โปรแกรมสเปรดชีตวินโดว์สอินเทอร์เฟส
Spreadsheet Windows Interface

โดย

1. นส.ยาใจ พรหมบุญ 32.1248

2. นส.ลาวัลย์ วชิรหาราภาคว 32.1268

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปริญญานิพนธ์ ปีการศึกษา 2535

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเขียนโปรแกรมสเปรดชีตโดว์สอินเทอร์เฟต

ผู้จัดทำ

1. นส.ยาใจ พรหมบุญ 32.1248
2. นส.ลาวัลย์ วชิรธาราภาดร 32.1268

ดร.บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา

(.....)

การเขียนโปรแกรมสเปรดชีตวินโดว์สอินเทอร์เฟต

1. นส.ยาใจ พรหมบุญ 32.1248

2. นส.ลาวัลย์ วชิรหาราภาคร 32.1268

อาจารย์ที่ปรึกษา

ดร.บุญชีร์ เครือตราชู

บทคัดย่อ

โครงงานนี้ เป็นการเขียนโปรแกรม spreadsheet interface บน Windows โดยจะแบ่งเป็น 2 ส่วนหลักคือ

1.Worksheet Interface

2.Graphics Interface

ในส่วน worksheet Interface จะเป็นการเขียนโปรแกรมเพื่อติดต่อกับผู้ใช้ที่ปฏิบัติการกับข้อมูลบน worksheet ทั้งหมด อันได้แก่ insert, delete , open file save file,การ split window และการพิมพ์ worksheet ออกทางเครื่องพิมพ์ เป็นต้น และอีกส่วนคือ Graphics Interface จะเป็นส่วนที่จัดการกับการแสดงผล Graphics ของข้อมูลในรูปแบบของกราฟต่างๆอันได้แก่ กราฟแท่ง กราฟเส้นตรง กราฟวงกลม กราฟจุด กราฟแท่งสะสม ตลอดจนการแสดงผลรูปกราฟเหล่านี้ออกทางเครื่องพิมพ์

การเขียนโปรแกรมทั้ง 2 ส่วนนี้ จะเขียนในเชิง Commercial คือเขียนในลักษณะที่โปรแกรมสามารถนำไปใช้งานได้จริงในแง่ประสิทธิภาพและความหลากหลายของฟังก์ชั่นต่างๆที่ใช้ในการทำงาน และความสะดวกสบายง่ายต่อการใช้ของผู้ใช้

Spreadsheet Windows Interface Program

By

1.Miss Yajai Promboon 32.1248

2.Miss Lawan Whachiratarapadorn 32.1268

Advisor:

Dr.Boontee Kruatrachue

Abstract

This Project is a spreadsheet Interface Windows Program. It is divided into two parts. That are

1. Worksheet Interface
2. Graphics Interface

The Worksheet Interface is the part which its function is interacting with users who operate data on worksheet such as insert, delete, open file, save file, split window and worksheet printing.

And the other,Graphics Interface is the part which presents data in form of graphs such as bar graph,circle graph,line graph, and dot graph and printing all of these graphs.

The objective of this project is to develop a spreadsheet program which can be use in a real world.

สารบัญ

หน้า

บทที่ 1

บทนำ

ในปัจจุบันนี้โปรแกรม spreadsheet เป็นโปรแกรมที่นิยมและแพร่หลายอย่างกว้างขวางในหมู่นักคอมพิวเตอร์และผู้ใช้คอมพิวเตอร์ในการประมวลผลทั้งหลาย โปรแกรม spreadsheet ที่ดีนั้น จะต้องมีความสามารถในการแสดงผลทางด้านกราฟฟิก, การประมวลผลข้อมูลทางคณิตศาสตร์และมีระบบจัดการอื่น ๆที่ดีเพื่อให้ผู้ใช้ได้ประโยชน์และประสิทธิภาพในการทำงานสูงสุด ส่วนโปรแกรม spreadsheet ที่ทำงานบน Windows นั้นนอกจากจะมีความสามารถเฉพาะตัวของมันเองแล้วยังได้รวมเอาอรรถประโยชน์ต่างๆ ที่ระบบ Windows มีให้เข้ามารวมไว้ด้วยกันอีก อาทิเช่น การทำงานแบบ Multitasking ความง่าย ความสะดวกสบายของผู้ใช้ในการใช้โปรแกรม เป็นต้น

โครงงานนี้เป็นการเขียนโปรแกรม spreadsheet Interface บน Windows โดยรายละเอียดของโครงงานทั้งหมดที่จะกล่าวในวิทยานิพนธ์ฉบับนี้ประกอบด้วย 3 ส่วนคือ

1. ส่วนทฤษฎีโดยทั่วไปและส่วนประกอบสำคัญของการเขียนโปรแกรมบน Windows
2. คุณสมบัติและการทำงานของโปรแกรมโครงงาน
3. ฟังก์ชั่นต่างๆที่เขียนขึ้นในโปรแกรมรวมทั้ง หน้าที่ อินพุต และ เอาต์พุต ของฟังก์ชั่นแต่ละอัน

บทที่ 2

ลักษณะโดยทั่วไปของโปรแกรมใช้งานบนสภาวะแวดล้อมของ Windows

การติดต่อผู้ใช้แอพพลิเคชันจะติดต่อผู้ใช้โดยผ่าน window โดย window มีหน้าตาเป็นสี่เหลี่ยมเล็กๆ ที่ปรากฏขึ้นบนจอมอนิเตอร์ในกรอบสี่เหลี่ยมอาจจะประกอบด้วยปุ่มต่างๆ ที่ถูกออกแบบไว้ให้เป็นภาพ 3 มิติเหมือนกับจะจับต้องได้ ขอบบนสุดจะเป็นชื่อของ window ซึ่งมักจะพบว่าเป็นชื่อเดียวกับโปรแกรม บรรทัดถัดมาจะเป็นเมนูบาร์ซึ่งเป็นที่รวบรวมชับเมนูของโปรแกรมมนั้นๆ เอาไว้ ในจุดนี้บางทีเราเรียกว่า พูลดาวน์เมนู ซึ่งนิยมใช้กันมานานแล้ว

การติดต่อของ wondows กับผู้ใช้ ระบบ windows เองติดต่อผู้ใช้โดยผ่านทางโปรแกรมมีชื่อว่า Program Manager ทุกๆ ครั้งที่รันโปรแกรม Windows มันจะรอคำสั่งจากผู้ใช้ที่จุดนี้เหมือนกับที่ระบบดอสติดต่อกับผู้ใช้โดยผ่านทางโปรแกรม command.com และจะรอรับคำสั่งถัดไปจาก prompt

ทั้ง command.com และ Program Manager ต่างก็เป็นแอพพลิเคชันที่ระบบใช้ติดต่อกับผู้ใช้ซึ่งก็ไม่ต่างไปจากแอพพลิเคชันอื่นๆ เท่าใดนัก แอพพลิเคชันทุกตัวบนระบบ Windows ต่างก็เรียกใช้บริการของระบบซึ่งประกอบด้วย User, Kernel, GDI ซึ่งจะกล่าวถึงรายละเอียดในภายหลัง

2.1 การติดต่อกับผู้ใช้

กรณีมีหลายแอพพลิเคชันกำลังทำงานอยู่ในขณะเดียวกันในระบบมัลติทาสกิง Windows จะอนุาตให้ทุกแอพพลิเคชันใช้หน้าจอร่วมกันได้ เพื่อให้ผู้ใช้สามารถเรียกใช้ได้ทุกๆ แอพพลิเคชัน ในทุกๆ ขณะ ในระบบอื่น บางระบบที่สามารถทำงานแบบมัลติทาสกิงได้ หน้าจอจะแสดงแอพพลิเคชันเพียงตัวเดียวเท่านั้น ถึงแม้แอพพลิเคชันตัวอื่นกำลังทำงานอยู่ข้างหลัง

ใน Windows สามารถทำดังนี้ได้ก็เพราะ Window ไม่ใช่แค่เพียงหน้าต่างธรรมดาๆ แต่ประกอบด้วย visual device เช่น menu, control, scroll bars ทำให้ผู้ใช้สามารถเข้าไปใช้ แอพพลิเคชันนั้นได้

Windows จะคอยจัดการหน้าจอให้ โดยจะควบคุมการแสดงผล การเลื่อน การเปลี่ยนขนาดของหน้าต่าง คอยกันไม่ให้แอพพลิเคชัน ใช้ส่วนของหน้าจอเดียวกันในขณะเดียวกัน อันจะก่อให้เกิดปัญหาตามมาได้

2.2 Queued Input

การรับข้อมูลจากอุปกรณ์ไม่ว่าจะเป็น ข้อมูลจากคีย์บอร์ด เวลาจากระบบ ตำแหน่งของเมาส์ ฯลฯ การรับข้อมูลระหว่างแอพพลิเคชันของ DOS และ Windows ใช้กรรมวิธีที่ต่างกันโดยสิ้นเชิงโดยแอพพลิเคชันของดอสจะรับข้อูลจากระบบ หรือบางทีจะรับจากฮาร์ดแวร์โดยตรง การรับแยกเป็นอย่างๆ เป็นตัวๆ ไปเช่น การใช้ฟังก์ชัน getchar รับอักษรจากคีย์บอร์ดมา 1 ตัว จะอ่านเวลาของระบบก็เรียกใช้ฟังก์ชันหนึ่ง จะอ่านตำแหน่งของเมาส์ก็เรียกอีกฟังก์ชันหนึ่ง แยกอิสระต่อกัน ซึ่งสิ่งเหล่านี้จะต่างจากแอพพลิเคชันของ Windows ตัวระบบ Windows จะคอยรับรวบรวมข้อมูลที่กระจัดกระจายเหล่านี้ไปบรรจุไว้ใน message queue และคอยให้ แอพพลิชันมารับไปประมวลผล

2.3 Device-Independent Graphics

ใน Windows มีระบบกราฟิกที่ไม่ขึ้นกับอุปกรณ์ใด (device independent) ซึ่งอุปกรณ์นั้นไม่ว่าจะเป็นจอภาพ เลเซอร์พรินเตอร์ หรือ ด็อตเมตริกซ์พรินเตอร์ก็ดี ระบบกราฟิกใหม่นี้อำนวยให้เราสามารถสร้างจุดภาพ ลากเส้นตรง วาดวงกลม เขียนรูปสามเหลี่ยม รูปวงรี บนอุปกรณ์ดังกล่าว

เมื่อถามว่าระบบ Windows ทำอย่างนั้นได้อย่างไร จะเกินความจริงหรือเปล่า ตอบว่าไม่ เพราะว่าภายใน Windows ถือว่ากราฟิกมีมาตรฐานเดียวกัน เมื่อต้องการส่งเอาต์พุตไปออกที่อุปกรณ์ตัวไหนจะต้องผ่านการแปลงโดย device driver ตัวนั้นๆ ตัวอย่างเช่นเมื่อต้องการออกที่จอภาพ ก็ต้องส่งผ่าน display driver ถ้าเราพิจารณาทางด้านอุปกรณ์จอมอนิเตอร์แล้ว ยังสามารถแบ่งออกได้อีกหลายชนิดเช่น จอโมโนฌครม จอ CGA, EGA, VGA ฯลฯ เมื่อจอมีอีกหลายชนิดแสดงว่า Display Driver ก็จะต้องมีอีกหลายตัวตามแต่ชนิดของจอ

ทางด้านอุปกรณ์เครื่องพิมพ์ก็จะมี device driver เรียกว่า Printer Driver เครื่องพิมพ์เองมีอีกหลายชนิดเช่นกัน ด็อตเมตริกซ์ และ เลเซอร์พรินเตอร์ ในเครื่องพิมพ์แบบเลเซอร์พอจะแบ่งได้เป็น 2 ค่ายคือ PCL laser printer และ PostScript laser printer โดย PCL เป็นของบริษัท Hewllet Packare และ PostScript เป็นของ Adobe System

2.4 Multitasking

Windows นั้นสามารถทำงานแบบมัลติทาสกิงได้ เพราะสามารถจะรันได้ถึง 3 โหมด คือ Real mode, Standard mode และ Enhance mode

ใน Real mode จะทำงานเหมือนกับดอสธรรมดา ส่วนใน Standard mode จะทำงานแบบมัลติทาสกิงที่จะต้องได้รับร่วมมือจากซอฟ์แวร์ ที่จริงแล้วในโหมดนี้ Windows ทำงานได้ดีกว่าในโหมดมัลติทาสกิงจริงๆ เสียอีก ทั้งนี้เพราะสาเหตุที่ว่าจะต้องได้รับความร่วมมือจากซอฟ์ทแวร์ ทำให้ Windows สามารถจัดสรรทรัพยากรของระบบได้เต็มประสิทธิภาพ แอพพลิเคชันส่วนใหญ่ถูกเขียนให้รันในโหมดนี้

ต่อไปคือ Enhance mode การทำงานในโหมดนี้ไม่ต้องอาศัยความช่วยเหลือจากซอฟ์ทแวร์ ที่สามารถทำให้อย่างนี้เพราะอาศัยความสามารถของ CPU 386 ในการทำมัลติทาสกิง จะเห็นว่าในโหมด Enhance mode จะใช้ได้เฉพาะเครื่องที่มี CPU เบอร์ 386 ขึ้นไป

ส่วน Standard mode จะทำงานได้กับ CPU เบอร์ 286 ขึ้นไป ข้อสำคัญในระบบมัลติทาสกิงก็คือแอพพลิเคชันจะต้องแบ่งกันใช้ทรัพยากรของระบบซึ่งประกอบด้วย อุปกรณ์อินพุต อุปกรณ์เอาท์พุต หน่วยความจำ และ CPU

การบริหารหน่วยความจำของ Windows ที่จะต้องระลึกไว้ประการแรกก็คือ ห้ามแกล้งระบบโดยการจองหน่วยความจำยังไว้มากๆ ควรจะจองไว้เท่าที่จำเป็นและควรจะยกเลิกทันทีที่ไม่ได้ใช้งาน ลักษณะการทำงานของ Windows จะออกมาในรูปแบบการเคลื่อนย้ายส่วนที่ต้องคงอยู่และยกเลิกส่วนที่ไม่จำเป็น โดยปกติ Windows จะอนุญาตให้แอพพลิเคชันสามารถจองหน่วยความจำได้ซึ่งสามารถจองได้ทั้งแบบ Global คือใช้ร่วมกับแอพพลิเคชันอื่นได้ และแบบ Local คือใช้ได้เฉพาะโปรแกรม

การจัดหน่วยความจำของระบบโดยทั่วๆ ไป เช่นดอสเป็นต้น เมื่อเราทำการจอง หรือทำการ allocate พื้นที่ความจำนั้นจะถูกสงวนไว้ไม่ให้โปรแกรมมาใช้ได้ และจะคงอยู่อย่างนั้นจนกว่าจะถูกยกเลิก แต่สำหรับ Windows ไม่เป็นเช่นนั้น ตัว Windows ยังคงสามารถเคลื่อนย้ายและยกเลิกหน่วยความจำในส่วนนั้นได้ การเคลื่อนย้ายหน่วยความจำ Windows ทำไปก็เพื่อต้องการให้มีพื้นที่ว่างต่อเนื่องเป็นพื้นที่ใหญ่ที่เดียวกัน กรณีทำการยกเลิกทำเพื่อนำหน่วยความจำที่ไม่จำเป็นกลับมาใช้ให้เป็นประโยชน์ และสามารถนำกลับมาได้กรณีแอพพลิเคชันต้องการใช้ความจำในส่วนนั้นขึ้นมา

ในระบบ Windows เมื่อเราทำการ allocate หน่วยความจำ ตัว Windows จะคืนหมายเลขของการ Handle กลับมาให้ ซึ่งหมายเลขจะใช้แก้ไขปัญหาทุกอย่างที่เกี่ยวกับหน่วยความจำ ตัวอย่างเช่น เมื่อเราต้องการเข้าถึงหรือเข้าไปใช้หน่วยความจำส่วนที่จองเอาไว้ ซึ่งเราต้องขอให้ Windows ทำการ lock หน่วยความจำตามหมายเลข handle ซึ่งในขณะนั้น Windows จำไม่สามารถเคลื่อนย้ายหรือยกเลิกพื้นที่เราได้ เมื่อใช้เสร็จก็ทำการปลด lock นั้นเพื่อให้ Windows ทำหน้าที่บริหารความจำต่อไป ในช่วงที่ lock และ unlock คล้ายกับการที่เราใช้คำสั่ง enable interrutpt และ disable interupt ของภาษาแอสเซมบลีเพื่อหยุดการขัดจังหวะของฮาร์ดแวร์

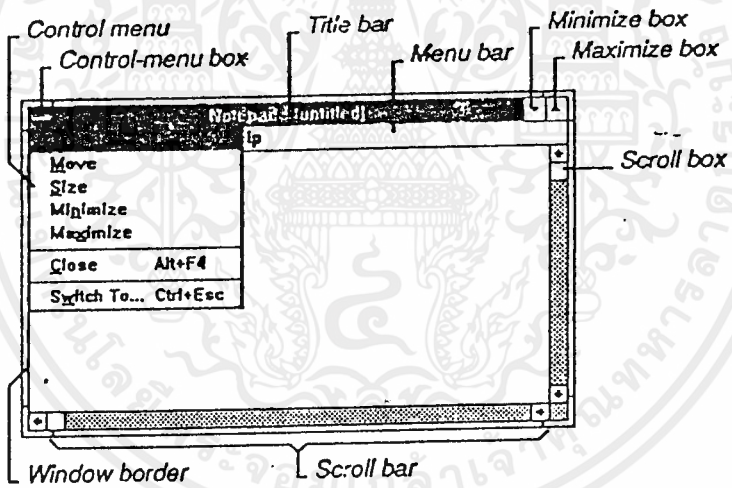
สังเกตุว่าเมื่อเราจองหน่วยความจำ เราจะได้หมายเลขของการ handle กลับคืนมาและก่อนที่จะเข้าไปต้องบอก Windows ด้วยการ lock ทั้งนี้เพื่อให้ Windows ได้เตรียมข้อมูลในส่วนที่ถูกยกเลิกกลับมาเข้าที่เสียก่อน ทั้งนี้จะเป็นไปตามหลักที่ว่า "ข้อมูลในหน่วยความจำไม่ว่าจะเป็น code หรือ data เมื่อไม่ถึงเวลาใช้มันก็ไร้ความหมาย จะมีความหมายก็ต่อเมื่อได้ใช้"

2.5 The Windows Programming Model

องค์ประกอบของโปรแกรมที่ใช้ติดต่อกับผู้ใช้ประกอบด้วย Window , Menu , Dialog box, Message loop ซึ่งจะพูดถึงตามลำดับ

2.5.1 Window

Window เป็นอุปกรณ์ที่ใช้ติดต่อกับผู้ใช้ทุกๆ แอพพลิเคชันภายใน Window จะประกอบด้วย Title bar, Menu bar, Scroll bar, Border นอกจากที่กล่าวมานี้แล้ว ยังมี Feature อื่นที่เราสามารถกำหนดให้ Windows เป็นผู้สร้างให้เราได้



รูปที่ 2.1 แสดงลักษณะโดยทั่วไปของ Window ที่ใช้ในการติดต่อกับผู้ใช้

แม้ว่าแอพพลิเคชันจะเป็นผู้กำหนด window ขึ้นมาใช่ว่าทำงานทั้งหมดถูกสั่งมาจากแอพพลิเคชัน แท้ที่จริงเกิดจากความร่วมมือระหว่างทั้ง 2 คือ aplication และ Windows ที่เกิดจากการเคลื่อนย้ายของผู้ใช้ นอกจากนั้งทำหน้าที่ควบคุมมาตรฐานต่างๆ ของ window อันได้แก่กรอบที่สามารถขยับขยายได้ ตำแหน่งของ Scroll bar และ Title รวมทั้งการตอบสนองกับผู้ใช้ นอกเหนือจากนั้นเป็นหน้าที่ของแอพพลิเคชัน โดยเฉพาะอย่างยิ่งพื้นที่ภายในกรอบทั้งหมดที่ใช้ในการแสดงข้อความ รูปภาพ ทั้งนี้จะต้องได้รับความร่วมมือจาก Windows ด้วย ตัวอย่างเช่นการขยับ window ใด window หนึ่งที่ทับกันแน่นอนว่าจะมีผลกระทบไปยัง window อื่นๆ แต่มันสามารถลงตัวกันได้ เพราะแอพพลิเคชัน กับ Windows มีความสัมพันธ์กันในส่วนของฟังก์ชันหนึ่งที่เรียกว่า window function กล่าวคือ window function ก็จะเป็นส่วนหนึ่งของ Windows เพราะว่าฟังก์ชันนี้จะถูกเรียกใช้โดย Windows ก็ถือว่าเป็นส่วนหนึ่งของ Windows มันจึงสามารถควบคุมให้ทุก window เข้ากันได้

2.5.2 MENU

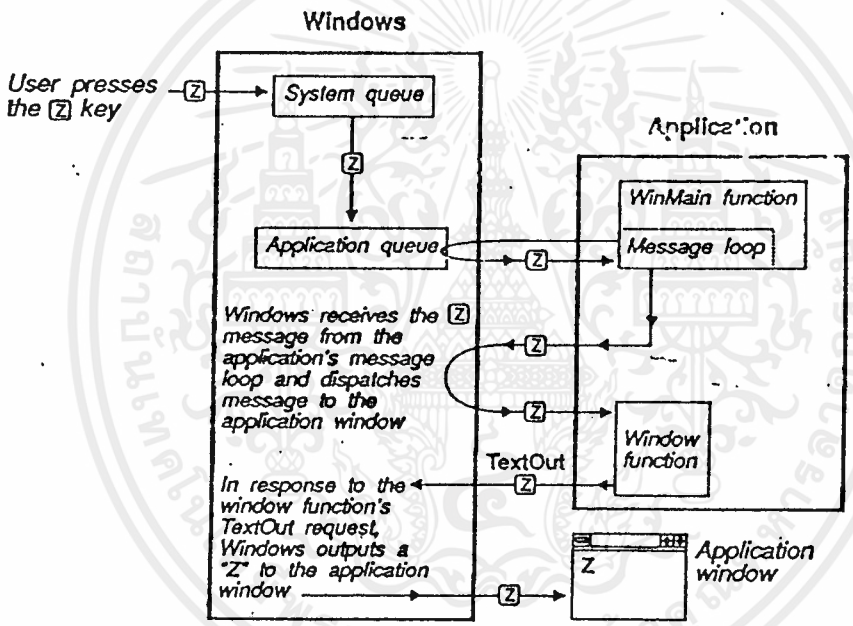
เมนู ถือว่าเป็นจุดสำคัญในการรับคำสั่งจากผู้ใช้ของบรรดาแอพพลิเคชันที่ถูกเขียนขึ้นมาสำหรับ Windows ทีเดียว ผู้ใช้สามารถดูคำสั่งที่สั่งผ่านทางคีย์บอร์ด หรือ จะออกคำสั่งจากเมนูได้โดยตรงก็ได้ การสร้างเมนูเหล่านี้ Windows จะเป็นผู้บริการเรา เพียงเราจ่ายรูปแบบของเมนูและชื่อคำสั่งที่ต้องการปรากฏในเมนูเท่านั้น เมื่อผู้ใช้เลือกคำสั่ง Windows จะเป็นผู้สั่งคำสั่งในรูปแบบของ message ไปยัง window ฟังก์ชัน message นี้ เป็นสัญญาณที่ทำให้แอพพลิเคชันทำงานในขั้นต่อไป

2.5.3 DIALOG BOXES

dialog boxes เป็นเพียง window ชั่วคราวซึ่งเราอาจจะแสดง และบังคับให้ผู้ใช้ตอบคำถามที่ปรากฏนั้นๆ ได้ dialog box จะใช้บรรจุ control อันหนึ่งหรือมากกว่าก็ได้ ซึ่ง control นี้เป็น window เล็กๆ มีหน้าที่รับอินพุตหรือส่งเอาต์พุต แบบง่ายๆ ตัวอย่างเช่น "edit control" เป็น window แบบง่ายๆ ให้ผู้ใช้พิมพ์ข้อความเข้าไป

2.5.4 MESSAGE LOOP

message loop เป็นเพียง กระบวนการสำคัญในการรับ message หรือ ข้อความของแอพพลิเคชัน กระบวนการนี้จะเกิดขึ้นภายในฟังก์ชันหลัก หรือ ฟังก์ชัน WinMain ในลักษณะการทำงานจะวนลูปมาดึง message จาก aplication queue อยู่บ่อยๆ message ที่ได้จะถูกส่งต่อไปยัง window function ซึ่งเป็นไปในลักษณะทางอ้อม



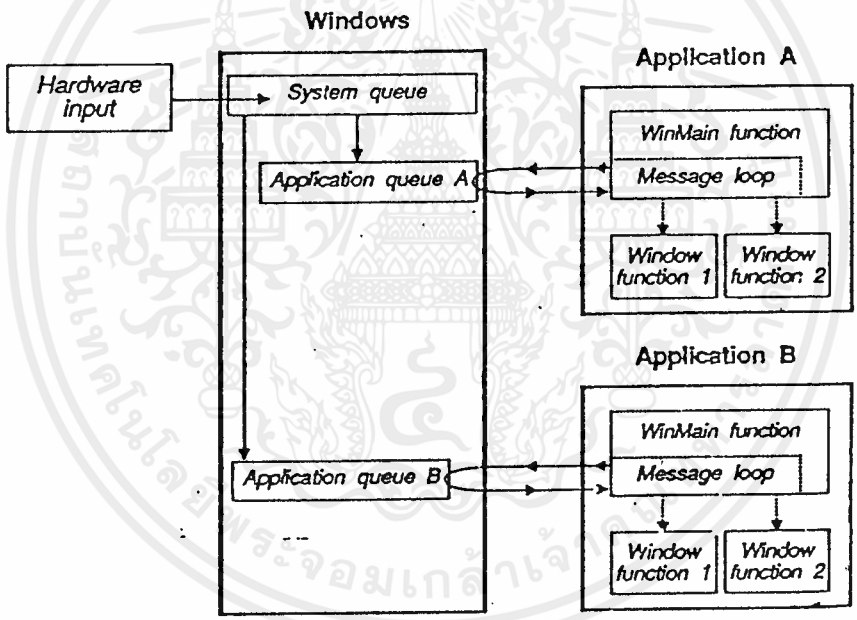
รูป 2.2 แสดงการ Processing Keyboard Input

จากรูปที่ 2.2 เป็นกระบวนการที่ Windows รับข้อมูลจากคีย์บอร์ด ซึ่งเป็นไปด้วย 2 ขั้นตอน คือ ขั้นตอนแรก window จะอ่านอักษรจากแป้นพิมพ์ในที่นี้เราสมมุติว่าเป็น z ซึ่ง Windows จะอ่านเข้ามาเก็บไว้ใน system queue ซึ่งเป็นที่พักเก็บชั่วคราวในส่วนกลาง ในขั้นตอนถัดมา Windows จะก๊อปปี้ข้อมูลนั้นไปให้ application queue เป็นอันว่าจบการทำงานของ Windows ในส่วนบริการรับคีย์

จากรูปจะเป็นว่าโปรแกรมจะถูกแบ่งออกเป็น 2 ส่วน คือ Windows กับ application ต่อมาดูการทำงานของแอพพลิเคชันบ้าง หลังจากที่ Windows นำอักษรไปเก็บไว้ใน queue เรียบร้อยแล้ว เมื่อฟังก์ชัน Winmain ทำงาน (ทำโดย Windows เป็นผู้เรียกขึ้นมา) ฟังก์ชัน Winmain เป็นจุดที่แอพพลิเคชันทำการดึงข้อมูลจาก application queue และส่งต่อไปยัง window function โดยส่งไปแบบทางอ้อม เมื่อ window function ได้รับ message ในที่นี้คือ ตัวอักษร z เพื่อจะนำอักษรนั้นพิมพ์ลงบน window

จากรูปจะเห็นว่า เป็นเส้นเชื่อมต่อระหว่าง Windows กับ application เส้นเหล่านี้เป็นการส่งผ่านค่าไปมา ซึ่งเกิดจากการที่ application เรียกใช้บริการ หรือเรียกใช้ฟังก์ชันที่ Windows จัดไว้ให้เช่น ฟังก์ชัน GetMessage() ใช้ดึง message จาก application queue ฟังก์ชัน DispatchMessage() ใช้ส่งผ่าน message ไปยัง window function เป็นต้น

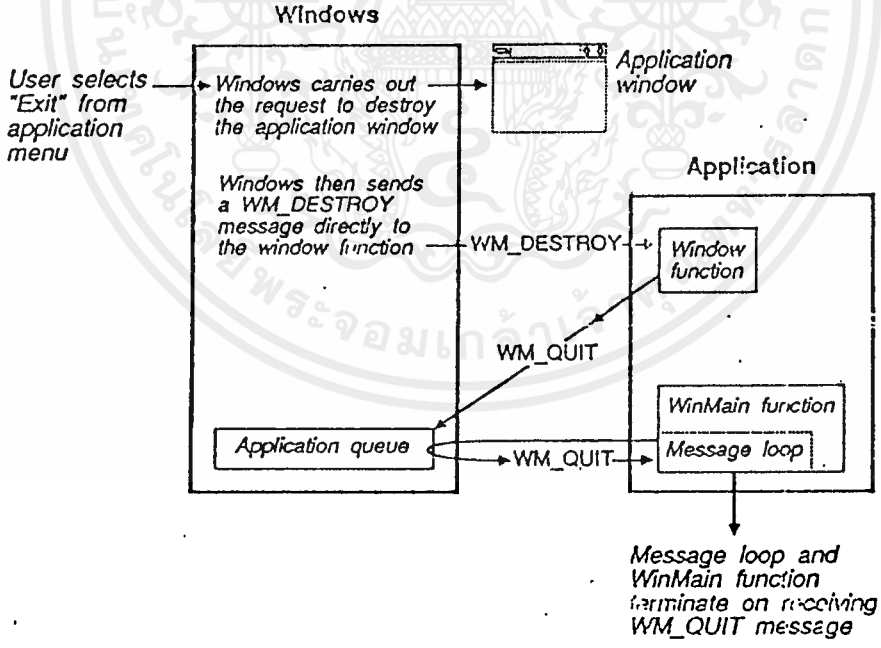
จากรูปที่แล้วเป็นตัวอย่างการรับอินพุตของแอพพลิเคชันเพียงตัวเดียว ต่อไปเราจะมาพิจารณาการทำงานร่วมกันระหว่าง Windows กับแอพพลิเคชัน 2 ตัวให้ชื่อว่า application A และ aplication B



