บทที่ 4

ตัวเลียนแบบเทอร์มินัล

ตัวเลียนแบบเทอร์มินัลในงานวิจัยนี้มีจุดมุ่งหมายเพื่อเป็นตัวเชื่อมต่อระบบปฏิบัติการยูนิกซ์เข้ากับซียูไรเตอร์และส่วนที่ทำหน้าที่เป็นหน้าฉาก (front-end) โดยผู้ใช้จำเป็นต้องใช้ตัวเลียนแบบเทอร์มินัลเพื่อลงบันทึกเข้า (login) สู่ระบบปฏิบัติการยูนิกซ์ก่อนจากนั้นจึงสามารถเรียกใช้ซียูไรเตอร์ผ่านตัวเลียนแบบเทอร์มินัล

การทำงานพื้นฐานของตัวเลียนแบบเทอร์มินัล

ตัวเลียนแบบเทอร์มินัลคือโปรแกรมชนิดหนึ่งที่ทำให้เครื่องไมโครคอมพิวเตอร์ทำหน้าที่เป็นเทอร์มินัลสำหรับติดต่อเข้าใช้บริการของคอมพิวเตอร์เครื่องอื่น ตัวเลียนแบบเทอร์มินัลมีหน้าที่ติดต่อกับแป้นพิมพ์และอุปกรณ์สื่อสารในขณะเดียวกัน โดยที่เมื่อมีอักขระส่งมาถึงอุปกรณ์สื่อสารตัวเลียนแบบเทอร์มินัลจะต้องพร้อมที่จะรับอักขระนั้นไปเก็บไว้เพื่อแสดงผลบนจอภาพ ในขณะเดียวกันเมื่อมีการกดแป้นพิมพ์ ตัวเลียนแบบเทอร์มินัลจะทำหน้าที่ส่งอักขระนั้นผ่านอุปกรณ์สื่อสารออกไป แม้ว่าตัวเลียนแบบเทอร์มินัลจะมีหลักการทำงานง่าย ๆ แต่ในทางปฏิบัติจะมีหลักการทำงานพื้นฐานอยู่สองวิธี คือวิธีหยั่งสัญญาณ และวิธีขัดจังหวะ

1. วิธีหยั่งสัญญาณ (polling)

โดยวิธีหยั่งสัญญาณ โปรแกรมจะทำหน้าที่คอยตรวจสอบพอร์ตสื่อสารเสมอเมื่อมีอักขระมาถึงพอร์ตสื่อสารโปรแกรมจะอ่านข้อมูลดังกล่าวไป ขณะเดียวกันต้องคอยรอรับการกดแป้นพิมพ์ของผู้ใช้โปรแกรมและแสดงผลไปพร้อมกัน เรียกว่าการวนรอบหยั่งสัญญาณ (round robin polling)

วิธีนี้มีข้อเสียคือจำเป็นจะต้องให้เวลาส่วนใหญ่อยู่กับการรอคอยข้อมูลที่จะมาถึงพอร์ตสื่อสารเพื่อไม่ให้ข้อมูลดังกล่าวถูกเขียนทับก่อนที่จะอ่านค่าออกมา ทำให้ไม่สามารถใช้อัตราการสื่อสารที่ความเร็วสูงได้

2. วิธีขัดจังหวะ (interrupt)

ตัวเลียนแบบเทอร์มินัลที่ออกแบบโดยใช้วิธีนี้จะมีโปรแกรมรับบริการขัดจังหวะ (interrupt service routine) ซึ่งจะเริ่มทำงานเมื่อมีสัญญาณขัดจังหวะจากพอร์ตสื่อสารเพื่อบอกว่ามีข้อมูลมาถึงพอร์ตสื่อสารแล้ว เมื่อเสร็จสิ้นการทำงานจะกลับไปทำงานในโปรแกรมหลัก โปรแกรมรับบริการขัดจังหวะนี้ทำหน้าที่อ่านข้อมูลจากพอร์ตสื่อสารแล้วนำไปเก็บไว้ในบัฟเฟอร์รับอักขระเพื่อรอให้โปรแกรมหลักนำข้อมูลไปใช้ต่อไป ตัวเลียนแบบเทอร์มินัล ที่ออกแบบโดยใช้หลักการพื้นฐานนี้สามารถรับข้อมูลได้ โดยไม่มีข้อมูลสูญหายเนื่องจากมีข้อมูลใหม่เข้ามาแทนที่ก่อนที่จะอ่านข้อมูลไป

ตัวเลียนแบบเทอร์มินัลภาษาไทยในวิทยานิพนธ์ฉบับก่อน

สมนึก เจียมเจริญเดช ได้พัฒนาตัวเลียนแบบเทอร์มินัลภาษาไทย เป็นวิทยานิพนธ์ปริญญามหาบัณฑิตจุฬาลงกรณ์มหาวิทยาลัยในปี พ.ศ. 2533 ตัวเลียนแบบเทอร์มินัลดังกล่าวเป็นการเลียนแบบเทอร์มินัลชนิด VT220 สามารถแสดงภาษาไทยโดยใช้รหัส ส.ม.อ.ซึ่งเป็นรหัส 8 บิต ในภาวะกราฟิก ตัวเลียนแบบเทอร์มินัลดังกล่าวใช้วิธีขัดจังหวะช่วยในการรับข้อมูลจากพอร์ตสื่อสาร

ตัวเลียนแบบเทอร์มินัลภาษาไทยในวิทยานิพนธ์ฉบับนี้

ตัวเลียนแบบเทอร์มินัลในวิทยานิพนธ์ฉบับนี้พัฒนาขึ้นโดยอาศัยหลักการสำคัญในการพัฒนาเช่นเดียวกับตัวเลียนแบบเทอร์มินัลภาษาไทยของสมนึก ตัวเลียนแบบเทอร์มินัลในวิทยานิพนธ์ฉบับนี้มีข้อแตกต่างจากโปรแกรมเดิม 6 ประการคือ

1. แบบอักขระใช้แบบอักขระขนาดเดียวกับที่ใช้ในซียูไรเตอร์ คือมีความ กว้าง 8 จุดและความยาว 20 จุดแต่เพื่อให้สามารถแสดงผลได้ 25 บรรทัดจึงแสดงอักขระในด้านยาวเพียง 19 จุดเท่านั้น ดังนั้นจึงสามารถใช้ซียูฟอนต์ซึ่งเป็นโปรแกรมออกแบบตัวอักษรของซียูไรเตอร์เพื่อใช้แก้ไขแบบอักขระได้

2. โปรแกรมรับบริการขัดจังหวะอสมวาร ได้รับการดัดแปลงจากเดิมให้สามารถทำงานได้เมื่อเกิดการขัดจังหวะทั้งจากการที่มีอักขระมาถึงพอร์ตสื่อสาร และเมื่อพอร์ตสื่อสารว่างพร้อมที่จะส่งอักขระ

3. ตัวเลียนแบบเทอร์มินัลเดิมจะส่งอักขระที่ได้จากการกดแป้นพิมพ์เป็นรหัส 8 บิต ในวิทยานิพนธ์ฉบับนี้จะแปลงอักขระที่รับมาจากแป้นพิมพ์เป็นรหัสกลาง 7 บิตก่อน

4. ในการแสดงอักขระจะแปลงรหัสกลางที่รับเข้ามาเป็นอักขระภาษาไทย เพื่อแสดงบนจอภาพโดยอัตโนมัติ

5. โปรแกรมหลักและโปรแกรมแปลลำดับอักขระหลีก ได้รับการจัดการทำงานใหม่ ให้มีการทำงานเป็นสถานะ (state) ชัดเจนขึ้น โดยมีการควบคุมการเปลี่ยนสถานะในการทำงานของตัวเลียนแบบเทอร์มินัลโดยใช้ข้อความ (message)

6. สามารถเรียกใช้ชียูไรเตอร์ผ่านตัวเลียนแบบเทอร์มินัลได้

ส่วนประกอบของตัวเลียนแบบเทอร์มินัล

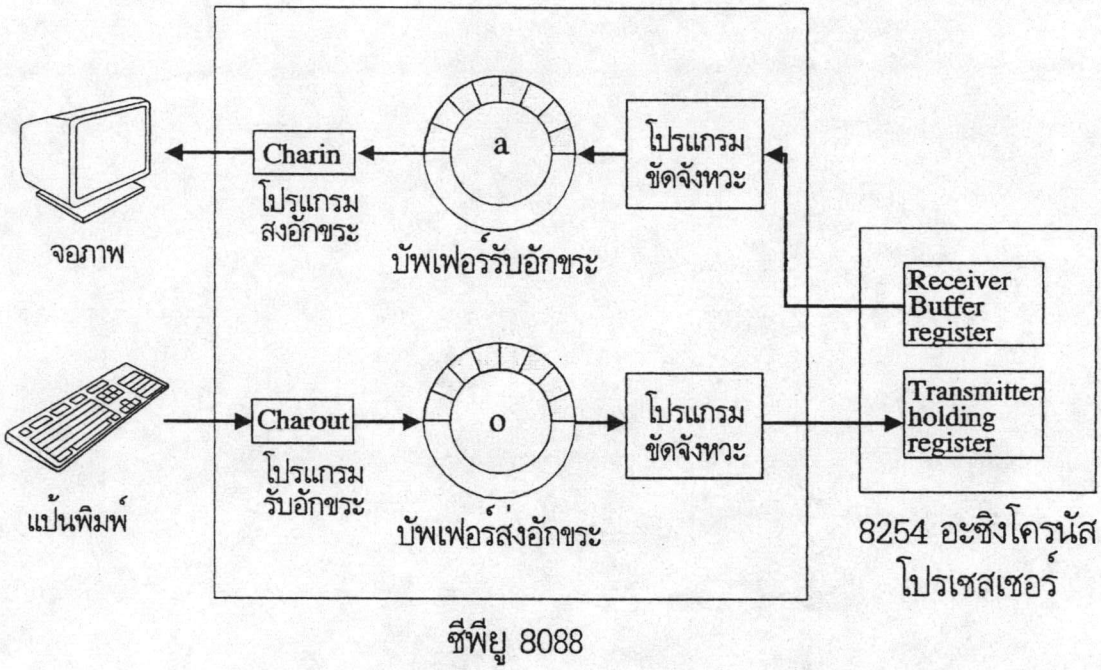
ส่วนประกอบสำคัญของตัวเลียนแบบเทอร์มินัลมีสองส่วนคือส่วนโปรแกรมหลักซึ่งจะทำหน้าที่อ่านข้อมูลที่มาจากพอร์ตสื่อสาร โดยผ่านบัฟเฟอร์รับอักขระและเขียนข้อมูลลงพอร์ตสื่อสารผ่านบัฟเฟอร์ส่งอักขระรวมทั้งการรับข้อมูลจากแป้นพิมพ์ การแสดงผลบนจอภาพ และโปรแกรมรับบริการขัดจังหวะรับและส่งข้อมูลที่อยู่ในบัฟเฟอร์กับพอร์ตสื่อสารส่วนประกอบและกระแสข้อมูลในตัวเลียนแบบเทอร์มินัลแสดงในรูป 4.1

1. โปรแกรมหลัก (main program)

