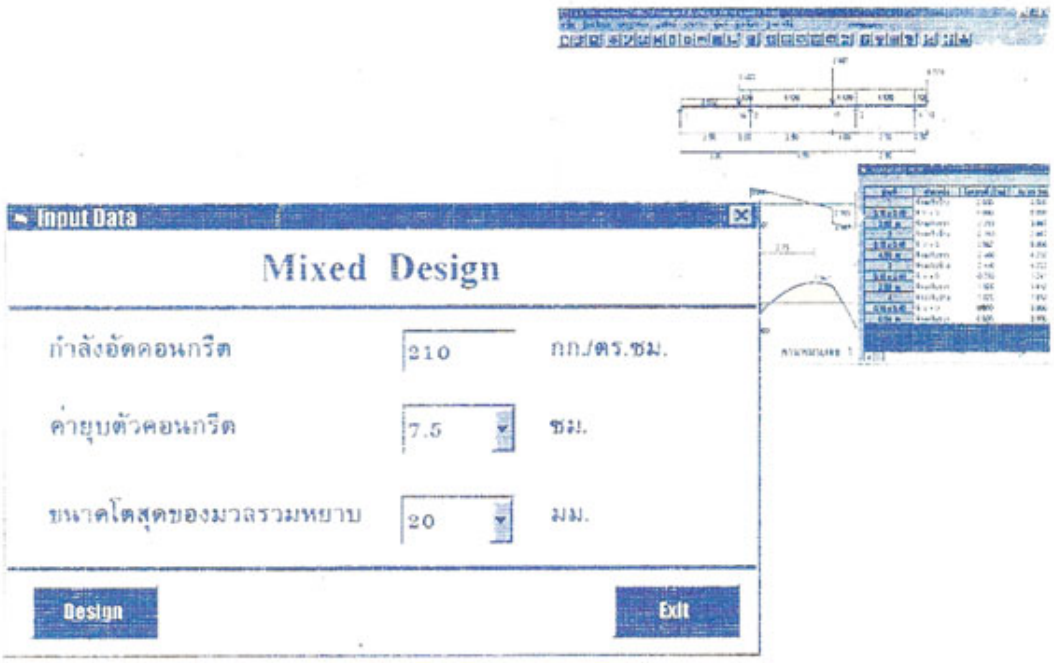
วิชวลเบสิก
ในงานวิศวกรรมโยธา

Visual Basic in Civil Engineering Work

ISBN 974-623-139-1

สรกานต์ ศรีตองอ่อน



วิทยาลัยเทคโนโลยีอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนำ

หนังสือเล่มนี้เรียบเรียงขึ้นด้วยวัตถุประสงค์หลักสองประการคือ ประการแรกใช้ประกอบการสอนวิชา 372354 Computer Application for Civil Engineering II ซึ่งเป็นรายวิชาที่อยู่ในหลักสูตรระดับปริญญาตรี สาขาวิชาเทคโนโลยีโยธา โครงการภาควิชาเทคโนโลยีโยธา วิทยาลัยเทคโนโลยีอุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ และประการที่สองสำหรับนักศึกษาทางด้านวิศวกรรมโยธาและผู้สนใจทั่วไป ที่สนใจเกี่ยวกับการเขียนโปรแกรมคอมพิวเตอร์ด้วยไมโครซอฟต์วิชวลเบสิก (Microsoft Visual Basic) โดยเน้นที่การประยุกต์ใช้ในงานทางด้านวิศวกรรมโยธา ในระดับเบื้องต้น ซึ่งเนื้อหาแบ่งออกเป็น 2 ภาค คือ

ภาคที่ 1 ว่าด้วยเรื่องความรู้พื้นฐานต่างๆ ของวิชวลเบสิก โดยนำมาเท่าที่เห็นว่าจำเป็นในการเขียนโปรแกรมระดับเบื้องต้น ดังนั้น รายละเอียดปลีกย่อยหลายอย่างจึงตั้งใจที่จะทอนลง ซึ่งผู้อ่านสามารถศึกษาเพิ่มเติมในระดับสูงได้จากหนังสือเกี่ยวกับวิชวลเบสิกทั่วไป

ภาคที่ 2 ว่าด้วยการปฏิบัติการสร้างโปรแกรม คือ ยกตัวอย่างงานทางด้านวิศวกรรมโยธามาอธิบายถึงขั้นตอนการสร้างโปรแกรมตั้งแต่เริ่มต้นออกแบบ จนกระทั่งแล้วเสร็จ อันเป็นพื้นฐานที่ผู้อ่านสามารถนำไปประยุกต์ใช้ในงานทางด้านวิศวกรรมอื่นๆ ได้ต่อไป

ผู้อ่านควรศึกษาเนื้อหาในภาคที่ 1 ให้เข้าใจภาพรวมก่อน แล้วจึงฝึกทำตามขั้นตอนที่ให้ไว้ในภาคที่ 2 อันเป็นการนำความรู้ในภาคที่ 1 มาประมวลเข้าสู่ภาคปฏิบัติ ซึ่งเป็นแนวทางที่ผู้เขียนได้ใช้มาแล้วได้ผลดีในระดับหนึ่ง ต่อจากนั้นจึงเป็นเรื่องของการฝึกฝนงานอื่นๆ ให้มาก เพื่อเพิ่มความชำนาญให้ยิ่งๆขึ้นต่อไป

สรกานต์ ศรีตองอ่อน
skng@kmitnb.ac.th
ตุลาคม 2544

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญแผนภูมิ

ภาคที่ 1
ความรู้พื้นฐาน

บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับไมโครซอฟต์วิชวลเบสิก

การเขียนโปรแกรมคอมพิวเตอร์

การเขียนโปรแกรมคอมพิวเตอร์ คือ การสั่งให้คอมพิวเตอร์ทำงานตามวัตถุประสงค์ต่างๆ ที่ต้องการ ซึ่งโปรแกรมคอมพิวเตอร์หรือที่ในปัจจุบันนิยมเรียกว่า ซอฟต์แวร์ หรือ แอพพลิเคชัน (application software) ที่มีจำหน่ายทั่วไป ก็เกิดจากการเขียนขึ้นมาด้วยภาษาโปรแกรมต่างๆ แต่เนื่องจากซอฟต์แวร์หลายโปรแกรม มักมีราคาแพง เพราะมีคุณลักษณะที่มากมาย ซึ่งเราไม่จำเป็นต้องใช้งานมากขนาดนั้น อีกทั้งบางทีก็ทำงานได้ไม่ตรงกับความต้องการ เราจึงควรที่จะเรียนรู้การเขียนโปรแกรมไว้เป็นพื้นฐานบ้าง อย่างน้อยก็เพื่อเขียนโปรแกรมขึ้นไว้ใช้งานเอง และหากชำนาญมากขึ้น ก็สามารถทำเป็นอาชีพหลักหรืออาชีพเสริมได้ ซึ่งสามารถสรุปข้อดีของการเขียนโปรแกรมขึ้นเองได้ดังนี้

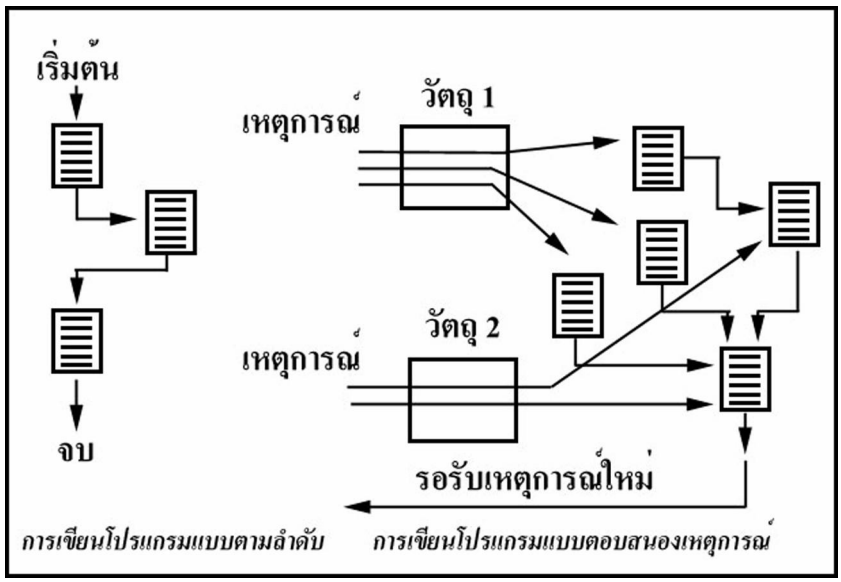
1. ทำให้เราสร้างโปรแกรมที่ตรงกับความต้องการได้
2. ฝึกการแก้ไขปัญหาต่างๆ
3. ทำให้เข้าใจการทำงานของคอมพิวเตอร์มากยิ่งขึ้น
4. สามารถใช้ประกอบอาชีพได้

ภาษาที่ใช้ในการเขียนโปรแกรม

ภาษาคอมพิวเตอร์ที่ใช้เขียนโปรแกรม ในปัจจุบันอาจแบ่งออกเป็น 2 แบบใหญ่ คือ

1. ใช้ในการเขียนโปรแกรมแบบตามลำดับขั้นตอน (procedural programming) เช่น Pascal, C, Fortran เป็นต้น
2. ใช้ในการเขียนโปรแกรมแบบตอบสนองเหตุการณ์ (event-driven) เช่น ไมโครซอฟต์วิชวลเบสิก (Microsoft Visual Basic), เดลไฟ (Delphi) เป็นต้น

ความแตกต่างของการเขียนโปรแกรมทั้ง 2 แบบนี้ แสดงดังรูปที่ 1-1



รูปที่ 1-1 ความแตกต่างของการเขียนโปรแกรม

ในที่นี้จะใช้ไมโครซอฟต์วิชวลเบสิก 6.0 (Microsoft Visual Basic 6.0) หรือเรียกสั้นๆ ว่า วิชวลเบสิก ในการเขียนโปรแกรมสำหรับทำงานบนระบบปฏิบัติการวินโดวส์

หลักการโปรแกรมเชิงภาพของวิชวลเบสิก

การเขียนโปรแกรมด้วยวิชวลเบสิก ใช้หลักการที่เรียกว่า การโปรแกรมเชิงภาพ (visual programming) คือการใช้หลักของภาพและการมองเห็น โดยเริ่มจาก

1. ออกแบบวินโดว์ย่อยที่เรียกว่า ฟอร์ม (form) ซึ่งในแต่ละฟอร์มสามารถเลือกนำส่วนประกอบต่างๆ ที่จำเป็นต่อการใช้งานมาวางไว้ เช่น ช่องรับข้อความ หรือ ปุ่มคำสั่ง เป็นต้น ส่วนประกอบต่างๆ นี้ เรียกโดยรวมว่า คอนโทรล (control) ซึ่งแต่ละคอนโทรลก็จะมีคุณสมบัติ (property) ต่างๆ ที่จำเป็นสำหรับโปรแกรม ขั้นตอนนี้รวมเรียกว่า **ขั้นตอนการออกแบบส่วนติดต่อกับผู้ใช้** (user interface)
2. เมื่อกำหนดคอนโทรลต่างๆ เรียบร้อยแล้ว จึงระบุว่าคอนโทรลแต่ละอย่างจะทำงานอย่างไร หรือจะทำงานเพื่อตอบสนองเหตุการณ์อะไร โดยเขียนโปรแกรมย่อยๆ เข้าไปในคอนโทรลต่างๆ ที่กำหนดขึ้น โปรแกรมย่อยๆ นี้ อาจเป็นวิธีการ (method) ของคอนโทรลเอง หรือเป็นคำสั่งที่เราเขียนขึ้นโดยใช้ภาษาเบสิก (BASIC) ขั้นตอนนี้รวมเรียกว่า **ขั้นตอนการเขียนคำสั่งโปรแกรมเพื่อให้ตอบสนองต่อเหตุการณ์ที่เกิดขึ้น**

ประวัติความเป็นมาของวิชวลเบสิก

วิชวลเบสิก เป็นภาษาคอมพิวเตอร์แบบหนึ่ง ที่พัฒนาโดยบริษัทไมโครซอฟต์ (Microsoft) ซึ่งเป็นบริษัทที่สร้างระบบปฏิบัติการวินโดวส์ ที่ใช้กันอย่างแพร่หลายในปัจจุบัน สำหรับคำสั่งที่ใช้ในวิชวลเบสิก มีรากฐานมาจากภาษา BASIC ซึ่งย่อมาจาก **B**eginner's **A**ll Purpose **S**ymbolic Instruction **C**ode แปลได้ว่า "รหัสคำสั่งสำหรับผู้เริ่มต้น" อันเป็นภาษาที่บริษัทไมโครซอฟต์เชี่ยวชาญเป็นอย่างดี เพราะพัฒนามาตั้งแต่ระบบปฏิบัติการ DOS

วิชวลเบสิก เวอร์ชัน (version) 1.0 เริ่มออกในปีพ.ศ. 2534 จนถึงล่าสุดคือ เวอร์ชัน 6.0 ออกในปีพ.ศ. 2541 มีรายละเอียดโดยสังเขปของแต่ละเวอร์ชันดังนี้

เวอร์ชัน 1.0 เริ่มต้นแนวทางการเขียนโปรแกรมแบบตอบสนองเหตุการณ์ โดยใช้หลักการโปรแกรมเชิงภาพ

เวอร์ชัน 2.0 ปรับปรุงความรวดเร็วในการทำงาน

เวอร์ชัน 3.0 เพิ่มเติมเครื่องมือ (tool) ต่างๆ ที่ช่วยในการเขียนโปรแกรม ทำให้การเชื่อมต่อกับระบบฐานข้อมูลดีขึ้น