รูป 2.3 แสดงการ Processing Input for Two Applications

เริ่มที่จาก Windows ไปรับข้อมูลมาจากฮาร์ดแวร์ไปเก็บไว้ในที่เก็บส่วนกลางคือ system queue ต่อจากนั้นจะแจกจ่ายข้อมูลไปยัง aplication queue ทั้ง A และ B เป็นอันว่าเสร็จขั้นตอนของ Windows ต่อมาเมื่อแอพพลิเคชันทั้งสองได้รับการทำงาน ในแต่ละตัวจะใช้กระบวนการ message loop ดึง message จาก application queue ของตัวเองและส่งไปยัง Window function ของตัวเองได้อย่างถูกต้อง

message ที่ปรากฏใน application queue ไม่ใช่ว่าจะเกิดจากฮาร์ดแวร์อย่างเดียว Windows สามารถกำหนด message ขึ้นมาเองโดยตรงก็ได้เช่น การสั่งให้ application จบงานทันที ตัวอย่างเช่นในรูปที่ 2.4 Windows ส่ง message WM_DESTROY ไปยัง window function และ window function ส่ง message กลับมายัง application queue เป็นผลให้ฟังก์ชัน WinMain ได้รับสัญญาณออก และจบโปรแกรมได้ทันที



รูป 2.4 แสดงการ Processing Window-Management message

2.6 The Windows Libraries

ไลบรารีของ Windows ต่างกับของ run-time C ตรงที่ไลบรารีของ Windows เป็น Dymanic Link Library (DLLs) ที่ระบบจะลิงค์กับแอพพลิเคชัน เมื่อมีการโหลดแอพพลิเคชันเท่านั้น ซึ่งจุดนี้เป็นข้อดีของ Windows เพราะจะทำให้ลดขนาดของข้อมูลให้เหลือเท่าที่จำเป็น

ไลบรารีที่สำคัญของ Windows มี 3 อย่าง คือ

USER : เป็นไลบรารีที่จะจัดการกับสภาพแวดล้อมทุกอย่างของ Windows และแอพพลิเคชัน

KERNEL : เป็นไลบรารีที่จัดการเกี่ยวกับการบริการระบบ เช่น การทำ multitasking การจัดการหน่วยความจำ การจัดการทรัพยากรของระบบ

GDI : เป็นไลบรารีที่คอยจัดการเกี่ยวกับ graphics device interfac

บทที่ 3
MESSAGE

3.1 ความหมายของ Message

การเขียนโปรแกรมในปัจจุบันที่นิยมมีหลายวิธี เช่น การเขียนเป็นลำดับ (Sequential) หรือ เป็นโปรแกรมย่อย (Procedural) ฯลฯ แต่การเขียนโปรแกรมให้ทำงานภายใต้สภาวะแวดล้อมของ Windows จะมีโครงสร้างที่ไม่ใช้ลักษณะดังกล่าว เป็นการเขียนโปรแกรมที่สนใจ message มากกว่าลำดับขั้นของการทำงาน

message คือข้อมูลที่เกี่ยวกับการเปลี่ยนแปลงของส่วนที่ใช้ติดต่อกับผู้ใช้ (User Interface) เช่น หน้าต่างถูกเลื่อน ลดขนาด ผู้ใช้มีการกดคีย์บอร์ด เลื่อน mouse ฯลฯ รวมถึงข้อมูลที่ Window Manager ใช้ติดต่อกับแอพพลิเคชันที่กำลังใช้งานอยู่

ในการเขียนโปรแกรมภายใต้สภาวะแวดล้อมแบบ Windows นี้ งานที่ทำส่วนใหญ่คือ ตัดสินใจว่า message ตัวใดที่จะถูกนำมาใช้งาน (Process) หรือ เพิกเฉย ซึ่งสิ่งที่จะต้องระลึกไว้เสมอคือ message ที่เกิดขึ้นไม่ได้เกิดขึ้นตามลำดับ จะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงส่วนที่ใช้ติดต่อกับผู้ใช้ หรือระบบปฏิบัติการต้องการติดต่อกับ Windows เท่านั้น

3.2 วิธีการรับข้อมูล และการแบ่งเวลาให้งานหลายๆ งาน (Multitasking Time Slice)

Windows เป็นระบบปฏิบัติการ ที่สามารถรันโปรแกรมได้หลายๆ งาน ในเวลาเดียวกันแต่เป็นระบบปฏิบัติการที่ไม่เหมือนกับระบบอื่น ตรงที่ Windows จะไม่ไปขัดจังหวะการทำงานของโปรแกรม (ระบบปฏิบัติการประเภทที่ขัดจังหวะการทำงานของโปรแกรมจะเรียกว่า Preemptive Multitasking) แต่ตัวโปรแกรมจะขัดจังหวะของตัวเอง เพื่อให้โปรแกรมอื่นทำงานได้

โปรแกรมที่วิ่งบนระบบจัดการ Windows จะรับข้อมูลจากผู้ใช้หรือจาก ส่วนที่ใช้ติดต่อกับผู้ใช้ผ่านทาง message โปรแกรมสามารถได้รับ message ผ่านทาง "ระบบส่ง message " (Message Delivery System) โดยจะรับส่ง message 1 message เมื่อมีการเรียกระบบส่ง message แต่ละครั้ง ถ้าโปรแกรมไม่มี message ที่จะรับ ระบบนี้ก็จะเริ่มส่ง message ให้กับโปรแกรมอื่นที่มี message ที่ส่งต่อไป

จากจุดนี้ทำให้มองเห็นว่า message ในระบบ Windows เป็นส่วนที่สำคัญที่สุด โดยจะทำหน้าที่ 2 อย่างที่สำคัญ คือ

1. ส่งข้อมูลให้กับโปรแกรมที่รับบน Windows
2. เป็นตัวกำหนดช่วงเวลาในการทำงานของโปรแกรมหนึ่งๆ

โปรแกรมระบบ Windows จะได้รับ message ได้ 2 ทางคือ

1. โดยการอ่านจาก Buffer (ที่พักข้อมูล) โดยใช้คำสั่ง GetMessage ซึ่ง Buffer ที่จะไปอ่านนี้จะมีอยู่ 2 ที่ได้แก่
   1.1 Hardware Event Queue เป็น Buffer ที่ใช้รวบรวมเหตุการณ์ต่างๆ ที่เกี่ยวกับ Mouse และ Keyboard (เช่น มีการกด keyboard เลื่อน mouse ฯลฯ)
   1.2 Application message Queue เป็น Bufferที่ใช้เก็บ message ชนิดอื่นๆทุกโปรแกรมจะมี message Queue เป็นของตัวเอง เมื่อคำสั่ง GetMessage เจอ message ก็จะส่ง message นี้ให้กับโปรแกรม แต่ถ้า ไม่พบก็จะหยุดการทำงานของโปรแกรม และเปลี่ยนการควบคุมการทำงาน ให้กับโปรแกรมอื่นซึ่งมี message รออยู่ ด้วยวิธีนี้จึงทำให้สามารถรันโปรแกรมหลายๆ โปรแกรมได้พร้อมกัน โดยไม่ต้องไปขัดจังหวะการทำงานของโปรแกรม
2. โดยการที่ระบบจัดการ Windows เรียกใช้ Window Procedure โดยตรง message ที่ส่งด้วยวิธีนี้จะไม่ไปรออยู่ใน message Queue แต่จะถูกนำไปใช้งานในทันที

จะเห็นว่า การรับ message โดยวิธีแรก โปรแกรมของเราจะเป็นตัวที่ทำหน้าที่ดึงเอา message มาจาก Queue เอง เรียกว่า Pull-model Processing ส่วนวิธีที่สองโปรแกรมของเราจะรอให้ระบบ Windows เป็นตัวส่ง message เข้ามา เรียกว่า push-model Processing

3.3 ชนิดของMessage

มี message ถึง 250 message ที่กำหนดในระบบ Windows แต่เราไม่จำเป็นที่จะต้องเรียนรู้ทั้งหมด เพราะ message ส่วนใหญ่ถูกกำหนดขึ้นมาเพื่อวัตถุประสงค์หนึ่งโดยเฉพาะเช่นเพื่อใช้ในการแลกเปลี่ยนข้อมูล (MDI: Multiple-Document Interface) นอกจากนี้ยังมี message ที่จะไม่อยู่ในความสนใจของเราเลย เช่น message ที่กำหนดขึ้นมาเพื่อใช้ภายในของระบบ Windows เอง

message จะมีชื่อเป็นของตัวเอง ชื่อของ message เหล่านี้จะถูกกำหนดใน Include File ที่ชื่อ Windows.h เช่น

```
#define    WM_COMMAND          0x0111
```

เพื่อความสะดวกในการศึกษา เราจะแบ่งประเภทของ message เป็น 8 ชนิดดังนี้

1. Hardware message : เป็น message ที่เกิดจาก Hardware ซึ่งมีอยู่ 3 ประเภทคือ timer, keyboard และ mouse ทั้ง 3 ชนิดนี้ต่างก็ทำการขัดจังหวะการทำงาน แต่เนื่องจากระบบ Windows เป็นระบบที่จะไม่ขัดจังหวะการทำงานของโปรแกรม ดังนั้นการขัดจังหวะของ hardware เหล่านี้จะถูกพักไว้ใน Buffer ก่อนที่จะถูกส่งให้กับโปรแกรมต่อไป ดังรูป 3.1

Mouse Messages: In a window's client area	
WM_LBUTTONDBLCLK	Left button double-click
WM_LBUTTONDOWN	Left button down
WM_LBUTTONUP	Left button up
WM_MBUTTONDBLCLK	Middle button double-click
WM_MBUTTONDOWN	Middle button down
WM_MBUTTONUP	Middle button up
WM_MOUSEMOVE	Mouse move
WM_RBUTTONDBLCLK	Right button double-click
WM_RBUTTONDOWN	Right button down
WM_RBUTTONUP	Right button up
Mouse Messages: In a window's non-client area	
WM_NCLBUTTONDBLCLK	Left button double-click
WM_NCLBUTTONDOWN	Left button down
WM_NCLBUTTONUP	Left button up
WM_NCMBUTTONDBLCLK	Middle button double-click
WM_NCMBUTTONDOWN	Middle button down
WM_NCMBUTTONUP	Middle button up
WM_NCMOUSEMOVE	Mouse move
WM_NCRBUTTONDBLCLK	Right button double-click
WM_NCRBUTTONDOWN	Right button down
WM_NCRBUTTONUP	Right button up
Keyboard Messages	
WM_CHAR	Character input
WM_DEADCHAR	Dead-character (umlaut, accent, etc.)
WM_KEYDOWN	Key has been depressed
WM_KEYUP	Key has been released
WM_SYSCHAR	System character input
WM_SYSDEADCHAR	System dead-character
WM_SYSKEYDOWN	System key has been depressed
WM_SYSKEYUP	System key has been released
Timer Message	
WM_TIMER	Timer has gone off.

รูป 3.1 แสดง Hardware Message

2. Window Maintenance Message : เป็น message ที่ถูกผลิตมาเพื่อดูแลส่วนที่ใช้ติดต่อกับผู้ใช้ให้มีลักษระตรงกับความเป็นจริงเสมอ แบ่งได้เป็น 3 ชนิดดังนี้

2.1 Notification Message : เป็น message ที่บอกว่าสถานะของหน้าต่างมีการเปลี่ยนแปลง เช่น หน้าต่างถูกเลื่อน ลดขนาด เป็นต้น

2.2 Request for Action Message : เป็น message ที่บอกให้ Window Procedure ทำสิ่งใดสิ่งหนึ่งมิฉะนั้นจะเกิดช่องว่างที่ส่วนที่ติดต่อ กับผู้ใช้ได้

2.3 Query Message : เป็น message ที่ต้องการคำตอบ ใช้เป็นการติดต่อกันระหว่างตัวโปรแกรม ระบบ Windows

3.User-Interface Message : เป็น message เกี่ยวกับส่วนที่ติดต่อกับผู้ใช้ในส่วนอื่นที่ประกอบด้วย menu, mousepointer, scrollbar, dialogbox และ Dialog box Controls นอกจากนี้ยังรวมถึง กลุ่มของ message ที่สนับสนุน MDI (Multiple Document Interface)

Window Messages: Notification

WM_ACTIVATE	Window is active.
WM_ACTIVATEAPP	Application is active.
WM_CREATE	Window has been created.
WM_DESTROY	Window has been destroyed.
WM_ENABLE	Input to the window has been enabled.
WM_KILLFOCUS	Window has lost keyboard control.
WM_MOUSEACTIVATE	Notifies a window that it is going to become active because of a mouse click.
WM_MOVE	Window has been moved.
WM_SETFOCUS	Window has gained keyboard control.
WM_SIZE	Window has changed size.

Window Messages: Request for action

WM_CLOSE	Close (destroy) window
WM_ERASEBKGND	Erase background
WM_ICONERASEBKGND	Erase background of iconic window
WM_NCACTIVATE	Change title bar to show active state
WM_NCCREATE	Create non-client area data
WM_NCDESTROY	Destroy non-client area data
WM_NCPAINT	Redraw non-client area
WM_PAINT	Redraw client area
WM_PAINTICON	Redraw iconic window client area
WM_SETREDRAW	Inhibit redrawing of window
WM_SETTEXT	Change window text
WM_SHOWWINDOW	Change window visibility

Window Messages: Query

WM_GETMINMAXINFO	What are min/max sizes for window?
WM_GETTEXT	What is the window text?
WM_GETTEXTLENGTH	What is the length of the window text?
WM_NCCALCSIZE	How big should the client area be?
WM_QUERYNEWPALETTE	Do you have a new palette?
WM_QUERYOPEN	Can iconic window be opened?

รูป 3.2 แสดง Window maintenance Message

Menu Messages

WM_COMMAND	Menu item has been selected.
WM_INITMENU	Initialize menu bar menu.
WM_INITMENUPOPUP	Initialize popup menu.
WM_MENUCHAR	Mnemonic key used to select menu.
WM_MENUSELECT	User is browsing through menus.

System Commands: System Menu, Min/Max Buttons, Titlebar, etc.

WM_SYSCOMMAND	A system command has been selected.

Mouse Pointer Messages

WM_NCHITTEST	Query: Where is mouse on the window?
WM_SETCURSOR	Request: Change pointer to correct shape.

Scroll Bar Messages

WM_HSCROLL	Horizontal scrollbar has been clicked.
WM_VSCROLL	Vertical scrollbar has been clicked.

Dialog Box and Dialog Box Control Messages

WM_COMMAND	Control communicating with Dialog Box.
WM_COMPAREITEM	Sent to the parent of an owner-draw dialog box control, asking to compare two items for the purpose of sorting.
WM_CTLCOLOR	Control asking for colors to be set.
WM_DELETEITEM	Notification to an owner-draw listbox or an owner-draw combobox that an item has been deleted.
WM_DRAWITEM	Request to the parent of an owner-draw control, or owner-draw menu, to draw.
WM_GETDLGCODE	Query control: Want keyboard input?
WM_GETFONT	Query control: What font are you using?
WM_INITDIALOG	Initialize dialog.
WM_MEASUREITEM	Request to the parent of an owner-draw control or an owner-draw item to provide the dimensions of the item that is going to be drawn.
WM_SETFONT	Request to control: Use this font.

รูป 3.3 แสดง User Interface Message

Multiple Document Interface Messages

WM_CHILDACTIVATE	Notifies a parent window that a child is active.
WM_MDIACTIVATE	Notifies an MDI child window that it is either gaining or losing activation.
WM_MDICASCADE	Request to arrange the open MDI child windows in a cascading, stair-step fashion.
WM_MDICREATE	Requests an MDI client window to create an MDI child window.
WM_MDIDESTROY	Request to an MDI client window to destroy an MDI child window.
WM_MDIGETACTIVE	Query an MDI client window for the currently active MDI child window.
WM_MDIICONARRANGE	Request to arrange the iconic MDI child windows in an orderly fashion.
WM_MDIMAXIMIZE	Request to maximize, or zoom, an MDI child window so that it occupies all of its parent's client area.
WM_MDINEXT	Request to activate the next MDI child window.
WM_MDIRESTORE	Request to restore an MDI child window to its previous state--iconic, normal, or zoomed.
WM_MDISETMENU	Adjusts the menu on an MDI frame window.
WM_MDITILE	Request to arrange the open MDI child windows in a tiled fashion in the MDI parent's client window.

รูป 3.3 ต่อ

4. Termination Message : เป็นกลุ่มของ message ที่เล็กที่สุด แต่ก็มี ความสำคัญมาก เพราะใช้ควบคุมการจบการทำงานของโปรแกรมของระบบด้วย

5. Private Message : เป็น message สำหรับใช้หน้าต่างที่มีชนิดพิเศษ Private Message ที่กำหนดให้ Windows.h ถูกกำหนดขึ้นมาเพื่อให้ใช้กับชนิดของหน้าต่างต่อไปนี้ คือ edit button listbox และ combo box

เราจะเห็นว่า message ทั้ง250 ตัวที่กำหนดขึ้นมาใน Windows.h กำหนดขึ้นมาเพื่อใช้ติดต่อระหว่างหน้าต่างต่าง ๆ ที่เราสร้างขึ้นมา และจากชนิดของ message ซึ่งถูกนิยามเป็น Unsigned Integer แสดงให้เห็นว่า เราสามารถที่จะกำหนด message ต่าง ๆ กันได้ 65,535 messages เนื่องจาก message ที่ถูกกำหนดมาแล้วมีเพียง 250 messages เท่านั้น เราจึงสามารถสร้าง message ที่ใช้เฉพาะงานพิเศษของเราขึ้นมาได้เอง

6. System Resource Notification : เป็น message ที่สร้างขึ้นมาเมื่อมีการเปลี่ยนแปลงของทรัพยากร (Resource) ภายในระบบ การตอบรับ message เหล่านี้ โดยมากจะเป็นการบันทึกถึงการเปลี่ยนแปลง

7. Data Sharing Message : เป็น message ที่สร้างขึ้นมาเพื่อใช้ในระบบการแลกเปลี่ยนข้อมูลระหว่างโปรแกรม (DDE : Dynamic Data Exchange) ซึ่งถือว่าเป็นหัวใจสำคัญของระบบ Windows

ตาราง 3.1 แสดง Termination Message

Application and System Termination	
WM_QUIT	Request that a program should terminate.
WM_QUERYENDSESSION	A Query: Ready for system shutdown?
WM_ENDSESSION	Notification of results of shutdown query.

ตาราง 3.2 แสดง System Resource Notification Message

System Resource Notification Messages	
WM_COMPACTING	Notification that system memory is low,and that the Memory Manager is trying to free up some memory.
WM_DEVMODECHANGE	Printer setup has changed.
WM_FONTCHANGE	Installed fonts in the system have changed.
WM_PALETTECHANGED	Hardware color palette has changed.
WM_SPOOLERSTATUS	Job has been removed from spooler queue.
WM_SYSCOLORCHANGE	One or more system colors has changed.
WM_TIMECHANGE	System time has changed.
WM_WININICHANGE	Initialization file, WIN.INI ,changed.

Clipboard Messages	
WM_ASKCBFORMATNAME	Asks for the name of a Clipboard format.
WM_CHANGECBCHAIN	Notification of a change in the viewing chain.
WM_DESTROYCLIPBOARD	Clipboard contents are being destroyed.
WM_DRAWCLIPBOARD	Clipboard contents have changed.
WM_HSCROLLCLIPBOARD	Horizontal scrolling of owner draw clipboard item.
WM_PAINTCLIPBOARD	Requests drawing of an owner draw clipboard item.
WM_RENDERALLFORMATS	Request to provide the data for all clipboard formats that have been promised.
WM_RENDERFORMAT	Request to provide data for a single clipboard format that has been promised.
WM_SIZECLIPBOARD	Notification to the owner of owner draw clipboard data that the size of the Clipboard viewer window has changed.
WM_VSCROLLCLIPBOARD	Vertical scrolling of an owner draw clipboard item.
Dynamic Data Exchange(DDE) Messages	
WM_DDE_ACK	Acknowledgment.
WM_DDE_ADVISE	Request from a DDE client to establish a permanent data link.
WM_DDE_DATA	Send a data item from a DDE server to a DDE client.
WM_DDE_EXECUTE	Request a DDE server to execute a series of commands.
WM_DDE_INITIATE	Logon to a DDE server.
WM_DDE_POKE	Request by a client for a server to update a specific data item.
WM_DDE_REQUEST	One-time request by a DDE client for a piece of information.
WM_DDE_TERMINATE	Logoff from a DDE server.
WM_DDE_UNADVISE	Terminate a permanent data link that was initiated with the WM_DDE_ADVISE message.

รูป 3.4 แสดง Data Sharing Messages

บทที่ 4
Graphic Device Interface

จากที่กล่าวมาแล้วว่าในระบบ Windows การแสดงผลจะแสดงใน Graphic Mode โดยผ่านทาง GDI (Graphic Device Interface) ซึ่งช่วยให้เราสามารถที่จะสร้างผลลัพธ์ โดยไม่ขึ้นกับว่าระบบของเราในขณะนั้นมีอุปกรณ์อะไรต่ออยู่

GDI คือ ส่วนที่เก็บ โปรแกรมย่อย (Routine) ที่ใช้แสดงผลแบบ Graphic ตัว GDI จะจัดการแสดงผลออกทางจอภาพ พรินเตอร์ พลอตเตอร์ ฯลฯ GDI สามารถสร้างเส้นตรง ตัวอักษรทรงกลม ให้ปรากฏบนอุปกรณ์ที่ GDI รู้จักได้

4.1 GDI Device

GDI สามารถที่จะวาดรูปลงไปในอุปกรณ์ได้หลาย ๆ ชนิด เช่น Laser Printer, Dot Metrix, plotter ฯลฯ โดยที่ GDI จะทำงานกับอุปกรณ์เหล่านี้ผ่านทาง Software ตัวหนึ่งเรียก Device Driver ซึ่งจะทำหน้าที่แปลงคำสั่งในการวาดรูปให้เป็น การกระทำ (Action) ที่อุปกรณ์จะต้องทำจริง

Device Driver จะบอกถึงความสามารถของอุปกรณ์ตัวนั้นให้กับ GDI ซึ่งประกอบด้วยการวาดเส้นตรง เส้นโค้ง รูปหลายเหลี่ยม รูปภาพ และตัวอักษร โดยอย่างน้อยอุปกรณ์ของ GDI จะต้องมีความสามารถที่จะทำ 2 สิ่งนี้ได้คือ กำหนดจุด และลากเส้นตรง ด้วยความสามารถนี้ ความสามารถอย่างอื่นก็สามารถใช้ความสามารถขั้นพื้นฐานนี้มาประกอบกันได้ ถ้าอุปกรณ์มีความสามารถมากเวลาใช้ในการวาดบนอุปกรณ์นั้นก็จะน้อยลง เพราะไม่ต้องเสียเวลาแตกคำสั่งเป็นการกระทำย่อย ๆ

เนื่องจาก GDI ติดต่อกับอุปกรณ์ต่าง ๆ หลายชนิด ผ่านทาง Device Driver ดังนั้นการติดต่อกับ GDI จึงเป็นการติดต่อแบบที่ไม่ขึ้นอยู่กับชนิดของอุปกรณ์ที่ต่ออยู่ เพราะ GDI จะเป็นผู้เลือก Device Driver ให้เหมาะสมกับอุปกรณ์ตัวนั้นเอง

นอกจาก GDI สามารถที่จะติดต่อกับอุปกรณ์ที่มีอยู่จริงได้แล้ว (เช่น printer, จอภาพ ฯลฯ) GDI ยังสนับสนุนอุปกรณ์เทียม (pseudo-device) อีกด้วย อุปกรณ์เทียมเหล่านี้ใช้สำหรับเก็บรูป ซึ่งไม่เหมือนกับอุปกรณ์ที่มีอยู่จริง อุปกรณ์เทียมจะเก็บรูปภาพเหล่านี้ลงบนหน่วยความจำ RAM หรือ Disk GDI สนับสนุนอุปกรณ์เทียม 2 ชนิด คือ Metafile และ Bitmap

ในระบบ windows Bitmap จะเก็บรูปภาพในหน่วยความจำ Bitmap จะช่วยให้

สามารถ Copy รูปภาพได้เร็วขึ้นดังนั้น Bitmap จึงใช้เก็บรูปภาพที่จะต้องมีการวาดเร็วๆ บนจอภาพ เช่น icon, cursor ฯลฯ

สำหรับ Metafile จะเก็บรูปภาพไว้ใน Disk สามารถนำกลับมาใช้ใหม่ได้ จะช่วยประหยัด memory แต่เวลาเรียกใช้จะช้ากว่าแบบ Bitmap ตัวอย่างการใช้ของ Metafile เช่น Clipboard จะใช้ Metafile ในการส่งข้อมูลไปมาระหว่างโปรแกรมต่าง ๆ

4.2 การติดต่อกับโปรแกรม (The Programming Interface)

ถึงแม้ว่า GDI สนับสนุนอุปกรณ์ที่ต่อรอบข้างมากมาย แต่ GDI ก็มี Routine เดียวที่ใช้จัดการเกี่ยวกับการแสดงผลทางอุปกรณ์เหล่านั้น เช่น SetPixel ใช้วาดจุด 1 จุดลงบนอุปกรณ์ทุกชนิดที่ GDI สนับสนุนอยู่ การที่จะเข้าใจถึงการแสดงผลลัพธ์ ทางอุปกรณ์ที่ GDI สนับสนุนจำเป็นจะต้องเข้าใจหัวข้อต่อไปนี้

1. **Drawing Coordinate :**

GDI อนุญาตให้เรากำหนดระบบพิกัดต่างๆ ได้มากมาย ตามแต่เราต้องการในการควบคุมการวาด แสดงผลลัพธ์ ดังแสดงในรูป GDI.1

GDI จะใช้คำว่า "Mapping Mode" ในการแสดงถึงระบบพิกัดที่แตกต่างกัน แต่มีระบบพิกัดหนึ่งที่น่าสนใจ คือ MM_TEXT หน่วยที่อ้างถึงคือ Pixel คือ องค์ประกอบในการวางรูป (Picture Element) ที่เล็กที่สุดที่อุปกรณ์นั้นสามารถวาดได้ ใน mode นี้สามารถควบคุมการแสดงผลลัพธ์ได้อย่างแม่นยำที่สุด นอกจากนี้ในระบบนี้ยังตรงกับระบบพิกัดที่ใช้อ้างตำแหน่งของ mouse อีกด้วย

ในระบบพิกัดนี้จะแตกต่างจากระบบ Cartesian ตรงที่ เมื่อเคลื่อนที่ลงมาข้างล่าง ค่าของแกน y จะเพิ่มขึ้นเรื่อย ๆ (ในระบบนี้อาจเรียกว่า "Client Area Coordinate")

ตารางที่ 4.1 แสดงระบบพิกัดต่าง ๆ ที่สามารถอ้างได้ใน GUI

Mapping Mode Name	1Logical Unit	Inches	Millimeters
MM_TEXT	1pixel	-	-
MM_HIMETRIC	0.01 mm	0.000394	0.01
MM_TWIPS	1/1440 inches	0.000694	0.0176
MM_HIENGLISH	0.001 inches	0.001	0.0254
MM_LOMETRIC	0.1 mm	0.00394	0.1
MM_LOENGLISH	0.01 inches	0.01	0.254
MM_ISOTROPIC	}Scaling based on ratio between two DC		
MM_ANISOTROPIC	}attribute values: window and viewport extents.		