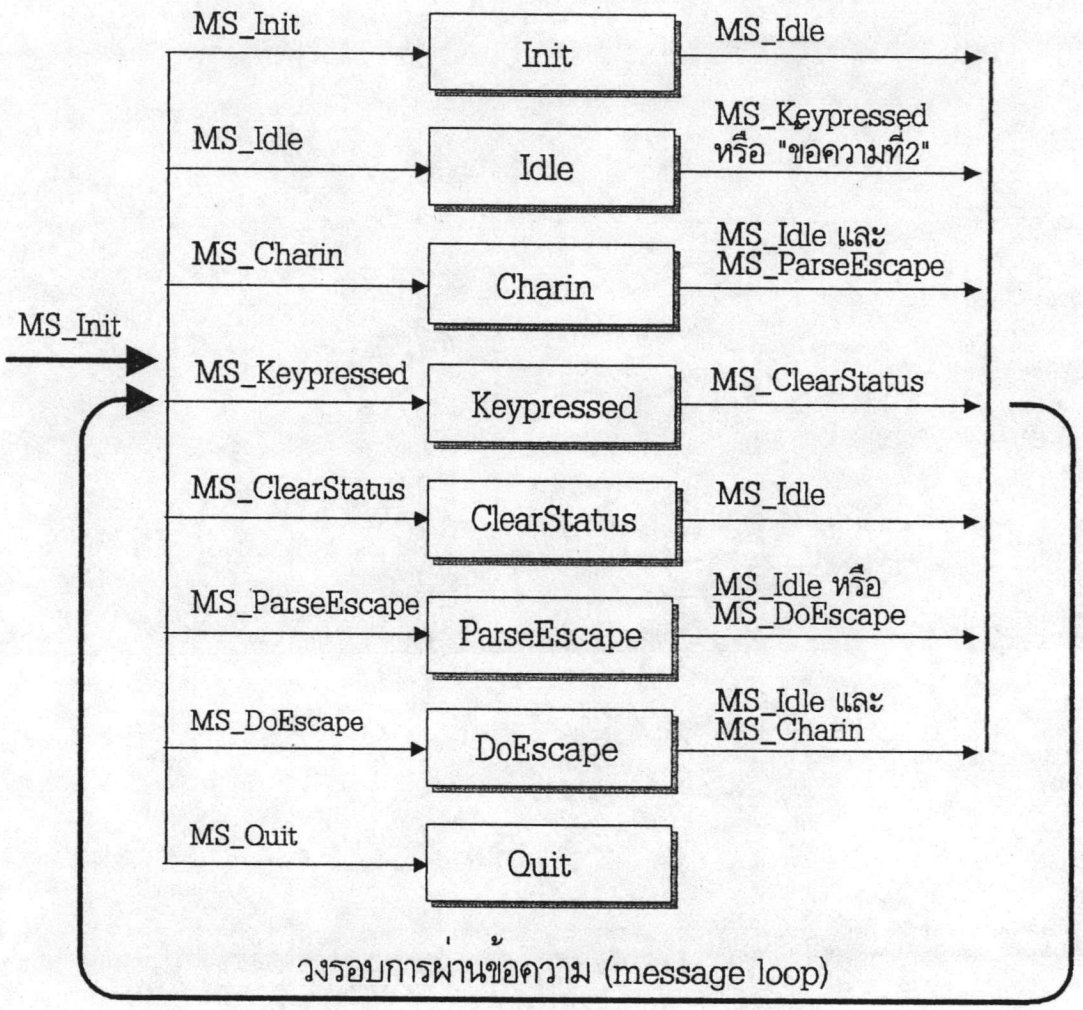
โปรแกรมหลักใช้สถาปัตยกรรมการผ่านข้อความ (message passing architecture) โดยจะประกอบด้วยวงรอบการทำงานหลักทำหน้าที่เป็นวงรอบข้อความ (message loop) การทำงานของโปรแกรมหลักจะเปลี่ยนสถานะ (state) ในการทำงานต่างกันไปขึ้นกับข้อความ (message) ที่ได้รับ รูป 4.2 แสดงข้อความที่ใช้ควบคุมสถานะของโปรแกรมหลัก และรูปที่ 4.3 แสดงการเปลี่ยนสถานะเมื่อได้โปรแกรมหลักได้รับข้อความแต่ละข้อความ

1.1 ข้อความ MS_Init

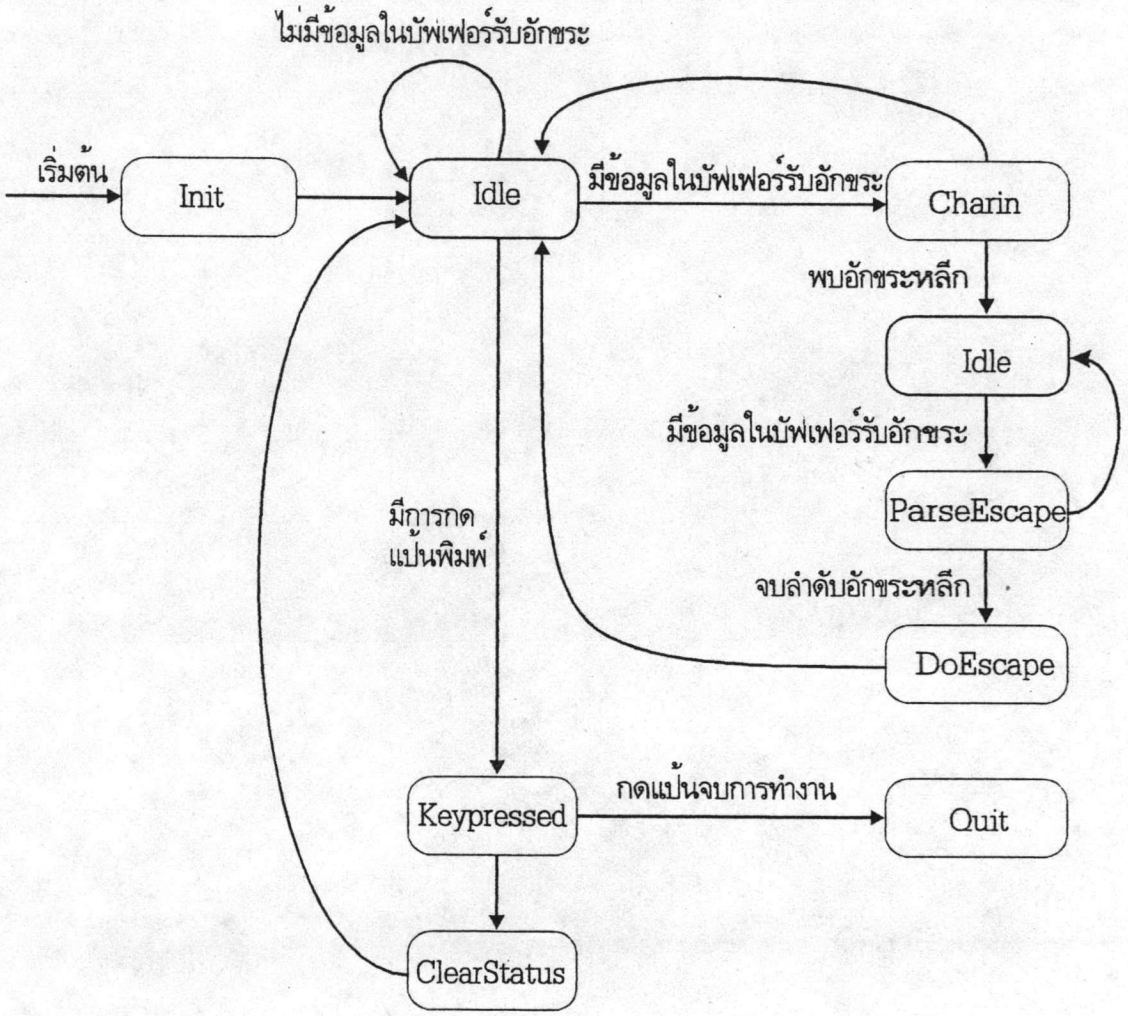
เมื่อได้รับข้อความนี้จะตั้งค่าของพอร์ตสื่อสารใหม่และลบล้างตัวแปรต่างๆ บัฟเฟอร์รับอักขระ และบัฟเฟอร์ส่งอักขระ รวมทั้งลบล้างจอภาพแล้วส่งข้อความ MS_Idle เพื่อบอกโปรแกรมให้รอรับการกดแป้นพิมพ์ในวงรอบการทำงานถัดไปจากนั้นจึงเข้าสู่วงรอบการทำงานถัดไป ข้อความนี้จะถูกส่งมาเป็นข้อความแรกก่อนเข้าสู่วงรอบการผ่านข้อความ



รูป 4.1 แสดงกระแสข้อมูลภายในตัวเลียนแบบเทอร์มินัล



รูป 4.2 แสดงข้อความและกระแสข้อความภายในวงรอบการผ่านข้อความ



แผนภาพแสดงการเปลี่ยนสถานะในโปรแกรมหลักของตัวเลียนแบบเทอร์มินัล

รูป 4.3 แผนภาพแสดงการเปลี่ยนสถานะภายในโปรแกรมหลัก

1.2 ข้อความ MS_Idle

มีผลทำให้โปรแกรมคอยรับค่าจากแป้นพิมพ์ เมื่อมีการกดแป้นพิมพ์ใด ๆ โปรแกรมในส่วนนี้จะส่งข้อความ MS_Keypressed แล้วเข้าสู่วงรอบการทำงานถัดไปเพื่อ ในกรณีที่ไม่มีการกดแป้นใด จะตรวจสอบบัฟเฟอร์สำหรับรอรับอักขระ ถ้ามีข้อมูลอยู่ในบัฟเฟอร์จะอ่านข้อมูลนั้นขึ้นมา จากนั้นจะถ่ายทอดข้อความที่สองที่ได้รับมาพร้อมกันออกไป ในกรณีที่บัฟเฟอร์ว่าง โปรแกรมจะส่งข้อความ MS_Idle ซ้ำออกไป ซึ่งจะมีผลทำให้ตัวเลียนแบบเทอร์มินัลหยุดการทำงานและรอคอยการกดแป้นพิมพ์หรือรอให้มีข้อมูลในบัฟเฟอร์สำหรับรอรับข้อมูล

1.3 ข้อความ MS_Charin

เมื่อโปรแกรมได้รับข้อความนี้จะแสดงอักขระบนจอภาพขึ้นมา ในกรณีที่พบว่าอักขระที่มีอยู่เป็น รหัสหลีก (escape code) จะส่งข้อความ MS_Idle เป็นข้อความแรก และ MS_ParseEscape เป็นข้อความที่ 2 โดยที่ข้อความแรกจะทำให้ตัวเลียนแบบเทอร์มินัลตรวจสอบการกดแป้นอักขระถัดไป ส่วนข้อความที่สองเพื่อให้ตีความชุดรหัสหลีกหลังจากตรวจสอบการกดแป้นพิมพ์แล้ว

1.4 ข้อความ MS_Keypressed

ข้อความนี้จะทำให้โปรแกรมในวงรอบการทำงานหลักตรวจสอบอักขระที่รับมาว่าเป็น แป้นสลับ (alternate key) แป้นควบคุม (control key) แป้นฟังก์ชัน (function key) หรือแป้นอักขระอื่น ๆ เมื่อทราบว่าเป็นแป้นอักขระใด โปรแกรมจะทำงานตามเงื่อนไขที่กำหนดในโปรแกรมต่อไป ในกรณีที่เป็นอักขระทั่วไปจะนำอักขระนั้นไปบรรจุลงในบัฟเฟอร์สำหรับเตรียมส่งอักขระ เมื่อโปรแกรมทำงานเสร็จจะส่งข้อความ MS_ClearStatus แล้วเข้าสู่วงรอบการทำงานถัดไป

1.5 ข้อความ MS_ClearStatus

ข้อความนี้มีผลทำให้โปรแกรมอ่านค่า รีจิสเตอร์สถานะของโมเด็ม และรีจิสเตอร์สถานะของสารสื่อสาร (modem & line status register) เพื่อเป็นการลบล้างสถานะของพอร์ตสื่อสาร จากนั้นจะส่งข้อความ MS_Idle