เวอร์ชัน 4.0 พัฒนาการเขียนโปรแกรมให้เต็มสมรรถนะของชิปแบบ 32 บิต

เวอร์ชัน 5.0 ปรับปรุงคุณลักษณะหลายอย่างให้ดีขึ้น ทำให้มีความแตกต่างจากเวอร์ชัน 4.0 พอสมควร

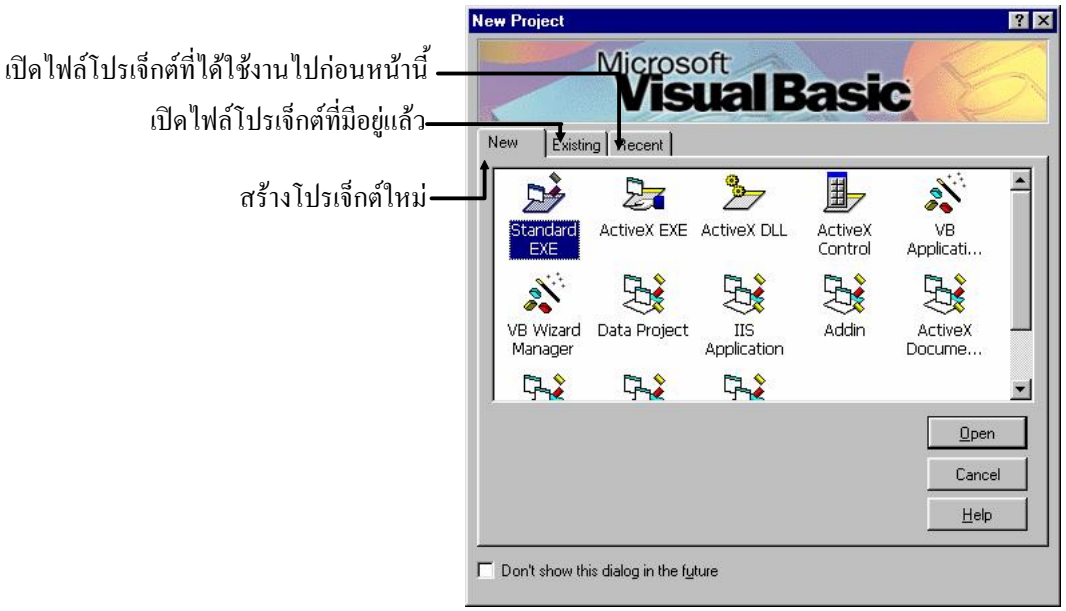
เวอร์ชัน 6.0 ปรับปรุงความสามารถในการเขียนโปรแกรมติดต่อกับเครือข่ายอินเตอร์เน็ต การเชื่อมต่อกับระบบฐานข้อมูล การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) และเพิ่มเติมคำสั่งต่างๆ

สำหรับวิชวลเบสิก เวอร์ชัน 6.0 แบ่งเป็น 3 Edition คือ Learning, Professional และ Enterprise ในคู่มือเล่มนี้ จะใช้ Professional Edition

บทที่ 2 การเขียนโปรแกรมคอมพิวเตอร์ด้วยวิชวลเบสิก

สภาพแวดล้อมของวิชวลเบสิก

เมื่อเข้าสู่วิชวลเบสิก หน้าต่างแรกที่พบคือ New Project ดังรูป ซึ่งจะเป็นการเลือกเขียนโปรแกรม โดยวิชวลเบสิกเรียกว่า โปรเจ็กต์ (project) ใน 3 รูปแบบ ดังรูปที่ 2-1 คือ



รูปที่ 2-1 การเลือกทำงานกับโปรเจ็กต์

มีรายละเอียดคือ

New เป็นการสร้างโปรเจ็กต์ใหม่ ที่มีการใช้งานให้เลือกตามไอคอน (icon) ที่ปรากฏ แต่ละไอคอนมีรูปแบบการใช้งานที่แตกต่างกัน อธิบายได้ดังตารางที่ 2-1

ตารางที่ 2-1 รูปแบบการสร้างโปรเจ็กต์ใหม่

ชื่อเรียก	รูปแบบการใช้งาน
Standard EXE	ใช้สร้างโปรแกรมทั่วไป
ActiveX EXE	ใช้สร้างโปรแกรมที่ใช้ติดต่อกับโปรแกรมอื่นในรูปแบบของ OLE Automation Server
ActiveX DLL	ใช้สร้างโปรแกรมเช่นเดียวกับ ActiveX EXE แต่จะเก็บอยู่ในไฟล์นามสกุล DLL แทน ซึ่งไม่สามารถรันได้ด้วยตนเอง จะต้องถูกเรียกใช้โดยโปรแกรมอื่น

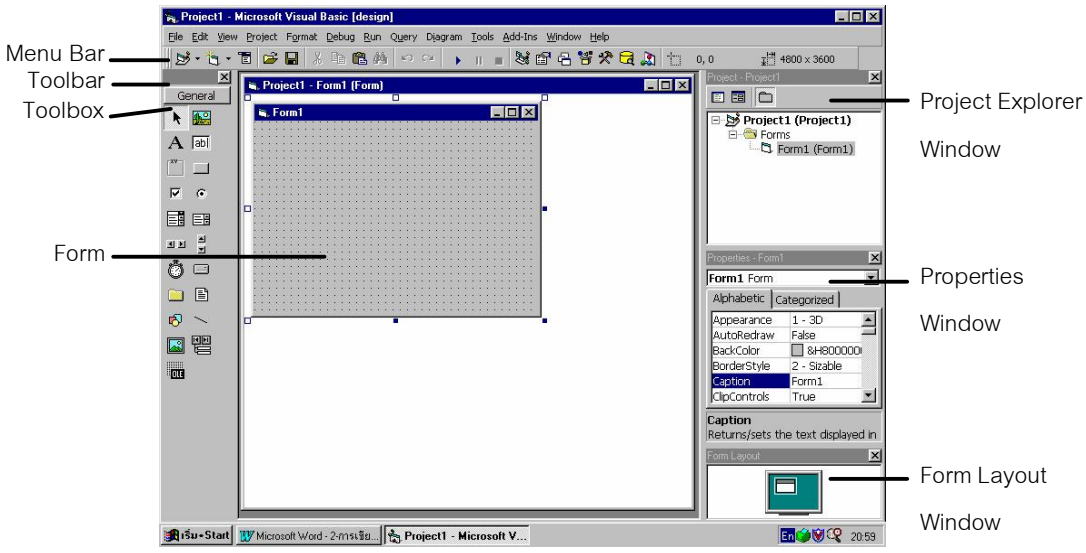
ตารางที่ 2-1 (ต่อ)

ชื่อเรียก	รูปแบบการใช้งาน
ActiveX Control	ใช้สร้าง ActiveX Control ขึ้นใช้งาน
VB Application Wizard	เป็นเครื่องมือที่ช่วยสร้างโปรแกรมขึ้นใช้งาน
Addin	ใช้เพิ่มเติม Utility อื่นเข้าไว้ในวิชวลเบสิก
ActiveX Document DLL	ใช้สร้างโปรแกรม ActiveX ที่อยู่ในรูปของไฟล์นามสกุล DLL
ActiveX Document EXE	ใช้สร้างโปรแกรม ActiveX ที่อยู่ในรูปของไฟล์ นามสกุล EXE
VB Wizard Manager	ใช้สร้างโปรแกรมที่ควบคุมการทำงานของ Wizard
Data Project	ใช้สร้างโปรแกรมที่ใช้ติดต่อกับฐานข้อมูลต่าง ๆ ผ่านทาง ODBC หรือ OLEDB
IIS Application	ใช้สร้างโปรแกรมที่ใช้งานบน Internet แบบ IIS
DHTML Application	ใช้สร้างโปรแกรมที่ใช้งานบน Internet แบบ Dynamic HTML
VB Pro Edition Control	ใช้สร้างโปรแกรมทั่วไป แต่มีคอนโทรลให้เลือกใช้มากขึ้น ทำให้ได้โปรแกรมที่ทำงานอย่างมีประสิทธิภาพมากขึ้น

Existing เป็นการเปิดไฟล์โปรเจ็กต์เดิมที่สร้างไว้แล้ว

Recent เป็นการเปิดไฟล์โปรเจ็กต์เดิมที่เพิ่งเปิดก่อนหน้า

สำหรับในการเขียนโปรแกรมเบื้องต้นนี้ เมื่อสร้างโปรเจ็กต์ใหม่ จะใช้การใช้งานรูปแบบ Standard EXE ซึ่งเมื่อคลิกเข้าไปแล้ว จะเกิดหน้าจอดังรูปที่ 2-2



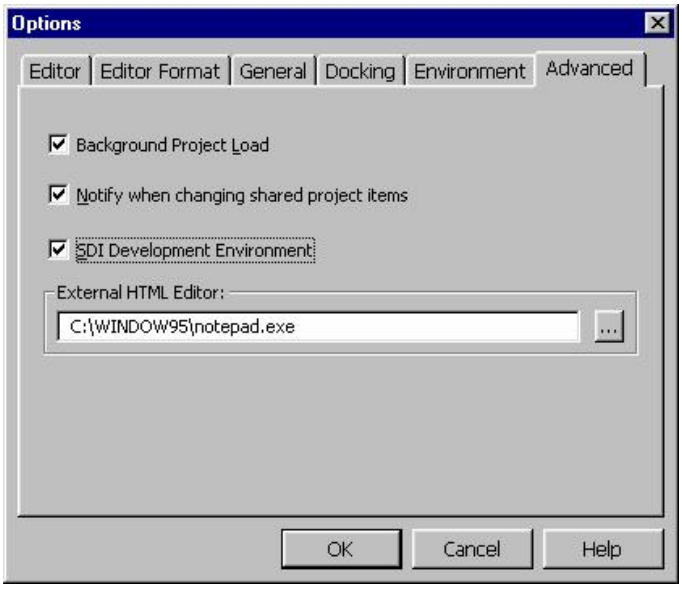
รูปที่ 2-2 หน้าจอแบบ Standard EXE

ส่วนประกอบในแต่ละส่วน มีรายละเอียดดังตารางที่ 2-2

ตารางที่ 2-2 รายละเอียดส่วนประกอบของหน้าจอแบบ Standard EXE

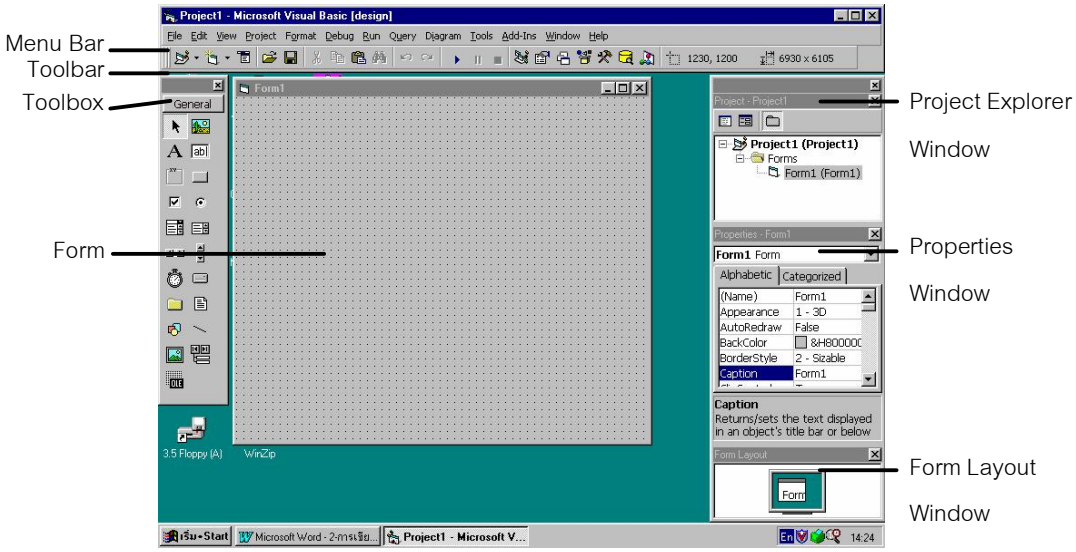
ส่วนประกอบ	รายละเอียด
Menu Bar	แถบเมนูในการเลือกทำงานต่าง ๆ
Toolbar	การเลือกทำงานโดยใช้ไอคอนแทนการไปที่แถบเมนู
Toolbox	ที่เก็บคอนโทรล (control) ต่าง ๆ
Form	ส่วนที่ใช้สร้างหน้าจอของโปรแกรมที่เขียนขึ้น
Project Explorer Window	ช่องหน้าต่างสำหรับแสดงจำนวนและชื่อของฟอร์ม (form) และโมดูล (module) ต่าง ๆ
Properties Window	ช่องหน้าต่างสำหรับแสดงรายละเอียดของคุณสมบัติ (property) ของฟอร์มและคอนโทรลต่าง ๆ
Form Layout Window	ช่องหน้าต่างสำหรับดูตำแหน่งของฟอร์มบนจอภาพ ซึ่งสามารถปรับตำแหน่งได้

สำหรับหน้าจอของวิชวลเบสิกที่เห็นนั้น เรียกว่า อยู่ในสภาพแวดล้อมแบบ MDI (Multiple Document Interface) เราสามารถเปลี่ยนให้เป็นแบบ SDI (Single Document Interface) โดยไปที่เมนู Tools / Options ซึ่งจะปรากฏหน้าต่างการแก้ไขสภาพแวดล้อมต่างๆ ของหน้าจอ ในที่นี้คือไปที่ Tab Advanced คลิกเลือกที่ SDI Development Environment ดังรูปที่ 2-3