2. Logical Drawing Object :

การที่ GDI สามารถจะแสดงผลลัพธ์โดยไม่ขึ้นกับอุปกรณ์ได้นั้น GDI จะอาศัยอุปกรณ์ที่เรียกว่า "Logical Drawing Object" ซึ่งเป็นอุปกรณ์ที่เราสร้างขึ้นเพื่อบรรยายลักษณะของ o/p ที่ต้องการ อุปกรณ์เหล่านี้ได้แก่ pen: ใช้สำหรับลากเส้น Brush: ใช้สำหรับระบายสี Font: ใช้สำหรับแสดงตัวอักษร Logical Color: ใช้สำหรับบรรยายลักษณะของสี

เมื่ออุปกรณ์เหล่านี้ถูกสร้างขึ้น เราสามารถใช้อุปกรณ์เหล่านี้กับอุปกรณ์ใด ๆ ก็ได้ แต่แน่นอนผลลัพธ์ที่ได้จะแตกต่างกัน เช่น Red Pen: จะลากเส้นสีแดงบนจอ EGA แต่จะลากเส้นดำบนเครื่องพิมพ์แบบ Dot Matrix ดังนั้นจึงเป็นหน้าที่ของ Device Driver ที่จะแปลง Logical Drawing Obfect นี้ให้เหมาะสมกับความสามารถของอุปกรณ์เหล่านั้น

3. Device Context:

Device Context เป็นการรวมของหน้าที่, Attribute, ข้อมูลต่าง ๆ ของอุปกรณ์แสดงผล เข้ามาอยู่ภายในที่เดียวกัน โดยปกติแล้วระบบ Windows จะไม่ยอมให้เราเข้าถึง Device Context นี้ได้โดยตรง ดังนั้นระบบ Windows จึงใช้ Handle เป็นตัวกำหนดถึง Device Context ที่ต้องการ หน้าที่ของ Device Context มีดังนี้

3.1 เป็นกล่องเครื่องมือ (Toolbox)

Device Context เป็นตัวเก็บลักษณะต่าง ๆ ของการวาด (Drawing Attribute) ประกอบด้วย pen, brush และ font เราสามารถเปลี่ยนอุปกรณ์เหล่านี้ได้ เช่น ปากกาสีแดง แปรงสีเหลือง เป็นต้น

เมื่อมีการเรียกใช้ Routine ของ GDI ในการแสดงผลทางอุปกรณ์ เช่น TexOut Routine เหล่านี้จะนำ Attribute ต่าง ๆ ภายใน Device Context นี้มาใช้ในการแสดงผลทางอุปกรณ์นั้น

Drawing Attribute	Default Value	Lines	Filled Areas	Text	Raster	Comments
Background Color	White	x	x	x		Styled pen, hatch brush
Background Mode	OPAQUE	x	x	x		On/Off switch
Brush Handle	White Brush		x		x	Filled areas
Brush Origin	(0, 0)		x		x	hatch and dithered brushes
Clipping Region Handle	Entire Surface	x	x	x	x	
Color Palette Handle	Default Palette	x	x	x		
Current Pen Position	(0, 0)	x				For LineTo routine
Drawing Mode	R2_COPYPEN	x	x			Boolean mixing
Font Handle	System Font			x		
Intercharacter Spacing	0			x		
Mapping Mode	MM_TEXT	x	x	x	x	One unit = 1 pixel
Pen Handle	Black Pen	x	x			
Polygon-Filling Mode	Alternate		x			For polygon routine
Stretching Mode	Black on White				x	For StretchBlt routine
Text Alignment	Left & Top			x		
Text Color	Black			x		
Viewport Extent	(1, 1)	x	x	x	x	Coordinate mapping
Viewport Origin	(0, 0)	x	x	x		Coordinate mapping
Window Extent	(1, 1)	x	x	x	x	Coordinate mapping
Window Origin	(0, 0)	x	x	x	x	Coordinate mapping

รูป 4.1 แสดง Attribute ต่าง ๆ ที่เก็บไว้ใน Device Context

3.2 เชื่อมต่อโปรแกรมเข้ากับอุปกรณ์ที่ต้องการจะวาด

โดยที่การต่อกันนี้เป็นการต่อแบบ Logical (คือไม่ใช้ต่อกันจริง ๆ) เนื่องจากระบบ Windows เป็นระบบที่สามารถรันโปรแกรมหลาย ๆ โปรแกรม พร้อมกัน ดังนั้นจึงต้องมีการจัดการเกี่ยวกับทรัพยากรต่าง ๆ ของระบบ ซึ่งรวมถึงอุปกรณ์ที่แสดงผลด้วย เพื่อรักษาให้ข้อมูลบนอุปกรณ์แสดงผลตรงกับความเป็นจริง โปรแกรมจะติดต่อกับอุปกรณ์แสดงผลได้โดยทาง Device Context

ดังนั้น Device Context จึงเป็นเสมือนกับใบอนุญาต (Permission slip) ซึ่งโปรแกรมจะต้องมีก่อนที่จะติดต่อกับอุปกรณ์แสดงผลได้ การติดต่อระหว่างโปรแกรมและอุปกรณ์แสดงผลมีอยู่ 2 วิธี ขึ้นอยู่กับชนิดของอุปกรณ์ที่ติดต่อคือ ถ้าเป็นอุปกรณ์จำพวก Hardcopy (ได้แก่ Printer, Plotter ฯลฯ) จะใช้การ Spooling ซึ่งเป็นเทคนิคที่ผลลัพธ์จะถูกนำลงใน Disk(Spool) ก่อนที่จะแสดงออกทางอุปกรณ์แสดงผล เป็นการป้องกันมิให้ ผลลัพธ์จากหลาย ๆ โปรแกรมมารวมกัน

ถ้าเป็นอุปกรณ์พวกจอภาพระบบ Windows จะใช้ระบบอนุญาตที่เรียกว่า "Clipping" ซึ่ง Clipping นี้จะเป็นเสมือนกรอบที่ผลลัพธ์ของโปรแกรมจะแสดงได้ โดยที่ระบบ Windows จะมีการ Check Clipping นี้ทุกครั้งที่มีการเรียกใช้ GDI ให้แสดงผลลัพธ์ โดยพื้นที่ส่วนนี้จะถูกเก็บไว้ใน Device Context นั่นเอง ตัวโปรแกรมจึงไม่ต้องทำหน้าที่ในการตรวจสอบว่าสามารถแสดงผลลงบนพื้นที่ส่วนนี้ได้หรือไม่ แต่จะเป็นหน้าที่ของ Windows Manager แทน (ซึ่งเป็นส่วนหนึ่งของระบบ Windows) จึงทำให้โปรแกรมหลาย ๆ โปรแกรมใช้จอภาพร่วมกันได้

4. Clipping and the Window Manager:

จากหัวข้อที่แล้วจะเห็นว่างานที่จัดการเกี่ยวกับการกำหนดขอบเขตการแสดงผลให้กับโปรแกรม จะถูกทำโดยส่วนระบบ Windows Manager (ซึ่งเป็นส่วนที่รับผิดชอบเกี่ยวกับส่วนที่ติดต่อกับผู้ใช้ซึ่งส่วนนี้จะอยู่บนจอภาพ)

Windows Manager จะมีชุดของ Device Context สำหรับการแสดงผลบนอุปกรณ์จอภาพอยู่ชุดหนึ่ง เมื่อโปรแกรม ต้องการวาดรูปบนหน้าต่าง โปรแกรมจะต้องยืม Device Context นี้จาก Program Manager

ก่อนที่ Program Manager จะให้โปรแกรมยืม Device Context ตัว Program Manager จะทำการกำหนดพื้นที่ Clipping ก่อน มี Routine อยู่ 3 ชุดที่โปรแกรมสามารถใช้ในการยืม Device Context จาก Windows Manager โดยที่แต่ละ Routine จะมีพื้นที่ Clipping แตกต่างกัน Routine เหล่านี้แสดงอยู่ใน ตารางที่ 4.2

ตารางที่ 4.2 แสดง Routine ที่ใช้ในการยืม Device Context

Borrowing Routine	Returning Routine	Description
BeginPaint	EndPaint	Clip to invalid part of client area
GetDC	ReleaseDC	Clip to entire client area
GetWindowDC	ReleaseDC	Clip to entire Window (Client and non_client areas)

Routine แรกคือ BeginPaint เป็น Routine ที่ใช้ตอบสนองต่อ คำเรียกร้องจาก Windows Manager ที่ให้โปรแกรมซ่อมแซมส่วนที่เสียหาย ซึ่งเป็นส่วนหนึ่งของหน้าต่างของโปรแกรมนั้นโดยที่ Windows Manager จะส่ง message WM_PAINT ออกมา เพื่อให้โปรแกรมรู้ว่า หน้าต่างต้องถูกทำการซ่อมแซม

Routine ที่สองคือ GetDC Routine นี้จะยอมให้โปรแกรมแสดงผลลัพธ์บนส่วนที่เรียกว่า Client Area ของหน้าต่างนั้นได้ โดยปกติ Routine นี้จะไม่อยู่ใน message WM_PAINT

Routine ที่สามคือ GetWindowDC เป็น Routine ที่ยอมให้มีการเขียนตรงส่วนไหนของหน้าต่างก็ได้ (รวมถึงส่วนที่เรียกว่า Non-Client Area) เป็น Routine ที่ถูกเรียกโดย Windows Manager เพื่อใช้ซ่อมแซมพื้นที่ส่วน Non-Client Area ของหน้าต่างของโปรแกรม

Routine ทั้งสามนี้ เมื่อมีการยืม Device Context แล้ว ก็จะต้องคืน Device Context นี้ให้กับ Windows Manager เพื่อให้ Windows Manager สามารถให้โปรแกรมอื่นยมใช้หรือ ตัว Window Manager เองใช้ในการซ่อมแซม ดูแลรักษาการแสดงบนจอภาพให้ถูกต้อง ถ้าโปรแกรมยืม Device Context แล้วไม่คืน ก็จะทำให้การดูแลการแสดงผลของ Window Manager ไม่สามารถทำงานต่อไปได้ ยังผลให้ระบบหยุดการทำงานได้

บทที่ 5

MENU

5.1 COMMAND

ในระบบ Windows จะสนับสนุนส่วนที่ต้องติดต่อกับผู้ใช้ในการรับคำสั่งจากผู้ใช้ 2 ที่คือ Menus และ accelerators Menus คือ รายการของสิ่งที่โปรแกรมสามารถทำได้ หรือข้อเลือก (Option) ต่าง ๆ นอกจากนี้ยังเป็นส่วนที่แสดงความสามารถทั้งหมดของโปรแกรมด้วย Menus ช่วยให้ผู้ใช้ไม่ต้องจำคำสั่ง แต่สำหรับผู้ใช้ที่ต้องการความรวดเร็วก็สามารถใช้ประโยชน์จาก accelerators ซึ่งจะเป็นตัวแปลผลจากการกด keyboard ให้ เป็นคำสั่งตามที่ผู้ใช้ต้องการ และเพื่อเป็นการลดช่องว่างระหว่าง Menu และ accelerator โปรแกรมส่วนมากจะแสดง accelerator key ภายใน Menu ด้วย

5.2 มาตรฐานในการติดต่อกับผู้ใช้

สิ่งที่ผู้ใช้จะพบใน Menu มี 2 อย่างคือ action และ option โดยที่ action คือการกระทำที่เราต้องการให้โปรแกรมนั้นแสดง เช่น เปิดไฟล์ แสดงผลทาง Printer เรียงตัวอักษร ฯลฯ และ option คือ ทางเลือกที่ผู้ใช้สามารถเลือกได้จาก Menu option จะมีผลในระยะยาวต่อการทำงานซึ่งต่างกัน action ที่จะมีผลสั้น ๆ ในช่วงเวลาหนึ่ง เท่านั้น โดยมาก option จะมีเครื่องหมายแสดงว่า option นั้นทำงาน (active) อยู่

สัญลักษณ์ต่าง ๆ ที่จะพบมากใน Menu ของระบบ Windows ซึ่งถือเป็นมาตรฐานในการติดต่อกับผู้ใช้มีดังนี้

- Accelerator Keystroke จะบอกให้ผู้ใช้ทราบถึง keyboard ที่จะใช้ให้เกิดผลลัพธ์ เช่นเดียวกับการเลือก Menu นั้น
- Arrow จะแสดงว่า Menu ตัวนั้น ยังประกอบด้วย Menu ย่อยลงไปอีก
- Separator จะแบ่ง Menu ที่มีขนาดยาวมาก ๆ ให้เป็นกลุ่มย่อย
- Check-Mark แสดงถึงว่า option นั้นกำลังทำงานอยู่
- Ellipsis หลังจากที่เลือก Menu ตัวนี้จะเกิด dialog box ขึ้นมาเพื่อให้ใส่ข้อมูลลงไป

- Exclamation Point จะแสดงว่า Menu ตัวนั้นไม่มี Menu popup อีก
- Underline Letter (ตัวอักษรที่ถูกขีดเส้นใต้ภายในชื่อของ Menu ตัวนั้น) ถ้ามีการกดตัวอักษรตัวนั้นพร้อมกัน Alternate จะทำให้มี Popup Menu ขึ้นมา เมื่อเกิด Popup Menu แล้ว ถ้าเรากดตัวอักษรที่ขีดเส้นใต้ใน Menu ที่ขึ้นมาเท่ากับเราเลือกให้ Menu ตัวนั้นทำงานต่อไป

โดยปกติแล้ว ทุก ๆ โปรแกรม จะต้องมี System Menu อยู่ด้วยเสมอ System Menu นี้จะเป็น Menu ที่แสดงขึ้นมาเมื่อโปรแกรมอยู่ในสภาวะ icon โดยทั่วไปโปรแกรมไม่ควรที่จะไปแก้หัวข้อต่าง ๆ ภายใน System Menu แต่สามารถที่จะเพิ่มหัวข้อเข้าไปใน System Menu ได้

5.3 การสร้าง Menu

เนื่องจากระบบ Windows เป็นระบบที่ติดต่อกับผู้ใช้ผ่านทาง Menu จึงไม่ใช่เรื่องแปลกเลยที่การสร้าง Menu เป็นเรื่องง่ายสำหรับ Programmer วิธีที่จะสร้าง Menu อย่างง่ายที่สุดคือการสร้างคำอธิบายเกี่ยวกับ Menu (Menu Description) ไว้ใน Resource File เมื่อ Windows Class ถูกกำหนด และหน้าต่างนั้นถึงสร้างขึ้น Menu จะถูกสร้างขึ้นมาด้วย สิ่งที่เหลือก็เพียงแต่จัดการกับ message WM_COMMAND ที่ Menu ส่งออกมาให้กับ Window Procedure เพื่อแสดงว่า ผู้ใช้ได้ทำการเลือก Menu แล้ว

ตาราง 5.1 แสดงขั้นตอนต่าง ๆ ที่เกี่ยวกับการใช้ Menu

Menu Operation	Mouse Action	Keyboard Action	Message
Initiate menu use	n/a	F10 or Alt key	WM_INITMENU
Display a popup	n/a	arrow or mnemonic keys	WM_INITEMENUPOPU
Initiate and Display popup	<Click>	Alt+menemonic key	WM_INITMENU and WM_INITMENUPOPUP
Browse a menu item	<Drag>	arrow keys	WM_MENUSELECT
Select a menu	<Release>	Enter key or mnemonic key	WM_COMMAND

message ที่สำคัญที่เกี่ยวข้องเมื่อ เราเลือกใช้ Menu มีอยู่ 3 message คือ

WM_INITMENU message นี้จะถูกส่งออกมาก่อนที่ popup Menu จะถูกสร้างขึ้น ดังนั้นถ้าเราจัดการ message นี้ เราสามารถที่จะกำหนดสภาวะค่าเริ่มต้น ให้กับ Menu ต่าง ๆ ภายใน popup Menu นั้นได้

WM_MENUSELECT บอกโปรแกรมถึง Menu ที่ผู้ใช้เลือกในแต่ละครั้งที่มีการเลือก Menu ใหม่โดยมาก message นี้จะใช้เพื่อ แสดงรายละเอียด คำอธิบายเกี่ยวกับ Menu นั้น ในส่วน Information Area

WM_COMMAND เป็น message ที่สำคัญที่สุดในการจัดการเกี่ยวกับ Menu ต่าง ๆ ภายในระบบ Windows เป็น message ที่ Menu ส่งให้ Windows Procedure ทุกครั้งที่มีการเลือก Menu นั้น ภายในส่วนของ message นี้ จะเป็นขั้นตอนที่จะต้องทำเพื่อตอบสนอง ต่อการเลือก Menu ของผู้ใช้

5.4 ต้นแบบของ MENU

ต้นแบบของ Menu (Menu Templete) จะกำหนด popup และ หัวข้อต่าง ๆ ของ Menu ที่ประกอบกันขึ้นมาเป็นระบบ Menu ของโปรแกรม ต้นแบบ Menu นี้จะมีโครงสร้างเป็นแบบลำดับชั้น (Hierarchy) โดยที่ส่วนที่อยู่ชั้นสูงสุด คือหัวข้อต่าง ๆ ของ Menu ที่แสดงอยู่บน Menu bar (บางครั้งเรียกว่า Active bar) Menu ที่แสดงในชั้นนี้อาจเป็นหัวข้อของ Menu เลย หรือ เป็นชื่อของ Popup Menu (หรือ บางทีเรียกว่า Pull down menu) Pupup Menu เป็นอีกระดับหนึ่งซึ่งต่ำลงมาจากระดับแรก เช่นเดียวกับระดับแรก ในระดับนี้สามารถที่จะมี pupup Menu ในระดับย่อยลงไปอีก โปรแกรมสามารถที่จะมี popup Menu ลงไปกี่ชั้นย่อยก็ได้ แต่โดยปกติจะนิยมให้ Menu มีระดับย่อยลงไปแค่ 3 ชั้นเท่านั้น มิฉะนั้นผู้ใช้อาจเกิดการสับสนในการเลือกใช้ Menu ได้

รูปแบบในการกำหนดต้นแบบของ Menu ใน Resource File จะมีลักษณะดังนี้

```
MenuID          MENU [Load Option] [Menu Option]
BEGIN
    MENUITEM or POPUP Statement
        :
        :
    MINUITEM  or POPUP Statement
END
```

[Load Option] เป็นข้อเลือกเกี่ยวกับการเรียกใช้ Menu ลงใน memory เช่น PRELOAD (หัวข้อ Menu จะถูกเรียกขึ้นมาก่อนที่จะรันโปรแกรม) LOADONCALL (หัวข้อ Menu จะถูกเรียกก็ต่อเมื่อต้องการใช้เท่านั้น ฯลฯ

[Memory Option] เป็นข้อเลือกที่จะให้ระบบ Windows จัดการหน่วยความจำ Menu อย่างไร เช่น FIXED, DISCARD, MOVEABLE ฯลฯ สำหรับค่า default ของระบบคือ Discardable และ LOADONCALL

ประโยค MENUITEM จะกำหนดหัวข้อของ Menu ที่เมื่อถูกเลือกแล้ว Menu จะส่ง message WM_COMMAND ออกมาให้กับ Windows Procedure สำหรับประโยค Popup จะเริ่มต้นด้วย BEGIN และ ลงท้ายด้วย END โดยจะรวมหัวข้อ Menu หรือ popup Menu เอาไว้

รูป 5.1 แสดงการกำหนดต้นแบบของ Menu ในโปรแกรม Resource File

```
1 MENU
 BEGIN
 POPUP  "&Files"
 BEGIN
  MENUITEM  "&Load",IDM_LOAD
 MENUITEM  "&Save", IDM_SAVE
END
 POPUP "&Caharcter"
 BEGIN
   POPUP  "&Font"
   BEGIN
      MENUITEM  "&System", IDM_SYS
      MENUITEM  "&Roman", IDM_ROMAN
      MENUITEM   "s&Cript", IDM_SCRIPT
    END
  END
END
```

ข้อกำหนดของประโยค Popup คือ

POPUP Text [,Option list]

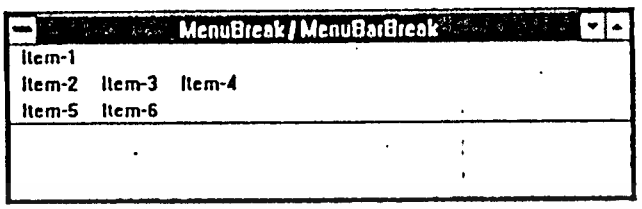
และของ MENUITEM คือ

MENUITEM Text, result-code [,Option list]

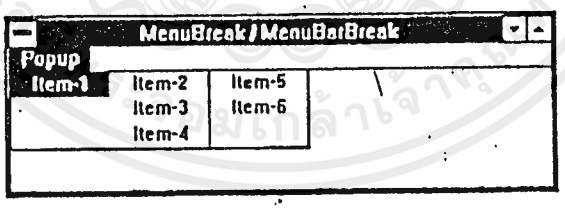
ข้อแตกต่างของทั้งสองคือ ใน MENUITEM จะมี Result Code ซึ่งเป็นตัวที่ใช้บอกความแตกต่างของแต่ละหัวข้อของ Menu แต่ใน Popup จะไม่มี

สำหรับ Option list เป็นข้อเลือกที่จะกำหนดคุณสมบัติของหัวข้อ Menu หรือ popup Menu นั้นจะมี 5 หัวข้อเลือกที่สามารถระบุได้ ดังนี้

- CHECKED จะแสดงเครื่องหมาย Check ในหัวข้อ Menu หรือ popup จะมีผลเฉพาะใน popup Menu เท่านั้น
- GRAYED ชื่อของหัวข้อจะเป็นสีเทา และไม่สามารถเลือกได้
- INACTIVE ชื่อของหัวข้อจะเป็นสีปกติ และไม่สามารถเลือกได้
- MENUBREAK จะแบ่ง Menu ออกเป็นกลุ่มย่อย โดยถ้าเป็น Menu บน Menu bar จะถูกแบ่งในแนวดิ่ง แต่ถ้าเป็นใน popup Menu จะถูกแบ่งในแนวนอน
- MENUBARBREAK จะแบ่ง Menu ออกเป็นกลุ่มย่อย ใน popup Menu จะแบ่งออกใน แนวดิ่ง



รูป 5.1 แสดงการใช้ MENUBARBREAK และ MENUBREAK ใน MENU Bar



รูป 5.2 แสดงการใช้ MENUBARBREAK และ MENUBREAK ใน Popup Menu

การกำหนดต้นแบบใน Resource File เป็นเพียงข้อกำหนดเท่านั้น การที่จะนำเอา Menu มาใช้ในโปรแกรมของเรา และให้ทำงานตามความต้องการ จะมีอยู่ 2 วิธี คือ กำหนดชื่อของ Menu ลงใน Windows Class Structure ใน Field IpszMenuName หรือ อาจจะอ้างถึงเมื่อมีการสร้างหน้าต่างในคำสั่ง CreateWindow การที่จะได้ Handle ของ Menu นั้นเราจะต้องเรียกใช้คำสั่ง LoadMenu เพื่อที่จะได้รับ Handle ของ Menu นั้นก่อนที่จะส่งต่อเป็น Parameter ของ CreatWindow ต่อไป ดังรูป 5.3

รูป 5.3 แสดงการอ้างอิงถึงต้นแบบ Menu ที่สร้างขึ้นใน Resource File

```
hmenu = LoadMunu (hInstance, "#1");
hwnd = CreateWindow("StanMenu:MAIN",/* Class name  */
                    "A Standard Menu", /* Title   */
                    WS_OVERLAPPEDWINDOW, /* Style bits */
                    CW_USEDEFAULT,  /* x - default  */
                    0, /* y - default */
                    CW_USEDEFAULT, /* cx - default  */
                    0,  /* cy - default  */
                    NULL, /* No parent    */
                    hmenu, /* Private menu   */
                    hInstance,  /*  Creator  */
                    NULL); /* Param  */
```

บทที่ 6

DIALOG BOX

Dialog Box เป็น หน้าต่างชนิดหนึ่งที่ใช้สำหรับให้ผู้ใช้ป้อนข้อมูลเพิ่มเติมให้กับโปรแกรมในตัว Dialog Box ยังประกอบด้วยหน้าต่างลูก (Child Window) ซึ่งหน้าต่างเหล่านี้เรียกว่า "Dialog Box Control" ระบบ Windows มี Class ของหน้าต่างอยู่ 6 Class ที่สนับสนุน Dialog Box Control นี้ ได้แก่ Button, Combo Box, Edit, List, Scroll Bar และ Static นอกจากหน้าต่างพวกนี้แล้ว เรายังสามารถสร้างหน้าต่างของเราเองเพื่อใช้เป็น Dialog Box Control ได้อีกด้วย (สามารถสร้างได้ง่ายเหมือนกับที่เราสร้างหน้าต่างของโปรแกรม)

Dialog Box มีอยู่ 2 ชนิด

1.Modal Dialog Box เป็น Dialog Box ที่เมื่อสร้างขึ้นมาแล้ว ผู้ใช้ไม่สามารถติดต่อกับหน้าต่างอื่นได้

2.Modeless Dialog Box จะตรงกันข้ามกับ Modal Dialog Box คือ ผู้ใช้สามารถเลือกติดต่อกับหน้าต่างอื่นได้

ดังนั้นการทำงานภายในของ dialog Box ทั้ง 2 แบบ จึงต่างกัน โดยที่แบบ Modal Dialog จะเปลี่ยนทางเดินของ message จาก mouse และ keyboard ของหน้าต่างพ่อ (Parent Windows) ของ Dialog Box มายัง Dialog Box นั้น ทำให้หน้าต่างพ่อหยุดการทำงานชั่วคราว เปลี่ยนการทำงานให้มาอยู่กับ Modal Dialog Box ตัว Modal Dialog Box นี้จะมี message Loop เป็นของตัวเอง ดังนั้นถ้าโปรแกรมมีการเรียกใช้ Accelerator (คือมีคำสั่ง Translate Accelerator ใน message Loop) ก็ไม่สามารถใช้งานใน Dialog Box แบบนี้ได้เช่นกัน

สำหรับการทำงานภายในของ Modeless Dialog Box จะต่างจาก Modal Dialog Box โดยที่ตัว Dialog Box จะไม่ไปเปลี่ยนทางเดินของ message ที่จะมาที่ Parent Window ดังนั้น Accelerator key ยังสามารถที่จะใช้งานได้อยู่ถึงแม้จะมีการเปิด Dialog Box แบบนี้อยู่ก็ตาม

การใช้งานส่วนมากของ Dialog Box จะเป็นแบบ Modal Dialog Box เพราะ Dialog Box แบบนี้จะให้ผู้ใช้ให้ความสนใจไปในคำสั่งที่เลือกเพียงอย่างเดียว เพื่อที่จะใส่ข้อมูล ให้คำสั่งนั้นทำงานได้อย่างถูกต้องตามต้องการ หรือ ยกเลิกการทำงานของคำสั่งนั้น

DialogBox ถือว่าเป็นส่วนที่ใช้ติดต่อกับผู้ใช้ที่สำคัญที่สุด และ Dialog Box ยังมีชุดของ Dialog Box Control ที่เป็นมาตรฐานอยู่เพื่อให้ผู้ใช้สามารถที่จะเรียนรู้การใช้งานได้อย่างรวดเร็ว

โปรแกรมที่มีการรับข้อมูลในแบบพิเศษซึ่งไม่สามารถใช้ Dialog Box Control ที่ระบบ Windows กำหนดมาให้ได้ ก็สามารถที่จะสร้างหน้าต่างขึ้นมาใหม่ได้ โดยที่ยังสามารถใช้รวมกับที่สร้างมาก่อนแล้วได้ด้วย

จะเห็นว่า Dialog Box Control พวกนี้เป็นเพียงหัวข้อที่ผู้ใช้จะต้องทำการเลือกหรือป้อน message เท่านั้น แต่ไม่สามารถจะจัดการเกี่ยวกับสิ่งที่ผู้ใช้เลือก หรือ ป้อนเข้ามาได้ เราจำเป็นต้องสร้างโปรแกรมย่อยขึ้นมาจัดการในส่วนนี้ต่อไป

6.1 มาตรฐานการติดต่อกับ Dialog Box

แรกสุดในการติดต่อกับ Dialog Box ผู้ใช้ต้องสามารถเลิกการติดต่อกับ Dialog ในเวลาใด ๆ ก็ได้ ดังนั้นทุก ๆ Dialog Box ต้องมีปุ่มกด (Push Botton) ที่เขียนไว้ว่า "Cancel" ผู้ใช้สามารถเลือกปุ่มนี้โดยใช้ Mouse หรือ Esc key เพื่อยกเลิกการทำงานของ Dialog Box นั้น

นอกจากนี้ Dialog Box อาจมีปุ่มอย่างอื่นอีกเช่น OK เพื่อ บอกว่าได้ป้อนข้อมูลครบเรียบร้อยแล้วให้โปรแกรมทำงานตามคำสั่งที่เลือกก่อนที่จะสร้าง Dialog Box อันนี้

นอกจากนี้ยังมี Push Button แบบอื่นที่ผู้ใช้สามารถสร้างขึ้นมาได้ ได้แก่ Push Button ที่มีชื่อของปุ่มนั้น และตามด้วยเครื่องหมาย Ellipsis (...) (เหมือนใน menu) เมื่อกดปุ่มนี้จะเกิด Dialog Box อีกอันหนึ่งแสดงขึ้นมา หรือ ปุ่มที่มีชื่อตามด้วยเครื่องหมาย Chevron (>>) เมื่อกดปุ่มนี้ Dialog Box จะแสดงส่วน Dialog Box Control ที่ซ่อนไว้ออกมา

Push Button แสดงถึงชุดของการกระทำ (Action) ที่ซึ่งผู้ใช้สามารถเรียกใช้ได้จาก Dialog Box เมื่อ Dialog Box ถูกเปิดจะมี Push Button อยู่ปุ่มหนึ่งที่เป็น Default อยู่ (ซึ่งหมายถึงว่า ปุ่มนั้นถูกเลือกอยู่) ถ้าผู้ใช้กด Enter Key Dialog Box ก็จะทำงานตามปุ่มที่ถูกเลือกอยู่ หรือ ถ้าผู้ใช้กด Tab ปุ่มที่ถูกเลือกก็จะเปลี่ยนเป็นปุ่มอื่น

ยังมีชนิดของปุ่มอยู่อีก 2 ชนิด คือ Radio Button และ Check Boxes โดยที่ Radio Button เป็นเสมือนข้อเลือกที่จะมีอยู่เพียงปุ่มเดียวเท่านั้นที่จะถูกเลือก ส่วน Check Box จะเป็นเหมือน Switch ถ้าถูกเลือกก็เปิด ถ้าไม่ถูกเลือกก็ปิด

Edit Control เป็นส่วนที่ผู้ใช้สามารถป้อน message เข้าไปได้ มีอยู่หลายแบบเช่น แบบบรรทัดเดียว หลายบรรทัด ซ่อนตัวอักษรที่ป้อน ฯลฯ

Static เป็น Class ของหน้าต่างที่ใช้แสดง icon, message รูปภาพต่าง ๆ ภายใน Dialog Box

List Box เป็นหน้าต่างที่ใช้แสดงหัวข้อต่าง ๆ ที่สามารถที่จะเลือกได้ โดยมากจะใช้คู่กับ Scroll Bar ในกรณีที่ตัวเลือก เหล่านั้นไม่สามารถแสดงได้พร้อมกัน หัวข้อที่จะเลือกนั้นโดยส่วนมากนักจะเป็นข้อความแต่เราก็สามารถที่จะใช้รูปภาพแทนก็ได้

Combo Box เป็นการรวมเอา Edit Control และ List Box เข้ากับ Static Text Control

6.2 การสร้าง Dialog Box

ในการสร้าง Dialog Box ไม่ว่าเป็น Modal หรือ Modeless ก็ตาม จะต้องประกอบด้วย 3 ส่วนต่อไปนี้ คือ ต้นแบบ Dialog Box, ส่วนที่สร้าง Dialog Box, ส่วนที่ดูแล Dialog Box

6.2.1 ต้นแบบ Dialog Box (Dialoge Template)

ต้นแบบของ dialog Box คือข้อมูลที่ใช้กำหนดขนาดของ Dialog Box ชนิดของการควบคุมใน Dialog Box และตำแหน่งของ Dialog Box Control ต้นแบบของ Dialog Box สามารถสร้างได้ 2 วิธี คือ จาก Graphic Editor (ชื่อว่า Dialog Box Editor) หรือใช้ Text Editor เพื่อบรรยายลักษณะของส่วนประกอบต่าง ๆ ภายใน Dialog Box ผลลัพธ์ของทั้งสองจะได้เหมือนกัน

ซึ่งการบรรยายลักษณะของส่วนประกอบใน Dialog Box นั้นจะคล้ายกับการบรรยายลักษณะของ MENU มาก

ข้อกำหนดของประโยค Dialog มีดังนี้

DialogID DIALOG [Load Option] [Memory Option] x, y, cx, cy

- DialogID เป็นตัวบ่งถึงต้นแบบของ Dialog Box
- [Load Option], [Memory Option] มีข้อเลือก ความหมาย เหมือนใน MenuOption
- x และ y เป็นตำแหน่งของหน้าต่างของ Dialog Box ซึ่งสัมพันธ์กับหน้าต่างพ่อของ Dialog Box นั้น
- cx และ cy เป็นความกว้าง และ ความสูงของ dialog Box

ข้อกำหนดของประโยค Control มีดังนี้

Control <Text>, NID, <Class>, <Styles>, x, y, cx, cy

- <Text> เป็นข้อความที่อยู่ใน Windows ที่สร้างขึ้น
- NID เป็นตัวเลขจำนวนเต็มที่ใช้บ่งถึง Control ตัวนี้
- <Class> เป็นชื่อของ Class ของหน้าต่างของ Dialog Box Control
- <Styles> เป็นลักษณะของหน้าต่างที่จะถูกสร้างขึ้น (มีได้หลายแบบ เช่น มี Horizontal Scroll ลักษณะคล้ายกรอบของหน้าต่าง ฯลฯ
- x และ y เป็นตำแหน่งของ Control ในหน้าต่างของ Dialog Box
- cx และ cy เป็นความกว้าง และ สูงของ Control ในหน้าต่างของ Dialog Box

ข้างล่างนี้คือส่วนของโปรแกรมที่ใช้สร้าง Modal Dialog Box

```
{
        FARPROC lpitAbout

        lpitAbout = MakeProcInstance (AboutDlgProc, hInstance);
        DialogBox (hInstance,
                "About",
                "hWnd,
                lpitAbout);
        FreeProcInstance (ipitAbout);
}
```