1.6 ข้อความ MS_ParseEscape

ข้อความนี้ทำให้โปรแกรมทำหน้าที่อ่านข้อมูลที่ตามมากับรหัสหลีกเพื่อตีความให้ทราบว่าเป็นชุดรหัสหลีก (escape sequence) ชุดใด ถ้าเป็นชุดรหัสหลีกที่ตัวเลียนแบบเทอร์มินัลให้บริการ โปรแกรมจะส่งข้อความ MS_DoEscape เพื่อทำงานตามชุดรหัสหลีกนั้น ในระหว่างที่กำลังตีความชุดรหัสหลีกอยู่โปรแกรมจะคอยส่งข้อความ MS_Idle เป็นระยะ ๆ เพื่อให้ตัวเลียนแบบเทอร์มินัลสามารถตอบสนองการกดแป้นพิมพ์ของผู้ใช้ได้เสมอ

1.7 ข้อความ MS_DoEscape

ข้อความนี้ทำให้โปรแกรมทำงานตามชุดรหัสหลีกที่พบ เมื่อทำงานเสร็จสิ้นจะส่งข้อความ MS_Idle เพื่อรอรับการกดแป้นพิมพ์ใหม่ พร้อมกับส่งข้อความ MS_Charin

1.8 ข้อความ MS_Quit

เมื่อโปรแกรมได้รับข้อความนี้จะจบวงรอบการทำงานหลักพร้อมกับจบการทำงานของตัวเลียนแบบเทอร์มินัล

2. โปรแกรมรับบริการขัดจังหวะ (interrupt survice routine)

เมื่อเกิดฮาร์ดแวร์อินเตอร์รัปท์ ซีพียูจะผลักค่าแฟลก (flags) เรจีสเตอร์ CS และเรจีสเตอร์ IP เข้าบนแสตก ทำให้ซีพียูอินเตอร์รัปท์แฟลก (if) ไม่ทำงาน (disable) เพื่อให้ซีพียูไม่สนใจสัญญาณขัดจังหวะอื่น จากนั้นอ่านค่าเรจีสเตอร์ CS และเรจีสเตอร์ IP จากตารางอินเตอร์รัปท์เวกเตอร์ (interrupt vector table) แล้วทำงานต่อโดยเริ่มต้นที่จุดตั้งต้นของโปรแกรมรับบริการขัดจังหวะ (interrupt service routine)

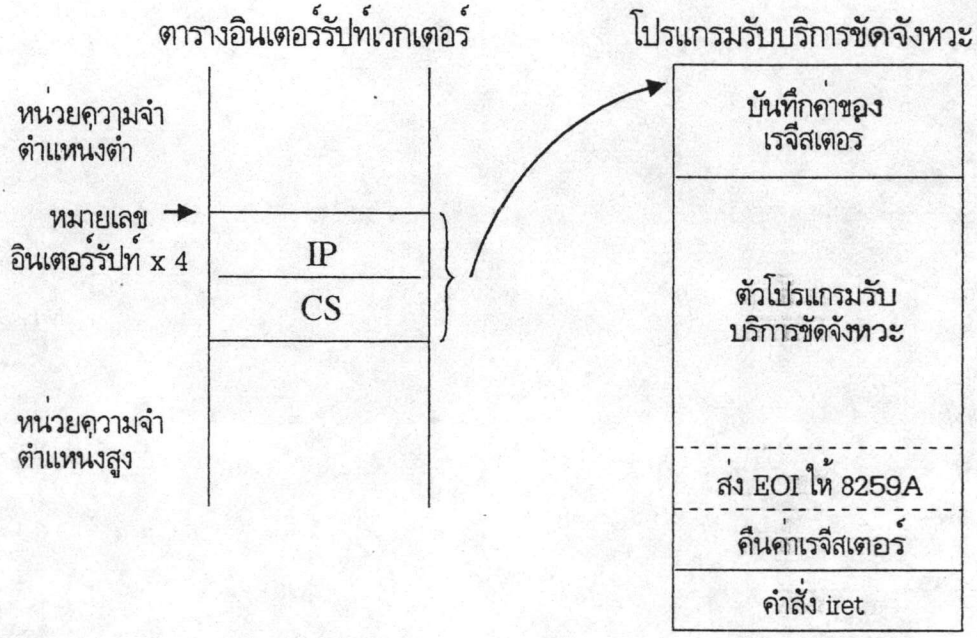
2.1 โปรแกรมรับบริการขัดจังหวะอสมวาร

โปรแกรมรับบริการขัดจังหวะจะบันทึกค่าเรจีสเตอร์ที่จะใช้ในโปรแกรมรับบริการขัดจังหวะทั้งหมดและจะทำงานต่าง ๆ ให้น้อยที่สุดเท่าที่จำเป็นเพื่อให้ซีพียูกลับไปอยู่ในสถานะที่ยอมให้มีสัญญาณขัดจังหวะเกิดขึ้นได้ นอกจากนี้โปรแกรมรับบริการขัดจังหวะอาจจะมีการกระทำบางอย่างเพื่อตั้งการควบคุมการขัดจังหวะใหม่ (reset interrupt control mechanisms) เช่นการอ่านค่าจากพอร์ต ของชิป 8250 ซึ่งทำหน้าที่เป็นยูอาร์ต (UART, Universal Asynchronous Receiver Transmitter) เพื่อให้ทราบชนิดของสัญญาณขัดจังหวะที่เกิดขึ้นจากชิป 8250 ในทำนองเดียวกันชิป 8259A ซึ่ง ทำหน้าที่เป็นตัวควบคุมการขัด

จังหวะ (interrupt controller) ในกรณีที่เป็นฮาร์ดแวร์อินเตอร์รัปท์ จำเป็นจะต้องออกคำสั่งจบการขัดจังหวะ (End Of Interrupt command, EOI) ให้กับพอร์ต 0x20 ของชิป 8259A สุดท้ายโปรแกรมรับบริการขัดจังหวะออกคำสั่งยอมให้ซีพียูรับสัญญาณขัดจังหวะอื่นได้แล้ว คืนค่าเรจีสเตอร์ที่บันทึกไว้กลับและทำคำสั่ง iret ซึ่งจะดึงค่าของเรจีสเตอร์ IP เรจีสเตอร์ CS และซีพียูแฟลกออกจากด้านบนของแสตก กระบวนการนี้ทำให้ซีพียูกลับไปทำงานในตำแหน่งที่ถูกต้องก่อนเกิดการขัดจังหวะขึ้น รูปแบบของโปรแกรมรับบริการขัดจังหวะแสดงในรูป 4.5

2.2 สาเหตุการเกิดการขัดจังหวะอสมวาร

การขัดจังหวะอสมวาร (asynchronous interrupt) เกิดขึ้นได้หลายชนิดซึ่งจะสามารถทราบชนิดของการขัดจังหวะที่เกิดขึ้นได้ โดยดูจากค่าของเรจีสเตอร์ระบุการขัดจังหวะ (Interrupt Identification Register) ค่าเหล่านี้แสดงในตาราง 4.1 เมื่อมีค่าเป็น 6 หมายถึงมีข้อผิดพลาดหรือมีการหยุดกระทันหัน ซึ่งอาจเกิดมาจากมีการเขียนทับ (overrun) หรือภาวะเสริมผิดพลาด (parity error) หรือ เฟรมข้อมูลผิดพลาด (framing error) หรือมีการหยุดกระทันหัน (break interrupt) จะลบล้างค่านี้โดยอ่านค่าเรจีสเตอร์สถานะสายสื่อสาร (LSR) ค่าสถานะสายสื่อสารจะเป็นตัวบอกชนิดของข้อผิดพลาดที่เกิดขึ้นดังแสดงในตาราง 4.2 การขัดจังหวะโดยสาเหตุนี้มีลำดับความสำคัญสูงสุดนั่นหมายถึงเมื่อเกิดการขัดจังหวะพร้อมกันกับการขัดจังหวะอสมวารอื่น8250 จะบอกเฉพาะค่านี้เท่านั้น เมื่อมีค่าเป็น 4 หมายถึงมีข้อมูลในเรจีสเตอร์บัฟเฟอร์ตัวรับ (RBR) การลบล้างค่าของเรจีสเตอร์ทำได้โดยการอ่านค่าจากเรจีสเตอร์บัฟเฟอร์ตัวรับ (RBR) การขัดจังหวะเนื่องจากสาเหตุนี้มีความสำคัญเป็นลำดับที่สอง เมื่อมีค่าเป็น 2 หมายถึงเรจีสเตอร์เก็บข้อมูลในการสื่อสาร (Transmitter Holding Register, THR) ว่าง จะลบค่านี้ได้โดยการอ่านหรือเขียนข้อมูลในเรจีสเตอร์เก็บข้อมูลในการสื่อสาร (THR) การขัดจังหวะเนื่องจากสาเหตุนี้มีความสำคัญในลำดับที่สาม เมื่อมีค่าเป็น 0 หมายถึงมีการเปลี่ยนสถานะของโมเด็มเนื่องมาจากชุดข้อมูลพร้อม(Data Set Ready, DSR) หรือพอร์ตว่างพร้อมส่งข้อมูล (Clear to Send, CTS) หรือตัวชี้บอกกริ่ง(Ring Indicator) หรือมาจากได้รับสายสัญญาณตรง (Received Line Signal Direct) จะลบล้างค่านี้ได้โดยการอ่านค่าเรจีสเตอร์สถานะโมเด็ม (Modem Status Register, MSR) ความหมายของบิตของเรจีสเตอร์สถานะโมเด็มแสดงในตาราง 4.3