รูปที่ 2-3 เลือกหน้าจอแบบ SDI

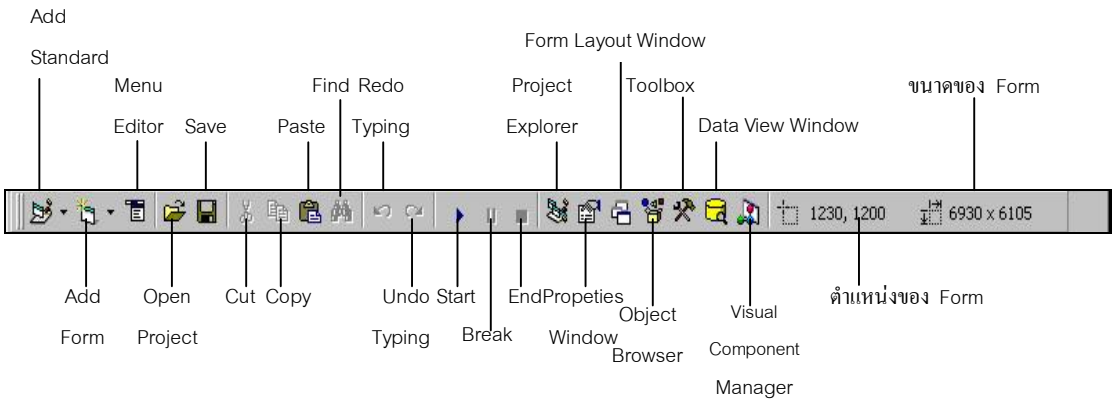
ซึ่งจะทำให้หน้าจอของวิชวลเบสิกเปลี่ยนไปดังรูปที่ 2-4



รูปที่ 2-4 หน้าจอแบบ SDI

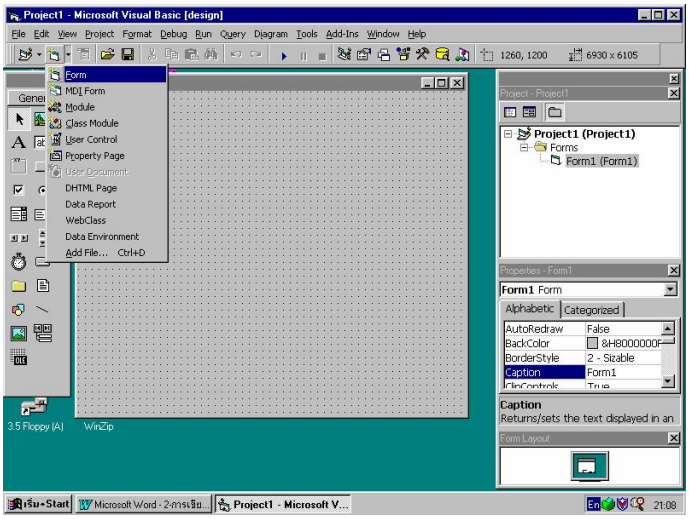
ในความเห็นของผู้เขียน หน้าจอแบบ SDI นี้ จะเขียนโปรแกรมได้คล่องตัวกว่าหน้าจอแบบ MDI จึงแนะนำให้ใช้หน้าจอแบบนี้

สำหรับรายละเอียดของแต่ละไอคอนใน Toolbar แสดงดังรูปที่ 2-5



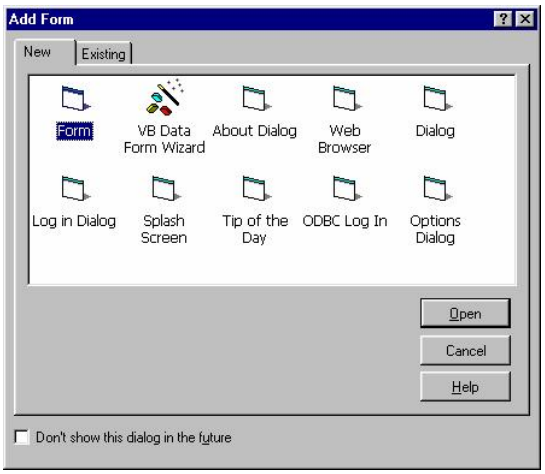
รูปที่ 2-5 รายละเอียดของแต่ละไอคอน

ในการเขียนโปรแกรมนั้น บ่อยครั้งที่จะต้องมีการเพิ่มฟอร์ม และโมดูลเข้าไปในโปรเจ็กต์ สามารถทำได้โดยคลิกที่ไอคอน Add Form ดังรูปที่ 2-6



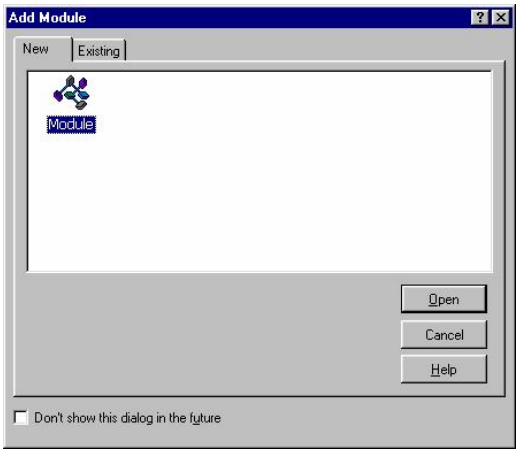
รูปที่ 2-6 เพิ่มฟอร์มโดยคลิกที่ไอคอน Add Form

จะเห็นว่าในเมนูย่อยนั้น มีการ Add หลายรูปแบบ แต่ที่ใช้ในเบื้องต้นคือ ฟอร์มและโมดูล ถ้าจะเพิ่มฟอร์มก็คลิกที่เมนูย่อยฟอร์ม ซึ่งจะแสดงหน้าต่าง Add Form ดังรูปที่ 2-7



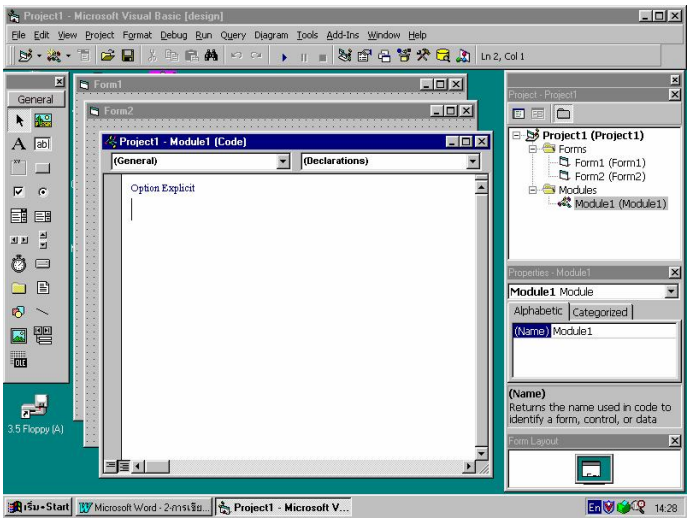
รูปที่ 2-7 หน้าต่าง Add Form

ถ้าจะเพิ่มโมดูลก็คลิกที่เมนูย่อยโมดูล ซึ่งจะแสดงหน้าต่าง Add Module ดังรูปที่ 2-8



รูปที่ 2-8 หน้าต่าง Add Module

เมื่อมีการเพิ่มฟอร์มและโมดูลเข้าไปในโปรเจ็กต์ ช่อง Project Explorer Window จะแสดงจำนวนและชื่อของฟอร์ม และโมดูล ตัวอย่างดังรูปที่ 2-9



รูปที่ 2-9 รายละเอียดในช่อง Project Explorer Window

<u>การจัดการไฟล์ของโปรแกรม</u>

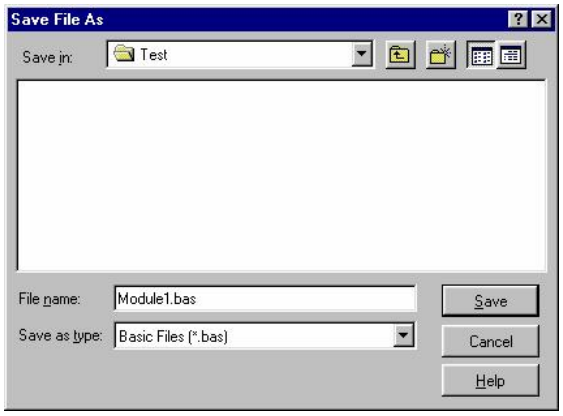
สำหรับการเขียนโปรแกรมด้วยวิชวลเบสิก ในเบื้องต้นนี้ โปรแกรมหนึ่งจะประกอบด้วย 3 ไฟล์หลักคือ

1. ไฟล์โปรเจ็กต์ นามสกุล .vbp คือไฟล์ที่เป็นชื่อของโปรแกรมนั้นๆ อีกทั้งมีหน้าที่ในการเป็นเสมือนผู้จัดการไฟล์ฟอร์ม และไฟล์โมดูลต่างๆ

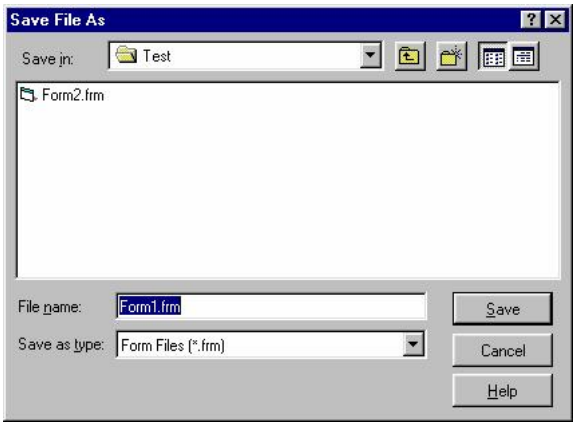
2. ไฟล์ฟอร์ม นามสกุล .frm คือไฟล์ของแต่ละฟอร์มที่ประกอบขึ้นเป็นโปรแกรม
3. ไฟล์โมดูล นามสกุล .bas คือไฟล์ที่เก็บตัวแปรส่วนกลาง (global variables) และเก็บโปรแกรมย่อยที่ใช้ได้กับทุกฟอร์ม (public procedure)

นอกจากนี้ ยังมีไฟล์อีกรูปแบบหนึ่ง เรียกว่า ไบนารีฟอร์ม (binary form) นามสกุล .frx ซึ่งวิชวลเบสิกจะบันทึกไฟล์นี้โดยอัตโนมัติให้เป็นชื่อเดียวกับ ไฟล์ฟอร์ม ที่มีการแทรกรูปภาพ หรือไอคอน ลงไปในฟอร์มนั้น

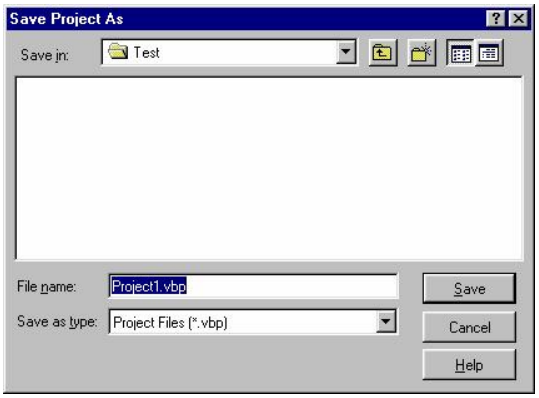
ในทางปฏิบัติ เมื่อมีการสร้างโปรเจ็กต์ใหม่ การบันทึกไฟล์ครั้งแรก วิชวลเบสิกจะแสดงหน้าต่างบันทึกไฟล์ในแต่ละนามสกุล ตัวอย่างดังรูปที่ 2-10 ถึง 2-12



รูปที่ 2-10 หน้าต่างบันทึกไฟล์โมดูล

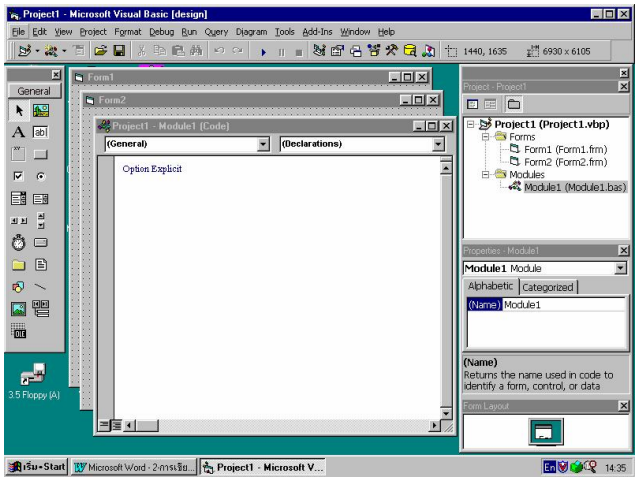


รูปที่ 2-11 หน้าต่างบันทึกไฟล์ฟอร์ม



รูปที่ 2-12 หน้าต่างบันทึกไฟล์โปรเจ็กต์

เมื่อบันทึกไฟล์ครั้งแรกเรียบร้อยแล้ว ช่อง Project Explorer Window จะแสดงชื่อไฟล์ และนามสกุลที่บันทึกไว้ ตัวอย่างดังรูปที่ 2-13