หลังจากที่มีการใช้ Dialog Box แล้วจะต้องมีการปล่อย Instance Handle ของ Dialog Box ให้กับโปรแกรมอื่นใช้งานได้ต่อไปโดย Routine FreeProcInstance

6.2.2 การดูแล Dialog Box (Maintaining the Dialog Box)

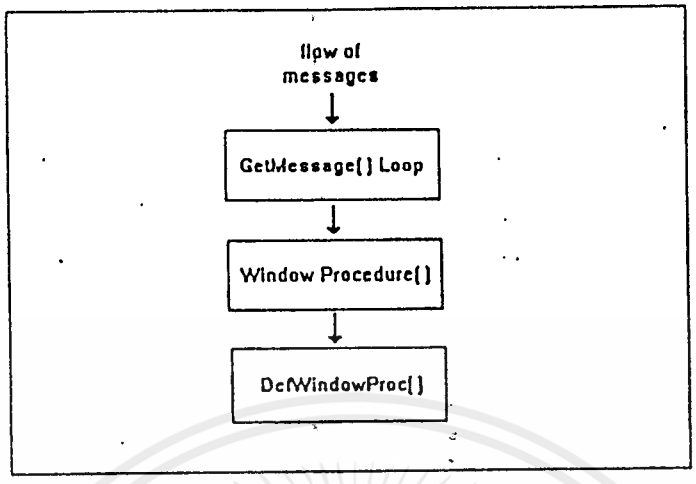
ส่วนที่ใช้ในการดูแล Dialog Box เราเรียกว่า "Dialog Box Procedure" ซึ่ง Dialog Box Procedure เหมือนกับ Window Procedure คือเป็นส่วนที่ใช้จัดการกับ message ดังนั้น เช่นเดียวกับ Window Procedure Dialog Box Procedure จะต้องถูกแสดงในส่วน EXPORT ใน ไฟล์ Module Definition ด้วย เช่น

```
        EXPORTS
                AboutdlgProc
```

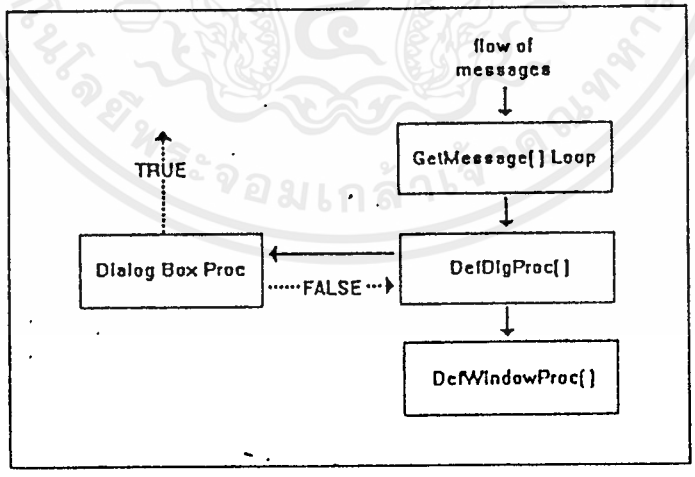
มิฉะนั้นข้อมูลที่ใช้ใน Dialog Procedure จะไม่ถูกต้อง

แต่ Dialog Box Procedure จะไม่ถูกต้อง

แต่ Dialog Box Procedure ก็ไม่ใช่ Window Procedure ดังนั้น การทำงานของทั้ง 2 Procedure นี้จึงแตกต่างกัน



รูป 6.1 แสดงการไหลของ message ใน Windows Procedure ธรรมดา



รูป 6.2 แสดงการไหลของ message ไปยัง Dialog Box Procedure

จากรูปจะเห็นว่า Dialog Box Procedure จะคืนค่ากลับมา 2 ค่าคือ True และ False ถ้าคือค่า True หมายความว่า Dialog Box Procedure ได้ทำงานประมวลผล message นั้นแล้ว ถ้าคืนค่า False แสดงว่า Dialog Box Procedure ยังไม่ได้ประมวลผล message นั้น DefDlgProc จะประมวลผล message นั้นเองถ้าทำได้ มิฉะนั้นก็จะส่ง message นั้นให้ DefWndProc ทำแทน

เพราะ dialog Box Procedure ประมวลผล message เหมือนกับ Window Procedure จึงใช้ Parameter เหมือนกับ Window Procedure ซึ่ง Dialog Box Procedure จะถูกนิยามดังนี้

```
BOOL FAR PASCAL DlgProc (HWND, WORD, WORD, LONG)
```

เพราะว่าเรามี Window Procedure ที่จะจัดการเกี่ยวกับการดูแลหน้าต่างอยู่แล้ว ดังนั้น message ที่ Dialog Box Procedure สนใจจึงมีอยู่ 2 message คือ WM_INIDIALOG และ WM_COMMAND

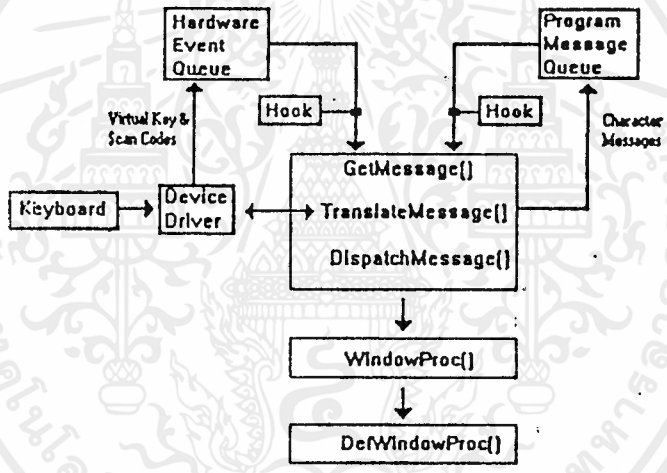
message WM_INITDIALOG ถูกส่งไปที่ Dialog Box Procedure หลังจากที่หน้าต่างของ Dialog Box Control ถูกสร้างเสร็จแล้ว แต่ก่อนที่ถูกแสดงออกมา ในการตอบรับ (Respond) ต่อ message นี้ โปรแกรมจะทำการกำหนดค่าเริ่มต้นของ Dialog Control แต่ละตัวใน Dialog Box

message WM_COMMAND ทำให้ Dialog Box ตอบสนองต่อการเปลี่ยนแปลงของ Dialog Box Control เหมือน message WM_COMMAND ที่ Menu ส่งออกมาเมื่อผู้ใช้มีการเลือก menu นั้น parameter ที่ส่งมาพร้อมกับ message นี้มีอยู่ 2 ชนิด คือ wParam เป็น Parameter ที่มีความยาว 1 word จะบรรจุข้อมูลที่บ่งถึงหัวข้อ (item) ของ control ใน Resource File และ lParam เป็น Parameter ที่มีความยาว 2 word โดยที่ Word บนจะเก็บ Notification Code และ Word ล่างจะเก็บ handle ของหน้าต่างของตัวควบคุมที่ส่ง message นั้นมา

สิ่งสุดท้ายที่ต้องทำหลังจากที่ผู้ใช้ได้ใส่ข้อมูลเรียบร้อยแล้ว หรือผู้ใช้ยกเลิกการทำงานของ Dialog Box คือการทำลาย Dialog Box โดยการเรียกใช้ EndDialog Modal Dialog Box จะถูกลบทิ้งไป และจะคืนการทำงานไปให้กับโปรแกรมที่สร้าง Dialog Box นั้นขึ้นมา

บทที่ 7
Keyboard Input

7.1โปรแกรมในระบบ Windows สามารถรับข้อมูลจาก keyboard ได้อย่างไร



รูป 7.1 แสดงการไหลของข้อมูลจาก Keyboard

ในรูป 7.1 แสดงถึงส่วนต่างๆ ที่เกี่ยวข้องในการรับข้อมูลจาก keyboard ในทุกโปรแกรมจะต้องมีคำสั่ง GetMessage ที่วนลูปคอยดึง message ของ keyboard จาก queue เข้ามาสู่ในโปรแกรมอยู่แล้ว แต่ข้อมูลที่ได้เป็นข้อมูลดิบ จะต้องผ่านกระบวนการอื่นที่แปลงข้อมูลดิบเหล่านั้นให้เป็นตัวอักษรเสียก่อน ซึ่งกระบวนการนี้เราก็มีอยู่แล้วใน Message Loop คือ Routine TranslateMessage

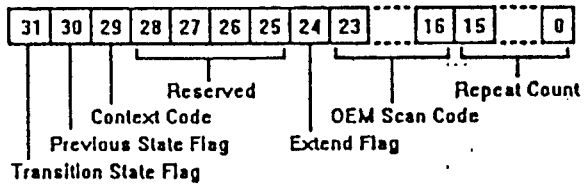
ในระหว่างที่มีการเริ่มใช้งานระบบ Windows ระบบ Windows จะทำการติดตั้งส่วนที่ใช้ในการจัดการเกี่ยวกับการรับ input จาก keyboard (เรียกว่า keyboard driver) ทุกๆ ครั้งที่มีการกดหรือปล่อยคีย์ ก็จะมีการเรียกใช้ส่วนนี้ทุกครั้ง หน้าที่ของส่วนนี้คือรับค่า Scan Code จาก keyboard และแปลงค่า Scan Code นั้นให้อยู่ในชุดของค่าๆ หนึ่ง ซึ่งเรียกว่า Virtual Key Code (หมายถึง ชุดมาตรฐานของคีย์ที่มีอยู่บน keyboard ที่มีขายอยู่ในปัจจุบัน นอกจากนี้ยังมีการกำหนดคีย์ที่ไม่มีอยู่ใน keyboard ที่มีขายอยู่ในปัจจุบันเพื่อไว้ใช้ในอนาคต)

หลังจากที่ keyboard driver ทำการแปลงค่า Scan Code ให้เป็น Virtual Key Code แล้ว ก็จะเรียกระบบ Windows ให้มาทำงานต่อไป ระบบ Windows จะนำค่า Scan Code และ Virtual Key Code ไปเก็บไว้ใน buffer ที่เรียกว่า Hardware Event Queue

จากที่เคยกล่าวมาแล้วว่าระบบ Windows เป็นระบบที่มีการทำงานเป็นแบบ Multitasking ชนิดที่ไม่ไปขัดจังหวะการทำงานของโปรแกรม (nonpreemptive multitasking) ดังนั้น input ทุกอย่างที่จะส่งให้กับโปรแกรมจึงถูกส่งไปในรูปของ message ซึ่งจะถูกเก็บไว้ใน Queue ก่อนที่จะส่งต่อไปให้โปรแกรมนั้น (ในกรณีของ input จาก keyboard message จะถูกเก็บอยู่ใน Hardware Event Queue)

ภายใน Hardware Event Queue จะเก็บ message ที่เกี่ยวกับ keyboard อยู่สองmessage คือ WM_KEYBOARD และ WM_KEYUP จากรูป KEY.1 แสดงให้เห็นว่าโปรแกรมสามารถรับ message เหล่านี้จาก Hardware Event Queue โดยการเรียกใช้ Routine GetMessage

สำหรับความหมายของ message จะถูกส่งมาพร้อมกับพารามิเตอร์สองตัวคือ wParam (เป็นจำนวนเต็มที่มีขนาด 2 ไบท์) จะเก็บค่าของ Virtual Key ของคีย์ถูกกดหรือปล่อย และ lParam (เป็นจำนวนเต็มที่มีขนาด 4 ไบท์) ซึ่งถูกแบ่งออกเป็น 6 ส่วน ตามรูป 7.2



รูป 7.2 แสดงส่วนต่างในพารามิเตอร์ lParam

รายละเอียดของแต่ละส่วนมีดังนี้

- Repeat Count เป็นความสามารถของ Keyboard HardWare ที่สามารถจะให้ตัวอักษรเกิดซ้ำกันได้โดยอัตโนมัติเมื่อมีการกดคีย์นั้นค้างไว้ในช่วงเวลาหนึ่ง เพื่อป้องกันมิให้เกิดการล้น (overflow) ใน Hardware Event Queue เนื่องจากสาเหตุนี้ ระบบ Windows จะทำการเพิ่มค่าของ Repeat Count ขึ้นหนึ่งทุกครั้งที่พบว่าคีย์ที่ส่งเข้ามาใหม่เหมือนกับคีย์ที่ส่งมาก่อนหน้านั้น และระบบ Windows จะทำการรวม message WM_KEYDOWN ใน Hardware Event Queue ให้เหลือเพียง message เดียว
- OEM Scan Code ใช้เก็บค่าของ Scan Code ที่ได้จาก Keyboard
- Extend Flag เป็นส่วนที่เพิ่มเติมจาก OEM Scan Code เพื่อแสดงว่าคีย์ที่ถูกกดนั้นเป็นคีย์ที่ตรงกับคีย์บน IBM'S Extended Keyboard
- Context Code มีค่าเป็น 1 เมื่อมีการกดคีย์ Alt ค้างอยู่
- Previous Key State ใช้ยกสถานะของคีย์ก่อนที่จะถูกกดซ้ำ มีค่าเป็น 1 เมื่อสถานะก่อนหน้าของคีย์เป็นการกด
- Transition State มีค่าเป็นหนึ่งเมื่อคีย์ถูกปล่อย เป็นศูนย์เมื่อคีย์ถูกกด

ในทุกๆ Message Loop จะต้องประกอบไปด้วย Routine 2 Routine คือ GetMessage และ TranslateMessage message ทุกๆ message จะไม่ถูกนำมาประมวลผลใน Routine TranslateMessage ยกเว้น WM_KEYDOWN และ WM_SYSKEYDOWN หน้าที่ของ Routine นี้คือทำการแปลง Virtual Key Code ให้เป็น ASCII Code แล้วสร้าง message WM_CHAR ส่งออกมาที่ Application Message Queue ของโปรแกรมนั้น

ลำดับของ message ที่ถูกสร้างขึ้นมาเมื่อมีการกดคีย์ w

KEY	wParam Contain
WM_KEYDOWN	Virtual Key Code W
WM_CHAR	ASCII Code w
WM_KEYUP	Virtual Key Code W

พารามิเตอร์ที่ส่งมากับ WM_CHAR จะมีความหมายเหมือนกับพารามิเตอร์ของ message WM_KEYDOWN แต่ต่างกันในส่วนของwParamซึ่งของ message WM_CHAR จะเก็บ ASCII Code ของคีย์ที่กด แต่ของ message WM_KEYDOWN จะเก็บ Virtual Key Code ของคีย์ที่กด

7.2 Keyboard และระบบ Multitasking

เพราะว่าระบบ windows เป็นระบบปฏิบัติการแบบ Multitasking ดังนั้นจึงจะต้องมีวิธีการที่จะใช้อุปกรณ์ร่วมกันเช่น Keyboard จึงมีแนวความคิด (CONCEPT) อยู่สองอย่างที่ระบบWindows ใช้ในการเลือกหน้าต่างที่จะส่ง input จาก Keyboard ไปให้ คือ Active Window และ Focus

เมื่อโปรแกรมสร้างหน้าต่างขึ้นมา หน้าต่างที่ถูกสร้างนี้จะเป็นส่วนที่ใช้ติดต่อกับผู้ใช้ (หน้าต่านี้ เราอาจจะเรียกว่า Top-Level Window) ซึ่งหน้าต่างนี้ก็คือ Active Window ที่ผู้ใช้เลือกที่จะติดต่อด้วยโดยปกติ Active Window จะอยู่บนสุดของหน้าต่างทั้งหมดในระบบ

เมื่อหน้าต่างชั้นบนสุดถูกเลือกให้ Active แล้ว routine Default Window Procedure (Routine ที่ใช้จัดการรับ message ที่ไม่ถูกนำไปใช้โดยโปรแกรม) จะให้ Focus ไปที่ Active Window นั้น ซึ่งมี Code ดังนี้

```
Case   WM_ACTIVE:
  if (wParam)
     SetFocus (hwnd);
```

ดังนั้น Focus จะเป็นเสมือนส่วนที่ใช้ระบุว่าหน้าต่างใดที่จะได้รับ input จาก Keyboard (หรือจะบอกได้ว่าหน้าต่างที่มี Focus จะครอบครองการติดต่อกับผู้ใช้ทาง Keyboard)

ในระบบ Windows สิ่งที่จะบอกว่าขณะนี้หน้าต่างใดได้ Focus อยู่เรียกว่า Caret (รูปสี่เหลี่ยมผืนผ้าที่กระพริบหน้าต่าง เหมือน Cursor ในระบบ Dos) เพราะว่า Caret เป็น Bitmap อย่างหนึ่ง จึงต้องมีการสร้างก่อนที่จะสามารถใช้งานได้ และจะต้องยกเลิกเมื่อเปลี่ยน Focus ไปที่หน้าต่างอื่น (โดยปกติโปรแกรมจะสร้าง Caret เมื่อมี message WM_SETFOCUS และยกเลิก Caret เมื่อมี message WM_KILLFOCUS เข้ามาในโปรแกรม)

บทที่ 8
โครงสร้างและหน้าที่ของโปรแกรมโครงงาน

โปรแกรมโครงงานในที่นี้คือโปรแกรม Spreadsheet ซึ่งประกอบไปด้วย 2 ส่วนหลักคือ

1.ส่วน Worksheet Interface

2.ส่วน Graphic Interface

โดยรายละเอียดและหน้าที่การทำงานของทั้ง 2 ส่วนมีดังต่อไปนี้

Worksheet Interface

โปรแกรมในส่วน worksheet interface มีส่วนประกอบต่าง ๆ ดังนี้

1.Window หลักที่มี System menu และ menu bar ของ user (pulldown menu)

2.Ribbon เป็นส่วนที่แสดงตำแหน่งของ cell ที่ active อยู่ในรูปแบบ column และ row

3.ส่วนที่ควบคุมการนำข้อมูลที่พิมพ์ไว้ใน buffer ลงสู่ cell ที่ active จะมีลักษณะเป็นกรอบที่มีเครื่องหมายถูก และผิด ถ้านำ mouse ไป click ที่เครื่องหมายถูกจะมีความหมายเท่ากับการกดคีย์ Enter เป็นการตอบตกลงการนำข้อมูลจาก global buffer ลงสู่ cell ที่ active อยู่ แต่ถ้านำ mouse ไป click ที่เครื่องหมายผิดจะเป็นการยกเลิกข้อมูลที่พิมพ์ลงไป

4.Edit buffer เป็น Window ที่รอรับการพิมพ์ข้อมูลเข้าไปทาง keyboard การทำให้ Window นี้ active คือ เอา mouse ไป click ภายใน Window หรือกดตัวอักษรใด ๆ เข้าไป

ข้อมูลที่พิมพ์ลงไปจะไปแสดงขึ้นที่ Edit buffer และจะเก็บข้อมูลนี้เอาไว้ใน global buffer ที่นี่จนกว่าจะกด Enter หรือ นำ mouse ไป click ที่เครื่องหมายถูกในกรอบควบคุม จึงจะนำข้อมูลใน global buffer นี้ถ่ายลงสู่ data structure ของ cell ที่ active นั้น และทำการ clear ค่าใน global buffer

แต่ถ้ากดเครื่องหมายผิดในกรอบควบคุม ก็จะทำการ clear ข้อมูลใน global buffer โดยไม่ยุ่งกับข้อมูลจริงของ cell นั้น ๆ

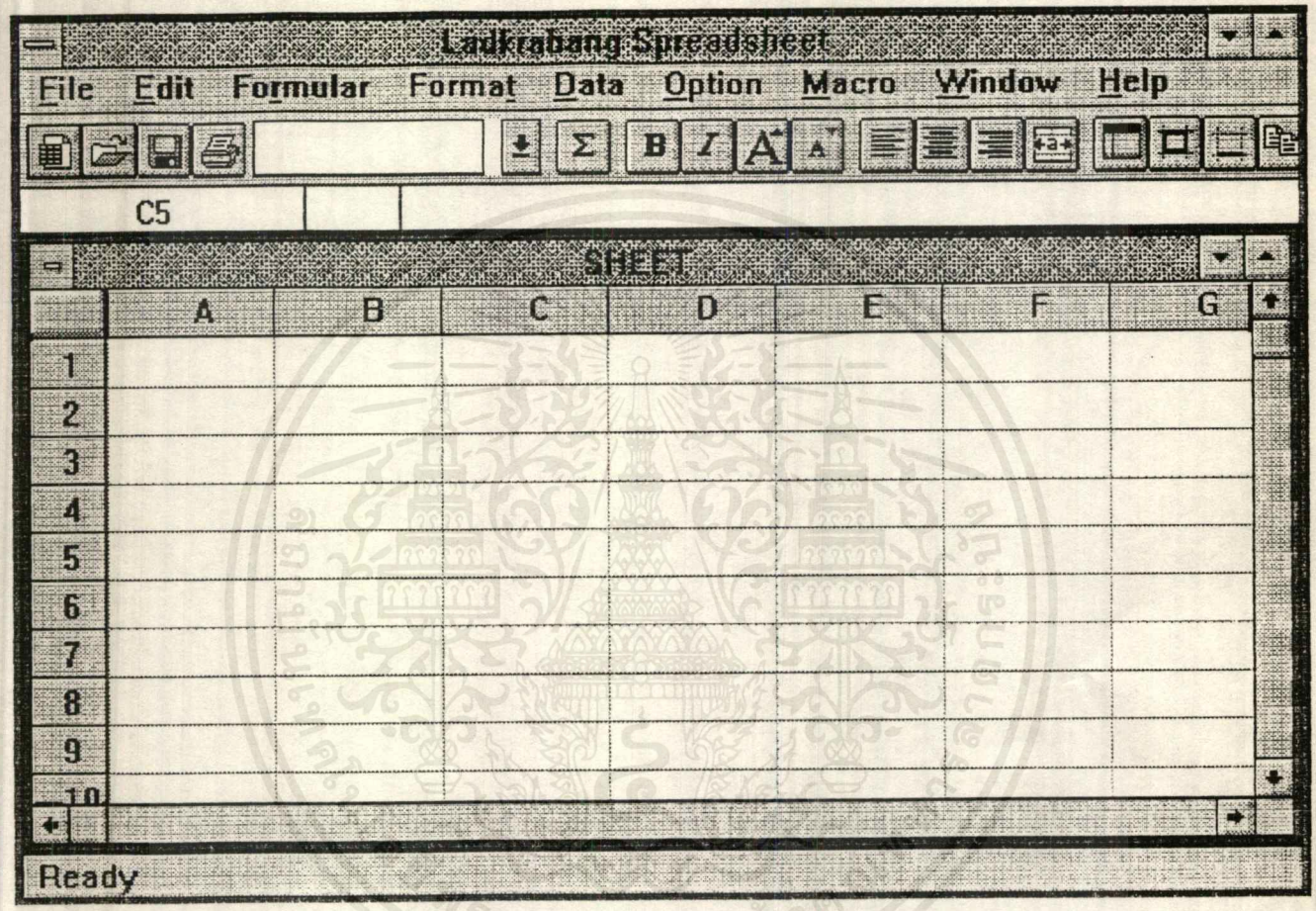
5.Toolbar เป็นปุ่มที่ใช้เป็นคำสั่งแทนการเลือก menu item ใน pulldown menu

6.Worksheet จะเป็น Window ที่มี horizontal scrollbar และ vertical scroll bar ไว้ควบคุมการเลื่อน cell เป็น sheet ที่ตีตารางเป็น row และ column ในแนว column มีช่วงตั้งแต่ A-ZZ และ ในแนว row มีช่วงตั้งแต่ 1-800

ส่วน worksheet นี้สร้างเป็น multi ducument สามารถเปิด sheet ได้พร้อมกันถึง 8 sheet และยังมี facility อื่นที่สามารถทำได้ซึ่งจะอธิบายในส่วนถัดไป

7.Status line เป็นแถบที่ใช้แสดงคำอธิบายแต่ละ menu item ที่มีการสร้างแถบสว่างผ่าน

รูปที่ 1 ในหน้าถัดไปจะแสดงถึงภาพรวม ๆ ของส่วนประกอบทั้ง 7 ส่วน



การทำงานในส่วน worksheet

เป็น Spread sheet ที่มีรูปแบบเป็น cell แต่ละ cell จะถูกระบุตำแหน่งโดย column และ row โดยใน column จะมีช่วงตั้งแต่ A-ZZ และ row จะมีช่วงตั้งแต่ 1-800 เมื่อนำ mouse ไป click ที่ cell จะแสดงตำแหน่งของ cell นั้น ๆ ที่ Ribbon ในรูปแบบ column x row และจะอ่านข้อมูลของ cell นั้น ๆ ขึ้นมาแสดงที่ Edit buffer เพื่อรอรับการแก้ไข

(ดังรูปที่ 2 cell ที่ active อยู่คือ cell C3 และข้อมูลของ cell C3 คือ "Spread")

สามารถขยายความกว้างของแต่ละ Column และความยาวในแต่ละ Row ได้ (ค่า default ของความกว้างของ column = 70 และ ค่า default ของความยาวของ Row = 20) โดยใช้ mouse หรือโดยใช้ menu ดังนี้

-ขยาย cell ผ่านทาง menu โดยเลือก Format\Column Width หรือ Format\Row Height ในการขยาย Column และ Row ตามลำดับ พอเลือก menu item ตามนั้นแล้ว ก็จะขึ้น Dialog box ให้กรอกขนาดที่ต้องการขยาย

-ขยาย cell โดยใช้ mouse ทำดังนี้ เคลื่อน mouse ไปที่แถบ Column หรือ แถบ Row จนกว่า cursor จะเปลี่ยนเป็นรูปกากบาท ให้ click mouse แล้ว drag ไปให้ได้ความกว้างที่ต้องการ

(ดังรูปที่ 2 ขยายความกว้างของ Column C และขยายความยาวของ Row 3)

สามารถใช้ Horizontal scrollbar และ Vertical scrollbar ควบคุมการเลื่อน cell และสามารถเลือก cell เป็นบล็อค โดยจะ mark สีดำเอาไว้ และสามารถ fill ข้อมูลลงทั้ง block ที่ mark เอาไว้ และสามารถ move cell , move block , copy cell , copy block

(ดังรูปที่ 3 จะเลื่อน Vscroll และ Hscroll และทำการ mark block ตั้งแต่ D5 ถึง D8 เพื่อกรอกข้อมูลลงไป และทำการขยายความสูงของ row ที่ 6 ด้วย)

สามารถ split window ออกเป็น 4 ส่วน และมี VScroll 2 อัน ไว้ควบคุม window ส่วนบน และส่วนล่าง และมี HScroll 2 อัน ไว้ควบคุม Window ส่วนซ้าย และส่วนขวา

(ดังรูปที่ 4 ทำการ Split ที่ cel D5 และทำการขยายความสูงของ row ที่ 6)
(ดังรูปที่ 5 ทำการ split window แล้ว scroll แยกกันในแต่ละส่วน จะเห็นได้ว่า cell H7 ไปปรากฏอยู่ใน Window ทั้ง 4 ส่วน)

สามารถ insert, delete ได้ทั้ง row และ column

สามารถ open file , save file

สามารถพิมพ์ worksheet ออกทาง printer

-โดยมี option ในการพิมพ์ดังนี้

1.สามารถพิมพ์ได้ไม่จำกัดจำนวน row หรือ column

2.สามารถเลือกจำนวน copies ที่จะพิมพ์ได้

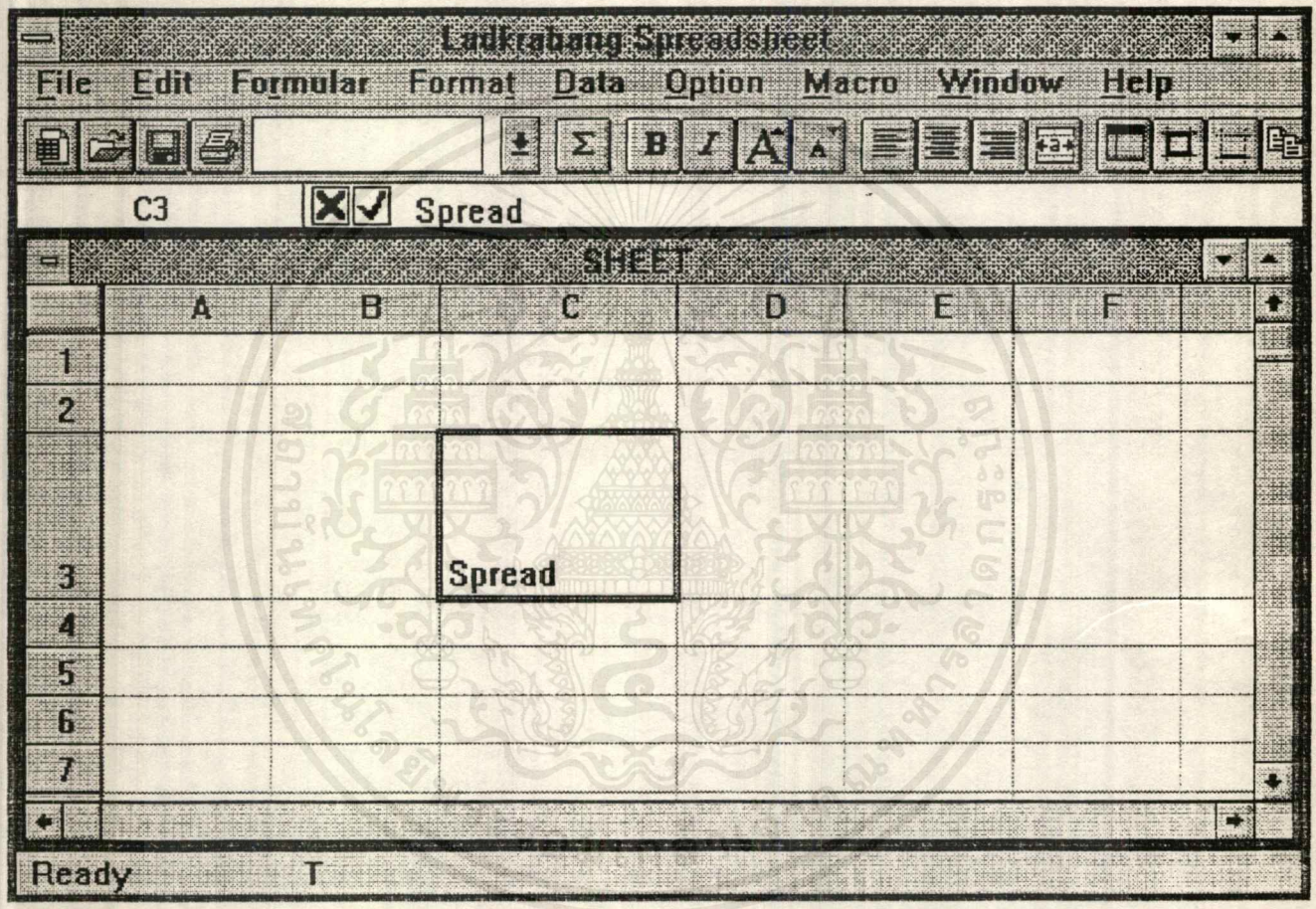
3.สามารถเลือกพิมพ์ในลักษณะ lanscape หรือ portait