รูป 4.5 แสดงรูปแบบของโปรแกรมรับบริการขัดจังหวะ

2.3 การทำงานของโปรแกรมรับบริการขัดจังหวะอสมวาร

โปรแกรมรับบริการขัดจังหวะอสมวารแสดงในรูป 4.6 โดยเริ่มจากหยุดการเกิดการขัดจังหวะทั้งหมดเพื่อออกคำสั่งหยุดเฉพาะการทำงานของการขัดจังหวะอสมวารเท่านั้นจากนั้นปล่อยให้สามารถเกิดการขัดจังหวะอื่นขึ้นได้ ต่อมาหาชนิดของการขัดจังหวะที่ทำให้โปรแกรมนี้ทำงานโดยอ่านค่าจากเรจีสเตอร์ระบุการขัดจังหวะ (IIR) แล้วแยกทำงานตามกรณีต่างๆ ที่เกิดขึ้น ได้แก่กรณีเมื่อเกิดข้อผิดพลาดหรือการหยุดกระทันหันโปรแกรมจะนับจำนวนครั้งที่ผิดบันทึกไว้ในตัวแปรส่วนกลางเพื่อให้ตัวเลียนแบบเทอร์มินัลสามารถทราบชนิดและจำนวนครั้งของข้อผิดพลาดที่เกิดขึ้นเมื่อตัวเลียนแบบเทอร์มินัลกลับมาทำงานตามปกติ กรณีมีข้อมูลในเรจีสเตอร์บัฟเฟอร์ตัวรับ (RBR) จะอ่านค่าจากเรจีสเตอร์บัฟเฟอร์ตัวรับมาเก็บไว้ในบัฟเฟอร์รับอักขระ กรณีเรจีสเตอร์เก็บค่าสำหรับสื่อสาร (THR) ว่างจะตรวจดูบัฟเฟอร์ส่งอักขระถ้ามีข้อมูลจะออกคำสั่งควบคุมโมเด็มให้ส่งสัญญาณว่าข้อมูลปลายทางพร้อม (Data Terminal Ready, DTR) และให้ส่งสัญญาณว่าโมเด็มว่างพร้อมที่จะส่งอักขระ (Clear to Send, CTS) โดยการเขียนคำสั่งลงไปที่เรจีสเตอร์ควบคุมโมเด็ม (MCR) คำสั่งที่ใช้ควบคุมโมเด็มแสดงในตาราง 4.4 จากนั้นคอยอ่านค่าเรจีสเตอร์สถานะโมเด็มดูสถานะของ RS232-C ว่าข้อมูลปลายทางพร้อม และว่างพร้อมจะส่งอักขระจึงจะอ่านอักขระจากบัฟเฟอร์ส่งอักขระหนึ่งอักขระไปเขียนลงเรจีสเตอร์เก็บค่าสำหรับสื่อสาร(THR) ถ้าข้อมูลปลายทางพร้อมไม่พร้อมหรือไม่ว่างที่จะส่งอักขระโปรแกรมจะพยายามอ่านค่าสถานะโมเด็มใหม่ซ้ำอีกสามครั้ง สุดท้ายก่อนจบโปรแกรมรับบริการขัดจังหวะหยุดการเกิดการขัดจังหวะทั้งหมดเพื่อออกคำสั่งจบการขัดจังหวะให้แก่8259A โดยนำคำสั่ง EOI ซึ่งมีค่าเป็นรหัส 20h ไปเขียนลงในตำแหน่ง 20h แล้วสั่งให้ปล่อยการขัดจังหวะให้เป็นไปตามปกติ โปรแกรมรับบริการขัดจังหวะในรูป 4.6 นี้เขียนด้วยภาษาซีและระบุชนิดของโปรแกรมย่อยเป็นอินเตอร์รัปท์ โดยมีคำว่า interrupt ไว้หน้าชื่อโปรแกรมย่อยซึ่งจะทำให้ตัวแปรภาษาซีแทรกคำสั่งเก็บค่าเรจีสเตอร์ทุกตัวลงแสตกก่อนทำคำสั่งใดในโปรแกรมรับบริการขัดจังหวะเมื่อจบโปรแกรมจะคืนค่าดังกล่าวให้เรจีสเตอร์ จากนั้นจะใช้คำสั่ง iret เพื่อคืนการทำงานมาที่โปรแกรมหลักแทนการใช้คำสั่ง ret ตามปกติ

```
void interrupt asynch_intrpt(void)
{
    BYTE iir;  //--- เรจีสเตอร์ระบุการขัดจังหวะ
               //    (Interrupt Identification Register)
    BYTE rbr;  //--- เรจีสเตอร์บัฟเฟอร์ตัวรับ (Receiver Buffer Register,
    BYTE lsr;  //--- เรจีสเตอร์สถานะสายสื่อสาร (Line Status Register,
    BYTE imr;  //--- เรจีสเตอร์อินเตอร์รัปท์มาสก์ (Interrup Mask Register,
               //--- พอร์ต 0x21)
    BYTE msr;  //--- เรจีสเตอร์สถานะโมเด็ม (Modem Status Register,
    BYTE mcr;  //--- เรจีสเตอร์ควบคุมโมเด็ม (Modem Control Register)
    BYTE c;
    WORD  timeout;
    int i;

    disable();
    disable_asynch();   //--- ป้องกันการเกิดการขัดจังหวะอสมวารซ้อน ซึ่งจะทำ
                        //    ให้ข้อมูลในแสตกเสียหาย
    enable();           //--- ยอมให้การขัดจังหวะอื่นชนิดอื่นเกิดขึ้นได้
    iir = inportb(IOBASE+IIR);  //--- หาชนิดของการขัดจังหวะโดยที่
    switch (iir) {
        case 0x06:    //--- กรณีเกิดข้อผิดพลาดหรือหยุดกระทันหัน
            ++interr;
            lsr = inportb(IOBASE+LSR);  //--- อ่านชนิดข้อผิดพลาดจาก LSR
            if (lsr & 0x02) ++overerr;  //--- นับจำนวนครั้งที่มีการเขียนทับ
            if (lsr & 0x04) ++parityerr;
```

รูป 4.6 แสดงโปรแกรมย่อยภาษาซีที่ทำหน้าที่เป็นโปรแกรมรับบริการขัดจังหวะอสมวาร

```
        if (lsr & 0x08) ++framingerr;
        if (lsr & 0x10) ++brkint;    //--- นับจำนวนครั้งที่หยุดกระทันหัน
        break;
    case 0x04:       //--- กรณีมีข้อมูลในเรจีสเตอร์บัฟเฟอร์ตัวรับ
        zero_dlab();
        rbr = inportb(IOBASE+RBR);  //--- อ่านข้อมูลจาก RBR
        puta(rbr);  //--- นำไปใส่ในบัฟเฟอร์รอรับอักขระ
        break;
    case 0x02:       //--- กรณี THR ว่างพร้อมที่จะส่งอักขระได้
        thrfree = 1;
        if (counto>0) {
            //--- เขียน MCR เพื่อให้เป็น DTR (-0010) และ RTS (-0001)
            outportb(IOBASE+MCR,0x0B);
            //--- อ่านสถานะโมเด็มเพื่อยืนยันว่า 8250 พร้อมที่จะรับข้อมูล นั่นคือ
            //    มีสัญญาณ DSR และ CTS ส่งมาจากโมเด็ม โดยจะพยายามอ่าน
            //    ซ้ำสามครั้ง
            for (timeout=3; timeout > 0; --timeout) {
                msr = inportb(IOBASE+MSR);
                if (msr & 0x30) {  //--- เมื่อมีสัญญาณ DSR และ CTS
                    c = nexto();    //--- อ่านข้อมูลจากบัฟเฟอร์ส่งอักขระ
                    outportb(IOBASE+THR,c); //--- เขียนลง THR
                    break;
                }
            }
        }
```

รูป 4.6 แสดงโปรแกรมย่อยภาษาซีที่ทำหน้าที่เป็นโปรแกรมรับบริการขัดจังหวะอสมวาร (ต่อ)

```
            break;
        case 0x00:      //--- กรณีสถานะโมเด็มเปลี่ยน
            msr = inportb(IOBASE+MSR);
            if (msr & 1)    //--- ถ้าเปลี่ยนสถานะ Clear To Send (CTS)
                cts = (msr & ctsmask) ? 1 : 0;
            if (msr & 2)    //--- ถ้าเปลี่ยนสถานะ Data Set Ready (DSR)
                dsr = (msr & dsrmask) ? 1 : 0;
            break;
    }
    disable();
    outportb(0x20,EOI); //--- ส่งคำสั่งจบการขัดจังหวะให้แก่พอร์ต 0x20
    enable_asynch();    //--- ยอมให้เกิดสัญญาณขัดจังหวะได้เมื่อกลับไปทำงานเดิม
    enable();
}
```

รูป 4.6 แสดงโปรแกรมย่อยภาษาซีที่ทำหน้าที่เป็นโปรแกรมรับบริการขัดจังหวะอสมวาร (ต่อ)

บิต	หน้าที่	ค่า	ความหมาย
0	Interrupt Pending	0 1	การขัดจังหวะที่เกิดจาก 8250 ถูกพักไว้ ไม่มีการพักการขัดจังหวะใด
1-2	Interrup ID	11 10 01 00	มีข้อผิดพลาดหรือมีการหยุดกระทันหัน มีข้อมูลที่รับเข้ามา THR เริ่มว่าง สถานะโมเด็มเปลี่ยน (CTS,DSR,RI,หรือ RLSD)
3-7		0000	บิตเหล่านี้มีค่าเป็นศูนย์เสมอ

ตาราง 4.1 แสดงความหมายของบิตของเรจีสเตอร์ระบุการขัดจังหวะ (IIR)

บิต	หน้าที่	ค่า	ความหมาย	การลบล้างค่า
0	ข้อมูลพร้อม	0 1	ข้อมูลใน RBR ไม่ครบ ข้อมูลใน RBR ครบ	อ่านหรือเขียนค่า กับ RBR หรือ LSR
1	รับข้อมูลไม่ทัน	0 1	ไม่เกิดการรับข้อมูลไม่ทัน รับข้อมูลไม่ทัน	อ่าน LSR
2	ภาวะเสริมผิดพลาด	0 1	ไม่เกิดภาวะเสริมผิดพลาด เกิดภาวะเสริมผิดพลาด	อ่าน LSR
3	เฟรมผิดพลาด	0 1	ไม่เกิดเฟรมผิดพลาด เกิดเฟรมผิดพลาด	อ่าน LSR
4	หยุดกระทันหัน	0 1	ไม่มีการหยุดกระทันหัน มีการหยุดกระทันหัน	อ่าน LSR
5	THR ว่าง	0 1	มีข้อมูลใน THR ไม่มีข้อมูลใน THR	เมื่อมีอักขระใหม่เข้า มาที่ THR
6	TSR ว่าง	0 1	มีข้อมูลใน TSR ไม่มีข้อมูลใน TSR	เมื่ออักขระถูกย้ายจาก THR ไปที่ TSR
7	-	0	บิตนี้จะมีค่าเป็นศูนย์เสมอ	

ตาราง 4.2 แสดงความหมายของบิตของเรจีสเตอร์สถานะสายสื่อสาร (LSR)

บิต	หน้าที่	ค่า	ความหมาย
0	DCTS	0	ไม่เปลี่ยนค่าของสัญญาณ CTS
		1	สัญญาณ CTS ถูกเปลี่ยนนับตั้งแต่ MSR ถูกอ่าน
1	DDSR	0	ไม่มีการเปลี่ยนค่าของสัญญาณ DSR
		1	สัญญาณ DSR ถูกเปลี่ยนนับตั้งแต่มีการอ่าน MSR
2	TERI	0	ไม่มีการเปลี่ยนค่าของสัญญาณ RI
		1	RI ถูกเปลี่ยนจากเปิดเป็นปิดนับตั้งแต่ MSR ถูกอ่าน
3	DRLSD	0	ไม่มีการเปลี่ยนค่าของสัญญาณ RLSD
		1	RLSD ถูกเปลี่ยนนับตั้งแต่ MSR ถูกอ่าน
4	CTS	0	ไม่ว่างที่จะส่งอักขระไปโมเด็ม
		1	อักขระอาจสามารถส่งไปโมเด็มได้
5	DSR	0	โมเด็มไม่อยู่ในสถานะพร้อม
		1	โมเด็มพร้อม
6	RI	0	ไม่พบว่าได้รับสัญญาณกริ่ง
		1	พบว่าได้รับสัญญาณกริ่ง
7	RLSD	0	ไม่ได้รับสัญญาณสาย (line signal) จากโมเด็ม
		1	พบว่าได้รับสัญญาณสายจากโมเด็ม

ตาราง 4.3 แสดงความหมายของบิตของเรจีสเตอร์สถานะโมเด็ม (MSR)

บิต	หน้าที่	ค่า	ความหมาย
0	DTR	0 1	ส่งสัญญาณบอกโมเด็มว่าข้อมูลปลายทางไม่พร้อม ส่งสัญญาณบอกโมเด็มว่าข้อมูลปลายทางพร้อม
1	RTS	0 1	บอกโมเด็มว่าพีซีไม่ร้องขอให้ส่งอักขระ ถามโมเด็มว่าว่างพร้อมที่จะส่งอักขระหรือไม่
2	OUT1		ไม่ใช้
3	OUT2	0 1	ไม่ใช้สายสัญญาณขัดจังหวะจาก 8250 ไป 8259A ใช้สายสัญญาณขัดจังหวะจาก 8250 ไป 8259A
4	LOOP	0 1	8250 จะไม่ทดสอบตนเอง 8250 จะทดสอบตนเองโดยลองส่งและรับข้อมูลทันที
5-7		000	บิตเหล่านี้เป็นศูนย์เสมอ