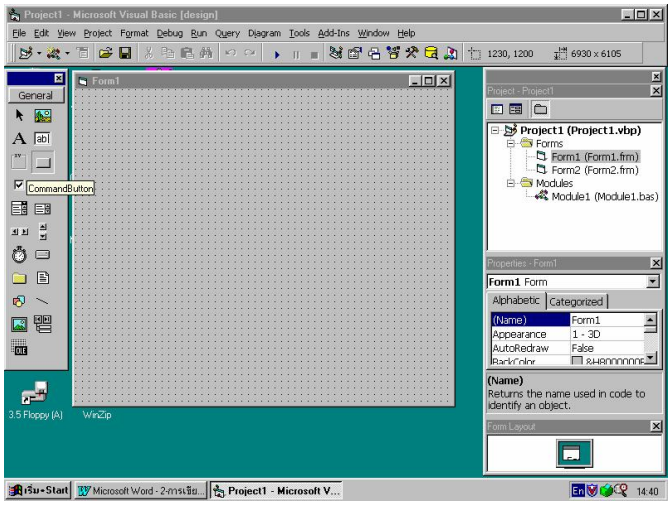


รูปที่ 2-13 ช่อง Project Explorer Window แสดงชื่อไฟล์ และนามสกุลที่บันทึกไว้

หลังจากนั้น หากไม่มีการเพิ่มฟอร์มหรือโมดูลใหม่ การบันทึกไฟล์ครั้งต่อไป วิชวลเบสิกจะบันทึกทุกไฟล์ที่มีอยู่แล้วโดยอัตโนมัติ และหากมีการเพิ่มฟอร์มหรือโมดูลใหม่เข้าไป ก็จะแสดงหน้าต่างให้บันทึกไฟล์นั้นทุกครั้งไป

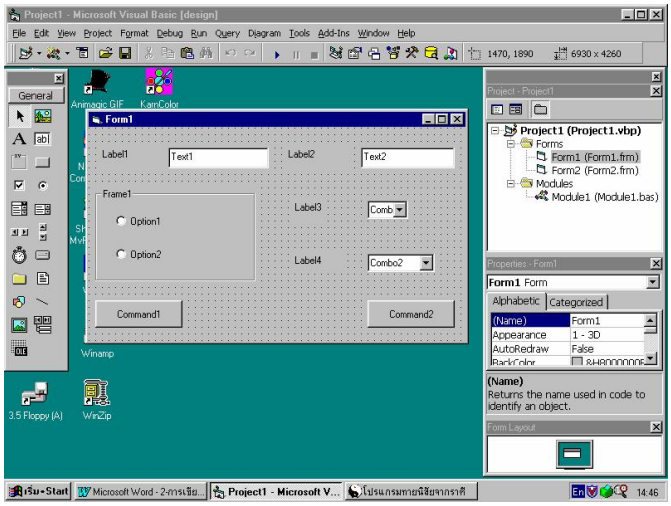
ขั้นตอนหลักในการเขียนโปรแกรม

1. ขั้นตอนออกแบบส่วนติดต่อกับผู้ใช้ คือการใส่คอนโทรลที่เหมาะสมลงไปในฟอร์ม โดยการเลือกใน Toolbox ตัวอย่างดังรูปที่ 2-14



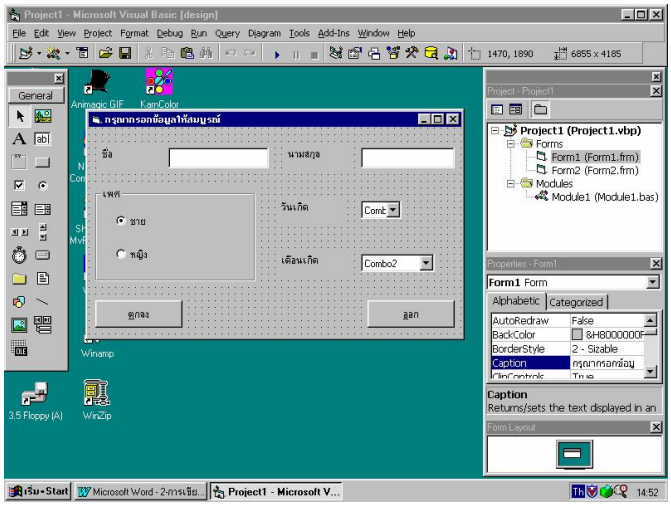
รูปที่ 2-14 เลือกคอนโทรลใน Toolbox

2. ผสมผสานคอนโทรลต่างๆ ให้เป็นไปตามที่ออกแบบไว้ ตัวอย่างดังรูปที่ 2-15



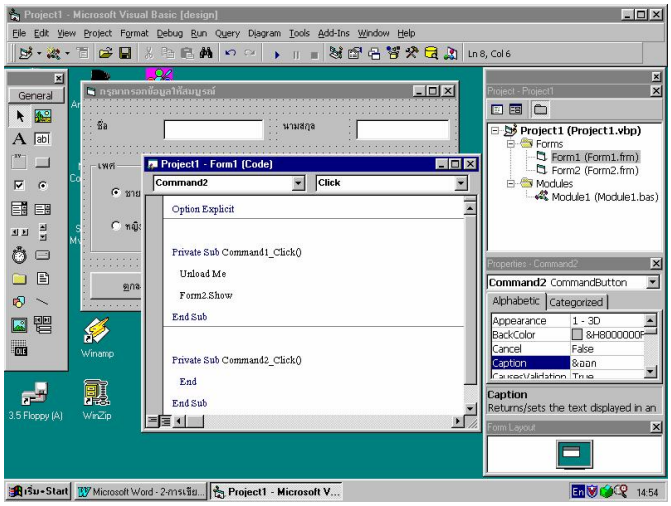
รูปที่ 2-15 ใส่คอนโทรลตามที่ออกแบบไว้ลงในฟอร์ม

3. กำหนดคุณสมบัติของฟอร์ม และคอนโทรลที่ใส่ลงไป ซึ่งคุณสมบัติบางอย่างคือสิ่งที่ผู้ใช้มองเห็น เช่น ข้อความต่างๆ อีกส่วนหนึ่งกำหนดไว้สำหรับใช้ในขั้นตอนการเขียนคำสั่ง ตัวอย่างดังรูปที่ 2-16



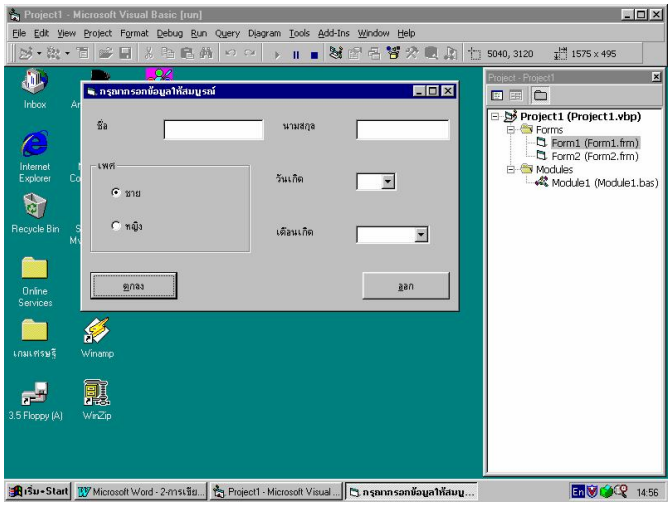
รูปที่ 2-16 กำหนดคุณสมบัติของฟอร์ม และคอนโทรลต่างๆ

4. ต่อมาเป็นขั้นตอนการเขียนโปรแกรมเพื่อตอบสนองเหตุการณ์ต่างๆ ตามที่เรากำหนดให้ผู้ใช้สามารถทำเหตุการณ์นั้นได้ เช่น คลิกปุ่มคำสั่ง เป็นต้น ลงในหน้าต่างโปรแกรมย่อย Code ตัวอย่างดังรูปที่ 2-17



รูปที่ 2-17 เขียนโปรแกรมที่หน้าต่างโปรแกรมย่อย Code

5. ในขณะเขียนโปรแกรม หรือเขียนโปรแกรมเรียบร้อยทั้งหมดแล้ว สามารถทำการรันโปรแกรมเพื่อดูการทำงานว่าเป็นไปตามที่ได้เขียนคำสั่งไว้หรือไม่ ในสภาพแวดล้อมของวิชวลเบสิกนั้นได้ ตัวอย่างดังรูปที่ 2-18



รูปที่ 2-18 ตัวอย่างการรันโปรแกรม

6. หลังจากที่เขียนโปรแกรมเสร็จแล้ว จึงทำให้เป็นโปรแกรมสำเร็จพร้อมที่จะแจกจ่าย หรือจำหน่ายต่อไป

<u>สรุปความหมายของคำเบื้องต้นที่ควรทราบ</u>

วัตถุ (object)

เป็นคำที่มีที่มาจากการเขียนโปรแกรมเชิงวัตถุ ซึ่งในวิชวลเบสิกจะหมายถึง ฟอร์ม และคอนโทรลต่างๆ รวมไปถึงการทำงานบางอย่างเช่น การทำงานกับเครื่องพิมพ์ (Printer) โดยในแต่ละวัตถุ จะประกอบด้วย คุณสมบัติประจำตัวของแต่ละวัตถุ และวิธีการที่วัตถุนั้นสามารถทำได้

ฟอร์ม (form)

วินโดว์หรือหน้าจอของโปรแกรม โดยในแต่ละโปรแกรมสามารถมีได้มากกว่า 1 ฟอร์ม แต่ละฟอร์มจะใช้บรรจุคอนโทรลและโปรแกรมย่อยในการตอบสนองเหตุการณ์ต่างๆ อาจอุปมาได้กับกระดาษเปล่าที่สามารถเขียนตัวหนังสือหรือวาดภาพลงไปได้

คอนโทรล (control)

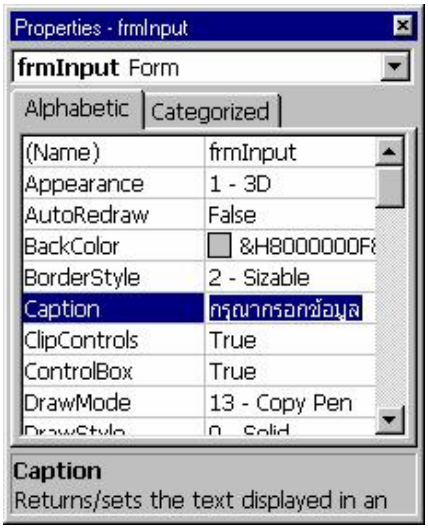
วัตถุที่มีรูปร่างและคุณสมบัติที่แตกต่างกันออกไปตามลักษณะการใช้งาน ใช้เป็นทางให้ผู้ใช้กำหนดเหตุการณ์ต่างๆ ซึ่งคอนโทรลจะรับรู้เหตุการณ์นั้นโดยอัตโนมัติ (วิชวลเบสิกเขียนคำสั่งส่วนนี้ไว้แล้ว) และตอบสนองโดยผ่านการเขียนคำสั่งที่โปรแกรมเมอร์จะเป็นผู้เขียนไว้

คุณสมบัติ (property)

หมายถึง ข้อมูลรายละเอียดของฟอร์ม และคอนโทรลต่างๆ ที่สามารถปรับแต่งเปลี่ยนแปลงได้ โดยวิชวลเบสิกได้กำหนดคุณสมบัติเริ่มต้นของฟอร์ม และแต่ละคอนโทรลไว้แล้ว ซึ่งในการเขียนโปรแกรม เรามักจะเปลี่ยนแปลงคุณสมบัติบางอย่างเท่านั้น

การกำหนดค่าคุณสมบัติ ทำได้ 2 แบบคือ

1. กำหนดในช่อง Properties Window ตัวอย่างดังรูปที่ 2-19



รูปที่ 2-19 การกำหนดคุณสมบัติในช่อง Properties Window

2. กำหนดโดยการเขียนคำสั่ง (code) ในโปรแกรมย่อย

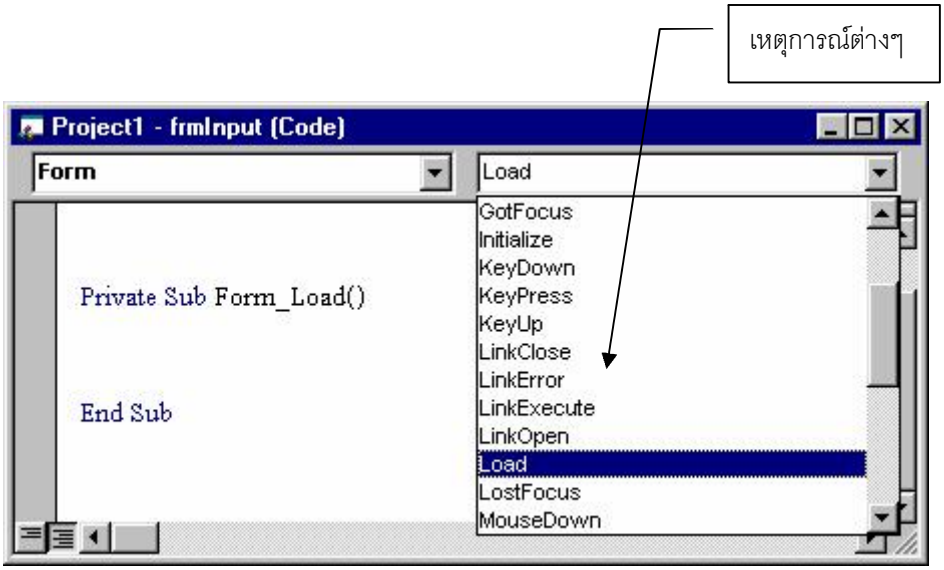
   รูปแบบ : <ชื่อของฟอร์มหรือคอนโทรล>.<คุณสมบัติ> = <ค่าที่กำหนด>

