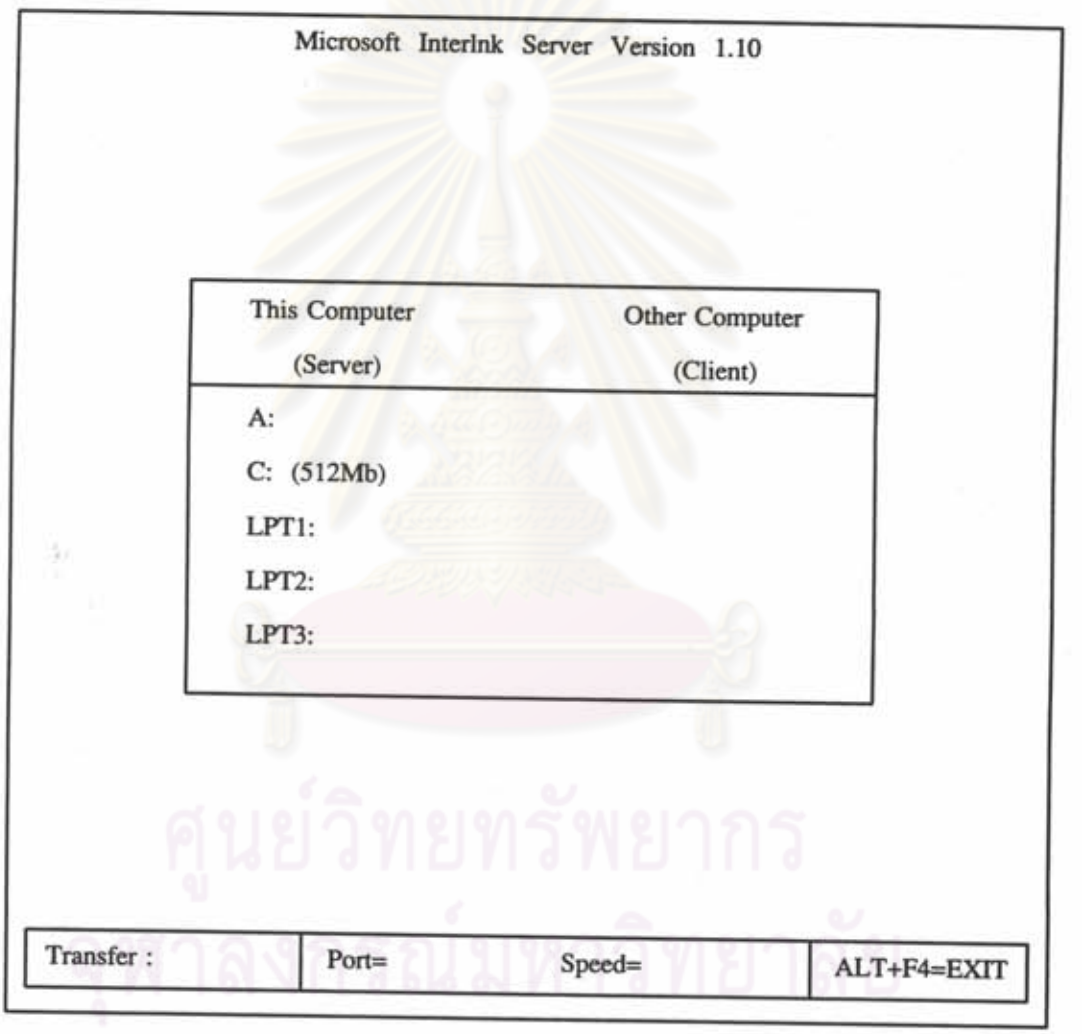
ส่วนการติดตั้งโปรแกรมที่เครื่องบริการ ไม่ต้องเปลี่ยนแปลงแก้ไขแฟ้ม CONFIG.SYS เพียงแต่ป้อนคำสั่ง INTERSVR โปรแกรมจะแสดงข้อมูลเกี่ยวกับหน่วยขับจาน และพอร์ตเครื่องพิมพ์บนจอภาพ ในคอลัมน์ "This Computer" จะแสดงรายการของเครื่องบริการ และคอลัมน์ "Other Computer" จะแสดงรายการของเครื่องผู้ใช้ ส่วนทางด้านล่างของจอภาพจะแสดงสถานะการเชื่อมต่อของ Interlnk ดังต่อไปนี้

Transfer แสดงว่าผู้ใช้กำลังอ่านหรือบันทึกข้อมูลที่เครื่องบริการ และแสดงเครื่องหมายดอกจัน (*) ถัดจากหน่วยขับจานที่กำลังอ่านหรือบันทึก

Port แสดงพอร์ตที่เชื่อมต่อกับเครื่องผู้ใช้

Speed แสดงความเร็วในการส่งข้อมูลระหว่างเครื่อง 2 เครื่อง

โปรแกรม INTERSVR นี้มีลักษณะเป็น dedicated เมื่อติดตั้งแล้วจะไม่สามารถทำงานอื่นได้ และถ้าต้องการหยุดการทำงานของโปรแกรมให้กดปุ่ม ALT+F4 ตัวอย่างจอภาพเป็นดังนี้



โปรแกรม Interlnk ของดอสที่กล่าวมาทั้งหมดนี้ พอสรุปได้ว่าในการนำไปประยุกต์ใช้งานนั้น ก็เพื่อวัตถุประสงค์ที่จะใช้หน่วยขับจานบันทึก และเครื่องพิมพ์ของเครื่องอื่นร่วมกันเท่านั้น และเชื่อมต่อได้เพียง 2 เครื่อง นอกจากนี้ยังไม่มีระบบควบคุมความมั่นคงข่ายงานและในการส่งแฟ้มไปพิมพ์ที่เครื่องบริการจะไม่เป็นแบบเก็บพัก แต่เป็นลักษณะของการพิมพ์ครั้งละงานจนเสร็จก่อนจึงจะสามารถรับแฟ้มจากเครื่องอื่นมาพิมพ์ต่อได้