ตาราง 4.4 แสดงความหมายของบิตของเรจีสเตอร์ควบคุมโมเด็ม (MCR)

3. การเริ่มต้น (initializing) โปรแกรมรับบริการขัดจังหวะอสมวาร

ในส่วนนี้จะกล่าวถึงการเริ่มต้นโปรแกรมรับบริการขัดจังหวะอสมวารเพื่อให้โปรแกรม ทำงานได้อย่างถูกต้องและสมบูรณ์จะต้องดำเนินการสามประการคือ การติดตั้งโปรแกรมรับบริการขัดจังหวะโดยกำหนดตำแหน่งของโปรแกรมรับบริการขัดจังหวะในตารางอินเตอร์รัปท์เวกเตอร์ การเริ่มต้นพอร์ต สื่อสาร และตั้งค่าตัวแปรส่วนกลาง (globla variable) ต่าง ๆ ที่ใช้ในการสื่อสารระหว่างโปรแกรมรับบริการขัดจังหวะกับโปรแกรมหลักของตัวเลียนแบบเทอร์มินัล

3.1 ตารางอินเตอร์รัปท์เวกเตอร์

ตารางอินเตอร์รัปท์เวกเตอร์เป็นตารางที่เก็บตำแหน่งของโปรแกรมรับบริการขัดจังหวะทั้งหลายของเครื่องคอมพิวเตอร์แบบไอบีเอ็มพีซี ตำแหน่งดังกล่าวมีขนาดสี่ไบต์โดยสองไบต์แรกเป็นตำแหน่งออฟเซตหรือค่าสำหรับเรจีสเตอร์ IP และสองไบต์ถัดมาเป็นตำแหน่งเซกเมนต์หรือค่าสำหรับเรจีสเตอร์ CS ตารางนี้สามารถเก็บตำแหน่งของโปรแกรมรับบริการขัดจังหวะได้ทั้งสิ้น 250 ตำแหน่งโดยเริ่มจากตำแหน่ง 0000:0000h จนถึงตำแหน่ง 0000:0400h

3.2 การติดตั้งโปรแกรมรับบริการขัดจังหวะ

การติดตั้งโปรแกรมรับบริการขัดจังหวะใช้โปรแกรม install โดยมีพารามิเตอร์สองตัวโดยตัวแรกจะบอกหมายเลขอินเตอร์รัปท์ที่จะติดตั้งในตารางอินเตอร์รัปท์เวกเตอร์ และตัวที่สองเป็นตำแหน่งของโปรแกรมรับบริการขัดจังหวะ เมื่อเขียนโดยใช้ภาษาซีจะได้โปรแกรมย่อยดังนี้ มีการทำงานโดยจะหาตำแหน่งของโปรแกรมรับบริการขัดจังหวะที่ติดตั้งอยู่ก่อนโดยใช้คำสั่ง getvect และบันทึกตำแหน่งนั้นไว้ที่ตัวแปรส่วนกลางชื่อ oldvec ซึ่งประกาศไว้เป็นชนิด interrupt เช่นเดียวกับโปรแกรมย่อย asynch_intrpt จากนั้นจะนำตำแหน่งของโปรแกรมรับบริการขัดจังหวะที่จะติดตั้งไปเปลี่ยนลงในตารางอินเตอร์รัปท์เวกเตอร์ โดยใช้คำสั่ง setvect ในภาษาซี

```
void interrupt (*oldvec)();

void install(int intrno, void interrupt (*service)())
{
        oldvec = getvect(intrno);
        setvect(intrno,service);
}
```

ในการเรียกใช้โปรแกรมติดตั้งโปรแกรมรับบริการขัดจังหวะทำได้โดยใช้คำสั่ง install เนื่องจากพอร์ตสื่อสารของเครื่องคอมพิวเตอร์แบบไอบีเอ็มพีซีตามปกติจะมีพอร์ตสื่อสาร 2 พอร์ตคือ พอร์ต COM1 และ COM2 ในการติดตั้งโปรแกรมรับบริการขัดจังหวะของแต่ละพอร์ตจะติดตั้งในตารางอินเตอร์รัปท์เวกเตอร์ต่างกันโดยCOM1 จะติดตั้งที่อินเตอร์รัปท์หมายเลข 0Ch และ COM1 จะติดตั้งที่อินเตอร์รัปท์หมายเลข 0Bh ดังนั้นการติดตั้งโปรแกรมรับบริการขัดจังหวะสำหรับพอร์ต COM1 เป็นดังนี้

```
        install(0x0C,asynch_intrpt);
```

และสำหรับพอร์ต COM2 จะเป็น

```
        install(0x0B,asynch_intrpt);
```

3.3 การยอมให้ใช้การขัดจังหวะ

ตามปกติโปรแกรมแม้จะมีการติดตั้งโปรแกรมรับบริการขัดจังหวะไปแล้วก็ตามจำเป็นจะต้องสั่งให้ 8250 ยอมให้เกิดสัญญาณขัดจังหวะได้เสียก่อนโปรแกรมรับบริการขัดจังหวะจึงจะเริ่มต้นทำงานได้ ในการนี้ทำได้โดยเลือกชนิดของการขัดจังหวะที่ต้องการใช้แล้วออกคำสั่งให้แก่เรจีสเตอร์เลือกการขัดจังหวะ (IER) ความหมายของการกำหนดค่าของบิตของเรจีสเตอร์เลือกการขัดจังหวะแสดงในตาราง 4.5

บิต	หน้าที่	ค่า	ความหมาย
0	Enable Data Available	0 1	ไม่มีการขัดจังหวะเมื่อได้รับข้อมูล มีการขัดจังหวะเมื่อข้อมูลถูกย้ายเข้ามาใน RBR
1	Enable THR Empty	0 1	ไม่มีการขัดจังหวะเมื่อ THR เริ่มว่าง มีการขัดจังหวะเมื่อ THR เริ่มว่าง
2	Enable Receive Line Status	0 1	ไม่มีการขัดจังหวะเมื่อสถานะสายสื่อสารเปลี่ยน มีการขัดจังหวะเกิดขึ้นเมื่อสถานะสายสื่อสารเปลี่ยน
3	Enable Modem Status	0 1	ไม่มีการขัดจังหวะเมื่อสถานะโมเด็มเปลี่ยน มีการขัดจังหวะเมื่อสถานะโมเด็มเปลี่ยน
4-7		0000	บิตเหล่านี้เป็นศูนย์เสมอ

ตาราง 4.5 แสดงความหมายของบิตของเรจีสเตอร์เลือกการขัดจังหวะ (IER)

ในงานวิจัยนี้ยอมให้เกิดการขัดจังหวะได้ทุกกรณี ได้แก่การมีอักขระมาที่เรจีสเตอร์บัฟเฟอร์ตัวรับ (RBR) เมื่อเรจีสเตอร์เก็บค่าสำหรับสื่อสารเริ่มว่าง (THR) เมื่อสถานะโมเด็มเปลี่ยนและเมื่อมีข้อผิดพลาดหรือสถานะสายสื่อสารเปลี่ยน ดังนั้นจึงกำหนดให้เรจีสเตอร์เลือกการขัดจังหวะมีค่าเป็น 0Fh ในการทำงานเมื่อเกิดการขัดจังหวะเนื่องจากมีอักขระมาที่เรจีสเตอร์บัฟเฟอร์ตัวรับแล้วหลังจากนั้นจะไม่มีการขัดจังหวะเนื่องจากเรจีสเตอร์เก็บค่าสำหรับสื่อสารเริ่มว่างเกิดขึ้นอีกดังนั้นจึงต้องมีการกำหนดค่าสำหรับเรจีสเตอร์เลือกการขัดจังหวะใหม่ทุกครั้งที่มีข้อมูลในบัฟเฟอร์ส่งอักขระเพื่อให้เกิดการขัดจังหวะ เมื่อเรจีสเตอร์เก็บค่าสำหรับสื่อสารเริ่มว่าง

3.4 การคืนการการทำงานให้กับโปรแกรมรับบริการขัดจังหวะเดิม

เมื่อจบการทำงานของตัวเลียนแบบเทอร์มินัล จำเป็นจะต้องคืนตำแหน่งของโปรแกรมรับบริการขัดจังหวะเดิมก่อนเริ่มตัวเลียนแบบเทอร์มินัลมิฉะนั้นจะเกิดผลที่ไม่สามารถคาดล่วงหน้าได้ เพราะเมื่อจบตัวเลียนแบบเทอร์มินัลแล้วจะยกเลิกการใช้งานตำแหน่งทุกตำแหน่งในโปรแกรมรวมทั้งโปรแกรมย่อยที่ทำหน้าที่รับบริการขัดจังหวะด้วยทำให้เมื่อมีสัญญาณขัดจังหวะเกิดขึ้น คอมพิวเตอร์จะพยายามนำคำสั่งที่ยังเหลือในตำแหน่งที่ยกเลิกแล้วนั้นไปทำงานซึ่งไม่ถูกต้อง การคืนตำแหน่งของโปรแกรมรับบริการขัดจังหวะเดิมทำโดยใช้คำสั่งภาษาซี setvect เพื่อนำค่าตัวแปรส่วนกลาง oldvec ที่เก็บไว้เมื่อตอนติดตั้งโปรแกรมรับบริการขัดจังหวะมาคืนสำหรับ COM1 ดังนี้

```
setvect(0x0C,oldvec);
```

และสำหรับ COM2 ใช้คำสั่ง

```
setvect(0x0B,oldvec);
```

4. การเริ่มต้นพอร์ตสื่อสาร

ก่อนการใช้งาน 8250 จำเป็นจะต้องเริ่มต้นด้วยการตั้งโปรแกรมต่าง ๆ ให้กับชิป 8250 ได้แก่การกำหนดอัตราบอด (baud rate) ชนิดของภาวะเสริม (parity) จำนวนบิตหยุด (stop bit) ที่ท้ายของแต่ละอักขระ และ จำนวนบิตต่ออักขระ ในการเริ่มต้นพอร์ตสื่อสารนี้สามารถทำได้สองวิธีคือการใช้บริการของไบออสโดยเรียกใช้อินเตอร์รัปท์ 14 และวิธีที่สองโดยการโปรแกรมเรจีสเตอร์ของชิป 8250 ในงานวิจัยนี้เลือกใช้วิธีที่สองเพราะเป็นวิธีที่สามารถโปรแกรม 8250 ได้โดยตรงจึงสามารถสั่งงานพอร์ตสื่อสารได้เต็มที่เช่นสามารถกำหนดอัตราบอดได้สูงถึง 19200 เป็นต้น

4.1 การกำหนดตำแหน่งเริ่มต้นของพอร์ตสื่อสาร

เนื่องจากโดยทั่วไปมีพอร์ตสื่อสารอยู่สองพอร์ตคือ COM1 และ COM2 ตำแหน่งของเรจีสเตอร์ภายของพอร์ตสื่อสารทั้งสองจะมีตำแหน่งเริ่มต้นหรือตำแหน่งฐาน (base address) ของพอร์ตสื่อสารต่างกันแต่มีออฟเซ็ตหรือตำแหน่งนับจากตำแหน่งเริ่มต้นของพอร์ตสื่อสารเหมือนกันเมื่อเป็นเรจีสเตอร์ชื่อเดียวกัน ตำแหน่งฐานสำหรับพอร์ตแสดงในตาราง 4.6 ซึ่งสามารถอ่านได้จากตารางเก็บข้อมูลของไบออสเริ่มจากตำแหน่ง 0040:0000h และค่าออฟเซ็ตของเรจีสเตอร์กำหนดในตาราง 4.7 ตำแหน่งเริ่มต้นของพอร์ตสื่อสารในสำหรับคอมในการหาค่าตำแหน่งของเรจีสเตอร์ใดทำได้โดยนำค่าฐานของพอร์ตนั้นบวกกับค่าออฟเซ็ต ตัวอย่างเช่นการหาตำแหน่งของเรจีสเตอร์สถานะโมเด็มของพอร์ต COM1 จะได้ 3FEh ซึ่งมาจากค่าฐานคือ 3F8h บวกกับค่าออฟเซ็ตคือ 6