   ตัวอย่างดังรูปที่ 2-20

รูปที่ 2-20 ตัวอย่างการกำหนดคุณสมบัติในโปรแกรมย่อย

เหตุการณ์ (event)

หมายถึง การตอบสนองในรูปแบบต่างๆที่เป็นไปได้ของฟอร์ม หรือคอนโทรล ที่มีต่อการทำงานของผู้ใช้ โดยวิชวลเบสิกได้กำหนดไว้แล้วว่าฟอร์มหรือคอนโทรลต่างๆ จะสามารถมีเหตุการณ์อะไรได้บ้าง แต่ในขั้นต้นโปรแกรมจะกำหนดได้เพียงว่ามีเหตุการณ์อะไรเกิดขึ้น ส่วนเมื่อมีเหตุการณ์เกิดขึ้นแล้ว โปรแกรมจะทำงานอย่างใดต่อไป เป็นหน้าที่ของโปรแกรมเมอร์ที่จะต้องกำหนดวิธีการ และ/หรือคำสั่ง ให้เป็นไปตามที่ออกแบบไว้ ตัวอย่างดังรูปที่ 2-21



รูปที่ 2-21 ตัวอย่างเหตุการณ์ต่างๆ

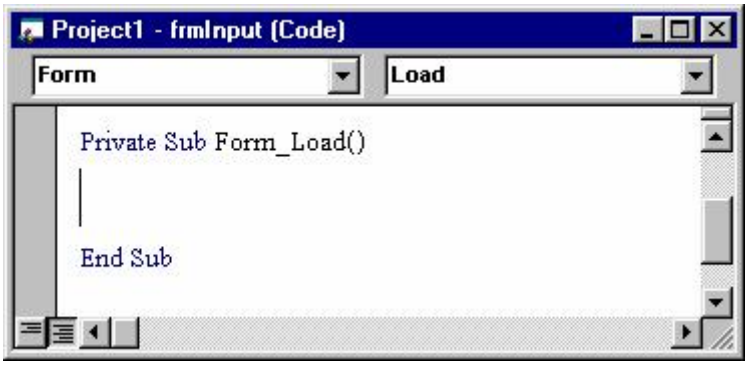
โปรแกรมย่อยเหตุการณ์ (event procedure)

ส่วนของโปรแกรมย่อยที่วิชวลเบสิกกำหนดขึ้น ใช้สำหรับเขียนคำสั่งเพื่อตอบสนองเหตุการณ์ต่างๆ

รูปแบบ : Private Sub <ฟอร์มหรือชื่อของคอนโทรล>_<เหตุการณ์>

End Sub

ตัวอย่างดังรูปที่ 2-22



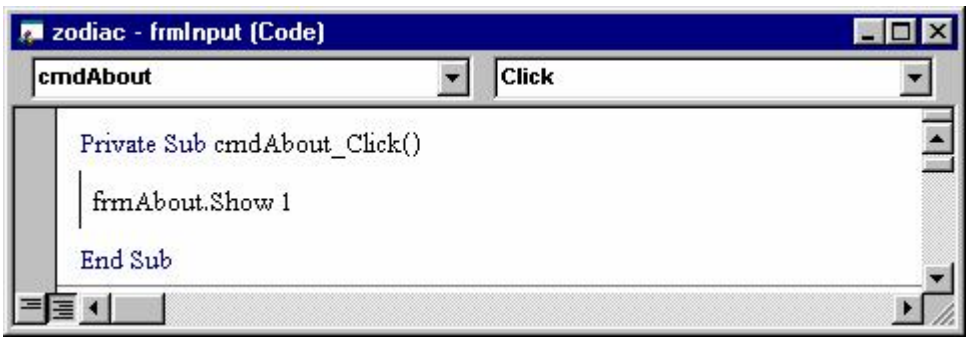
รูปที่ 2-22 ตัวอย่างโปรแกรมย่อยเหตุการณ์

วิธีการ (method)

หมายถึง โปรแกรมย่อยที่วิชวลเบสิกกำหนดไว้แล้วในฟอร์ม และคอนโทรลต่างๆ เป็นการสั่งให้ฟอร์มหรือคอนโทรลนั้นทำงานบางอย่างที่สามารถทำได้ เปรียบเหมือนพฤติกรรมที่วัตถุนั้นสามารถแสดงออกได้

รูปแบบ : <ชื่อของฟอร์มหรือคอนโทรล>.<ชื่อวิธีการ> (<ค่าพารามิเตอร์ที่ส่ง>)

ตัวอย่างดังรูปที่ 2-23



รูปที่ 2-23 ตัวอย่างวิธีการ

บทที่ 3 คุณสมบัติของฟอร์มและคอนโทรลพื้นฐาน

คุณสมบัติของฟอร์ม

มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-1

ตารางที่ 3-1 คุณสมบัติของฟอร์มที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของฟอร์ม
BorderStyle	กำหนดรูปแบบกรอบของฟอร์ม
Caption	แสดงข้อความบนแถบไตเติลของฟอร์ม
Height	กำหนดขนาดความสูงของฟอร์ม
MaxButton	กำหนดให้ปรากฏหรือไม่ปรากฏปุ่ม Maximize บนฟอร์ม
MinButton	กำหนดให้ปรากฏหรือไม่ปรากฏปุ่ม Minimize บนฟอร์ม
Width	กำหนดขนาดความกว้างของฟอร์ม
WindowState	กำหนดให้แสดงรูปแบบต่างๆของฟอร์ม เมื่อทำการรันโปรแกรม

คุณสมบัติของคอนโทรลพื้นฐาน

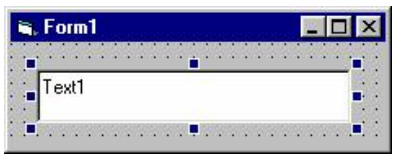
1. TextBox 

ใช้สำหรับรับหรือแสดงข้อความต่างๆ มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-2

ตารางที่ 3-2 คุณสมบัติของ TextBox ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Text	เก็บข้อความที่แสดงในช่องพิมพ์ของคอนโทรล

ตัวอย่าง TextBox แสดงดังรูปที่ 3-1



รูปที่ 3-1 ตัวอย่าง TextBox

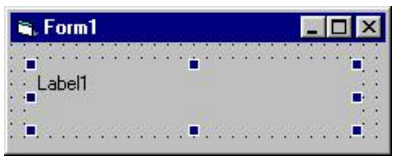
2. Label 

ใช้สำหรับแสดงข้อความที่ผู้ใช้ไม่สามารถแก้ไขได้ โดยมากมักนำมากำกับหน้าคอนโทรลอื่น เพื่อบอกความหมายของคอนโทรลนั้นๆ มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-3

ตารางที่ 3-3 คุณสมบัติของ Label ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Alignment	รูปแบบการจัดข้อความ ชิดซ้าย กลาง หรือ ชิดขวา
AutoSize	ให้ปรับขนาดตามความยาวของข้อความหรือไม่
Caption	แสดงข้อความบนคอนโทรล
Font	กำหนดชนิด ขนาด และลักษณะพิเศษของตัวอักษร
WordWrap	กำหนดให้มีการตัดคำขึ้นบรรทัดใหม่หรือไม่

ตัวอย่างดังรูปที่ 3-2



รูปที่ 3-2 ตัวอย่าง Label

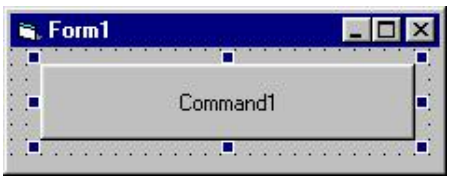
3. CommandButton 

ใช้สำหรับให้สั่งทำงานต่างๆ เช่น กดปุ่มเพื่อประมวลผล หรือเพื่อจบการทำงาน เป็นต้น มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-4

ตารางที่ 3-4 คุณสมบัติของ CommandButton ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Caption	แสดงข้อความบนคอนโทรล

ตัวอย่างดังรูปที่ 3-3



รูปที่ 3-3 ตัวอย่าง CommandButton

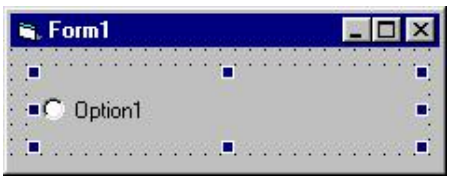
4. OptionButton 

ใช้สำหรับสร้างตัวเลือกภายในฟอร์ม ถ้าหากต้องการให้ผู้ใช้เลือกตัวใดตัวหนึ่งเท่านั้น มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-5

ตารางที่ 3-5 คุณสมบัติของ OptionButton ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Caption	แสดงข้อความบนคอนโทรล
Index	ใช้ระบุลำดับที่ของ Control Array
Value	ระบุว่าคอนโทรลนี้ถูกเลือก (True) หรือไม่ถูกเลือก (False)

ตัวอย่างดังรูปที่ 3-4



รูปที่ 3-4 ตัวอย่าง OptionButton

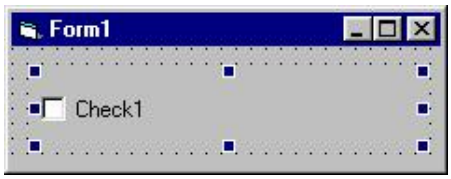
5. CheckBox 

ใช้สำหรับสร้างตัวเลือกภายในฟอร์ม ถ้าหากต้องการให้ผู้ใช้เลือกตัวเลือกตามจำนวนที่ต้องการ มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-6

ตารางที่ 3-6 คุณสมบัติของ CheckBox ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Caption	แสดงข้อความบนคอนโทรล
Index	ใช้ระบุลำดับที่ของ Control Array
Value	มีค่าได้ 3 ค่าคือ 0 ไม่ถูกเลือก, 1 ถูกเลือก และ 2 แสดงเป็นสีเทา ซึ่งหมายถึงถูกเลือกบางส่วน

ตัวอย่างดังรูปที่ 3-5



รูปที่ 3-5 ตัวอย่าง CheckBox

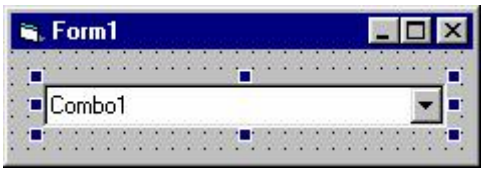
6. ComboBox 

ใช้สำหรับกรอกข้อความลงในช่องของ ComboBox มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-7

ตารางที่ 3-7 คุณสมบัติของ ComboBox ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
List	รายการข้อมูล
Text	เก็บข้อความที่แสดงในช่องพิมพ์ของคอนโทรล

ตัวอย่างดังรูปที่ 3-6



รูปที่ 3-6 ตัวอย่าง ComboBox

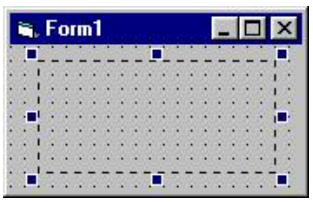
7. Image 

ใช้สำหรับแสดงรูปภาพ ซึ่งรูปภาพที่ปรากฏใน Image สามารถย่อหรือขยายได้ มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-8

ตารางที่ 3-8 คุณสมบัติของ Image ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Picture	สำหรับระบุภาพที่ต้องการนำมาแสดง (กลุ่มไฟล์ภาพที่สามารถนำมาแสดง ได้แก่ไฟล์นามสกุล .BMP, .DIB, .GIF, .JPG, .WMF, .EMF, .ICO และ .CUR)
Stretch	กำหนดให้สามารถย่อ หรือขยายภาพในคอนโทรลนั้นๆ ได้

ตัวอย่างดังรูปที่ 3-7



รูปที่ 3-7 ตัวอย่าง Image

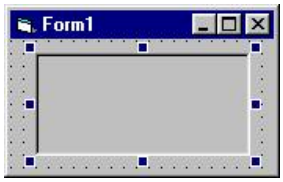
8. PictureBox 

ใช้สำหรับแสดงรูปภาพ ต่างกับ Image ที่ PictureBox ไม่มีคุณสมบัติ ย่อ ขยายภาพ แต่สามารถผสมข้อความ การวาดเส้น หรือ ลวดลายต่างๆ ลงในรูปภาพ โดยผ่านวิธีการ เช่น Line, Circle เป็นต้น มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-9

ตารางที่ 3-9 คุณสมบัติของ PictureBox ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
AutoSize	กำหนดให้ปรับขนาดของคอนโทรลให้เท่ากับภาพปรากฏในคอนโทรล
Picture	สำหรับระบุภาพที่ต้องการนำมาแสดง (กลุ่มไฟล์ภาพ เหมือนกับ Image)
ToolTipText	กำหนดข้อความที่ต้องการแสดงเมื่อเลื่อนลูกศรของเมาส์มาชี้บนคอนโทรล
BorderStyle	กำหนดลักษณะของเส้นขอบรูป

ตัวอย่างดังรูปที่ 3-8



รูปที่ 3-8 ตัวอย่าง PictureBox

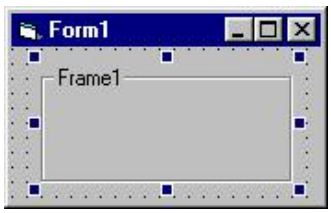
9. Frame 

ใช้สำหรับจัดกลุ่มหรือขอบเขตให้กับคอนโทรลอื่นๆ ที่อยู่ใน Frame ซึ่งคอนโทรลเหล่านั้น จะกลายเป็นคอนโทรลลูกของ Frame ทันที ทำให้ถ้า Frame นั้นถูกเปลี่ยนตำแหน่งบนฟอร์ม คอนโทรลต่างๆ ใน Frame ก็จะเปลี่ยนตำแหน่งตามไปด้วย เป็นต้น มีคุณสมบัติที่ใช้บ่อย ดังตารางที่ 3-10