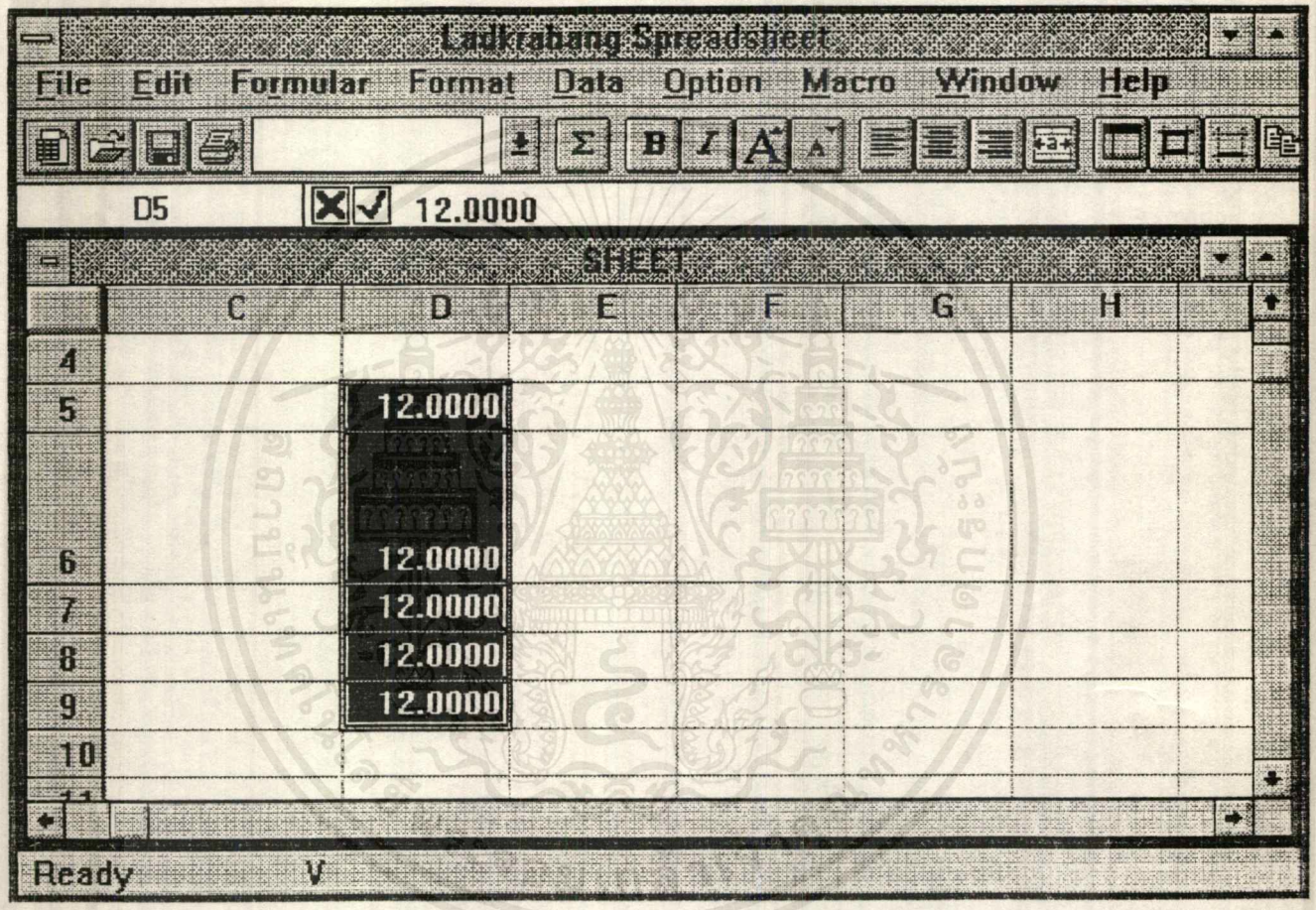
4.สามารถเลือกความกว้างของกระดาษที่จะพิมพ์

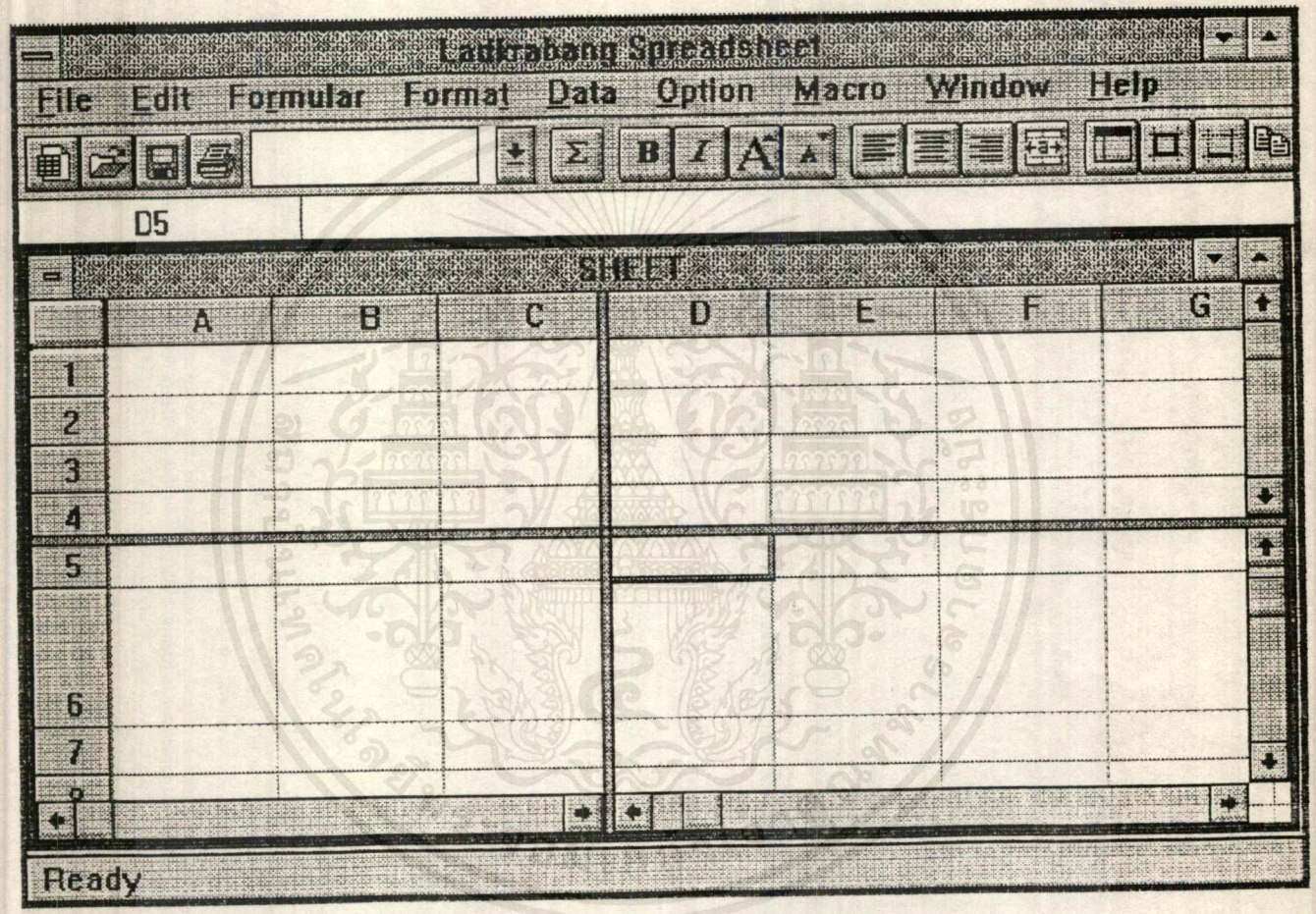
รูปที่ 6.1-6.4 แสดง sheet ที่พิมพ์ออกมา โดยผู้ใช้เลือกพิมพ์ 20 row และ 5 column (การเลือกข้อมูลที่จะพิมพ์ทำได้โดยใช้ mouse drag ข้อมูลที่ส่วนนั้น แล้วเลือก menu item "Print Report")

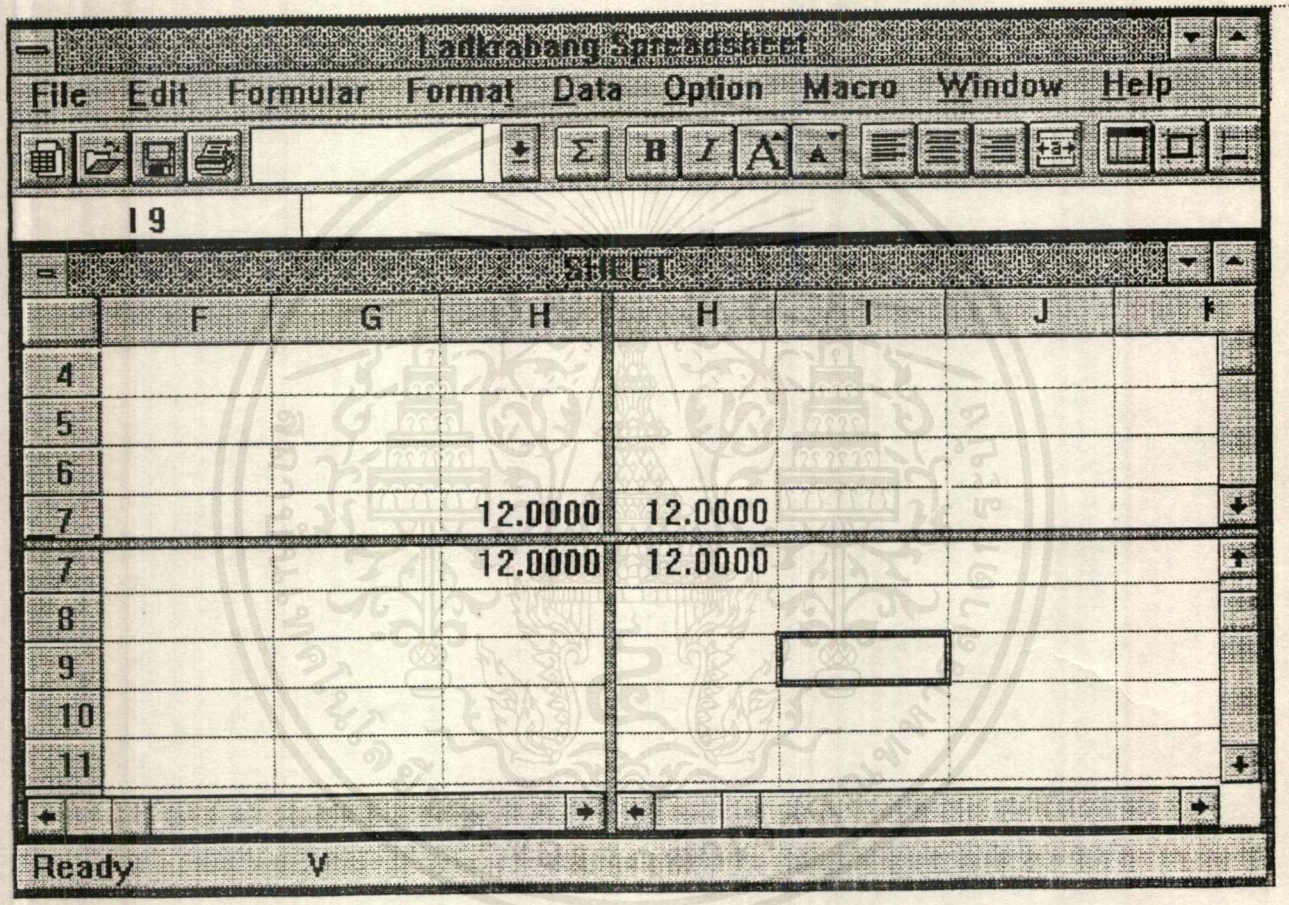
รูปที่ 6.5-6.6 แสดง sheet ที่พิมพ์ออกมา โดยผู้ใช้เลือกพิมพ์ 20 row และ 5 column และเลือกพิมพ์แบบ landscape

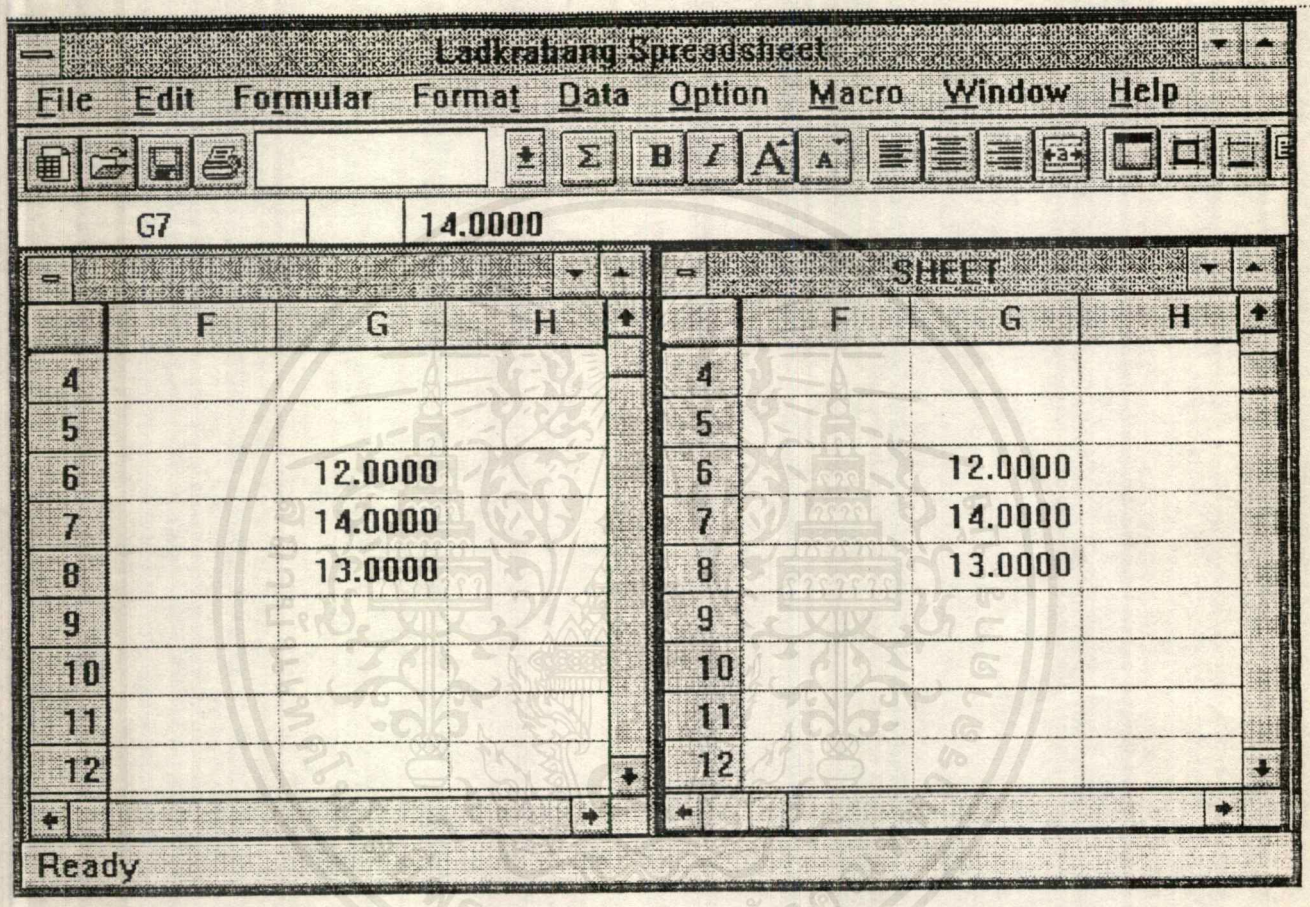
สามารถเปิด sheet ได้พร้อมกัน 8 sheet (ดังรูปที่ 7 แสดงการเปิด sheet 2 sheet ให้ขึ้นมา active พร้อมกัน)











0	March	March	March	March	March
1	March	March	March	March	March
2	March	March	March	March	March
3	March	March	March	March	March
4	March	March	March	March	March
5	March	March	March	March	March
6	3200	3200	3200	3200	3200
7	3200	3200	3200	3200	3200
8	3200	3200	3200	3200	3200
9	3200	3200	3200	3200	3200
10	March	March	March	March	March
11	March	March	March	March	March
12	March	March	March	March	March

13	March	March	March	March	March
14	March	March	March	March	March
15	2100	2100	2100	2100	2100
16	2100	2100	2100	2100	2100
17	2100	2100	2100	2100	2100
18	2100	2100	2100	2100	2100
19	2100	2100	2100	2100	2100
20	March□	March□	March□	March□	March□

Graphics Interface

ส่วน Graphics Interface นี้ประกอบไปด้วยส่วนที่ทำหน้าที่หลัก 2 อย่างด้วยกัน คือ

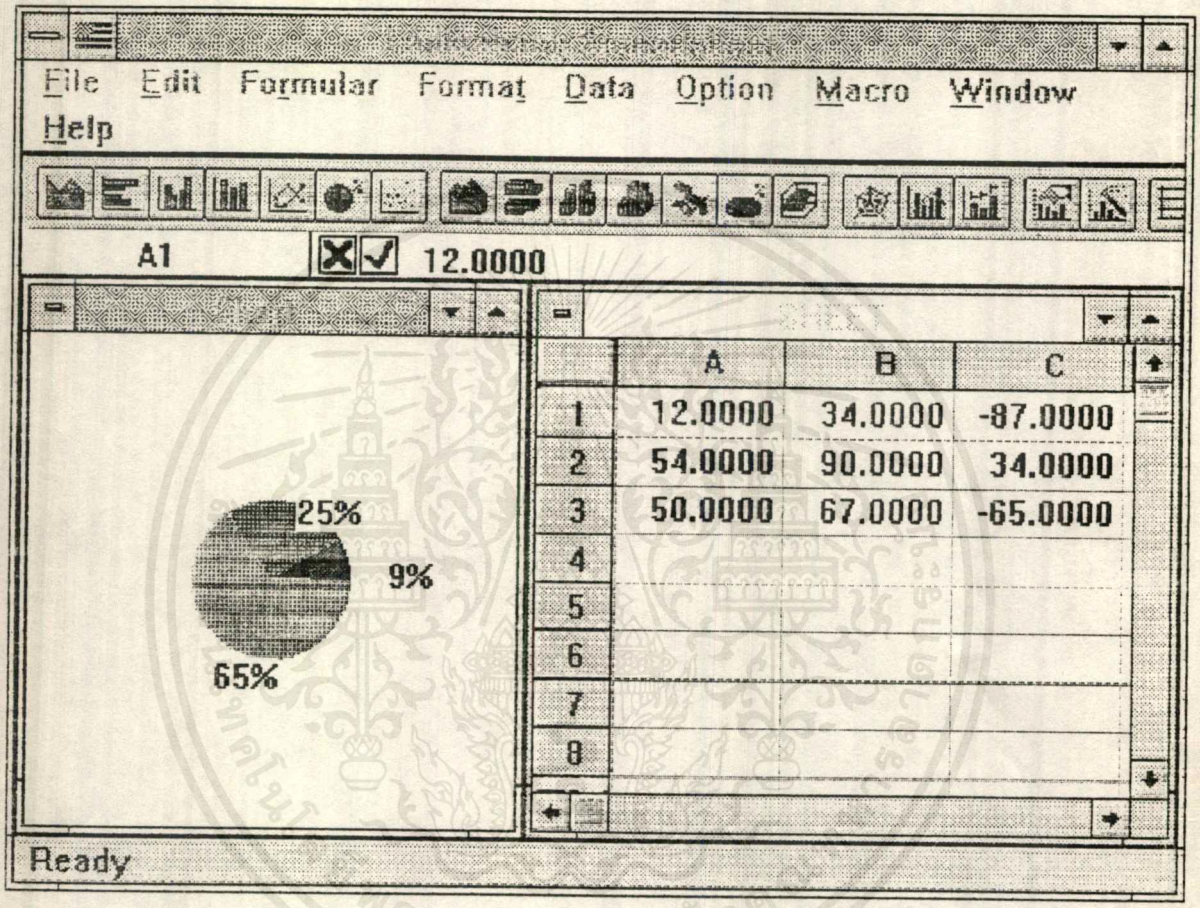
1. ส่วน Mutidocument ที่แสดงข้อมูลในรูปแบบกราฟชนิดต่างๆ อันได้แก่
   - กราฟแท่ง
   - กราฟแท่งสะสม
   - กราฟวงกลม
   - กราฟจุด
   - กราฟเส้น

2. ส่วน พิมพ์รูปกราฟของแต่ละ document ออกทาง Printer

ซึ่งรายละเอียดของหัวข้อทั้ง 2 มีดังนี้

1. ส่วน Mutidocument ที่แสดงข้อมูลในรูปแบบกราฟชนิดต่างๆ

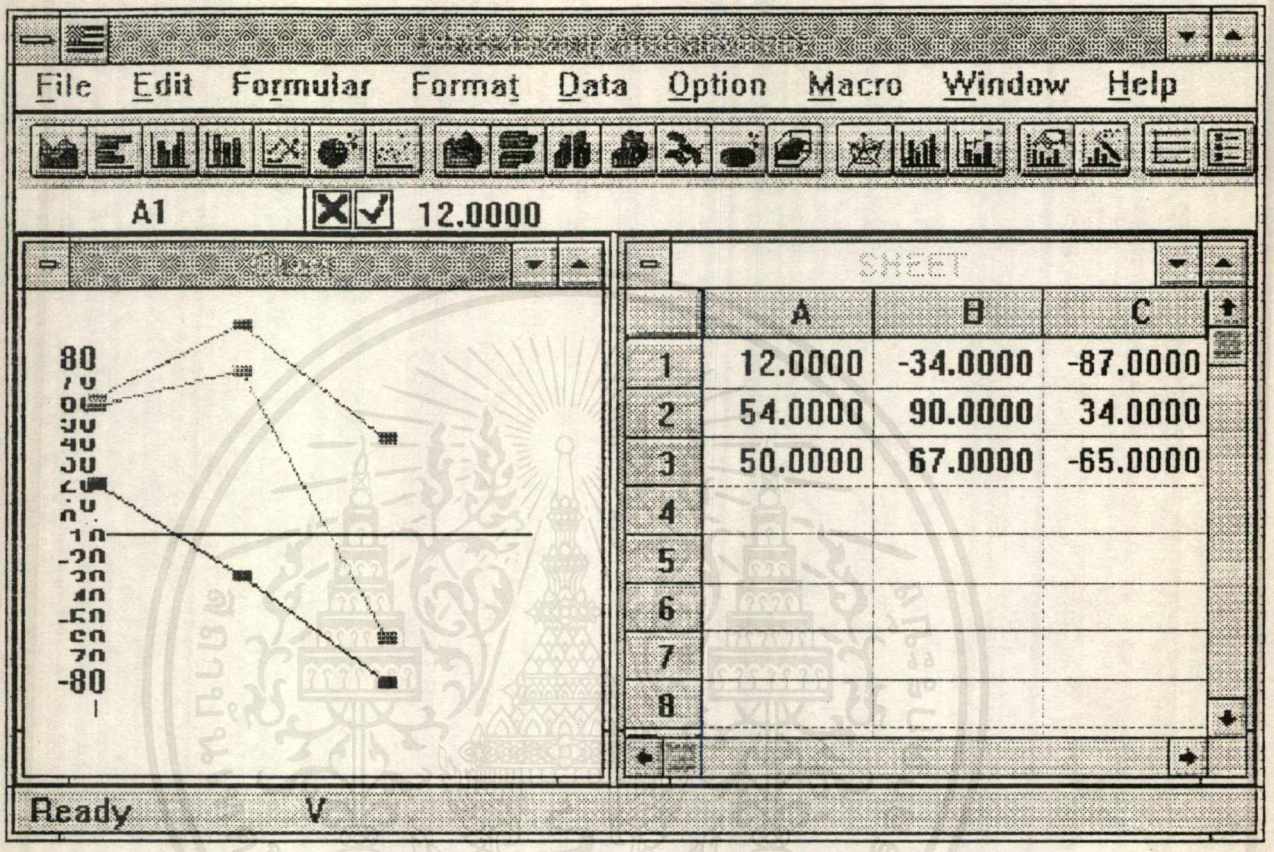
ส่วนนี้แต่ละ document สามารถที่จะแสดงรูปกราฟที่สัมพันธ์ข้อมูลใน worksheet โดยเป็นอิสระต่อกัน เมื่อผู้ใช้ต้องการให้ plot กราฟของข้อมูลในส่วนไหน ก็สามารถทำได้โดย ที่document worksheet ให้ผู้ใช้ทำการ drag mouse แล้วเลื่อนให้แถบสีดำทับข้อมูลทั้งหมดที่ต้องการ plot จากนั้นก็เปิด window ใหม่จาก menu item "New Chart" ดังแสดงในรูป G.1 ก็จะได้ window ที่แสดงรูปกราฟที่สัมพันธ์กับข้อมูลที่ได้เลือกไว้



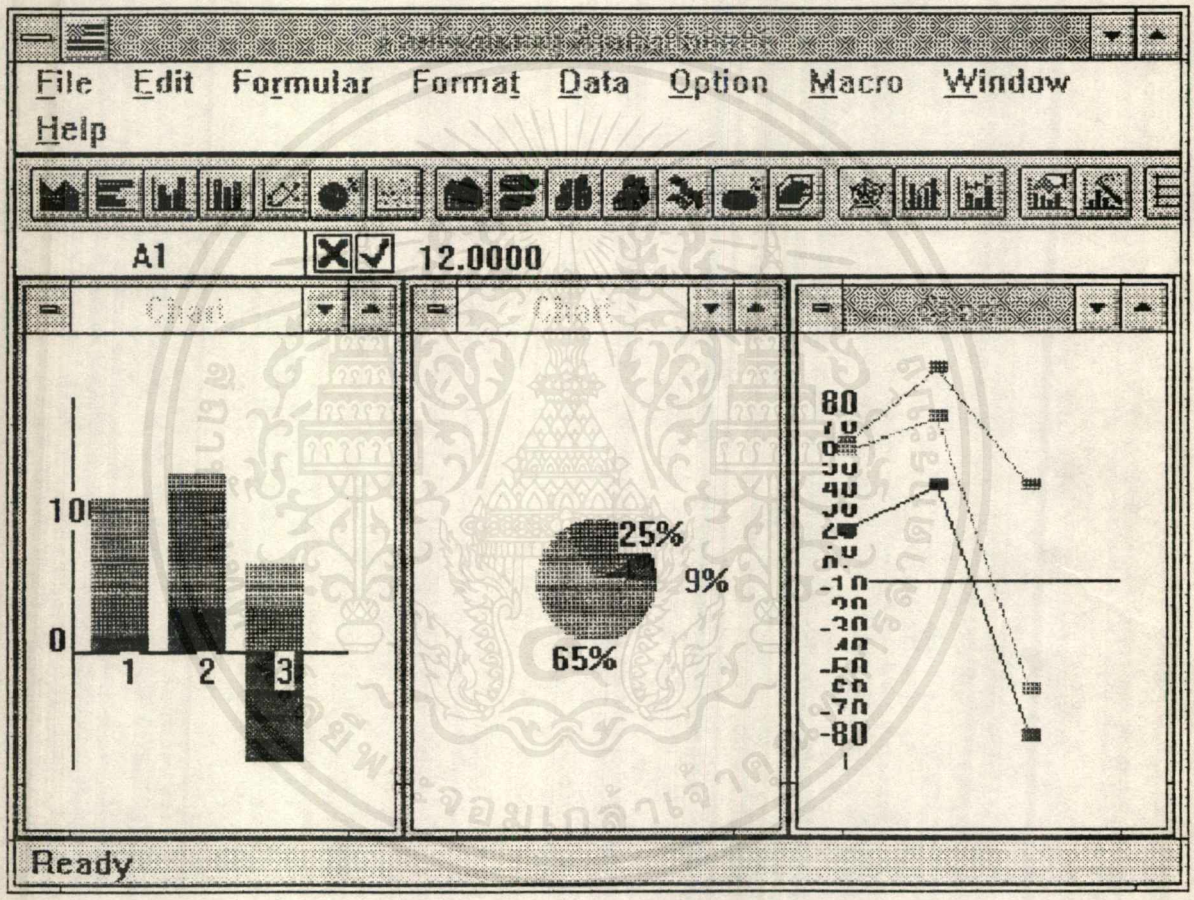
รูป G.1 แสดงsheet window และ chart window ที่แสดงรูปกราฟที่สัมพันธ์กับข้อมูลใน she

จากรูป เมื่อผู้ใช้ต้องการเปลี่ยนรูปแบบกราฟก็สามารถทำได้โดย นำ mouse ไป click ที่ปุ่ม Toolbar บนแถบ Toolbar จากนั้นรูปกราฟบน window ที่ active อยู่ ก็จะเปลี่ยนไปเป็นรูปแบบตามที่ผู้ใช้เลือก

ตัวอย่างแสดงดังรูปที่ G.2 คือจากรูป G.1 เมื่อผู้ใช้ นำ mouse ไป click ที่ปุ่ม Toolbar ปุ่มที่ 5 รูปกราฟจากกราฟแท่งก็จะเปลี่ยนเป็นรูปกราฟแบบเส้นตรงทันที



รูปที่ G.2



รูปที่ G.3 แสดงMultidocument ของกราฟหลายรูปแบบ

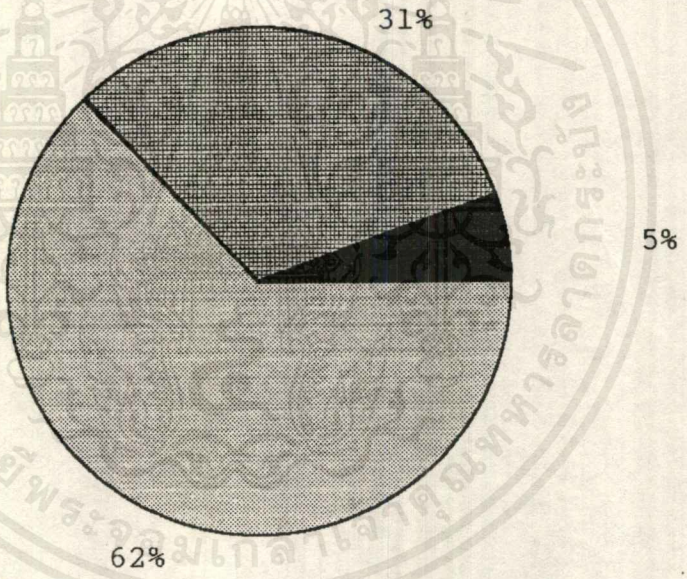
2. ส่วน พิมพ์รูปกราฟของแต่ละ document ออกทาง Printer