ชื่อพอร์ต	ตำแหน่งที่เก็บค่าในไบออส	ค่าตำแหน่งเริ่มต้น
COM1:	0040:0000h	3F8h
COM2:	0040:0002h	2F8h
COM3:	0040:0004h	3E8h
COM4:	0040:0006h	2E8h

ตาราง 4.6 แสดงตำแหน่งเริ่มต้นของพอร์ตสื่อสารสำหรับเครื่องคอมพิวเตอร์แบบพีซี

ตำแหน่งนับจากตำแหน่งเริ่มต้นของพอร์ตสื่อสาร	ค่า DLAB	เรจีสเตอร์ของ 8250
0	0	เรจีสเตอร์เก็บค่าสำหรับสื่อสาร (Transmitter Holding Register, THR)
0	0	เรจีสเตอร์บัฟเฟอร์ตัวรับ (Receiver Buffer Register, RBR)
0	1	เรจีสเตอร์เก็บบิตตัวหารที่มีความสำคัญน้อยสุด (Devisor Latch LSB, DLL)
1	1	เรจีสเตอร์เก็บบิตตัวหารที่มีความสำคัญมากสุด (Devisor Latch MSB, DLM)
1	0	เรจีสเตอร์เลือกการขัดจังหวะ (Interrupt Enable Register, IER)
2		เรจีสเตอร์ระบุการขัดจังหวะ (Interrupt Identification Register, IIR)
3		เรจีสเตอร์ควบคุมสายสื่อสาร (Line Control Register, LCR)
4		เรจีสเตอร์ควบคุมโมเด็ม (Modem Control Register, MCR)
5		เรจีสเตอร์สถานะสายสื่อสาร (Line Status Register, LSR)
6		เรจีสเตอร์สถานะโมเด็ม (Modem Status Register, THR)

ตาราง 4.7 แสดงตำแหน่งเรจีสเตอร์ของพอร์ตสื่อสารสำหรับชิป 8250

4.2 การคำนวณหาค่าสำหรับเรจีสเตอร์ตัวหาร

ค่าเรจีสเตอร์ตัวหารเป็นค่าที่ใช้เพื่อกำหนดอัตราที่ 8250 ใช้ส่งข้อมูล โดยค่าดังกล่าวเป็นตัวเลขที่ใช้หารกับความถี่ของสัญญาณนาฬิกาของ 8250 ซึ่งมีค่า 1.8432 เมกะเฮิรตซ์ เพื่อให้ได้ผลลัพธ์เป็น 16 เท่าของอัตราบอด ดังนั้นค่าตัวหารดังกล่าวจึงใช้สูตรคำนวณดังนี้

$$\text{ตัวหาร} = \frac{1843200}{16 \times \text{อัตราบอด}}$$

ดังนั้นค่าตัวหารเมื่ออัตราบอดที่ต้องการเป็น 2400 จะคำนวณได้จากดังนี้

$$\text{ตัวหาร} = \frac{1843200}{16 \times 2400}$$

$$= \frac{115200}{2400}$$

$$= 48$$

ดังนั้นตัวหารจึงมีค่าเป็น 48 หรือเป็น 30h ตาราง 4.8 จะแสดงค่าตัวหารสำหรับหาอัตราบอดต่าง ๆ ค่าเหล่านี้จะใช้เพื่อกำหนดอัตราบอดของพอร์ตสื่อสาร

อัตราบอด	ตัวหาร		คู่เรจีสเตอร์ตัวหาร	
	ฐานสิบ	ฐานสิบหก	DLM	DLL
110	1047	0417h	04h	17h
150	768	0300h	03h	00h
300	384	0180h	01h	80h
600	192	00C0h	00h	C0h
1200	96	0060h	00h	60h
2400	48	0030h	00h	30h
4800	24	0018h	00h	18h
9600	12	000Ch	00h	0Ch
19200	6	0006h	00h	06h
38400	3	0003h	00h	03h

ตาราง 4.8 แสดงค่าตัวหารสำหรับหาอัตราบอด (baud rate divisors)

4.3 โปรแกรมเริ่มต้นพอร์ตสื่อสาร (port_init)

โปรแกรมเริ่มต้นพอร์ตสื่อสารจะกำหนดตัวแปรแถวลำดับเพื่อเก็บค่าเริ่มต้นสำหรับพอร์ตสื่อสารได้แก่แถวลำดับระบุบิตเสริม (parity) ซึ่งมีค่าเป็น 0 หมายถึงไม่มีบิตเสริม 1 หมายถึงบิตเสริมคี่และ 2 หมายถึงบิตเสริมคู่ แถวลำดับระบุบิตหยุด (stopbit) ซึ่งมีค่าเป็น 0 หมายถึงมี 1 บิตหยุด มีค่าเป็น 1 หมายถึงมี 2 บิตหยุด แถวลำดับระบุความยาวของแต่ละไบต์ มีค่าเป็น 2 หมายถึงมีความยาวเป็น 7 บิต และมีค่าเป็น 3 หมายถึงมีความยาวเป็น 8 บิต แถวลำดับระบุพอร์ตสื่อสารที่ใช้ มีค่าเป็น 0 หมายถึง COM1 และมีค่าเป็น 1 หมายถึง COM2 ตามลำดับ สุดท้ายแถวลำดับระบุค่าเรจีสเตอร์ตัวหารประกอบด้วยแถวลำดับสองชุดคือ dlm และ dll โดยแถวลำดับ dlm เก็บค่าที่มีนัยสำคัญสูงสุดของตัวหาร และแถวลำดับ dll เก็บค่าที่มีนัยสำคัญต่ำสุดของตัวหาร แถวลำดับเหล่านี้จะใช้ค่าตัวเลือกซึ่งอ่านมาจากแฟ้มเก็บตัวเลือกสำหรับพอร์ตสื่อสารที่ถูกอ่านเข้ามาในขณะที่เริ่มตัวเลียนแบบเทอร์มินัลเป็นตัวชี้ค่าในแถวลำดับ ตัวเลือกจะเก็บไว้ในตัวแปร parm ซึ่งเป็นตัวแปรแบบโครงสร้างโดยมีแบบข้อมูลเป็น parmrec ซึ่งแสดงโครงสร้างบางส่วนของแบบข้อมูล parmrec ดังนี้

```
typedef struct {
        .
        .
        .
        int     br;
        int     pr;
        int     sb;
        int     wl;
        int     cp;
        .
        .
        .
} parmrec;
```

จากค่าตัวเลือกซึ่งจะใช้เป็นตัวชี้ในแถวลำดับที่เก็บค่าสำหรับพอร์ตสื่อสาร เราสามารถกำหนดค่าของเริ่มต้นของพอร์ตได้โดยการเขียนลงไปที่เรจีสเตอร์ควบคุมสายสื่อสาร (LCR) โดยการกำหนดค่าของบิตต่างๆ ของ เรจีสเตอร์ควบคุมสายสื่อสารให้ทำงานตามตัวเลือกที่กำหนดไว้ ตาราง 4.9 แสดงความหมายของการกำหนดค่าบิตต่าง ๆ ของเรจีสเตอร์ควบคุมสายสื่อสาร ตัวอย่างการกำหนดค่าของเรจีสเตอร์ควบคุมสายสื่อสารเมื่อต้องการให้มีมีค่าอัตราบอด 2400 มีบิตหยุด 1 บิต บิตเสริมเป็นชนิดคี่ และให้ความยาวของไบต์เป็น 7 บิต จากตาราง 4.9 จะได้ว่า บิตที่หนึ่งและศูนย์จะมีค่าเป็น 10 หรือ 2 ที่ฐานสิบ บิตที่สองมีค่าเป็น 0 บิตที่สามมีค่าเป็น 1 หมายถึงมีการใช้บิตเสริม บิตที่สี่มีค่าเป็น 0 หมายถึงมีภาวะเสริมคี่ บิตที่เจ็ดมีค่าเป็น 1 เสมอเพื่อบอกให้ทราบว่าเรจีสเตอร์ในตำแหน่งออฟเซ็ตที่ศูนย์ตามตาราง 4.7 ทำหน้าที่เก็บบิตตัวหารที่มีความสำคัญน้อยสุด และออฟเซ็ตที่หนึ่งทำหน้าที่เก็บบิตตัวหารที่มีความสำคัญสูงสุด จากตัวอย่างโปรแกรมในรูป 4.7 จึงมีการ "หรือ" ค่าสำหรับสายสื่อสารด้วย dlabmask ซึ่งมีค่าเป็น 80h เพื่อทำให้บิตที่เจ็ดของเรจีสเตอร์ควบคุมสายสื่อสารมีค่าเป็น 1 จากนั้นกำหนดค่าบิตตัวหารตามข้อมูลในตัวแปรแถวลำดับให้แก่เรจีสเตอร์เก็บบิตตัวหารที่มีความสำคัญน้อยที่สุดและเรจีสเตอร์เก็บบิตตัวหารที่มีความสำคัญมากที่สุดตามลำดับ สุดท้ายจะเปลี่ยนค่าบิตที่เจ็ดของเรจีสเตอร์ควบคุมสายสื่อสารให้มีค่าเป็น 0 เพื่อให้เรจีสเตอร์ในตำแหน่งออฟเซ็ตที่ศูนย์ทำหน้าที่เป็นเรจีสเตอร์เก็บค่าสำหรับสื่อสาร (THR)หรือเรจีสเตอร์บัฟเฟอร์ตัวรับ (RBR) และเรจีสเตอร์ในตำแหน่งออฟเซ็ตที่หนึ่งทำหน้าที่เป็นเรจีสเตอร์เลือกการขัดจังหวะ (IER)

```
WORD port_init()
{
    BYTE    bLCR;
    BYTE    pr[3]={0,1,3};     //--- แถวลำดับระบุบิตเสริมเป็น ไม่มี คี่ หรือ คู่
    BYTE    sb[2]={0,1};       //--- แถวลำดับระบุบิตหยุดเป็น 1 หรือ 2 บิตหยุด
    BYTE    wl[2]={2,3};       //--- แถวลำดับระบุความยาวเป็น 7 หรือ 8 บิต
    BYTE    cp[4]={0,1,2,3};   //--- แถวลำดับระบุพอร์ตสื่อสาร 1 2 3 หรือ 4
    //--- แถวลำดับระบุค่าเรจีสเตอร์ตัวหารเมื่ออัตราบอดเป็น 110 150 300 600
    //    1200 2400 4800 9600 หรือ 19200
    BYTE    dlm[9]={0x04,0x03,0x01,0x00,0x00,0x00,0x00,0x00,0x00};
    BYTE    dll[9]={0x17,0x00,0x80,0xC0,0x60,0x30,0x18,0x0C,0x06};

    //--- เลือกพอร์ตสื่อสารและตำแหน่งเริ่มต้นโดยดูจากส่วนเก็บข้อมูลของไบออส
    COMPORT = cp[parm.cp];
    IOBASE  = *(biosdta+COMPORT);
    bLCR = dlabmask | (pr[parm.pr]<<3) | (sb[parm.sb]<<2) | wl[parm.wl];
    outportb(IOBASE+LCR,bLCR);
    //--- ตั้งค่าอัตราบอด
    outportb(IOBASE+DLM,dlm[parm.br]);  //--- ตั้ง DLMSB ---
    outportb(IOBASE+DLL,dll[parm.br]);  //--- ตั้ง DLLSB ---
    //--- หยุดการใช้พอร์ตเพื่อทำหน้าที่เรจีสเตอร์เก็บบิตตัวหาร
    zero_dlab();
    return (0);
}
```