ตารางที่ 3-10 คุณสมบัติของ Frame ที่ใช้บ่อย

คุณสมบัติ	**หน้าที่**
Name	ชื่อของคอนโทรล
Caption	แสดงข้อความบริเวณมุมบนซ้ายของคอนโทรล

ตัวอย่างดังรูปที่ 3-9



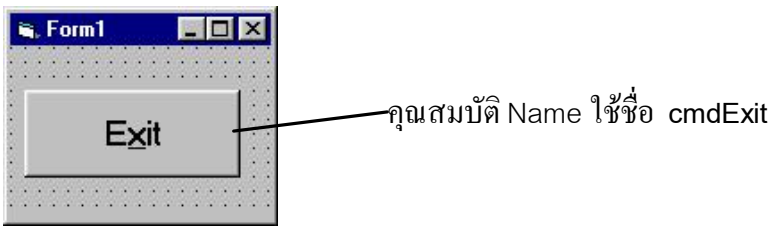
รูปที่ 3-9 ตัวอย่าง Frame

<u>แนวทางการตั้งชื่อของคุณสมบัติ</u> Name

ในการเขียนโปรแกรมแบบตอบสนองเหตุการณ์นี้ เนื่องจากการเขียนคำสั่งต่างๆ ไม่ได้เรียงลำดับเหมือนกับการเขียนโปรแกรมแบบตามลำดับขั้นตอน แต่เป็นการเขียนลงในโปรแกรมย่อยเหตุการณ์ต่างๆ ของแต่ละฟอร์มหรือคอนโทรล ดังนั้น การตั้งชื่อคุณสมบัติ Name ให้สื่อถึงฟอร์มหรือชนิดของคอนโทรลจึงเป็นเรื่องสำคัญ ข้อแนะนำในการตั้งชื่อคือ

> 3 ตัวอักษรแรกเป็นอักษรย่อของฟอร์มหรือชนิดของคอนโทรล ใช้ตัวพิมพ์เล็ก ตัวอักษรถัดไปเป็นคำที่สื่อถึงการใช้งาน โดยเริ่มต้นด้วยตัวพิมพ์ใหญ่

ตัวอย่างการตั้งชื่อ Name ของคอนโทรล CommandButton ใช้สำหรับออกจากโปรแกรม ดังรูปที่ 3-10



รูปที่ 3-10 ตัวอย่างการตั้งชื่อ Name ของคอนโทรล CommandButton

สำหรับอักษรย่อต่างๆ ของวัตถุฟอร์มและคอนโทรล แสดงดังตารางที่ 3-11

ตารางที่ 3-11 อักษรย่อต่างๆ ของวัตถุฟอร์มและคอนโทรล

วัตถุ	อักษรย่อ
Form	frm
TextBox	txt
Label	lbl
CommandButton	cmd
OptionButton	opt
CheckBox	chk
ComboBox	cbo
Image	img
PictureBox	pic
Frame	fra

การตั้งชื่อของคุณสมบัติ Name แบบแถวลำดับ

ในการทำงานกับคอนโทรล บ่อยครั้งที่เราใช้คอนโทรลชนิดเดียวกัน ในการทำงานที่คล้ายคลึงกัน เช่น ใช้ OptionButton ให้ผู้ใช้เลือกตอบว่าใช้วัสดุคอนกรีตหรือเหล็ก ดังรูปที่ 3-11

รูปที่ 3-11 การกำหนด OptionButton เพื่อเลือกทำงาน

เมื่อผู้ใช้คลิกตัวเลือกใดตัวเลือกหนึ่ง เราก็จะต้องเขียนโปรแกรมเพื่อตอบสนองเหตุการณ์คลิกนี้ ซึ่งจะเห็นว่า ทั้งสองปุ่มบอกถึงความเป็นวัสดุใดวัสดุหนึ่ง กรณีทำนองนี้ มักนิยมการกำหนด

คุณสมบัติ Name แบบแถวลำดับ (array) คือ ชื่อเดียวกัน ต่างกันที่คุณสมบัติ Index ดังตัวอย่างด้านบน กำหนดคุณสมบัติ Name ชื่อเดียวกันคือ optMaterial แต่ตัวเลือกคอนกรีต กำหนดคุณสมบัติ Index เท่ากับ 0 และตัวเลือกเหล็ก Index เท่ากับ 1 และการอ้างถึงชื่อของแต่ละตัวเลือกเขียนได้ว่า

ตัวเลือกคอนกรีต คือ optMaterial(0)

ตัวเลือกเหล็ก คือ optMaterial(1)

การกำหนดคอนโทรลที่มีชนิดเดียวกัน ให้ Name เหมือนกัน แต่ต่างกันที่ Index นี้ เรียกว่าแถวลำดับของคอนโทรล (control array)

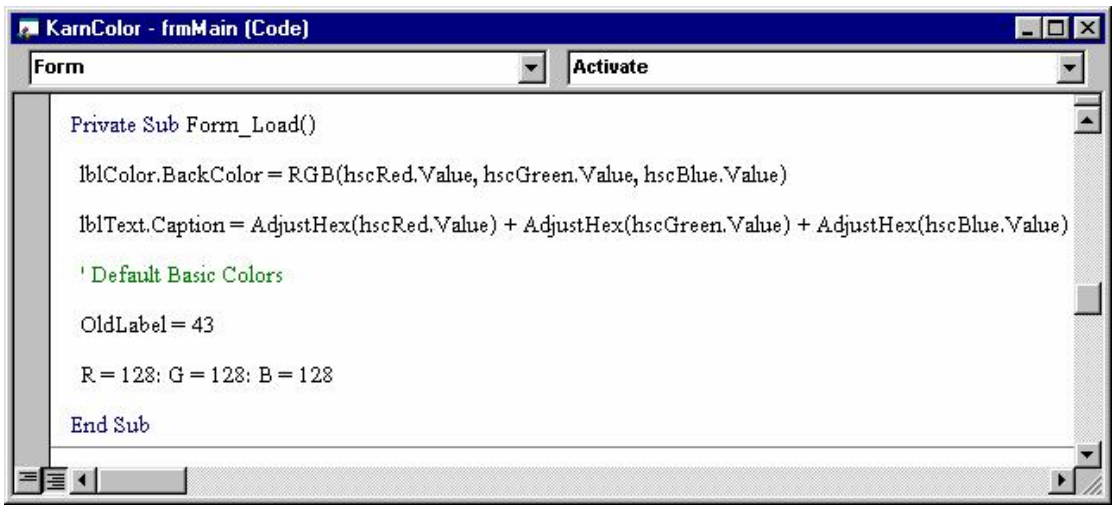
บทที่ 4 เหตุการณ์และวิธีการ

เหตุการณ์และวิธีการของฟอร์ม

เมื่อมีการเรียกฟอร์มใดๆ ให้ปรากฏบนจอภาพ จะมีลำดับการเกิดเหตุการณ์ของฟอร์มคือ

1. Initialize เกิดขึ้นเมื่อโปรแกรมกำหนดหน่วยความจำขึ้นมาสำหรับฟอร์มนั้น
2. Load เกิดขึ้นเมื่อฟอร์มแสดงบนจอภาพ
3. Resize เกิดขึ้นเมื่อมีการปรับขนาดของฟอร์ม
4. Activate เกิดขึ้นเมื่อฟอร์มนั้นถูกเรียกใช้งานอยู่ (เรียกว่าฟอร์มนั้นได้รับการ Focus)

ซึ่งหมายความว่า เหตุการณ์ 4 แบบนี้เกิดขึ้นทุกครั้ง แต่จะเขียนคำสั่งให้ตอบสนองเหตุการณ์หรือไม่เป็นอีกเรื่องหนึ่ง เช่นเดียวกับเหตุการณ์อื่นๆ ที่ฟอร์มมีอยู่ ก็ขึ้นอยู่กับการออกแบบโปรแกรมว่าจะกำหนดให้ตอบสนองเหตุการณ์นั้นหรือไม่ แต่โดยทั่วไปของโปรแกรมส่วนใหญ่แล้ว เรามักเขียนคำสั่งในเหตุการณ์ Load ของฟอร์ม เพื่อเป็นการปรับคุณสมบัติและตัวแปรต่างๆ ให้เป็นค่าเริ่มต้น เรียกได้ว่า เหตุการณ์ Load เป็นเหตุการณ์ที่ใช้บ่อยที่สุดของฟอร์ม เช่นตัวอย่างดังรูปที่ 4-1



รูปที่ 4-1 เหตุการณ์ Load ของฟอร์ม

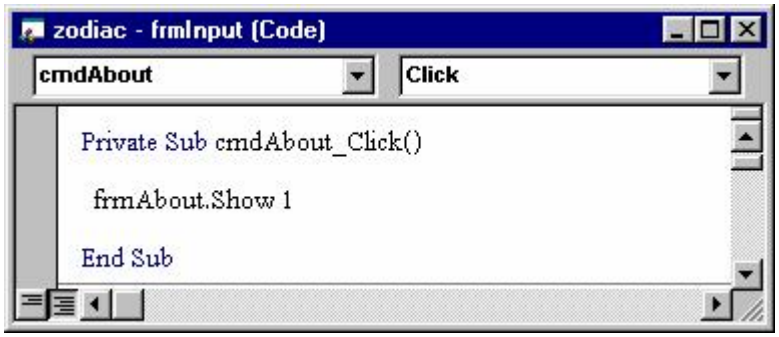
สำหรับวิธีการและคำสั่งที่ใช้บ่อยของฟอร์ม คือ การเปิด-ปิดฟอร์ม และออกจากโปรแกรม ซึ่งมีรายละเอียดดังนี้

วิธีการ Show เปิดฟอร์มขึ้นมาแสดงผลบนจอภาพ

รูปแบบ : <Name ของฟอร์ม>.Show (<ค่าพารามิเตอร์>)

สำหรับค่าพารามิเตอร์ ถ้าใส่ค่า 1 หมายถึง ฟอร์มที่เปิดมานี้ จะต้องมีการตอบสนองก่อนจึงจะไปทำงานฟอร์มอื่นได้

ตัวอย่างดังรูปที่ 4-2



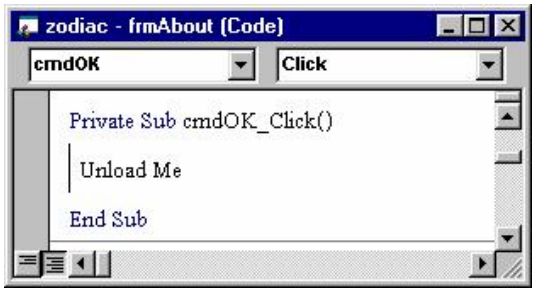
รูปที่ 4-2 ตัวอย่างการใช้วิธีการ Show

คำสั่ง Unload ปิดฟอร์ม

รูปแบบ :Unload <Name ของฟอร์ม>

หากเป็นการสั่งปิดฟอร์มตัวเอง Name ของฟอร์มสามารถใช้คำว่า Me แทนได้

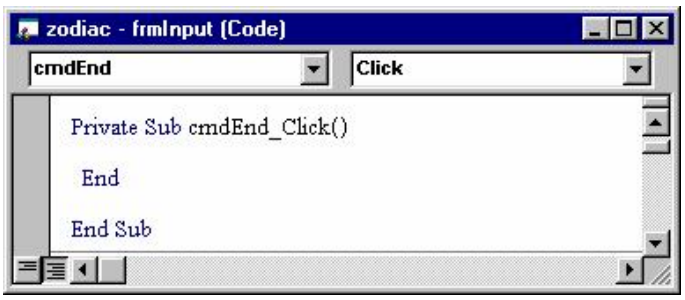
ตัวอย่างดังรูปที่ 4-3



รูปที่ 4-3 ตัวอย่างการใช้คำสั่ง Unload

คำสั่ง End ออกจากโปรแกรม

ตัวอย่างดังรูปที่ 4-4



รูปที่ 4-4 ตัวอย่างการใช้คำสั่ง End

เหตุการณ์ของคอนโทรล

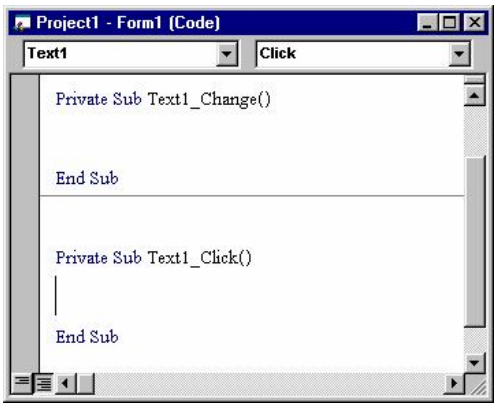
แบ่งเหตุการณ์ที่ใช้บ่อยออกเป็น 3 กลุ่มของคอนโทรล คือ

1. กลุ่มประเภทรับข้อความ ได้แก่ คอนโทรล TextBox มักเขียนโปรแกรมเพื่อตอบสนองเหตุการณ์ ดังนี้