การพิมพ์รูปกราฟนั้นสามารถทำได้โดย นำ mouse ไป click ให้ window ที่ต้องการพิมพ์รูปกราฟนั้น active เสียก่อน จากนั้นให้เลือกคำสั่งพิมพ์จาก menu item ที่ชื่อ "Print Chart" จากนั้นก็จะปรากฏ dialog box เพื่อให้ผู้ใช้เลือก option ในการพิมพ์ดังนี้

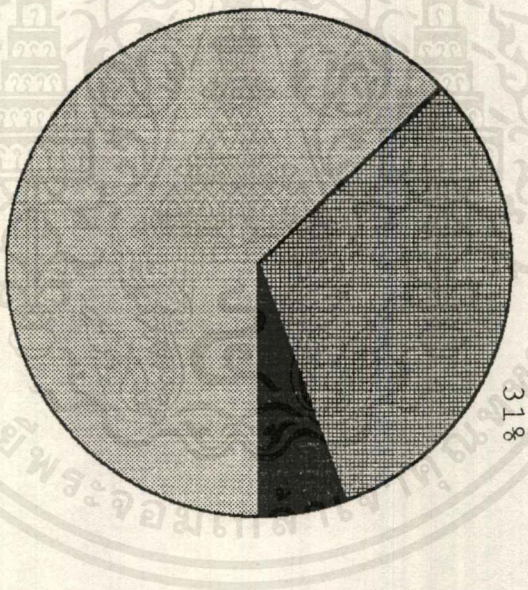
1. เลือกรูปแบบการพิมพ์ว่าจะเป็นแบบ landscape หรือ แบบ portrait
2. เลือกขนาดของกระดาษที่จะพิมพ์
3. เลือกความเข้มสว่างของการพิมพ์

ฯลฯ

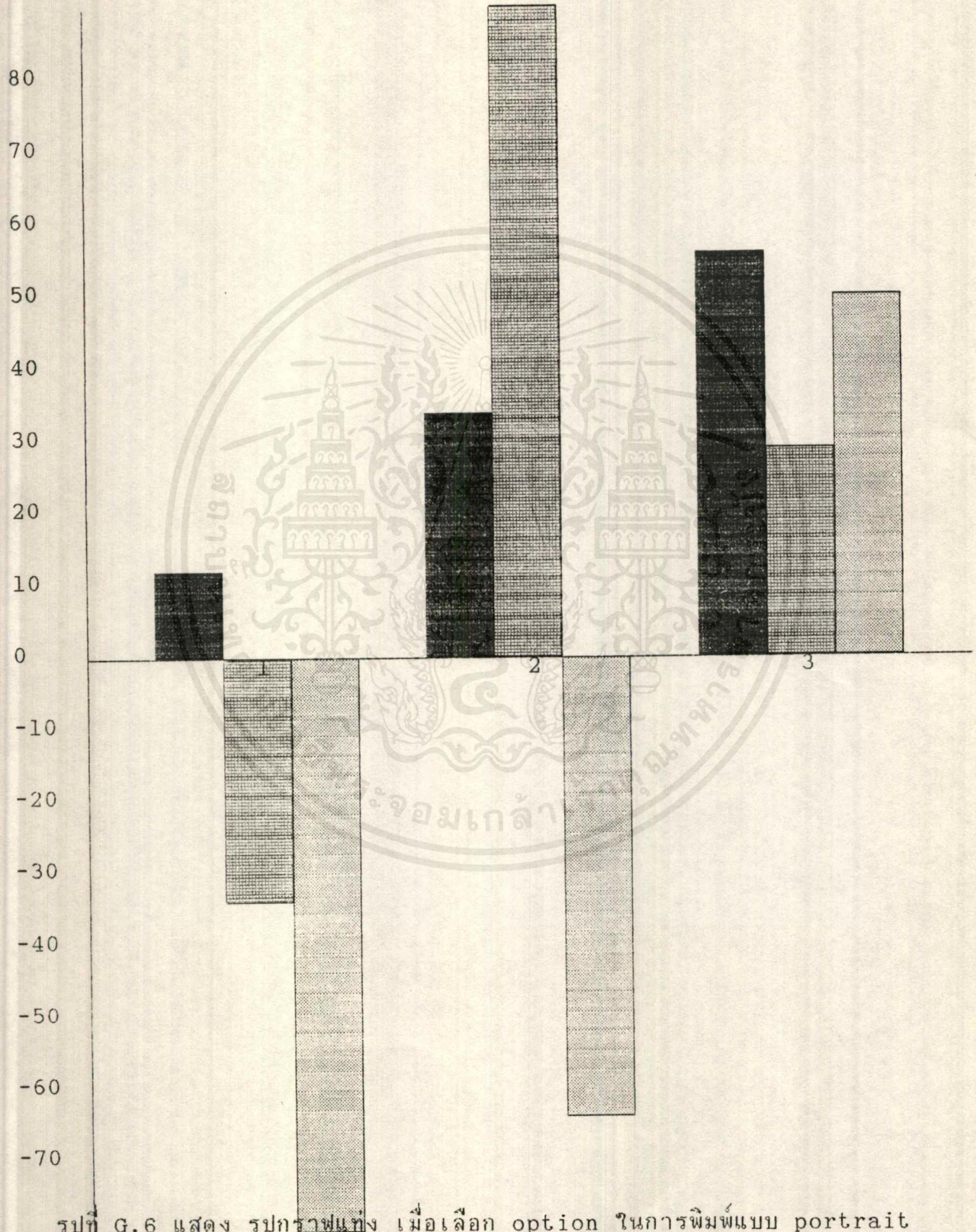
รูปที่ G.4 - G.11 จะแสดงกราฟที่พิมพ์ออกมาในรูปแบบต่างๆ



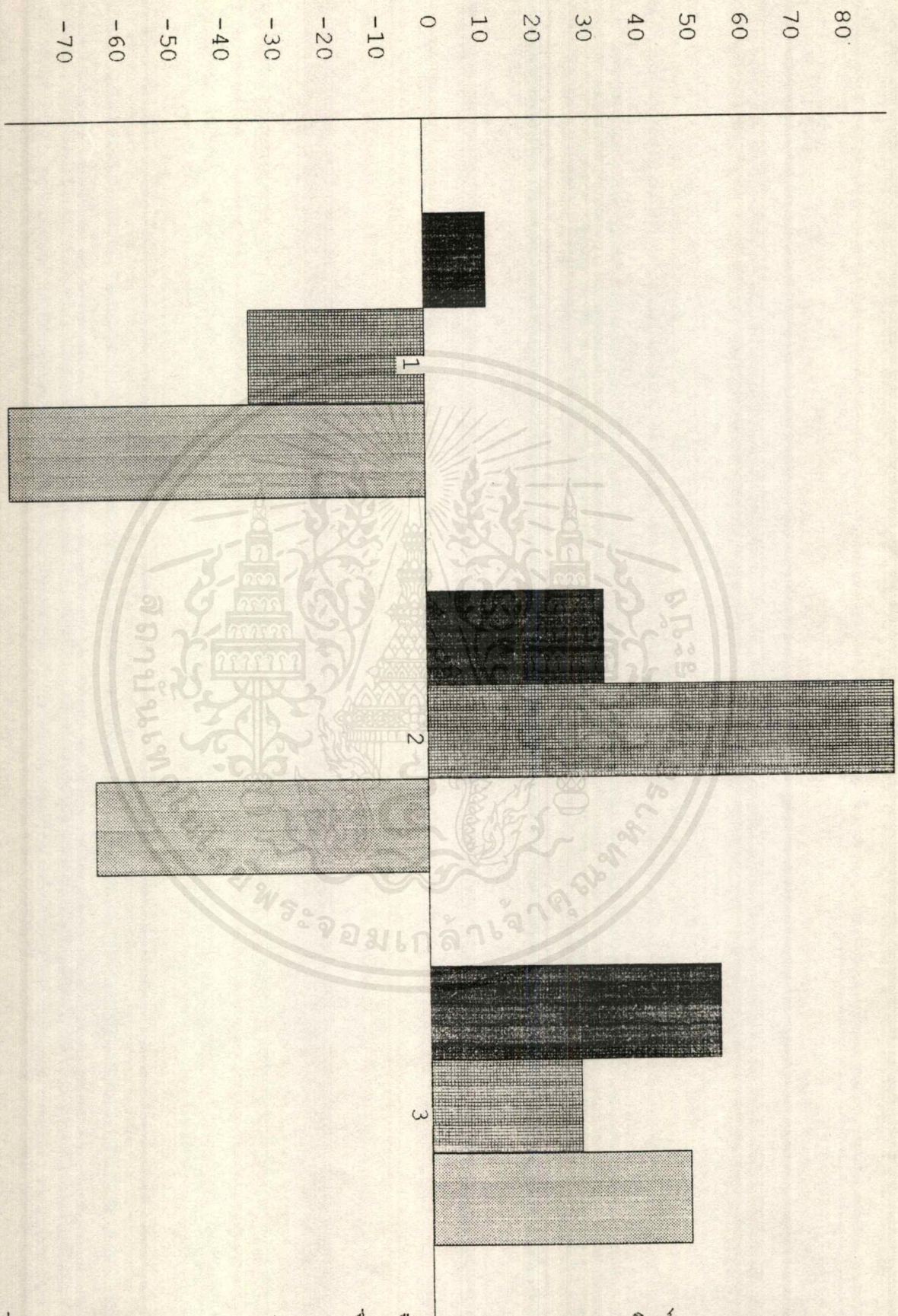
รูปที่ G.4 แสดง รูปกราฟวงกลม เมื่อเลือก option ในการพิมพ์แบบ portrait



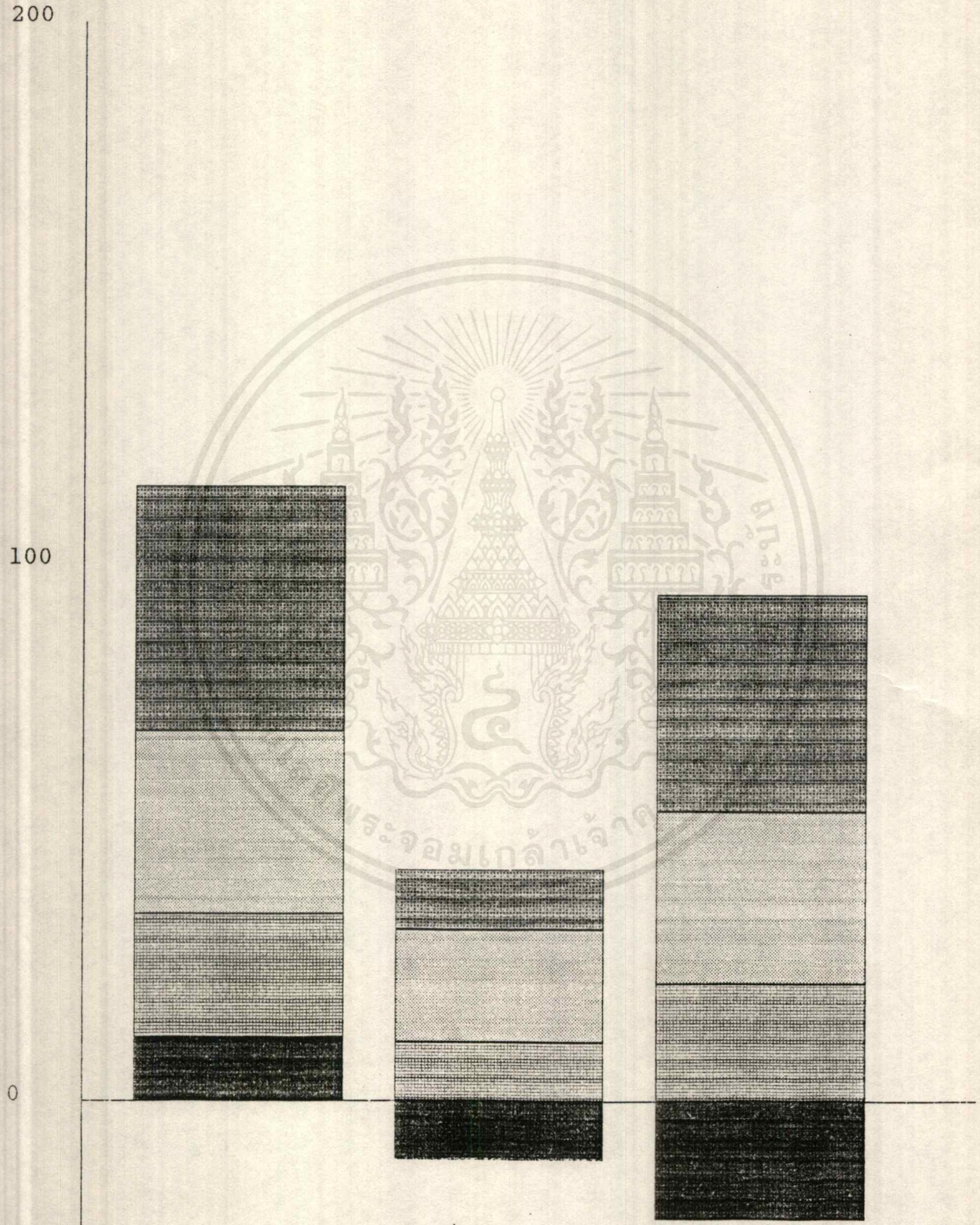
รูปที่ G.5 แสดง รูปกราฟวงกลม เมื่อเลือก option ในการพิมพ์แบบ landscape



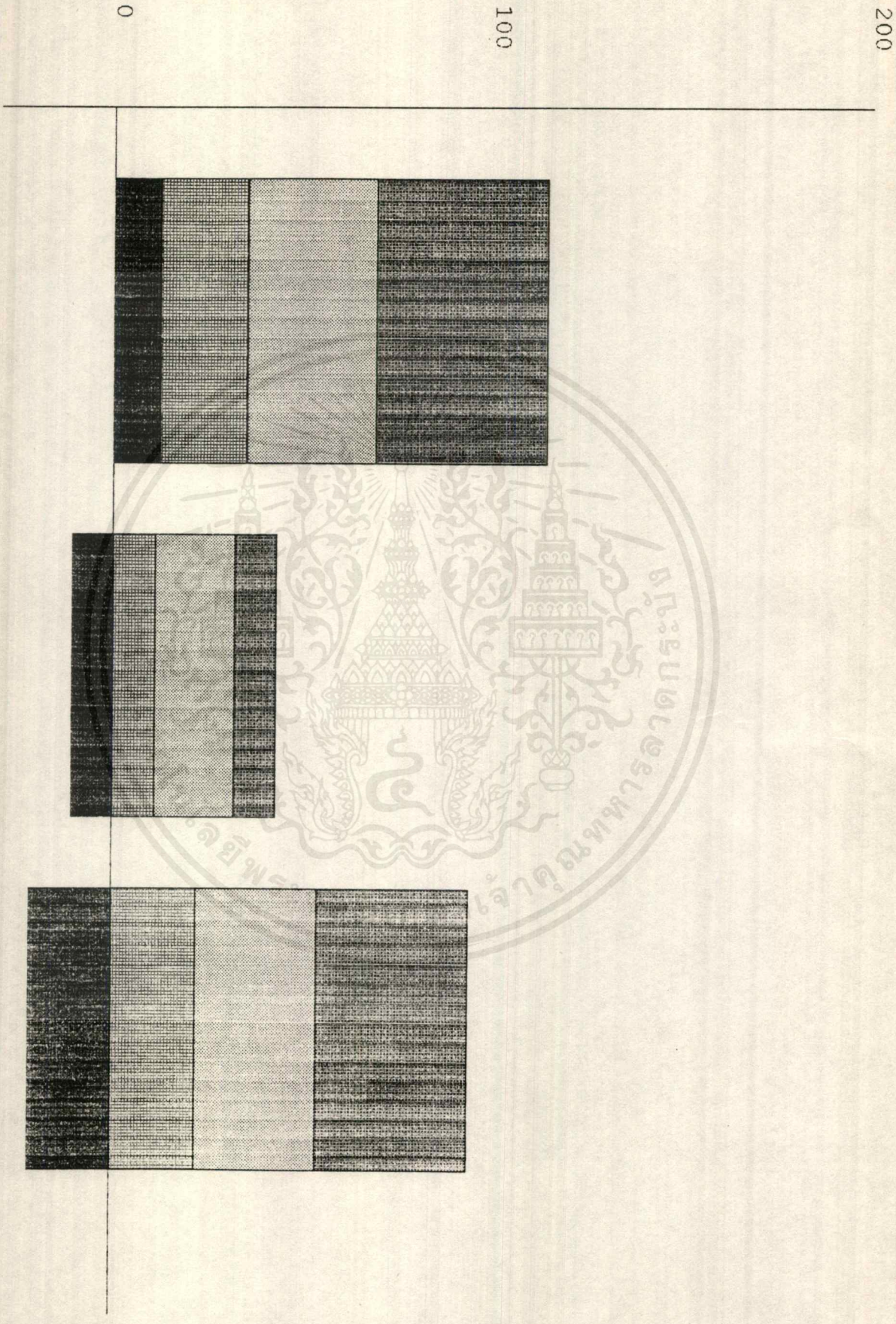
รูปที่ G.6 แสดง รูปกราฟแท่ง เมื่อเลือก option ในการพิมพ์แบบ portrait



รูปที่ G.7 แสดง รูปกราฟแท่ง เมื่อเลือก option ในการพิมพ์แบบ landscape



รูปที่ G.6 แสดง รูปกราฟแท่งสะสม เมื่อเลือก option ในการพิมพ์แบบ portrait



รูปที่ G.7 แสดง รูปกราฟแท่งสะสม เมื่อเลือก option ในการพิมพ์แบบ landscape

บทที่ ๑
ฟังก์ชันย่อยภายในโปรแกรม

About DlgProc (hWnd, num, wParam, lParam)

```
HWND    hWnd;
UINT    num;
WPARAM  wParam;
LPARAM  lParam;
```

<u>หน้าที่</u> เป็น Procedure ที่ process เกี่ยวกับ dialog box เมื่อเลือก คำสั่งจาก menu ดังนี้ Help/About

<u>input</u>

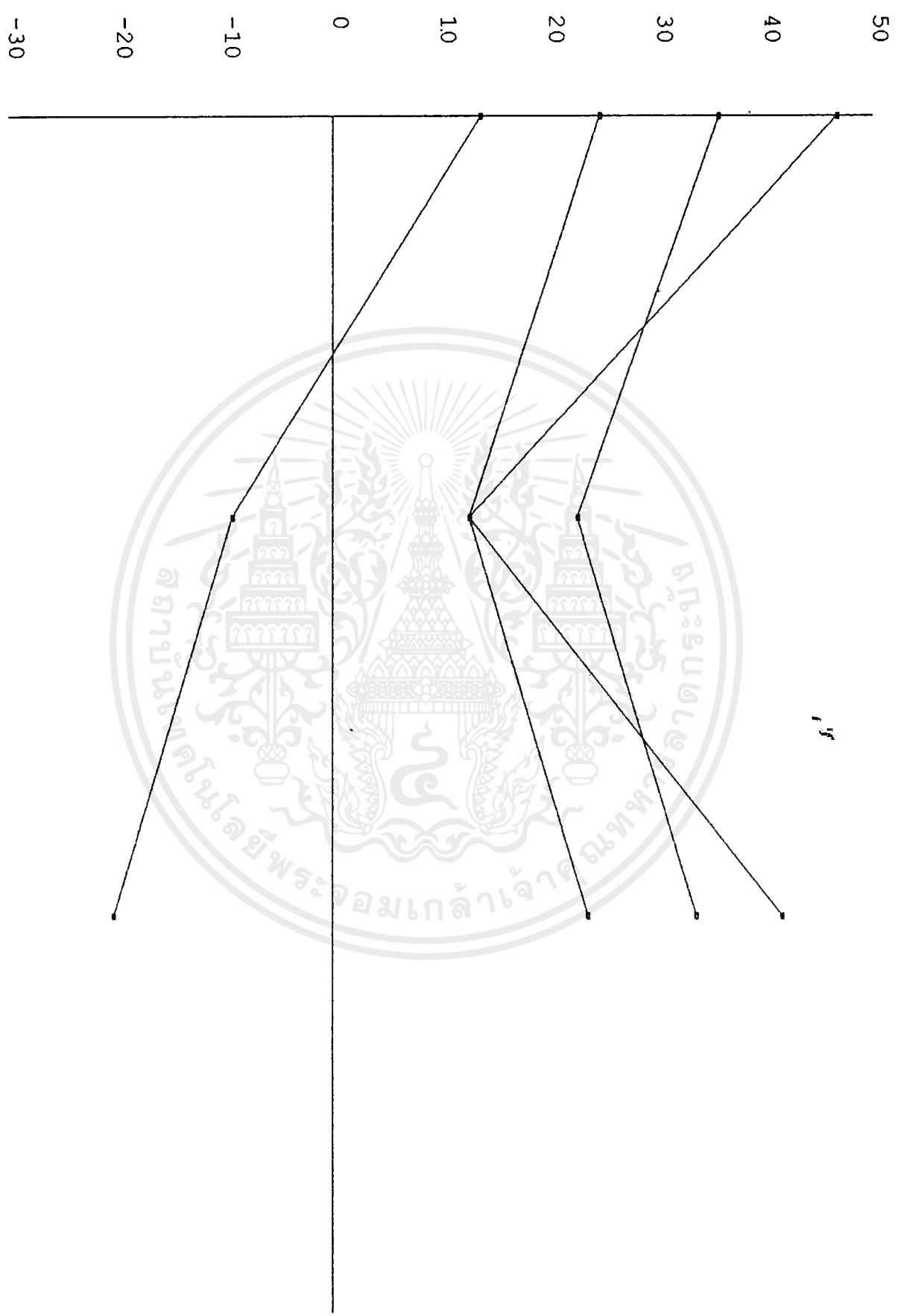
1. hWnd = ค่า handle ประจำ dialog box
2. num = ค่าเลขประจำตัว dialog box
3. wParam = ค่า WORD PARAMETER ที่ระบบส่งผ่านมา
4. lParam = ค่า LONG PARAMETER ที่ระบบส่งผ่านมา

<u>return</u> TRUE ถ้าสร้าง dialog สำเร็จ
FALSE ถ้าสร้าง dialog ไม่สำเร็จ

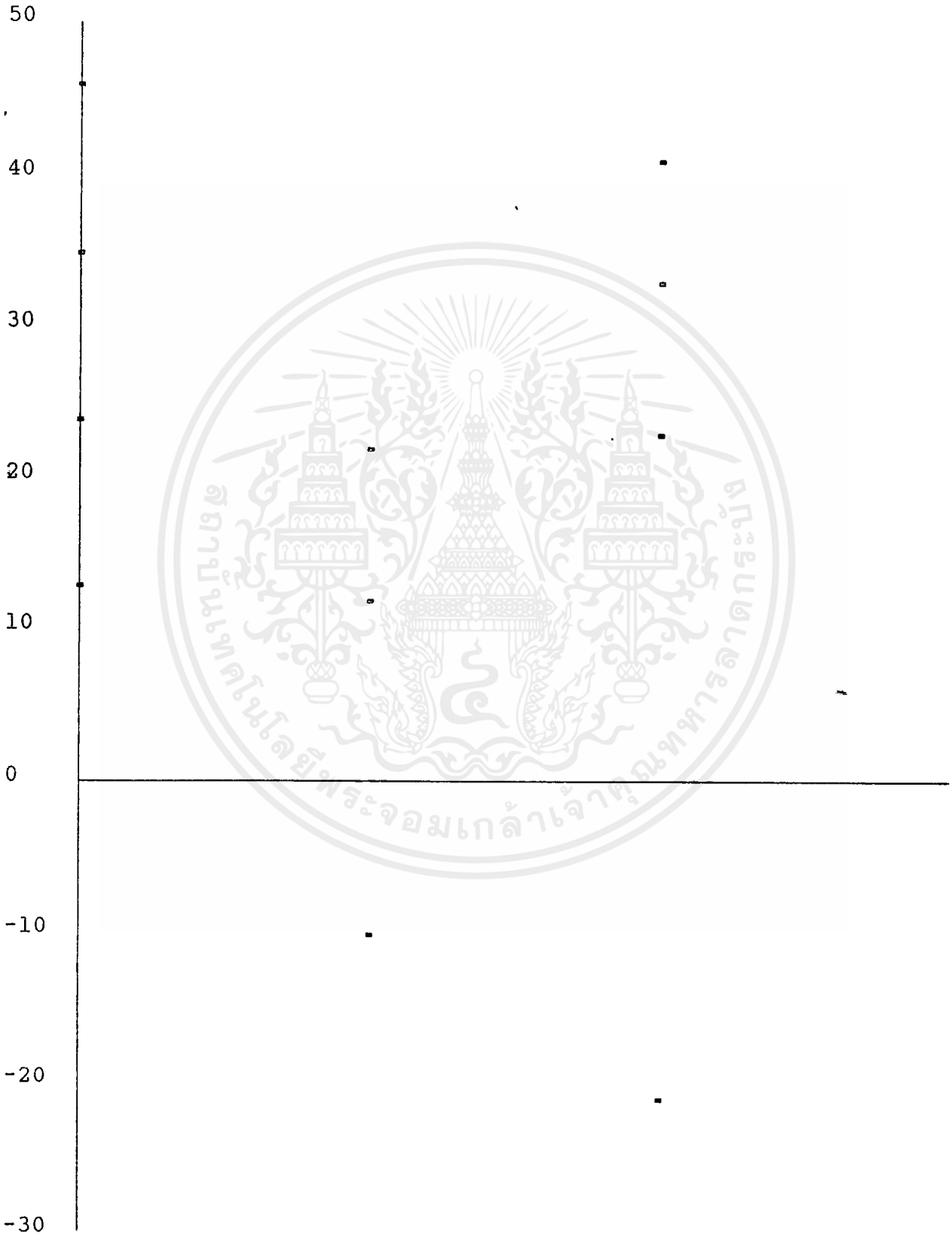
ROWHELGMsgProc (hWnd, num, wParam, lParam)

```
HWND    hWnd;
UINT    num;
WPARAM  wParam;
LPARAM  lParam;
```

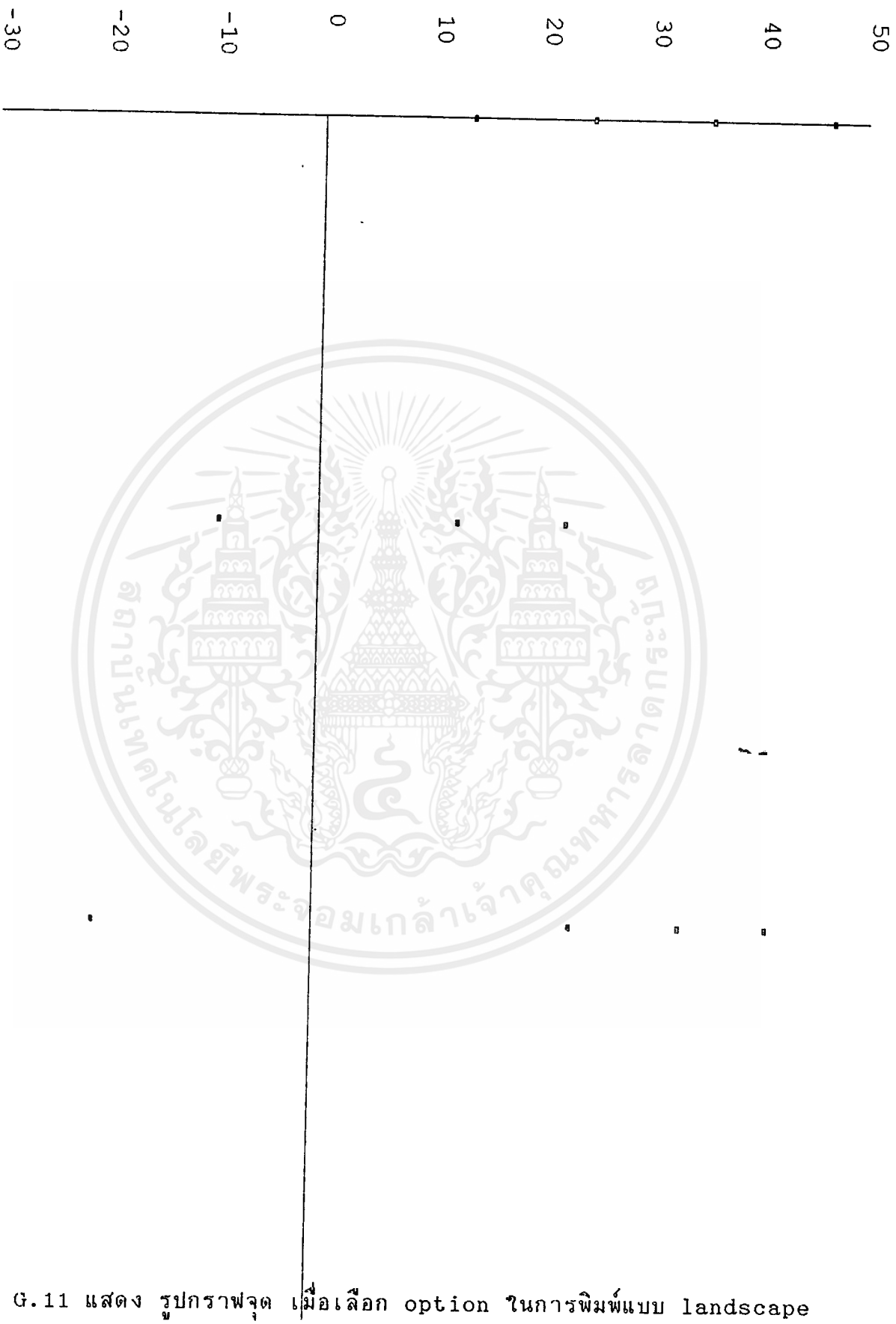
<u>หน้าที่</u> เป็น Procedure ที่ process เกี่ยวกับ dialog box ของการขยายความ