รูป 4.7 แสดงโปรแกรมย่อยที่ใช้ในการเริ่มต้นพอร์ตสื่อสาร

บิต	หน้าที่	ค่าของบิต	ความหมาย
1,0	บอกความยาวคำ	00 01 10 11	5 บิตต่อคำ 6 บิตต่อคำ 7 บิตต่อคำ 8 บิตต่อคำ
2	บอกจำนวนบิตหยุด	0 1	1 บิตหยุด 2 บิตหยุด
3	การใช้ภาวะเสริม	0 1	ไม่ใช้ภาวะเสริม ใช้ภาวะเสริม
4	บอกภาวะเสริมคู่	0 1	เป็นภาวะเสริมคี่ เป็นภาวะเสริมคู่
7	DLAB	0 1	ใช้พอร์ตเป็น THR และ RBR ใช้พอร์ตเป็นเรจีสเตอร์เก็บบิตตัวหาร

ตาราง 4.9 แสดงความหมายของบิตของเรจีสเตอร์ควบคุมสายสื่อสาร (LCR)

5. การจัดการบัฟเฟอร์ และเขตวิกฤติ

5.1 บัฟเฟอร์ (buffer)

บัฟเฟอร์คือที่พักข้อมูล ทำหน้าที่ประสานจังหวะ (synchronize) กับพอร์ตสื่อสาร ในงานวิจัยนี้ใช้บัฟเฟอร์สองชุดเพื่อทำหน้าที่ดังกล่าวโดยชุดแรกทำหน้าที่รอรับข้อมูลที่มาจากพอร์ตสื่อสารเรียกบัฟเฟอร์นี้ว่าบัฟเฟอร์สำหรับรับอักขระ ส่วนบัฟเฟอร์ชุดที่สองทำหน้าที่เก็บข้อมูลที่จะส่งให้แก่พอร์ตสื่อสารเรียกบัฟเฟอร์นี้ว่า บัฟเฟอร์ส่งอักขระโดยใช้ตัวแปรแถวคอยเป็นวง (circular queue) เพื่อทำหน้าที่เป็นบัฟเฟอร์ ฟังก์ชันเกี่ยวข้องกับบัฟเฟอร์ทั้งสองมี 4 ฟังก์ชันได้แก่ nexta, nexto, puta และ puto โดยที่ nexta และ puta จะเกี่ยวข้องกับบัฟเฟอร์รับอักขระโดยที่ฟังก์ชัน nexta จะอ่านอักขระในลำดับแรกของแถวคอยและลบอักขระนั้นออกจากแถวคอย puta จะนำอักขระเข้าไปเพิ่มต่อท้ายแถวคอย ส่วน nexto และ puto จะทำหน้าที่เช่นเดียวกันกับข้างต้นแต่จะกระทำกับบัฟเฟอร์ส่งอักขระ ฟังก์ชัน puto และ nexta จะถูกเรียกใช้ในโปรแกรมหลัก เพราะ puto ทำหน้าที่นำข้อมูลที่รับมาจากแป้นพิมพ์ส่งเข้าไปในบัฟเฟอร์รอที่จะส่งผ่านพอร์ตสื่อสารเช่นเดียวกันกับ nexta ซึ่งโปรแกรมหลักจะเรียกเมื่อต้องการนำข้อมูลที่รับมาจากภายนอกไปแสดงผล ส่วน puta และ nexto จะถูกเรียกใช้โดยโปรแกรมรับบริการขัดจังหวะ โดยที่ puta จะถูกเรียกเมื่อโปรแกรมรับบริการขัดจังหวะได้รับอักขระจากพอร์ตสื่อสาร และ nexto จะถูกเรียกเมื่อโปรแกรมรับบริการขัดจังหวะพร้อมที่จะส่งข้อมูล

5.2 เขตวิกฤติ (critical section)

เนื่องจากโปรแกรมหลักและโปรแกรมรับบริการขัดจังหวะจะมีการทำงานสลับกันไปมาตลอดและทั้งสองโปรแกรมใช้บัฟเฟอร์ร่วมกัน ดังนั้นจึงอาจให้ตัวแปรและข้อมูลของบัฟเฟอร์ผิดพลาด เช่นเมื่อโปรแกรมหลักกำลังเรียกใช้ฟังก์ชัน nexta เพื่ออ่านอักขระออกจากบัฟเฟอร์ ในขณะนั้นโปรแกรมหลักจะทราบว่าในบัฟเฟอร์สำหรับรับอักขระมีข้อมูลอยู่จำนวนหนึ่ง ในขณะนั้นถ้ามีอักขระมาที่พอร์ตสื่อสารทำให้เกิดการขัดจังหวะขึ้น โปรแกรมรับบริการขัดจังหวะจะเข้ามาทำงานแทนโปรแกรมหลัก โดยเรียกใช้ฟังก์ชัน puta เพื่อนำอักขระใหม่เข้าไปเก็บในบัฟเฟอร์แล้วตั้งค่าจำนวนอักขระในบัฟเฟอร์ใหม่ เมื่อจบการทำงานของโปรแกรมรับบริการขัดจังหวะกลับไปสู่โปรแกรมหลักอีกครั้ง โปรแกรมหลักจะไม่ทราบว่าจำนวนอักขระในบัฟเฟอร์รับอักขระขณะนี้ได้เปลี่ยนไปแล้ว เมื่อทำงานโดยใช้ฟังก์ชัน nexta ต่อจนเสร็จจะปรับปรุง

บัฟเฟอร์โดยไม่สนใจอักขระใหม่เพิ่งรับเข้ามาเมื่อมีการขัดจังหวะทำให้อักขระนั้นสูญหาย

เพื่อการป้องกันความสับสนเนื่องจากการใช้งานบัฟเฟอร์ร่วมกันของโปรแกรมทั้งสองจึงไม่ยอมให้เกิดการขัดจังหวะในขั้นในระหว่างการทำงานของ nexta และ puta และระหว่างการทำงานของ nexto และ puto โดยใช้คำสั่ง disable ในตอนเริ่มต้นการทำงานของฟังก์ชัน nexta, puta, nexto และ puto เพื่อห้ามการขัดจังหวะใด ๆ และเพิ่มคำสั่ง enable ก่อนที่จะจบการทำงานของฟังก์ชัน เพื่อยอมให้มีการขัดจังหวะเกิดขึ้นได้หลังจากที่ทำงานในฟังก์ชันเหล่านี้เสร็จสิ้นแล้ว ส่วนของโปรแกรมที่ไม่สามารถยอมให้มีการขัดจังหวะเกิดขึ้นได้นี้เรียกว่าเขตวิกฤติ

6. การจัดระดับการแสดงผลภาษาไทย

อักขระไทยเมื่อแสดงบนจอภาพจะจัดเป็นระดับซ้อนกัน 4 ระดับในหนึ่งบรรทัด โดยระดับแรกเพื่อแสดงวรรณยุกต์และทัณฑฆาต ระดับที่สองสำหรับสระบน ระดับที่ สาม สำหรับพยัญชนะ ตัวเลข และสระข้าง ระดับที่สี่ได้แก่สระล่าง แม้ว่าจะมีจัดวางอักขระถึง 4 ระดับแต่การแสดงข้อความในภาวะกราฟิก (graphics mode) นั้นสามารถวางอักขระซ้อนกันโดยนำข้อมูลแบบตัวอักขระ (font) มากระทำกันทางคณิตศาสตร์โดยใช้ตัวดำเนินการทางตรรกศาสตร์ "หรือ" (logic operator or) ทำให้สามารถจำแนกชนิดของอักขระได้เป็นสองชนิดคือ อักขระที่ไม่มีความกว้าง(zero width character) และอักขระที่มีความกว้าง (non-zero width character)

6.1 ประเภทของอักขระ

6.1.1 อักขระที่ไม่มีความกว้าง

หมายถึงอักขระที่ต้องแสดงในระดับที่ 1 2 และ 4 ได้แก่ วรรณยุกต์ ทัณฑฆาต สระบน และสระล่าง การแสดงอักขระเหล่านี้จะวางทับในตำแหน่งเดียวกับอักขระที่มีอยู่ก่อน โดยใช้ตัวดำเนิกการ "หรือ" กับบิตต่าง ๆ บนจอภาพ เมื่อแสดงอักขระประเภทนี้แล้ว จะไม่มีการเลื่อนตัวชี้ตำแหน่ง (cursor) ไปตำแหน่งถัดไป

6.1.2 อักขระที่มีความกว้าง

คืออักขระที่ต้องแสดงในระดับที่ 3 ได้แก่ พยัญชนะและสระข้างสระประเภทนี้เมื่อแสดงบนจอภาพแล้วจะต้องเลื่อนตัวชี้ตำแหน่งไปหนึ่งตำแหน่ง จึงเรียกอักขระประเภทนี้ว่าเป็นอักขระที่มีความกว้าง

6.2 การแยกประเภทอักขระ

ในการบอกตำแหน่งสำหรับวางอักขระไทย สามารถใช้ส่วนของโปรแกรมภาษาซีพร้อมกับตัวแปรแถวลำดับ (array) ดังแสดงในรูป 4.8 เพื่อแยกประเภทของอักขระ

เมื่อกำหนดให้รหัสแอสกีของอักขระที่ต้องการทราบประเภทเป็นพารามิเตอร์ของแมคโคร TypeOfChar() จะให้ค่ากลับมาเป็น 1 หรือ 0 โดยที่หนึ่งหมายถึงเป็นอักขระประเภทไม่มีความกว้างและศูนย์หมายถึงเป็นอักขระประเภทมีความกว้างแมคโครนี้จะทำหน้าที่ตรวจว่ารหัสแอสกีของอักขระอยู่ในช่วง 48 ตัวสุดท้ายของตารางรหัส มอก.620-2533 หรือไม่ เพราะว่าอักขระประเภทไม่มีความกว้างจะอยู่ในช่วงรหัสดังกล่าวเท่านั้น ในกรณีที่อยู่ในช่วงรหัสดังกล่าวจะนำรหัสไปค้นหาประเภทของอักขระในตารางประเภทของอักขระอีกครั้งหนึ่ง

```
#define ZeroWidth    1
#define NonZeroWidth 0
#define TypeOfChar(i) (i < 0xD0) ? 0 : TypeTable[i-0xD0]
int TypeTable[48] = {
        0,1,0,0,1,1,1,1, 1,1,1,1,0,0,0,0,
        0,0,0,0,0,0,0,1, 1,1,1,1,1,1,1,0,
        0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0
};
```