   Change เกิดขึ้นเมื่อมีการเปลี่ยนแปลงข้อความในช่องรับข้อความ

   Click เกิดขึ้นเมื่อมีการคลิกเมาส์ในช่องรับข้อความ

   ตัวอย่างดังรูปที่ 4-5

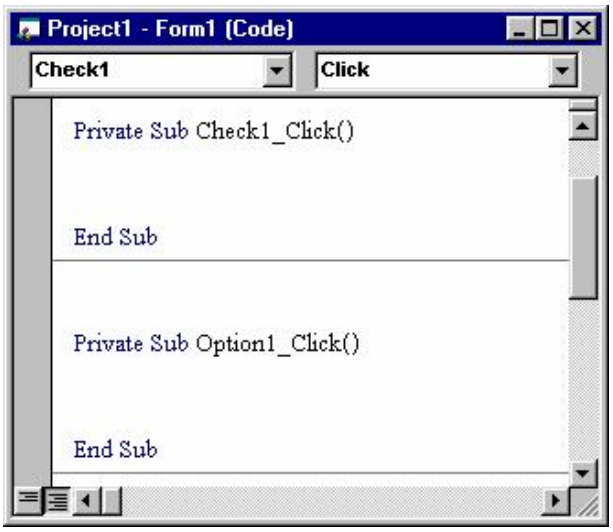


รูปที่ 4-5 ตัวอย่างเหตุการณ์ Change และ Click ของคอนโทรลประเภทรับข้อความ

2. กลุ่มประเภทตัวเลือก ได้แก่ OptionButton, CheckBox มักเขียนโปรแกรมเพื่อตอบสนองเหตุการณ์ ดังนี้

Click เกิดขึ้นเมื่อมีการคลิกเมาส์ในช่องตัวเลือก

ตัวอย่างดังรูปที่ 4-6

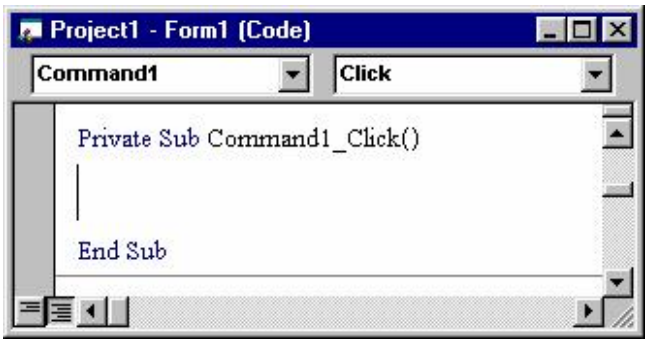


รูปที่ 4-6 ตัวอย่างเหตุการณ์ Click ของคอนโทรลประเภทตัวเลือก

3. กลุ่มประเภทปุ่มคำสั่ง ได้แก่ CommandButton มักเขียนโปรแกรมเพื่อตอบสนองเหตุการณ์ ดังนี้

   Click เกิดขึ้นเมื่อมีการคลิกเมาส์บนปุ่มคำสั่ง

   ตัวอย่างดังรูปที่ 4-7



รูปที่ 4-7 ตัวอย่างเหตุการณ์ Click ของคอนโทรลประเภทปุ่มคำสั่ง

ถ้าอธิบายในเชิงการเขียนคำสั่งคือ เรามักจะเขียนคำสั่งเพื่อตอบสนองเหตุการณ์ของคอนโทรลต่างๆ ในโปรแกรมย่อยเหตุการณ์ที่กล่าวมาข้างต้น

บทที่ 5 ภาษาโปรแกรมของวิชวลเบสิก

หลังจากที่เขียนโปรแกรมส่วนติดต่อกับผู้ใช้เรียบร้อยแล้ว ก็มาถึงขั้นตอนการเขียนโปรแกรมให้ทำงานได้ตามวัตถุประสงค์ ซึ่งต้องรู้ถึงภาษาโปรแกรมของวิชวลเบสิกในการเขียนคำสั่งต่างๆ และก่อนที่จะเขียนคำสั่งเพื่อให้โปรแกรมทำงานนั้น เราจะต้องกำหนดขั้นตอนจากการที่มนุษย์ทำงานไปเป็นคอมพิวเตอร์ทำงานให้เรียบร้อย ซึ่งพื้นฐานความรู้ที่จำเป็นคือ อัลกอริทึม (algorithm) และผังงาน (flow chart)

อัลกอริทึม

อัลกอริทึม หมายถึง ขั้นตอนหรือวิธีการที่เป็นระบบ ซึ่งกำหนดขึ้นเพื่อให้คอมพิวเตอร์ทำงานตามที่เราต้องการ นั่นหมายถึงการแบ่งงานออกเป็นขั้นตอนย่อยๆ ที่มีการทำงานแน่นอน อัลกอริทึมบางอย่างก็จะคล้ายคลึงกับกระบวนการทำงานของมนุษย์ตามปกติ แต่บางอัลกอริทึมจะต้องพัฒนาขึ้นโดยเฉพาะ เนื่องจากการทำงานของคอมพิวเตอร์กับการทำงานของมนุษย์มีความแตกต่างกัน

ตัวอย่างง่ายๆ ในชีวิตประจำวัน เช่น คนที่เกิดอยู่ในช่วงวันที่ 21 มีนาคม ถึง 20 เมษายน เป็นคนราศีเมษ เมื่อมีคนมาบอกว่าเกิดวันที่ 30 มีนาคม เราตอบได้ว่า คนนั้นเป็นคนราศีเมษ เนื่องจากความคิดบอกว่าวันที่ 30 มีนาคม อยู่ในช่วงระหว่างวันและเดือนของราศีเมษ แต่หากเราจะสั่งให้คอมพิวเตอร์ตอบคำถามนี้ เราจะทำอย่างไร

อัลกอริทึมที่เป็นไปได้คือ

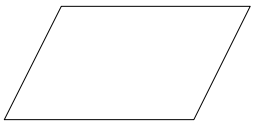
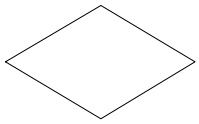
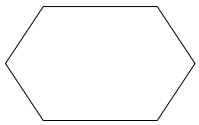
ถ้า เกิดวันที่ >= 21 **และ** เกิดเดือนมีนาคม **เป็น** คนราศีเมษ

ถ้า เกิดวันที่ <= 20 **และ** เกิดเดือนเมษายน **เป็น** คนราศีเมษ

อาจกล่าวได้ว่า **การพัฒนาอัลกอริทึมในแต่ละโปรแกรม เป็นหัวใจสำคัญของโปรแกรมนั้น** ที่จะทำให้เขียนเสร็จและมีความถูกต้องหรือไม่ ซึ่งจะต้องอาศัยความรู้ในงานที่เราจะเขียนโปรแกรม และประสบการณ์ในการเขียนโปรแกรม ประกอบเข้าด้วยกัน

ผังงาน

ผังงาน หมายถึง แผนภาพแบบหนึ่ง ที่แสดงถึงอัลกอริทึมการทำงานของโปรแกรม หรือกล่าวอีกอย่างหนึ่งว่า ผังงาน คือ การเขียนอัลกอริทึมให้อยู่ในรูปของแผนภาพที่เป็นสากล ทำให้มีประโยชน์อื่นอีกคือ เป็นเสมือนเครื่องมือสื่อสารระหว่างผู้เขียนโปรแกรมด้วยกัน หรือระหว่างผู้เขียนโปรแกรมกับผู้ใช้ ว่าแผนงานหรือการประมวลผลของโปรแกรมจะมีลักษณะขั้นตอนตามนี้ การเขียนผังงาน จะประกอบด้วยสัญลักษณ์หลัก ดังรูปที่ 5-1

สัญลักษณ์	ความหมาย
	**เริ่มต้น และหยุดการทำงาน**
	**อินพุต และเอาต์พุต แบบไม่ระบุอุปกรณ์แสดงผล**
	**ประมวลผล**
	**ทางเลือกเส้นทางการทำงาน**
	**จัดเตรียมข้อมูล**
	**จุดเชื่อมต่อ**
	**จุดเชื่อมต่อของผังงานที่ไม่จบในหน้าเดียว**

รูปที่ 5-1 สัญลักษณ์หลักของผังงาน

ตัวแปร

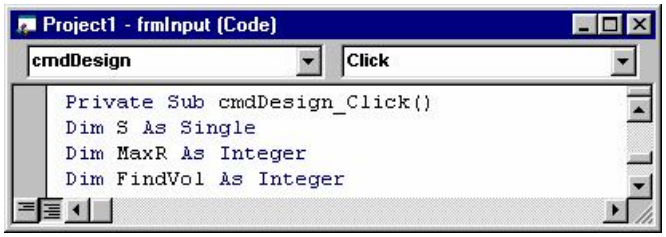
ตัวแปร (variable) ใช้ในการประมวลผลของโปรแกรม (รับข้อมูล คำนวณ แสดงข้อมูล) ซึ่งตัวแปรในวิชวลเบสิก แบ่งออกเป็นแบบข้อมูลพื้นฐาน ดังตารางที่ 5-1

ตารางที่ 5-1 แบบข้อมูลพื้นฐานของตัวแปร

แบบข้อมูล	**ขอบเขต**
Byte	0-255
Integer	–32,767 ถึง 32,768
Long	–2,147,483,648 ถึง 2,147,483,647
Single	1.401298E-45 ถึง 3.402823E38 เลขบวก -3.402823E38 ถึง -1.401298E-45 เลขลบ 0
Double	4.94065645841247E-324 ถึง 1.79769313486232E308 เลขบวก -1.79769313486232E308 ถึง -4.94065645841247E-324 เลขลบ 0
Boolean	True กับ False
String	ใช้ในการเก็บข้อความที่เป็นชุดของตัวอักขระ
Variant	กำหนดข้อมูลให้เป็นแบบที่มีใน VB แบบใดก็ได้ แล้วแต่การใช้งาน

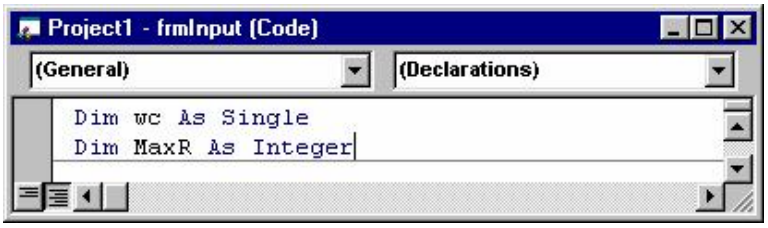
ขอบเขตของตัวแปร

1. ใช้เฉพาะในโปรแกรมย่อย
   รูปแบบ Dim ชื่อตัวแปร As แบบข้อมูล
   ประกาศในโปรแกรมย่อยที่ต้องการใช้ตัวแปรนั้น
   ตัวอย่างดังรูปที่ 5-2



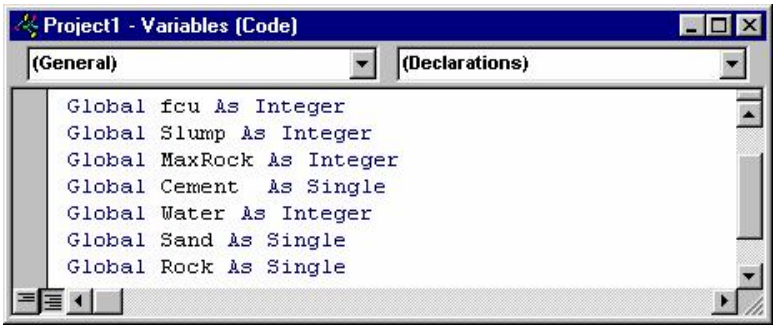
รูปที่ 5-2 ตัวอย่างการประกาศตัวแปรในโปรแกรมย่อย

2. ใช้ในแต่ละฟอร์ม สามารถใช้ตัวแปรได้กับโปรแกรมย่อยที่มีในฟอร์มนั้น
   รูปแบบ Dim ชื่อตัวแปร As แบบข้อมูล
   ประกาศใน Declarations ของฟอร์มนั้น
   ตัวอย่างดังรูปที่ 5-3



รูปที่ 5-3 ตัวอย่างการประกาศตัวแปรในฟอร์ม

3. ใช้กับทุกฟอร์ม สามารถใช้ตัวแปรได้กับโปรแกรมย่อยของทุกฟอร์ม
   รูปแบบ Global ชื่อตัวแปร As แบบข้อมูล
   ประกาศในโมดูล (Module)
   ตัวอย่างดังรูปที่ 5-4



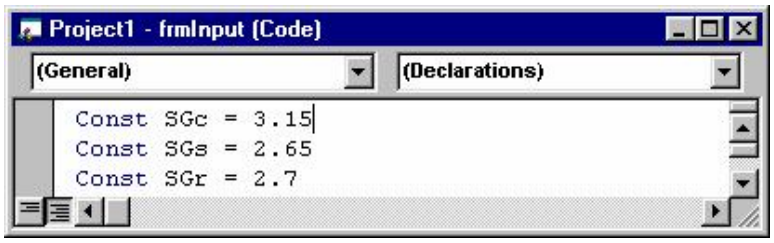
รูปที่ 5-4 ตัวอย่างการประกาศตัวแปรในโมดูล

ค่าคงที่

เราสามารถกำหนดตัวแปรให้เป็นค่าคงที่ (constant) ได้ 2 รูปแบบคือ

1. Const ชื่อตัวแปร = ค่าที่กำหนด (กำหนดในโปรแกรมย่อย หรือในแต่ละฟอร์ม)
2. Global Const ชื่อตัวแปร = ค่าที่กำหนด (กำหนดในโมดูล)

ตัวอย่างดังรูปที่ 5-5



รูปที่ 5-5 ตัวอย่างการกำหนดค่าคงที่

ขอบเขตของค่าคงที่ มีลักษณะเดียวกับขอบเขตของตัวแปร

โอเปอเรเตอร์

โอเปอเรเตอร์ (operator) คือ ตัวดำเนินการเพื่อให้ได้ผลลัพธ์ออกมาตามที่ต้องการ แบ่งออกเป็น 4 แบบ ได้แก่ คำนวณ เปรียบเทียบ ตรรกะ และเชื่อมข้อมูล รายละเอียดและลำดับการทำงานก่อนหลังของแต่ละโอเปอเรเตอร์ แสดงดังตารางที่ 5-2 ถึง 5-5