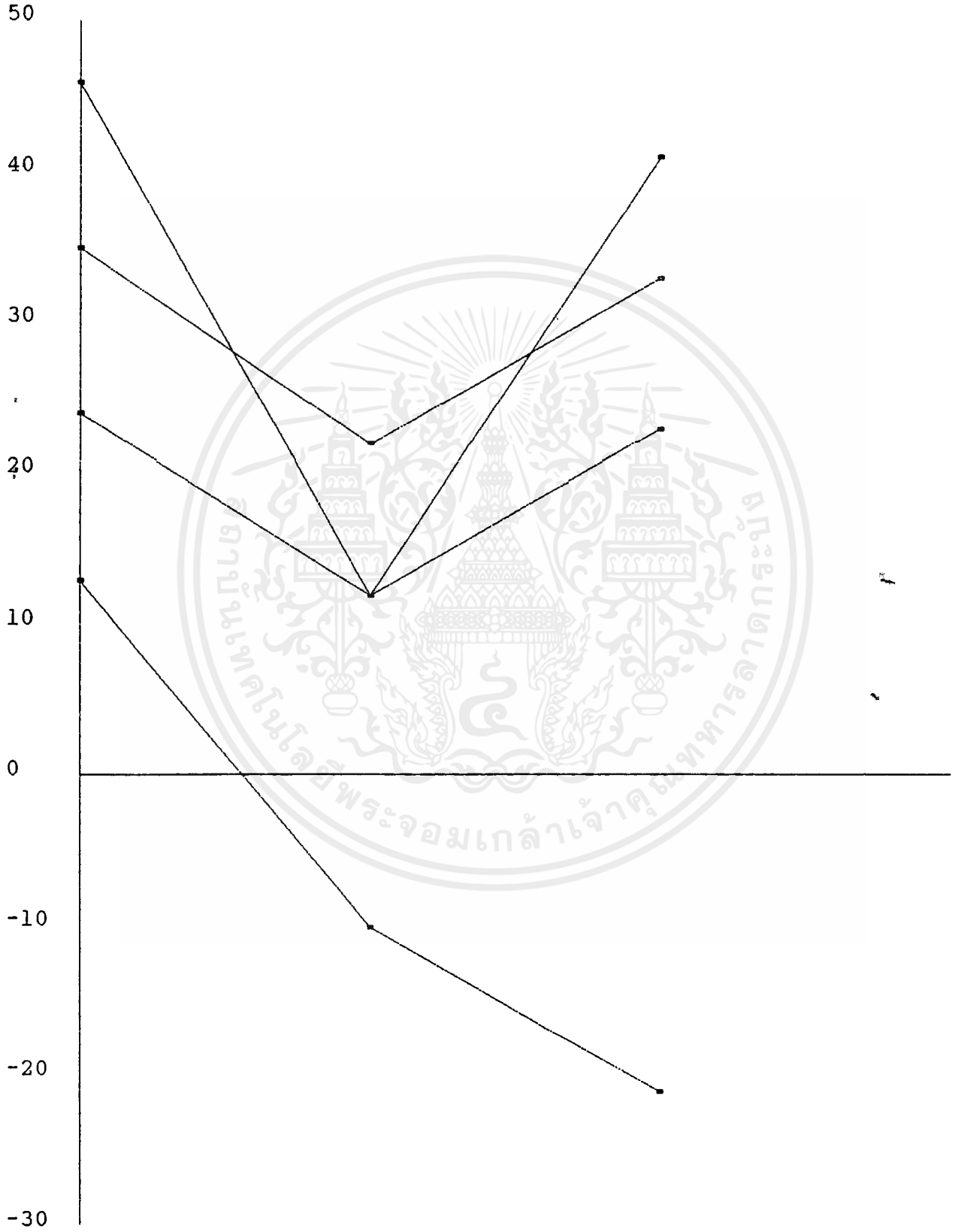
รูปที่ G.9 แสดง รูปกราฟเส้นตรง เมื่อเลือก option ในการพิมพ์แบบ landscape



รูปที่ G.10 แสดง รูปกราฟจุด เมื่อเลือก option ในการพิมพ์แบบ portrait



รูปที่ G.11 แสดง รูปกราฟจุด เมื่อเลือก option ในการพิมพ์แบบ landscape



รูปที่ G.8 แสดง รูปกราฟเส้นตรง เมื่อเลือก option ในการพิมพ์แบบ portrait

สูง ของ ROW ซึ่ง dialog box นี้เกิดจากการเลือกคำสั่งจาก menu ดังนั้น Format / Row Height เป็น dialog สำหรับให้ใส่ค่าความสูงของ row ที่ต้องการ เมื่อนำ mouse ไป click ที่ปุ่ม OK มันจะอ่านค่าที่เต็มเข้าไป และส่ง message ไปให้ Window ของ Client จัดการขยายความสูงของ row นั้นให้

input

1. hWnd = ค่า handle ประจำ dialog box
2. num = ค่าเลขประจำตัว dialog box
3. wParam = ค่า WORD PARAMETER ที่ระบบส่งผ่านมา
4. lParam = ค่า LONG PARAMETER ที่ระบบส่งผ่านมา

return TRUE ถ้าสร้าง dialog สำเร็จ
FALSE ถ้าสร้าง dialog ไม่สำเร็จ

COLWIDTHMsgProc (hWnd, num, wParam, lParam)

```
HWND    hWnd;
UINT    num;
WPARAM  wParam;
LPARAM  lParam;
```

หน้าที่ เป็น Procedure ที่ process เกี่ยวกับ dialog box ของการขยายความกว้างของ colum ซึ่ง dialog box นี้เกิดจากการเลือกคำสั่งจาก menu ดังนี้ Format/Column Width เป็น dialog box สำหรับใส่ค่าความกว้างของ column ที่ต้องการหรือ เลือกที่ปุ่ม Best Width มันจะคำนวณค่าความกว้างที่เหมาะสมสำหรับ Column นั้น ๆ โดยคัดจากค่า cell ใน column นั้นที่มีตัวอักษรมากที่สุด

input

1. hWnd = ค่า handle ประจำ dialog box

2. num = ค่าเลขประจำตัว dialog box

3. wParam = ค่า WORD PARAMETER ที่ระบบส่งผ่านมา

4. lParam = ค่า LONG PARAMETER ที่ระบบส่งผ่านมา

return TRUE ถ้าสร้าง dialog สำเร็จ
FALSE ถ้าสร้าง dialog ไม่สำเร็จ

INSERTMsgProc (hWnd, num, wParam, lParam)

```
HWND    hWnd;
UINT    num;
WPARAM  wParam;
LPARAM  lParam;
```

หน้าที่ เป็น Procedure ที่ process เกี่ยวกับ dialog box ของการ insert column หรือ row ที่ cell ที่ active อยู่ โดย insert ทั้ง row หรือ ทั้ง column และ row หรือ column ที่ insert เข้าไปนี้ จะมีค่าความสูง หรือ ความกว้างเป็นค่า de fault (ค่า default สำหรับ row = 20, ค่า default สำหรับ column = 70)

input

1. hWnd = ค่า handle ประจำ dialog box

2. num = ค่าเลขประจำตัว dialog box

3. wParam = ค่า WORD PARAMETER ที่ระบบส่งผ่านมา

4. lParam = ค่า LONG PARAMETER ที่ระบบส่งผ่านมา

return TRUE ถ้าสร้าง dialog สำเร็จ
FALSE ถ้าสร้าง dialog ไม่สำเร็จ

DELETEMsgProc (hWnd, num, wParam, lParam)

```
HWND    hWnd;
UINT    num;
WPARAM  wParam;
LPARAM  lParam;
```

หน้าที่ เป็น Procedure ที่ process เกี่ยวกับ dialog box ของการ delete ทั้ง column หรือทั้ง row ที่ cell active อยู่

input

1. hWnd = ค่า handle ประจำ dialog box
2. num = ค่าเลขประจำตัว dialog box
3. wParam = ค่า WORD PARAMETER ที่ระบบส่งผ่านมา
4. lParam = ค่า LONG PARAMETER ที่ระบบส่งผ่านมา

return TRUE ถ้าสร้าง dialog สำเร็จ
FALSE ถ้าสร้าง dialog ไม่สำเร็จ

Init Aplication (VOID)

หน้าที่ ลงทะเบียนทุก ๆ Window

input NONE

return TRUE ถ้าลงทะเบียนสำเร็จ
FALSE ถ้าลงทะเบียนไม่สำเร็จ

Init Instance (LPSTR, WORD)

หน้าที่ สร้างทุก ๆ window ด้วยคำสั่ง Create Window และจะได้ค่า Handle ของแต่ละ window เก็บไว้เป็นค่า global ดังนี้

1.hWnd Status เป็น handle ของ Window Status line

2.hWnd Main เป็น handle ของ Window Main Window

3.hWnd Child1 เป็น handle ของ Window แถบที่อยู่ของ tool bars

4.hWnd Ribbon เป็น handle ของ Window Ribbon

5.hWnd Edit เป็น handle ของ Window ที่ให้ใส่ข้อมูลที่ edit ลง cell

input

1. WORD เป็นคำ stage การโชว์ window (สำหรับ main window)

return TRUE ถ้าสร้าง และ show ทุก ๆ window สำเร็จ
FALSE ถ้า สร้างหรือ show window ใด window หนึ่งไม่สำเร็จ

DrawControl (hWnd, lpDraw)

```
HWND                hWnd;
LPDRAWITEMSTRUCT    lpDraw;
```

หน้าที่ เป็น Function ที่ใช้ในการ load ภาพ bitmap ของ toolbar ขึ้นมาแปะที่แถบ toolbar โดยภาพนั้นจะมีลักษณะเป็น toggle mode คือ กดครั้งแรกจะนำภาพปุ่มขณะปุ่มขึ้นมาแปะ เมื่อกดครั้งที่ 2 จะนำภาพ ปุ่มปกติขึ้นมาแปะ

input

1. hWnd = handle ของ window ที่เรียก function นี้
2. lpDArw = pointer ที่ชี้ไปยัง structure ของภาพ bitmap

return -

FillGray (hWnd, hDC, X, Y, W, L)

```
HWND  hWnd;
```

```
HDC   hDC;
int   X;
int   Y;
int   W;
int   L;
```

หน้าที่ เป็น fuction ที่ใช้ในการ fill สีเทา

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.hDC = handle ของ device context
3.X = ค่าตำแหน่งที่จุดซ้าย (left)
4.Y = ค่าตำแหน่งที่จุดบน (top)
5.W = ค่าความกว้างที่จะ fill
6.L = ค่าความสูงที่จะ fill

return -

PrintCell (hWnd, Col, Row, Mode)

```
HWND hWnd;
int  Col;
int  Row;
int  Mode;
```

หน้าที่ เป็น function ที่ใช้ในการพิมพ์ข้อมูลลง cell ลงเพียง 1 cell

input

1. hWnd = handle ของ window ที่เรียก function นี้
2. Col = ค่า Column
3. Row = ค่า row

4. Mode = เป็นค่า mode ว่าจะให้พิมพ์ text ลงไปด้วยปากกาสีอะไร ซึ่งถ้าค่า = 0 แสดง ว่าเลือกปากกาสีดำ ถ้าค่า = 1 แสดงว่าเลือกปากกาสีขาว (ใช้พิมพ์ในกรณีที่มีการ mark block)

return -

PrePrintCell (hWnd, Col, Row)

```
HWND  hWnd;
int   Col;
int   Row;
```

หน้าที่ เป็น function ที่ใช้ พิมพ์ข้อมูลลง cell เพียง 1 cell ซึ่งจะต่างจาก Print Cell ดังนี้ Pre print cell จะเป็นการพิมพ์ ข้อมูลลง cell ก่อนมีการตอบตกลงเอาข้อมูลนั้น (โดยการกด enter หรือ click ที่เครื่องหมายถูกที่ส่วน control) โดยจะพิมพ์ข้อมูลชิดขอบซ้ายของ cell เสมอ แต่เมื่อมีการตอบตกลง จะใช้ Print Cell ในการพิมพ์ข้อมูลลง cell โดยจะมีการพิจารณาข้อมูลว่า เป็น text หรือ ตัวเลข ถ้าเป็น text ก็จะยังคงพิมพ์ชิดซ้าย แต่ถ้าเป็นตัวเลขจะย้ายไปพิมพ์ชิดขวา

input

1. hWnd = handle ของ window ที่เรียก function นี้
2. Col = ค่า Column ของ cell
3. Row = ค่า row ของ cell

return -

PrintRibbon (hWnd, Col, Row, Alpha)

```
HWND  hWnd;
```

```
int   Col;
int   Row;
char *Alpha;
```

หน้าที่ เป็น function ที่ใช้ ที่พิมพ์ค่าตำแหน่งของ cell ที่ active ในรูปแบบ col, row

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.Col = ค่า Column ของ cell
3.Row = ค่า row ของ cell
4.Alpha = เป็น pointer ที่ชี้ไปที่เก็บค่า cloumn

return -

ClearEditBuffer (VOID)

หน้าที่ เป็น function ที่ใช้ clear ค่าที่อยู่ใน buffer คือ pEditBuffer ให้มีค่า เป็น NULL

input NULL
return -

DrawFrame (hWnd, Col, Row, X1 , Y1 , Mode)

```
HWND  hWnd;
int   Col;
int   Row;
int   X1;
int   Y1;
```

```
int   Mode;
```

หน้าที่ เป็น function ที่ใช้วาดกรอบสีดำ ที่เป็นตัวบอกไว้ว่า cell ไหน active

input

1.hWnd : handle ของ window ที่เรียก function นี้
2.Col : ค่า Column
3.Row : ค่า row
4.X1 : ค่าที่บอกถึง column แรกสุด
5.Y1 : ค่าที่บอกถึง row แรกสุด
6.Mode : บอกค่า mode ว่าเป็นการ draw frame ใหม่ หรือวาดค่ากรอบเทาเก่า

return -

ManageVSCroll (hWnd, wParam, lParam, VPos , VMax , rect, Mode)

```
HWND    hWnd;
WPARAM  wParam;
LPARAM  lParam;
SHORT   VPos;
SHORT   VMax;
RECT    rect;
int     Mode;
```

หน้าที่ เป็น function ที่ใช้ในการจัดการควบคุม Vertical scroll bar ทั้งเวลาที่ไม่มีการ split window และมีการ split window ซึ่งจะต้องมีการควบคุม Vertical scroll bar 2 อัน หน้าที่ในการควบคุมมีดังนี้

1.จัดการการเลื่อน sheet ขึ้นลง (โดยใช้คำสั่ง ScrollWindow)
2.จัดตำแหน่งของ scroll box (โดยใช้คำสั่ง SetScrollPos)

input

1.hWnd : handle ของ window ที่เรียก function นี้

2.wParam : ค่า Word Parameter ของ window ที่เรียก function นี้

3.lParam : ค่า Long Parameter ของ window ที่เรียก function นี้

4.VPos : ค่าตำแหน่งเดิมของ scroll box ที่อยู่บน scroll bar

5.VMax : ค่า range ที่มากที่สุดที่จะให้ scroll box เลื่อม

6.rect : structure ที่กำหนดตำแหน่งมุมสี่มุมที่จะให้ทำการ scroll

7.Mode : เป็นค่าที่บอกว่าอยู่ใน mode split window รึเปล่า

return ค่าที่ใช้ในการเปลี่ยนแปลงค่า indexY โดยจะนำค่าที่ return นี้มาบวกกับ indexY ถ้าค่านี้มีค่าบวกก็จะทำให้ indexY เพิ่มขึ้น ถ้าค่านี้มีค่าลบก็จะทำให้ indexY ลดลง

ManageHScroll (hWnd, wParam, lParam, HPos , HMax , rect, Mode)

```
HWND    hWnd;
WPARAM  wParam;
LPARAM  lParam;
SHORT   HPos;
SHORT   HMax;
RECT    rect;
int     Mode;
```

หน้าที่ เป็น function ที่ใช้ในการจัดการควบคุม Horizontal scroll bar ทั้งเวลาที่ไม่มีการ split window และมีการ split window ซึ่งจะต้องมีการควบคุม Vertical scroll bar 2 อัน หน้าที่ในการควบคุมมีดังนี้

1.จัดการการเลื่อน sheet ไปขวาซ้าย (โดยใช้คำสั่ง ScrollWindow)

2.จัดตำแหน่งของ scroll box (โดยใช้คำสั่ง SetScrollPos)

input

1.hWnd : handle ของ window ที่เรียก function นี้

2.wParam : ค่า Word Parameter ของ window ที่เรียก function นี้

3.lParam : ค่า Long Parameter ของ window ที่เรียก function นี้

4.HPos : ค่าตำแหน่งเดิมของ scroll box ที่อยู่บน scroll bar

5.HMax : ค่า range ที่มากที่สุดที่จะให้ scroll box เลื่อม

6.rect : structure ที่กำหนดตำแหน่งมุมสี่มุมที่จะให้ทำการ scroll

7.Mode : เป็นค่าที่บอกว่าอยู่ใน mode split window รึเปล่า

return ค่าที่ใช้ในการเปลี่ยนแปลงค่า indexX โดยจะนำค่าที่ return นี้มาบวกกับ indexX ถ้าค่านี้มีค่าบวกก็จะทำให้ indexX เพิ่มขึ้น ถ้าค่านี้มีค่าลบก็จะทำให้ indexX ลดลง

DrawGrid(hWnd, hDC, X1 , Y1 , X2 , Y2 , idX, idY)

```
HWND  hWnd;
HDC   hDC;
int   X1;
int   Y1;
int   X2;
int   Y2;
int   idX;
int   idY;
```

หน้าที่ เป็น function ที่ใช้ในการ ตีตาราง grid สีเทา

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.hDC = handle ของ device context

3.X1 = ค่า column เริ่มต้น

4.Y1 = ค่า row เริ่มต้น

5.X2 = ค่า column สุดท้าย

6.Y2 = ค่า row สุดท้าย

7.idX = ค่า indexX ประจำ sheet

8.idY = ค่า indexY ประจำ sheet

return -

DrawTitleX(hWnd, hDC, X1 , X2 , idX)

```
HWND  hWnd;
HDC   hDC;
int   X1;
int   X2;
int   idX;
```

หน้าที่ เป็น function ที่ใช้ในการวาดรูปให้เป็นปุ่มในแกน X (column)

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.hDC = handle ของ device context

3.X1 = ค่า column เริ่มต้น

4.X2 = ค่า column สุดท้าย

5.idX = ค่า indexX ประจำ sheet

return -

DrawTitleY (hWnd, hDC, Y1 , Y2 , idY)

```
HWND  hWnd;
HDC   hDC;
```

```
int   Y1;
int   Y2;
int   idY;
```

หน้าที่ เป็น funciton ที่ใช้วาดรูปให้เป็นปุ่มในแกน Y (row)

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.hDC = handle ของ device context
3.Y1 = ค่า column เริ่มต้น
4.Y2 = ค่า column สุดท้าย
5.idY = ค่า indexX ประจำ sheet

return -

WriteTitleX (hWnd, hDC, X, idX)

```
HWNd   hWnd;
HDC    hDC;
int    X;
int    idX;
```

หน้าที่ เป็น function ที่ใช้ในการ เขียนค่าของ column (A,B,C,...) ลงใน ปุ่ม ที่วาดโดย Draw Title X

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.hDC = handle ของ device context
3.X = ค่า column เริ่มต้น
4.idX = ค่า indexX ประจำ sheet

return -

WriteTitleY (hWnd, hDC, Y, idY)

```
HWNd    hWnd;
HDC     hDC;
int     Y;
int     idY;
```

หน้าที่ เป็น function ใช้ในการเขียนค่าของ row (1, 2, 3, 4,...) ลงในปุ่มที่วาดโดย Draw Title Y

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.hDC = handle ของ device context
3.Y = ค่า row เริ่มต้น
4.idY = ค่า indexY ประจำ sheet

return -

ManageKey (hWnd, wParam)

```
HWND    hWnd;
WPARAM wParam;
```

หน้าที่ เป็น function ที่ใช้ในการจัดการ key ลูกศร ซ้าย, ขวา, บน, ล่าง เป็น function ผ่านเท่านั้นจะส่งค่าไปให้ function Manage HSCroll หรือ Manage VScroll ทำงานอีกที ถ้าเป็นลูกศร ซ้ายขวา จะส่งต่อให้ function ManageHScroll ถ้าเป็นลูกศรขึ้นลง จะส่งต่อให้ function ManageVScroll

input

1.hWnd = handle ของ window ที่เรียก function นี้

4.wParam = ค่า Word Parameter ที่ส่งมาจาก window ที่เรียก function นี้ ในที่นี้จะหมายถึงค่า scan code ของคีย์ที่กด

return -

DisplayScreen (hWnd, X1 , X2 , Y1 , Y2 , Mode, idX, idY)

```
HWND  hWnd;
int   X1;
int   X2;
int   Y1;
int   Y2;
int   Mode;
int   idX;
int   idY;
```

หน้าที่ เป็น function ที่ใช้ในการแสดงค่าข้อมูลของ cell หลาย ๆ cell โดยมีการทำงานดังนี้

1.ตรวจสอบค่า color (input ตัวที่ 6) ว่าจะใช้ปากกา และ background สีอะไร

2.วนลูปตามค่า column และ row ที่ให้มาเพื่อทำการหาค่าข้อมูลของแต่ละ cell

3.เมื่อทราบ structure ของข้อมูลแล้วให้พิมพ์ข้อมูลลง cell ตาม format ของข้อมูลดังนี้

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X1 = ค่า Column แรกสุด

3.X2 = ค่า Column สุดท้าย

4.Y1 = ค่า Row แรกสุด

5.Y2 = ค่า Row สุดท้าย

6.Mode = ค่า color เป็นตัวกำหนดสีปากกา และสี background

7.idX = ค่า indexX ประจำ sheet

8.idY = ค่า indexY ประจำ sheet

return -

SaveBuffer (hWnd)

```
HWNd   hWnd;
```

หน้าที่ เป็น function ที่ใช้ในการ save ข้อมูลของแต่ละ cell (Cell ที่ active อยู่) ลงใน data structure ของมัน จะมีการทำงานดังนี้

1.ตรวจสอบค่า Feature ของ cell นั้น ๆ

2.call function editcell เพื่อทำการเก็บข้อมูลลง data structure ของแต่ละ cell

input

1.HWND = handle ของ window ที่เรียก function นี้

return -

DrawGhostLineX (hWnd, x , rect)

```
HWND  hWnd;
int   x;
```

```
RECT  rect;
```

หน้าที่ เป็น function ที่ใช้ในการวาดเส้นประในแนวแกนตรง ขณะที่มีการกด mouse เลื่อนขยายความกว้างของ Column มีขั้นตอนการทำงานดังนี้

1. ทำการวาดภาพเก่าลงทับตำแหน่งที่ Ghost line วิ่งผ่าน (ดดย function BitBlt)
2. ลากเส้นประจากตำแหน่ง Y บนสุดมายังตำแหน่งล่างสุด โดยตำแหน่ง X คือตำแหน่งที่ click mouse ผ่าน

input

1. hWnd = handle ของ window ที่เรียก function นี้
2. x = ตำแหน่งในแนวแกน X
3. rect = เป็น struct รูปสี่เหลี่ยมบอกถึงบริเวณที่ให้วาดรูปเก่าทับด้วย function Bitblt

return -

DrawGhostLineY (hWnd, y , rect)

```
HWND  hWnd;
int   y;
RECT  rect;
```

หน้าที่ เป็น function ที่ใช้ในการวาดเส้นประในแนวแกนขวางขณะที่มีการกด mouse เลื่อนขยายความสูงของ row มีขั้นตอนการทำงานดังนี้

1. ทำการวาดภาพเก่าลงทับตำแหน่งที่ Ghost line วิ่งผ่าน (ดดย function BitBlt)
2. ลากเส้นประจากตำแหน่ง X ขวาสุดมายังตำแหน่งซ้ายสุด โดยตำแหน่ง Y คือตำแหน่งที่ click mouse ผ่าน

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.y = ตำแหน่งในแนวแกน Y

3.rect = เป็น struct รูปสี่เหลี่ยมบอกถึงบริเวณที่ให้วาดรูปเก่าทับด้วย function Bitblt

return -

PaintScreenX (hWnd, rect)

```
HWND  hWnd;
RECT  rect;
```

หน้าที่ เป็น function ที่ใช้วาด แถบปุ่มของ column ใหม่หมด โดยมีขั้นตอนทำงานดังนี้

1.call function FillGray เพื่อทำการ fill สีเทาลงในแนวแถว Column (TitleX)

2.call function DrawTitleX เพื่อทำการวาดปุ่มปุ่ม ลงในแนวแถว Column (TitleX)

3.call function WriteTitleX เพื่อทำการเขียนตัวอักษรประจำ Column (A, B, C, ..) ลงในแนวแถว Column (TitleX)

4.call function DrawGrid เพื่อทำการตีตารางเส้น grid ใหม่หมด

5.ตรวจสอบว่าอยู่ใน split mode รึเปล่า ถ้าใช่ก็ทำการย้าย Hscroll bar ทั้ง 2 อันให้อยู่ในตำแหน่งที่เหมาะสม

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.rect = บอกบริเวณที่ให้ PaintScreen

return -

PaintScreenY (hWnd, rect)

```
HWND  hWnd;
RECT  rect;
```

หน้าที่ เป็น function ที่ใช้วาด แถบปุ่มของ row ใหม่หมด โดยมีขั้นตอนทำงานดังนี้

1.call function FillGray เพื่อทำการ fill สีเทาลงในแนวแถว Row (TitleY)

2.call function DrawTitleY เพื่อทำการวาดปุ่มปุ่ม ลงในแนวแถว Row (TitleY)

3.call function WriteTitleY เพื่อทำการเขียนตัวเลขประจำ Row (1, 2, 3, ..) ลงในแนวแถว Row (TitleY)

4.call function DrawGrid เพื่อทำการตีตารางเส้น grid ใหม่หมด

5.ตรวจสอบว่าอยู่ใน split mode รึเปล่า ถ้าใช่ก็ทำการย้าย Vscroll bar ทั้ง 2 อันให้อยู่ในตำแหน่งที่เหมาะสม

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.rect = บอกบริเวณที่ให้ PaintScreen

return -

Move_Data (Scol1, Srow1 , Scol2, Srow2, Dcol1 ,Drow1)

```
int Scol1;
int Srow1;
int Scol2;
int Srow2;
int Dcol1;
int Drow1;
```

หน้าที่ เป็น function ที่ใช้ในการ move cell จาก source ที่ให้มายัง destination

input

1.Scol1 = ค่า Column เริ่มต้น (source)

2.Srow1 = ค่า Row เริ่มต้น (source)

3.Scol2 = ค่า Column สุดท้าย (source)

4.Srow2 = ค่า Row สุดท้าย (source)

5.Dcol1 = ค่า Column เริ่มต้น (destination)

6.Drow1 = ค่า Row สุดท้าย (destination)

return -

DeleteCell (hWnd, Col, Row)

```
HWND  hWnd;
int   Col;
int   Row;
```

หน้าที่ เป็น function ใช้ในการลบคำข้อมูล cell นั้น ๆ โดยแค่ fill สีขาว บง ใน cell เท่านั้น ไม่ได้ไปลบที่ตัว structure จริง

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.Col = Column

3.Row = Row

return -

MarkBLock (hWnd, X, Y)

```
HWND hWnd;
int  X;
int  Y;
```

หน้าที่ เป็น function ใช้ในการ fill แถบสีดำในกรณีที่ mark blockมีขั้นตอนการทำงานดังนี้

1.คำนวณหา Column และ Row ที่ต้องการ

2.call function FillColor เพื่อทำการ fill สีลง block

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X จุดที่ mouse อยู่

3.Y = ตำแหน่ง Y จุดที่ mouse อยู่

return -

ClearBlock (hWnd)

```
HWND  hWnd;
```

หน้าที่ เป็น function ที่ใช้ในการ clear แถบดำที่ได้ mark block เอาไว้

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

ChKTooBars (VOID)

หน้าที่ เป็น function ที่ใช้ในการพิจารณาว่ามีปุ่ม toolbar ปุ่มไหนอยู่ใน state

ที่ได้รับการกดค้างเอาไว้

input NULL

return -

ChkBuffer (VOID)

หน้าที่ เป็นการตรวจสอบว่า cell ที่ active นั้นมี data อะไรอยู่ โดยการ call function search(เพื่อนำไปพิมพ์ลง cell)

input NULL

return -

ReadrawGrid (hDC, X1 , Y1 , X2 , Y2 , idX, idY)

```
HDC  hDC;
int  X1;
int  Y1;
int  X2;
int  Y2;
int  idX;
int  idY;
```

หน้าที่ จะคล้ายกับการ Draw Grid แต่จะเป็นการวาดเส้น grid ใหม่บางส่วนเท่านั้น จะมีขั้นตอนทำงานดังนี้

1.คำนวณหา Column และ Row ทั้ง 4 มุม
2.ลากเส้น grid สีเทาตามขอบเขตที่กำหนด
3.ถ้าอยู่ใน mode split window ให้วาดเส้น split ด้วย

input

1.hDC = handle ของ device context

2.X1 = Column แรก

3.Y1 = Row แรก

4.X2 = Column สุดท้าย

5.Y2 = Row สุดท้าย

6.idX = indexX ประจำ sheet

7.idY = indexY ประจำ sheet

return -

```
DrawBigFrame (hDC, X  , Y  , X1 , Y1 , X2 , Y2 ,Mode, idX, idY)

        HDC  hDC;
        int  X;
        int  Y;
        int  X1;
        int  Y1;
        int  X2;
        int  Y2;
        int  Mode;
        int  idX;
        int  idY;
```

หน้าที่ จะคล้ายกับ DrawFrame แต่จะเป็นการวาด frame สำหรับ block ที่ mark เอาไว้ จะเป็นการวาดกรอบสีดำ 2 กรอบซ้อนกัน

input

1.hDC = handle ของ device context

2.X = ค่า Column ณ.จุดที่นำ mouse ไป click

3.Y = ค่า Row ณ.จุดที่นำ mouse ไป click

4.X1 = ค่าตำแหน่ง X เริ่มต้น

5.Y1 = ค่าตำแหน่ง Y เริ่มต้น

6.X2 = ค่าตำแหน่ง X สุดท้าย

7.Y2 = ค่าตำแหน่ง Y สุดท้าย

8.Mode = ค่า mode ที่บอกว่าให้วาดกรอบด้วยปากกาสีอะไร

-0 ใช้ปากกาสีดำ

-1 ใช้ปากกาสีขาว

9.idX = indexX ประจำ sheet

10.idY = indexY ประจำ sheet

return -

DeleteBigFrame (hWnd, X1 , Y1 , X2 , Y2 , idX, idY)

```
HWND  hWnd;
int   X1;
int   Y1;
int   X2;
int   Y2;
int   idX;
int   idY;
```

หน้าที่ จะลบกรอบที่วาดไว้โดย function DrawBigFrame โดยใช้ปากกาสีขาว ลากทับกรอบสีดำที่วาดไว้โดย function DrawBigFrame

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X1 = Column แรก

3.Y1 = Row แรก

4.X2 = Column สุดท้าย

5.Y2 = Row สุดท้าย

6.idX = indexX ประจำ sheet

7.idY = indexY ประจำ sheet

return -

DrawFrameMove (hWnd, X , Y , X1 , Y1 , Mode)

```
HWND  hWnd;
int   X;
int   Y;
int   X1;
int   Y1;
int   Mode;
```