รูป 4.8 แสดงส่วนของโปรแกรมภาษาซีที่ใช้ในการแยกประเภทของอักขระไทย

7. ภาวะภาษาไทยและภาวะภาษาอังกฤษ

ภาวะภาษาไทยเกิดเมื่อตัวเลียนแบบเทอร์มินัลได้รับชุดรหัสหลีก <ESC> { T โดยตัวเลียนแบบเทอร์มินัลจะกำหนดค่าตัวแปรภายใน SOmode เป็น 1 ส่วนภาวะภาษาอังกฤษเกิดเมื่อตัวเลียนแบบเทอร์มินัลได้รับชุดรหัสหลีก <ESC> { L โดยตัวเลียน แบบเทอร์มินัลจะกำหนดค่าตัวแปรภายใน SOmode เป็น 0 ในภาวะภาษาไทยตัวเลียนแบบเทอร์มินัลจะแปลงรหัสภาษาไทยที่ได้รับจากการกดแป้นพิมพ์เป็นรหัสกลางแล้วส่งเข้าบัฟเฟอร์ สำหรับเตรียมส่งอักขระส่วนในการแสดงผลจะถือว่าข้อมูลที่อ่านเข้ามาจากบัฟเฟอร์สำหรับรอรับข้อมูลเป็นรหัสกลางดังนั้นตัวเลียนแบบเทอร์มินัลจะถอดรหัสกลับเป็นรหัสเดิมแล้วนำไปแสดงผลบนจอภาพ ผู้ใช้สามารถกำหนดภาวะภาษาไทยและภาษาอังกฤษได้โดยการกดแป้นพิมพ์ F10 โดยที่เมื่อกดแป้นพิมพ์นี้ครั้งแรกตัวเลียนแบบเทอร์นินัลจะส่งรหัส <ESC> { T เข้าสู่บัฟเฟอร์สำหรับเตรียมส่งอักขระในการการกดครั้งที่สองจะส่งรหัส <ESC> { L แทนโดยสลับกันเช่นนี้เรื่อยไป

8. การแปลงข้อมูลเป็นรหัสกลาง

การแปลงข้อมูลเป็นรหัสกลางของตัวเลียนแบบเทอร์มินัลทำโดยโปรแกรมในส่วน ParseEscape โดยจะทำหน้าที่ตรวจสอบข้อมูลที่รับเข้ามาจากพอร์ตสื่อสาร ในกรณีที่เป็นลำดับรหัสหลีกในตาราง 3.1 ชุดคำสั่ง DoEscape จะทำหน้าที่กำหนดค่าตัวแปรภายในเพื่อบอกให้ตัวเลียนแบบเทอร์มินัลทราบถึงหน้าที่ที่จะต้องปฏิบัติกับข้อมูลที่รับเข้ามาต่อไปตัวแปรภายในนี้คือ ตัวแปร SOmode และตัวแปร SCOmode โดยที่ตัวแปร SOmode มีค่าเป็นหนึ่งหมายถึงอยู่ในภาวะภาษาไทย และมีค่าเป็นศูนย์หมายถึงอยู่ในภาวะภาษาอังกฤษ ส่วนตัวแปร SCOmode เป็นตัวแปรเสริมเพื่อช่วยในการแปรงรหัส ผลของชุดรหัสทั้งสี่ที่มีต่อตัวแปรภาวะของตัวเลียนแบบเทอร์มินัลแสดงในตาราง 3.2

8.1 การเข้ารหัสของตัวเลียนแบบเทอร์มินัล

การเข้ารหัสของตัวเลียนแบบเทอร์มินัลเกิดขึ้นเมื่อตัวเลียนแบบเทอร์มินัลอยู่ในภาวะภาษาไทย (Thai mode) โดยที่ตัวเลียนแบบเทอร์มินัลจะตัดบิตที่ 7 ออกจากข้อมูลโดยใช้ตัวเลข 7Fh ในฐานสิบหกมาดำเนินการทางตรรก "และ" กับอักขระที่จะตัดบิตที่ 7 จะมีผลทำ ให้อักขระนั้นบิตที่ 7 มีค่าเป็นศูนย์ ในที่นี้ตัวเลียนแบบเทอร์มินัลนี้จะถือว่าอักขระที่จะนำเข้าได้ต้องอยู่ในช่วงรหัสที่กำหนดโดยรหัส มอก.602-2533 นั่นคือตั้งแต่ A1h ถึง

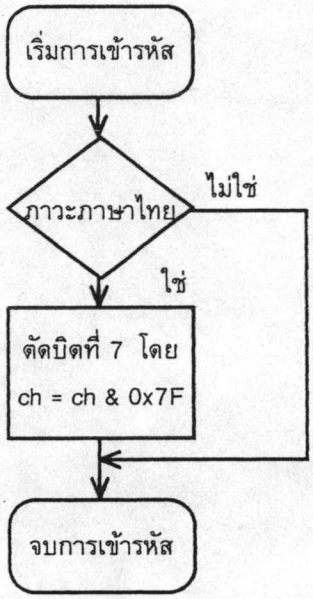
FBh เท่านั้น ดังนั้นจึงละการเข้ารหัสสำหรับอักขระในช่วง 80h ถึง 9Fh ซึ่งทำให้ขั้นตอนการเข้ารหัสง่ายขึ้นกว่ากรณีทั่วไปที่กล่าวถึงในบทก่อน ผังงานของการเข้ารหัสที่ใช้ในตัวเลียนแบบเทอร์มินัลแสดงในรูป 4.9

8.2 การถอดรหัสของตัวเลียนแบบเทอร์มินัล

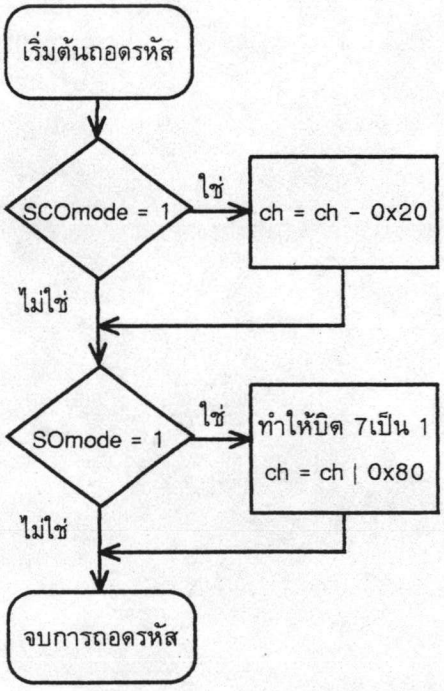
การถอดรหัสกลับเป็นอักขระเดิมเพื่อการแสดงผลของตัวเลียนแบบเทอร์มินัลจะพิจารณาจากตัวแปรภาวะสองตัวคือ SCOmode และ SOmode โดยที่เมื่อ SCOmode มีค่าเป็นหนึ่งหมายถึงรหัสนั้นอยู่ในช่วงอยู่ในช่วง 00h ถึง 1Fh หรือ 80h ถึง 9Fh ตัวเลียนแบบเทอร์มินัลจะนำรหัสกลางมาลบออกด้วย 20h ในฐานสิบหก จากนั้นพิจารณาตัวแปรภาวะ SOmode ถ้ามีค่าเป็นหนึ่งหมายถึงเป็นรหัสที่เพิ่มเติมจากรหัส ISO 646 จึงต้องทำให้บิตที่ 7 มีค่าเป็นหนึ่ง รูป 4.10 แสดงผังงานการถอดรหัสของตัวเลียนแบบเทอร์มินัล

9. การเรียกใช้ชีฐูไรเตอร์ผ่านตัวเลียนแบบเทอร์มินัล

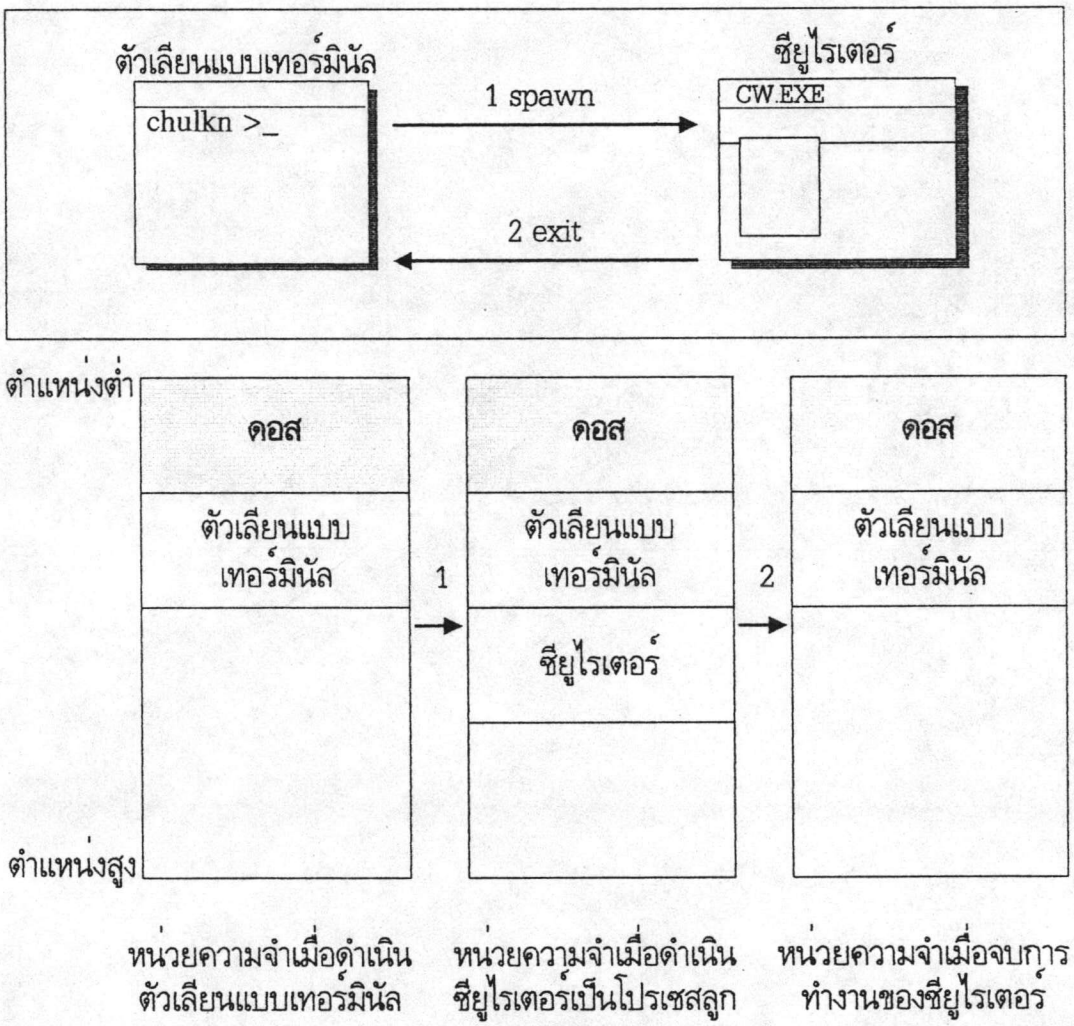
ตัวเลียนแบบเทอร์มินัลจะให้กำเนิดชีฐูไรเตอร์เป็นโปรเซสลูก (child process) โดยใช้ฟังก์ชัน spawnvpe ในภาษาซี ซึ่งมีการทำงานดังแสดงในรูป 4.11 โดยเริ่มต้นเมื่อเรียกใช้ฟังก์ชันนี้ดอสจะพักการทำงานของตัวเลียนแบบเทอร์มินัลไว้แล้วผ่านการควบคุมมาให้ชีฐูไรเตอร์ทำงาน โดยโปรแกรมแรกยังคงค้างอยู่ในหน่วยความจำรอจนชีฐูไรเตอร์จบการทำงานจะยกเลิกการใช้งานหน่วยความจำของชีฐูไรเตอร์ แล้วคืนการควบคุมมาให้ตัวเลียนแบบเทอร์มินัล



รูป 4.9 ผังงานแสดงการแปลงเป็นรหัสกลางของตัวเลียนแบบเทอร์มินัล



รูป 4.10 ผังงานแสดงการถอดรหัสกลางเพื่อแสดงผลบนจอ



รูป 4.11 แสดงขั้นตอนการดำเนินซียูไรเตอร์เป็นโปรเซสลูก