หน้าที่ จะวาดกรอบสำหรับการ move โดยกรอบนี้จะเลื่อนไปมาตาม mouse จะเป็นกรอบสีเทาเข้มที่มีเส้นความหนา = 3

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X ณ.จุดที่ click mouse

3.Y = ตำแหน่ง Y ณ.จุดที่ click mouse

4.X1 = Column แรก

5.Y1 = Row แรก

6.Mode = ค่าที่บอกว่าควรใช้ปากกาสีอะไร

-0 ใช้ปากกาสีขาว

-1 ใช้ปากกาสีเทา (RGB(122, 88, 124))

return -

RedrawGridMove (hWnd, X , Y , X1 , Y1)

```
HWND  hWnd;
int   X;
int   Y;
int   X1;
int   Y1;
```

หน้าที่ จะวาด grid ใหม่ตามหลังการเรียก function drawFrameMove

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.X = ตำแหน่ง X ณ.จุดที่ click mouse
3.Y = ตำแหน่ง Y ณ.จุดที่ click mouse
4.X1 = Column แรก
5.Y1 = Row แรก

return -

DrawNewBlock (hWnd, X , Y , X1 , Y1)

```
HWND  hWnd;
int   X;
int   Y;
int   X1;
int   Y1;
```

หน้าที่ จะทำการ Mark Block โดย fill แถบสีดำ จะมีขั้นตอนการทำงานดังนี้

1.คำนวณหา ตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย

2.call function FillColor เพื่อทำการ fill สี

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X ณ.จุดที่ click mouse

3.Y = ตำแหน่ง Y ณ.จุดที่ click mouse

4.X1 = Column แรก

5.Y1 = Row แรก

return -

ControlCopy (hWnd, X , Y , SX1, SY1, SX2, SY2, DX1, DY1, DX2, DY2)

```
HWND hWnd;
int  X;
int  Y;
int  SX1;
int  SY1;
int  SX2;
int  SY2;
int  DX1;
int  DY1;
int  DX2;
int  DY2;
```

หน้าที่ จะควบคุมการทำงานขณะอยู่ใน mode copy โดยมีขั้นตอนการทำงานดังนี้

1.คำนวณหา Column และ Row เริ่มต้นของการ copy

2.call function Copy เพื่อทำการ copy ข้อมูลลง destination

3.call function DrawNewBlock เพื่อทำการ fill แถบสีดำลงบริ

เวณของ destination

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X ณ.จุดของ Column แรกที่จะ copy

3.Y = ตำแหน่ง Y ณ.จุดของ Row แรกที่จะ copy

4.SX1 = Column แรก ของ Source

5.SY1 = Row แรกของ Source

6.SX2 = Column สุดท้ายของ Source

7.SY2 = Row สุดท้ายของ Source

8.DX1 = Column แรก ของ Destination

9.DY1 = Row แรกของ Destination

10.DX2 = Column สุดท้ายของ Destination

11.DY2 = Row สุดท้ายของ Destination

return -

SplitWindow (hWnd)

HWND hWnd;

หน้าที่ จะทำการวาดเส้น split ถ้าเลือกคำสั่ง window/split และทำการ save ค่า column และ row ที่ split เอาไว้ในค่าตัวแปร Global SplitX และ SplitY ตามลำดับ

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

DrawSplit (hWnd)

HWND hWnd;

หน้าที่ จะทำการวาดเส้น split กรณีที่มีการกระทำอื่นมาทับเส้น split ซึ่งเส้น splitจะมีลักษณะเป็นเส้นสีเทาที่มีเส้นความหนา 3 สองเส้นตัดกัน

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

RemoveSplitWindow (hWnd)

HWND hWnd;

หน้าที่ จะลบเส้น split ทั้งแนว แกน X และ แกน Y ออกโดยใช้ปากกาสีขาวลากทับบริเวณที่ทำการ DrawSplit และทำการวาด grid ที่ส่วนนั้นโดยมีขั้นตอนการทำงานดังนี้

1.ทำลาย VScroll และ HScroll สำหรับการ split (โดยคำสั่ง DestroyWindow)

2.ทำการ show VScroll และ HScroll อันเก่า (โดยคำสั่ง SetScrollRange)

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

MarkTitle (hWnd, X)

```
HWND hWnd;
int  X;
```

หน้าที่ จะทำการ mark block ที่ column ที่ต้องการ ทั้ง column โดยมีขั้นตอนการทำงานดังนี้

1.หาตำแหน่งมุมทั้ง 4 มุม

2.call function DrawNewBlock เพื่อทำการ fill สีดำลงทั้ง Column ที่เลือกเอาไว้

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ค่า Column

return -

Copy (hWnd, SX1, SY1, SX2, SY2, DX1, DY1, DX2, DY2)

```
HWND hWnd;
int  SX1;
int  SY1;
int  SX2;
int  SY2;
int  DX1;
int  DY1;
int  DX2;
int  DY2;
```

หน้าที่ จะทำการ fill ข้อมูลจาก source ลง destination โดยมีการ step ขั้น โดยมีขั้นตอนการทำงานดังนี้

1.call function search หาข้อมูลจาก source

2.call function fill ทำการกรอกข้อมูลลงใน data structure ของ destination แบบมีการ step ขั้น

input

1.HWND = handle ของ window ที่เรียก function นี้

2.SX1 = Column เริ่มต้น (source)

3.SY1 = Row เริ่มต้น (source)

4.SX2 = Column สุดท้าย (source)

5.SY2 = Row สุดท้าย (source)

6.DX1 = Column เริ่มต้น (destination)

7.DY1 = Row เริ่มต้น (destination)

8.DX2 = Column สุดท้าย (destination)

9.DY2 = Row สุดท้าย (destination)

return -

DrawFrameSplit (hWnd, Col, Row)

```
HWND  hWnd;
int   Col;
int   Row;
```

หน้าที่ จะทำการวาด frame ขณะที่อยู่ใน mode split window จะต่างจาก function DrawFrame คือจะมีการพิจารณาว่า frame ของ cellที่ active อยู่นั้นอยู่ติดกับเส้น split รึเปล่า

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.Col = Column ที่ active

3.Row = Row ที่ active

return -

ControlTwinCell (hWnd, Mode)

```
HWND hWnd;
int  Mode;
```

หน้าที่ จะควบคุมการพิมพ์ข้อมูล แสดงข้อมูลขณะที่อยู่ใน mode split window โดยจะพิจารณาว่าcell ที่ active อยู่ใน window ที่ active นั้นยังไปปรากฏอยู่ใน window ช่องอื่นรึเปล่า ถ้าใช่ให้จัดการพิมพ์ข้อมูลนั้นลงไปด้วย

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.Mode = mode ที่เลือกรูปแบบการพิมพ์
-0 เป็นการพิมพ์แบบ PrePrintCell
-1 เป็นการพิมพ์แบบ PrintCell

return -

WInsertColumn (hWnd)

```
HWND hWnd;
```

หน้าที่ จะทำการ insert column ใหม่มีค่าความกว้างเท่ากับค่า default = 70 โดยมีขั้นตอนการทำงานดังนี้

1.call function UpdataW เพื่อทำการแก้ไขค่าความกว้างของแต่ละ Column ลงในตัวแปร WIDTH[]
2.call function PaintScreenX เพื่อทำการวาดหน้าจอในแนวแกน X ใหม่

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

WInsertRow (hWnd)

```
HWND hWnd;
```

หน้าที่ จะทำการ insert row ใหม่มีค่าความสูง เท่ากับค่า default = 20
โดยมีขั้นตอนการทำงานดังนี้

1.call function UpdataL เพื่อทำการแก้ไขค่าความสูงของแต่ละ Row ลงในตัวแปร LENGTH[]

2.call function PaintScreenY เพื่อทำการวาดหน้าจอในแนวแกน X ใหม่

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

WDeleteColumn (hWnd)

```
HWND hWnd;
```

หน้าที่ จะทำการ ลบ column ที่ cell นั้น active อยู่โดยมีขั้นตอนการทำงานดังนี้

1.call function UpdataW เพื่อทำการแก้ไขค่าความกว้างของแต่ละ Column ลงในตัวแปร WIDTH[]

2.call function PaintScreenX เพื่อทำการวาดหน้าจอในแนวแกน X ใหม่

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

WDeleteRow (hWnd)

```
HWND hWnd;
```

หน้าที่ จะทำการ ลบ row ที่ cell นั้น active อยู่โดยมีขั้นตอนการทำงานดังนี้

1.call function UpdataL เพื่อทำการแก้ไขค่าความสูงของแต่ละ Row ลงในตัวแปร LENGTH[]

2.call function PaintScreenY เพื่อทำการวาดหน้าจอในแนวแกน X ใหม่

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

UpdateW(hWnd, X, Mode)

```
HWND hWnd;
int  X;
int  Mode;
```

หน้าที่ ทำการคำนวณความกว้างทุก Column ใหม่ ในกรณีที่มีการ insert หรือ delete column

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = Column ที่ทำการ delete หรือ insert

3.Mode = mode

-0 เป็น insert mode

-1 เป็น delete mode

return -

UpdateL(hWnd, Y, Mode)

```
HWND hWnd;
int  Y;
int  Mode;
```

หน้าที่ ทำการคำนวณความกว้างทุก row ใหม่ ในกรณีที่มีการ insert หรือ delete row

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.Y = Row ที่ทำการ delete หรือ insert

3.Mode = mode

-0 เป็น insert mode

-1 เป็น delete mode

return -

SetStatusLine(chptr)

```
char _far *chptr;
```

หน้าที่ จะพิมพ์ค่า coment ที่ต้องการลงในแถบ status line

input

1.chptr = pointer ที่ชี้ไปยังประโยคที่ต้องการพิมพ์

return -

PopFileInitailize (hWnd)

HWND hWnd;

หน้าที่ จะ initial ค่าที่จำเป็นใน structure OPENFILENAME ก่อนจะมีการ open file

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

PopFontInitialize (hWnd)

HWND hWnd;

หน้าที่ จะ initial ค่าที่จำเป็นใน structure font

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

PopfileOpenDlg (hWnd, lpstrFilename, lpStrTitleName)

```
        HWND    hWnd;
        LPSTR   lpStrFilkename;
        LPSTR   lpStrTitleName;
```

หน้าที่ จะสร้าง common dialog สำหรับการ open file

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.lpStrFileName = pointer ที่ชี้ไปยัง FileName

3.lpStrTitleName = pointer ที่ชี้ไปยัง TitleName

return -

PopFileSaveDlg (hWnd, lpStrFileName, lpStrTitleName)

```
        HWND    hWnd;
        LPSTR   lpStrFilkename;
        LPSTR   lpStrTitleName;
```

หน้าที่ จะสร้าง common dialog สำหรับการ save file

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.lpStrFileName = pointer ที่ชี้ไปยัง FileName

3.lpStrTitleName = pointer ที่ชี้ไปยัง TitleName

return -

PopPrntPrintFile (hWnd, lpStrTitlename)

```
        HWND    hWnd;
        LPSTR   lpStrTitleName;
```

หน้าที่ จะสร้าง common dialog สำหรับการ print

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.lpStrTitleName = pointer ที่ชี้ไปยัง TitleName

return -

PopFontChooseFont (hWnd)

```
HWND  hWnd;
```

หน้าที่ จะสร้าง common dialog สำหรับการเลือก font

input

1.hWnd = handle ของ window ที่เรียก function นี้

return -

MakeNewChild (name)

```
char * name;
```

หน้าที่ สร้าง Multidocument อันใหม่

input

1.name = ชื่อของ multi document อันใหม่

return handle ของ multi document อันใหม่

CreateHScrool (hWnd, X, Y, W, L)

```
HWND hWnd;
int  X;
```

```
        int  Y;
        int  W;
        int  L;
```

หน้าที่ สร้าง horizontal scrool bar ขณะที่ split window

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X ของ Scroll bar

3.Y = ตำแหน่ง Y ของ Scroll bar

4.W = ความกว้างของ Scroll bar

5.L = ความสูงของ Scroll bar

return Handle ของ Scroll bar

```
CreateVScroll (hWnd, X, Y, W, L)

        HWND hWnd;
        int  X;
        int  Y;
        int  W;
        int  L;
```

หน้าที่ สร้าง vertical scrool bar ขณะที่ split window

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.X = ตำแหน่ง X ของ Scroll bar

3.Y = ตำแหน่ง Y ของ Scroll bar

4.W = ความกว้างของ Scroll bar

5.L = ความสูงของ Scroll bar

return Handle ของ Scroll bar

CalPositionX (hWnd, x)

```
HWND hWnd;
int  x;
```

หน้าที่ คำนวณค่า Column ในตำแหน่งที่ click mouse

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.x = ตำแหน่ง X ณ.จุดที่ click mouse

return ค่า Column

CalPositionY (hWnd, y)

```
HWND hWnd;
int  y;
```

หน้าที่ คำนวณค่า row ในตำแหน่งที่ click mouse

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.y = ตำแหน่ง Y ณ.จุดที่ click mouse

return ค่า Row

ChkBorderX (hWnd, x)

```
HWND hWnd;
int  x;
```

หน้าที่ ตรวจสอบว่า ค่าตำแหน่ง X นั้น เป็นขอบของแต่ละ column รึเปล่า

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.x = ตำแหน่ง X ณ.จุดที่ลาก mouse ผ่าน

return 0 ถ้าจุดที่ลาก mouse ผ่านเป็นขอบ

1 ถ้าจุดที่ลาก mouse ผ่านไม่เป็นขอบ

ChkBorderY (hWnd, y)

```
HWND hWnd;
int  x;
```

หน้าที่ ตรวจสอบว่า ค่าตำแหน่ง Y นั้น เป็นขอบของแต่ละ row ริเปล่า

input

1.hWnd = handle ของ window ที่เรียก function นี้

2.y = ตำแหน่ง Y ณ.จุดที่ลาก mouse ผ่าน

return 0 ถ้าจุดที่ลาก mouse ผ่านเป็นขอบ

1 ถ้าจุดที่ลาก mouse ผ่านไม่เป็นขอบ

FindStartX (x, idX)

```
int x;
int idX;
```

หน้าที่ หาตำแหน่งเริ่มต้นของ column (X)

input

1.x = ตำแหน่ง X

2.idX = indexX ประจำ sheet

return ค่า Column ที่สอดคล้องกับตำแหน่ง x

FindStartY (y, idY)

```
int y;
int idY;
```

หน้าที่ หาตำแหน่งเริ่มต้นของ row (Y)

input

1.y = ตำแหน่ง Y

2.idY = indexY ประจำ sheet

return ค่า Row ที่สอดคล้องกับตำแหน่ง y

Different (num1, num2)

```
int num1;
int num2;
```

หน้าที่ หาค่าความแตกต่างระหว่างเลข 2 ตัว (เป็นค่า absolute)

input

1.num1 = เลขตัวแรก

2.num2 = เลขตัวที่สอง

return ค่าความแตกต่างระหว่างเลข 2 ตัว (เป็นค่า absolute)

Toggle (OldNum)

```
int OldNum;
```

หน้าที่ ทำการสลับค่าตัวเลข ดังนี้ ถ้า OldNum = 0 จะ toggle ให้เป็น 1
ถ้า OldNum = 1 จะ toggle ให้เป็น 0

input

1.OldNum = ตัวเลขที่เป็น i/p ที่ส่งมาให้ toggle

return ค่าตัวเลขที่ทำการ toggle ค่า OldNum แล้ว

PressTitle (hWnd, x, y)

```
HWND hWnd;
int  x;
int  y;
```

หน้าที่ ตรวจสอบว่ามีการกดที่ Titlex หรือ Title Y รึเปล่า

input

1.hWnd = handle ของ window ที่เรียก function นี้
2.x = ตำแหน่ง x ที่มีการ click mouse
3.y = ตำแหน่ง y ที่มีการ click mouse

return 0 ถ้าเป็นการกดที่ ขอบColumn หรือ ขอบ Row
1 ถ้าไม่ได้กดที่ ขอบColumn หรือ ขอบ Row

Between (num1, num2, num3)

```
int num1;
int num2;
int num3;
```

หน้าที่ ตรวจสอบว่าค่า num2 มีค่าอยู่ระหว่าง num1 และ num3 รึเปล่า

input

1.num1 = ตัวเลขขอบตัวแรก
2.num2 = ตัวเลขที่ต้องการตรวจสอบ
3.num3 = ตัวเลขขอบตัวสุดท้าย

return TRUE ถ้า num2 อยู่ระหว่าง num1 และ num2
FALSE ถ้า num2 ไม่อยู่ระหว่าง num1 และ num2

ฟังก์ชัน DrawBitmapP(hdc,xStart,yStart,pic)

```
HDC     hdc;
short   xStart;
short   yStart;
int     pic;
```

หน้าที่ เป็นฟังก์ชันสำหรับนำภาพ Bitmap ที่ได้สร้างไว้แล้ว (โดยใช้ Imagedit Tool หรือ อื่น ๆ) นำมาติดทับลงบน window ณ.ตำแหน่งที่ต้องการ

input 1.hdc

คือค่า handle device context

2.xStart

คือค่า x-position ของตำแหน่งที่ต้องการนำภาพ Bitmap ไปติด โดยค่า x-position จะมีค่า reference กับตำแหน่งกับจุดบนซ้ายสุด ของ window ที่เราต้องการติดภาพ ซึ่ง ณ.จุดซ้ายสุดของ window นั้น ๆ จะมีค่า x-positionเป็น 0 เสมอ

3.yStart

คือค่า y-position ของตำแหน่งที่ต้องการนำภาพ Bitmap ไปติด โดยค่า y-position จะมีค่า reference กับตำแหน่งกับจุดบนซ้ายสุด ของ window ที่เราต้องการติดภาพ ซึ่ง ณ.จุดซ้ายสุดของ window นั้น ๆ จะมีค่า y-positionเป็น 0 เสมอ

4.เป็นค่า ID ของ Bitmap

return value -

Drawbotton(hdc)

```
HDC hdc;
```

หน้าที่ เป็นเป็นฟังก์ชันที่นำภาพ Bitmap 23 ภาพ ไปติดบน window โดยจะเรียกใช้ฟังก์ชัน DrawBitmapPอีกที (จุดประสงค์หลักของ ฟังก์ชันนี้ก็เพื่อติดภาพปุ่ม Toolbar ทั้ง 23 ปุ่มลงบน window ที่ทำหน้าที่เป็นแถบ Toolbar)

input 1.hdc

คือค่า handle device context

return value -

G_Bar(hdc,cxPage,cyPage,P_hPen,p_hBrush,P_Value,Br,Bc,Er,Ec)

```
HDC     hdc;
short   cxPage;
short   cyPage;
HPEN    *P_hPen;
HBRUSH  *p_hBrush;
float   P_Value;
short   Br;             /*begin row*/
short   Bc;             /*begin column*/
short   Er;             /*end row*/
short   Ec;             /*end column*/
```

หน้าที่ แสดงค่าข้อมูลให้อยู่ในรูปแบบของกราฟแท่งบน output device เช่น บนจอภาพ, บน printer , บน ploter หรือ output device อื่น ๆ

input 1.hdc

เป็น handle device context ของ output device ที่ต้องการแสดงรูปกราฟ ตย.เช่น handle device context อาจจะเป็น ของ จอภาพ หรือ Printer ก็ได้

2.cxPage

เป็นความกว้างของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

3.cyPage

เป็นความยาวของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

4.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

5.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

6.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

7.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

8.Bc

เป็นค่าของตำแหน่ง columnเริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

9.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

10.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value -

G_Bar_PLUS (hdc,cxPage,cyPage,P_hPen,p_hBrush,P_Value,Br,Bc,Er,Ec)

```
HDC     hdc;
short   cxPage;
short   cyPage;
HPEN    *P_hPen;
HBRUSH  *p_hBrush;
float   P_Value;
short   Br;             /*begin row*/
short   Bc;             /*begin column*/
short   Er;             /*end row*/
short   Ec;             /*end column*/
```

หน้าที่ แสดงค่าข้อมูลให้อยู่ในรูปแบบของกราฟแท่งสะสม บน output device เช่น บนจอภาพ, บน printer , บน ploter หรือ output device อื่น ๆ

input 1.hdc

เป็น handle device context ของ output device ที่ต้องการแสดงรูปกราฟ ตย.เช่น handle device context อาจจะเป็น ของ จอภาพ หรือ Printer ก็ได้

2.cxPage

เป็นความกว้างของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

3.cyPage

เป็นความยาวของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

4.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

5.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

6.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

7.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

8.Bc

เป็นค่าของตำแหน่ง columnเริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

9.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

10.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value -

G_5_1 (hdc,cxPage,cyPage,P_hPen,p_hBrush,P_Value,Br,Bc,Er,Ec)

```
HDC     hdc;
short   cxPage;
short   cyPage;
HPEN    *P_hPen;
HBRUSH  *p_hBrush;
float   P_Value;
short   Br;             /*begin row*/
short   Bc;             /*begin column*/
short   Er;             /*end row*/
short   Ec;             /*end column*/
```

หน้าที่ แสดงค่าข้อมูลให้อยู่ในรูปแบบของกราฟเส้นบน output device เช่น บนจอภาพ, บน printer , บน ploter หรือ output device อื่น ๆ

input 1.hdc

เป็น handle device context ของ output device ที่ต้องการแสดงรูปกราฟ ตย.เช่น handle device context อาจจะเป็น ของ จอภาพ หรือ Printer ก็ได้

2.cxPage

เป็นความกว้างของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

3.cyPage

เป็นความยาวของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

4.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

5.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

6.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

7.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

8.Bc

เป็นค่าของตำแหน่ง columnเริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

9.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

10.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value -

G_5_1noline(hdc,cxPage,cyPage,P_hPen,p_hBrush,P_Value,Br,Bc,Er,Ec)

```
HDC       hdc;
short     cxPage;
short     cyPage;
HPEN      *P_hPen;
HBRUSH    *p_hBrush;
float     P_Value;
short     Br;              /*begin row*/
short     Bc;              /*begin column*/
short     Er;              /*end row*/
short     Ec;              /*end column*/
```

หน้าที่ แสดงค่าข้อมูลให้อยู่ในรูปแบบของกราฟจุดบน output device เช่น บนจอภาพ, บน printer , บน ploter หรือ output device อื่น ๆ

input 1.hdc

เป็น handle device context ของ output device ที่ต้องการแสดงรูปกราฟ ตย.เช่น handle device context อาจจะเป็น ของ จอภาพ หรือ Printer ก็ได้

2.cxPage

เป็นความกว้างของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

3.cyPage

เป็นความยาวของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

4.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

5.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

6.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

7.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

8.Bc

เป็นค่าของตำแหน่ง columnเริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

9.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

10.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value -

```
G_Pie1(hdc,cxPage,cyPage,P_hPen,p_hBrush,P_Value,Br,Bc,Er,Ec)
        HDC       hdc;
        short     cxPage;
        short     cyPage;
        HPEN      *P_hPen;
        HBRUSH    *p_hBrush;
        float     P_Value;
        short     Br;              /*begin row*/
        short     Bc;              /*begin column*/
        short     Er;              /*end row*/
        short     Ec;              /*end column*/
```

<u>หน้าที่</u> แสดงค่าข้อมูลให้อยู่ในรูปแบบของกราฟจุดบน output device เช่น บนจอภาพ, บน printer , บน ploter หรือ output device อื่น ๆ

<u>input</u> 1.hdc

เป็น handle device context ของ output device ที่ต้องการแสดงรูปกราฟ ตย.เช่น handle device context อาจจะเป็น ของ จอภาพ หรือ Printer ก็ได้

2.cxPage

เป็นความกว้างของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

3.cyPage

เป็นความยาวของเนื้อที่แสดงผลของ output device แต่ละชนิด โดยมีหน่วยเป็น pixel

4.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

5.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

6.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

7.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

8.Bc

เป็นค่าของตำแหน่ง column เริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

9.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

10.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value -

```
PrintGraph(hwnd,P_hPen,p_hBrush,whichgraph,P_value,Br,Bc,Er,Ec)
        HDC     hwnd;
        HPEN    *P_hPen;
        HBRUSH  *p_hBrush;
        short   whichgraph;
        float   P_Value;
        short   Br;             /*begin row*/
        short   Bc;             /*begin column*/
        short   Er;             /*end row*/
        short   Ec;             /*end column*/
```

<u>หน้าที่</u> พิมพ์รูปกราฟออกทาง Printer

โดยสามารถเลือกรูปแบบกราฟที่จะพิมพ์ออกได้ดังนี้

1.กราฟแท่ง

2.กราฟแท่งสะสม

3.กราฟเส้น

4.กราฟจุด

5.กราฟวงกลม

และมี option อื่นๆ ให้ user เลือกรูปแบบในการพิมพ์อีกดังนี้

1.เลือกพิมพ์แบบ lanscape หรือ potrait

2.เลือกขนาดกระดาษ

ฯลฯ

<u>input</u> 1.hwnd

เป็น handle ของ window

2.P_hPen

เป็น pointer ที่ชี้ไปยัง Pen handle ตัวแรก ของ Pen ที่ใช้ในการวาดรูปกราฟทั้งหมด โดย handle ของ Pen ทั้งหมดจะเก็บเป็น array ไว้

3.P_hBrush

เป็น pointer ที่ชี้ไปยัง Brush handle ตัวแรก ของ Brush ที่ใช้ใน การระบายรูปกราฟทั้งหมด โดย handle ของ HBrush ทั้งหมดจะเก็บเป็น array ไว้

4.whichgraph

เป็นค่า ID ที่ระบุถึงชนิดของกราฟที่จะพิมพ์

ค่า ID ต่างๆ มีดังนี้

BUTTON_THREE	หมายถึง กราฟแท่ง
BUTTON_FOUR	หมายถึง กราฟแท่งแบบสะสม
BUTTON_FIVE	หมายถึง กราฟเส้น
BUTTON_SIX	หมายถึง กราฟวงกลม
BUTTON_SEVEN	หมายถึง กราฟจุด

และในกรณีที่ input ที่ส่งมาเป็น

BUTTON_THREE	ฟังก์ชันนี้ก็จะทำการเรียกฟังก์ชัน G_BAR
BUTTON_FOUR	ฟังก์ชันนี้ก็จะทำการเรียกฟังก์ชัน G_BAR_PLUS
BUTTON_FIVE	ฟังก์ชันนี้ก็จะทำการเรียกฟังก์ชัน G_5_1
BUTTON_SIX	ฟังก์ชันนี้ก็จะทำการเรียกฟังก์ชัน G_Pie1
BUTTON_SEVEN	ฟังก์ชันนี้ก็จะทำการเรียกฟังก์ชัน G_5_1noline

5.P_Value

เป็น pointer ที่ชี้ไปยังค่าของข้อมูลตัวแรกของข้อมูลที่จะนำมา plot graph (ซึ่งข้อมูลเฉพาะที่จะนำมา plot graph จะเก็บเป็น array ของข้อมูลแบบ float ไว้)

6.Br

เป็นค่าของตำแหน่ง row เริ่มต้นของข้อมูลบน worksheetที่จะนำมา Plot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

7.Bc

เป็นค่าของตำแหน่ง column เริ่มต้นของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

8.Er

เป็นค่าของตำแหน่ง row สุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง row บน worksheet

A,B,C,D,... = 0,1,2,3,...

9.Ec

เป็นค่าของตำแหน่ง columnสุดท้ายของข้อมูลบน worksheetที่จะนำมาPlot graph

โดยให้ตำแหน่ง column บน worksheet

1,2,3,4,... = 0,1,2,3,...

return value ถ้า พิมพ์สำเร็จก็จะ return ค่า TRUE

ถ้า ไม่สำเร็จก็จะ return ค่า FALSE

---

printSheet(hwnd)

HWND hwnd;

หน้าที่ พิมพ์ส่วน sheet ออกทาง Printer

option ของการพิมพ์มีดังนี้

1.สามารถพิมพ์ได้ไม่จำกัดจำนวน row หรือ column

2.สามารถเลือกจำนวน copies ที่จะพิมพ์ได้

3.สามารถเลือกพิมพ์ในลักษณะ lanscape หรือ portait

4.สามารถเลือกความกว้างของกระดาษที่จะพิมพ์

input hwnd

เป็นค่า handle ของ window ที่เรียกฟังก์ชันนี้

return value ถ้า พิมพ์สำเร็จก็จะ return ค่า TRUE

ถ้า ไม่สำเร็จก็จะ return ค่า FALSE

สรุป

การทำงานในโครงงานนี้ก็เป็นไปตามจุดประสงค์ที่ได้ตั้งไว้ แต่ก็จะมีปัญหาบ้างดังต่อไปนี้

1. เนื่องจากโครงงานนี้เป็นการเขียนโปรแกรมที่มีขนาดใหญ่ ดังนั้นจึงต้องมีความพร้อมในหลายปัจจัย อันได้แก่
   - เนื้อที่ harddisk สำหรับ install โปรแกรมในการทำงานอย่างน้อย 60 เมกกะไบต์
   - หน่วยความจำสำหรับขั้นตอนการทำงานของโปรแกรมอย่างน้อย 8 เมกกะไบต์
2. เนื่องจากโครงงานนี้ได้แบ่งโปรแกรมที่เขียนขึ้นออกเป็น 2 ส่วนใหญ่ ซึ่งขั้นตอนในการเชื่อมโปรแกรมทั้ง 2 ส่วนนี้เข้าด้วยกันจะยุ่งยากและมีปัญหามาก ดังนั้นหากผู้ที่สนใจในการเขียนโปรแกรมใหญ่ๆ บน Windows และทำงานกันหลายคนควรที่จะคำนึงถึงปัญหานี้ด้วย

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ ดร.บุญธี เครือตราชู
อาจารย์ กฤตวัน เครือตราชู
อาจารย์ อภิเนตร อุนากูล
ที่ให้คำปรึกษาและให้ความช่วยเหลือในการทำโครงงานครั้งนี้

เอกสารอ้างอิง

1. วีรศักดิ์ กาญจนรจิต,"Windows Application",วรสาร Computer Review,ฉบับที่ 81,2534, หน้า 211-216.
2. วีรศักดิ์ กาญจนรจิต,"Windows ApplicationII",วรสาร Computer Review, ฉบับที่ 82,2534, หน้า 189-197.
3. วีรศักดิ์ กาญจนรจิต,"Windows ApplicationIII",วรสาร Computer Review, ฉบับที่ 83,2534, หน้า 254-258.
4. Charles Petzold,"Programming Windows 3.1",Microsoft Press, Washington,1992.
5. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 1, Microsoft Press,Washington,1992.
6. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 2, Microsoft Press,Washington,1992.
7. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 3, Microsoft Press,Washington,1992.
8. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 4, Microsoft Press,Washington,1992.
9. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 5, Microsoft Press,Washington,1992.
10. Microsoft Corporation,Windows 3.1 Progammer's Reference Volume 6, Microsoft Press,Washington,1992.