ตารางที่ 5-2 โอเปอเรเตอร์คำนวณ

ลำดับการทำงาน	โอเปอเรเตอร์คำนวณ	คำอธิบาย
1	^	ยกกำลัง
2	-	การแสดงค่าลบของตัวเลข
3	*, /	คูณ, หาร
4	\	หารจำนวนเต็ม
5	Mod	หาเศษจากการหาร
6	+, -	บวก, ลบ

ตารางที่ 5-3 โอเปอเรเตอร์เปรียบเทียบ

ลำดับการทำงาน	โอเปอเรเตอร์เปรียบเทียบ	คำอธิบาย
1	=	เท่ากับ
2	<>	ไม่เท่ากับ
3	>=	มากกว่าหรือเท่ากับ
4	<=	น้อยกว่าหรือเท่ากับ
5	<	น้อยกว่า
6	>	มากกว่า

ตารางที่ 5-4 โอเปอเรเตอร์ตรรกะ

ลำดับการทำงาน	โอเปอเรเตอร์ตรรกะ
1	Not
2	And
3	Or
4	Xor
5	Eqv
6	Imp

ตารางที่ 5-5 โอเปอเรเตอร์เชื่อมข้อมูล

ลำดับการทำงาน	โอเปอเรเตอร์เชื่อมข้อมูล	คำอธิบาย
1	+	เชื่อม String กับ String
2	&	เชื่อม String กับ String หรือ Numeric ก็ได้

ชุดคำสั่ง

ชุดคำสั่ง (statement) เป็นชุดข้อความที่สั่งให้โปรแกรมทำงานตามที่ผู้เขียนโปรแกรมต้องการ ซึ่งมีชุดคำสั่งหลักดังนี้

1. **ชุดคำสั่งกำหนดค่า** ใช้สัญลักษณ์ "=" กำหนดค่าซ้ายมือให้เท่ากับค่าขวามือ

   ตัวอย่าง

```
fcu = txtFcu.Text
SandVol = FindVol - (Cement / SGc)
cboSlump.Text = cboSlump.List(Slump)
```

2. **ชุดคำสั่งเลือกเส้นทางการทำงาน** แบ่งเป็น

   2.1 If…Then…Else

   รูปแบบ

```
If เงื่อนไข Then
  ชุดคำสั่ง
[ElseIf เงื่อนไข Then
  ชุดคำสั่ง]
[Else
  ชุดคำสั่ง]
End If
```

หมายเหตุ : [] หมายถึง มีหรือไม่มีก็ได้แล้วแต่เงื่อนไขว่ามากกว่าหนึ่งเงื่อนไขหรือไม่

ตัวอย่าง

```
If WindowState = 0 Then
   Move (Screen.Width - Me.Width) \ 2, (Screen.Height - Me.Height) \ 2
End If


If MaxRock = 0 Then
   lblMaxRock.Caption = 20
ElseIf MaxRock = 1 Then
   lblMaxRock.Caption = 25
End If
```

2.2 Select Case

รูปแบบ

Select Case ตัวแปรที่ใช้ตรวจสอบ
  Case ค่าที่ 1
   ชุดคำสั่ง
  Case ค่าที่ 2
   ชุดคำสั่ง
  Case ค่าที่ n
   ชุดคำสั่ง
End Select

ตัวอย่าง

```
Select Case MaxRock
  Case 0
    lblMaxRock.Caption = 20
  Case 1
    lblMaxRock.Caption = 25
End Select
```

3. **ชุดคำสั่งควบคุมการทำงานซ้ำ** แบ่งเป็น

3.1 For…Next

รูปแบบ

For ตัวแปรแบบตัวเลข = ค่าเริ่มต้น To ค่าสิ้นสุด
  ชุดคำสั่ง
Next ตัวแปรแบบตัวเลข (ตัวเดิม)

ตัวอย่าง

```
For I = 1 To NumM
  NodeM(I).Head = Elem(I).H
  NodeM(I).Tail = Elem(I).t
  If NMat = 1 Then
    TypeM(I) = 1
```

```
        Else
            TypeM(I) = Elem(I).Set
        End If
    Next  I
```

3.2 Do…Loop

รูปแบบ *(มีหลายรูปแบบ แต่รูปแบบนี้ใช้บ่อย)*

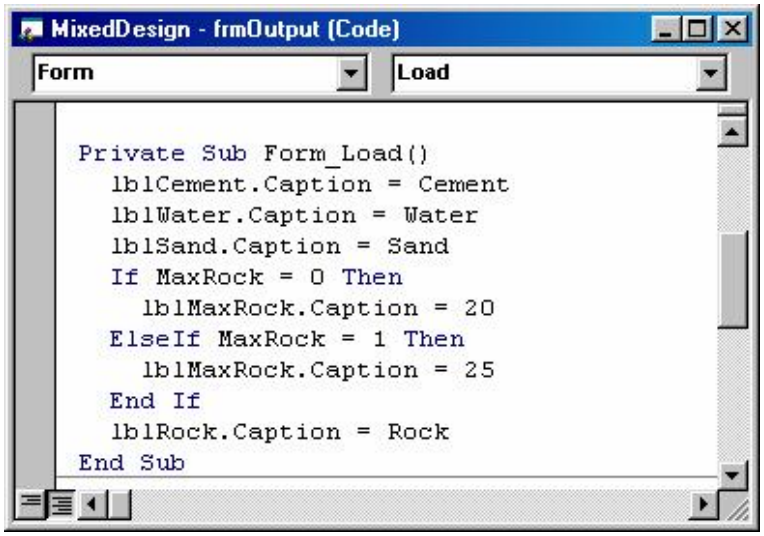
```
    Do
      ชุดคำสั่ง
    Loop Until เงื่อนไข
```

ตัวอย่าง

```
    Num = 0
    Do
      P = P + Num
      Num = Num + 1
    Loop Until Num = 21
```

บทที่ 6 โปรแกรมย่อยและฟังก์ชัน

ในบทก่อน เราทราบแล้วว่า การเขียนชุดคำสั่งในวิชวลเบสิก จะเขียนลงในโปรแกรมย่อยเหตุการณ์ต่างๆ ของฟอร์มหรือคอนโทรล ตัวอย่างดังรูปที่ 6-1



รูปที่ 6-1 ตัวอย่างโปรแกรมย่อยเหตุการณ์

ในทางปฏิบัติแล้ว ชุดคำสั่งในโปรแกรมย่อยเหตุการณ์ หลายครั้งที่มีบรรทัดเป็นจำนวนมาก เราสามารถแก้ปัญหาได้โดย สร้างโปรแกรมย่อยขึ้นมา แล้วเรียกโปรแกรมย่อยนั้นซ้อนอยู่ในโปรแกรมย่อยเหตุการณ์อีกที ซึ่งโปรแกรมย่อยที่วิชวลเบสิกกำหนดให้สร้างขึ้นมาเองได้นั้น แบ่งได้เป็น 2 รูปแบบคือ

1. โปรแกรมย่อย Sub
2. โปรแกรมย่อย Function

โปรแกรมย่อย Sub

เป็นโปรแกรมย่อยที่ไม่มีการส่งค่ากลับมา รูปแบบคือ

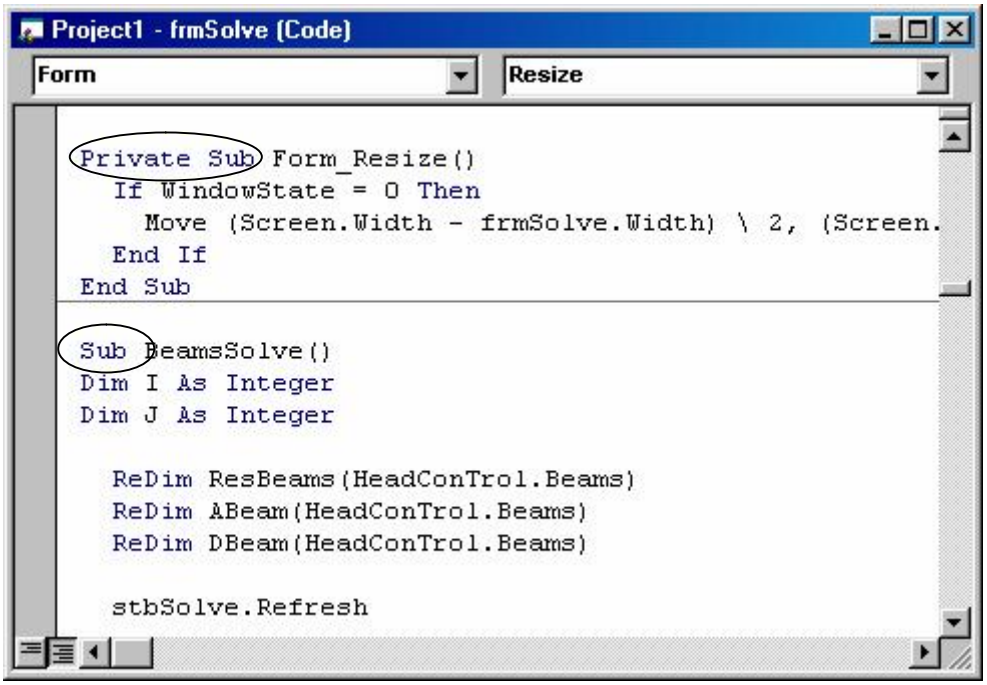
[Private | Public] Sub <ชื่อของโปรแกรมย่อย>(พารามิเตอร์ที่ส่งมาด้วย)

ชุดคำสั่ง

End Sub

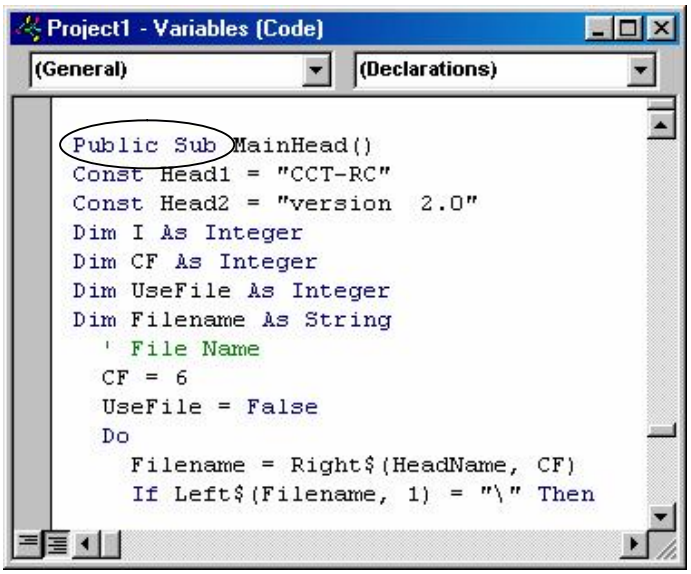
โดยที่

Private หมายถึง โปรแกรมย่อยที่สร้างขึ้น ใช้ได้กับฟอร์มที่อยู่ของโปรแกรมย่อยเท่านั้น ซึ่งเราจะละไว้ไม่เขียนก็ได้ ตัวอย่างดังรูปที่ 6-2



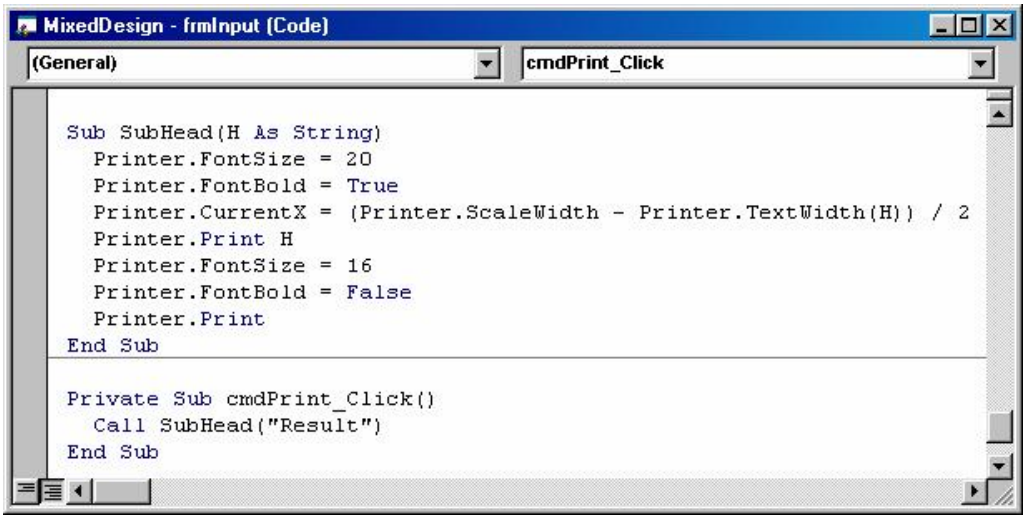
รูปที่ 6-2 แสดงตัวอย่าง Private Sub และ Sub

Public หมายถึง การกำหนดให้โปรแกรมย่อยนั้นใช้ได้กับทุกฟอร์ม โดยการสร้างโปรแกรมย่อยไว้ในโมดูล ตัวอย่างดังรูปที่ 6-3



รูปที่ 6-3 แสดงตัวอย่าง Public Sub

พารามิเตอร์ที่ส่งมาด้วย หมายถึง โปรแกรมย่อยที่สามารถรับข้อมูลส่งมาด้วย และการเรียกโปรแกรมย่อยในกรณีนี้ จะใช้คำสั่ง Call <ชื่อโปรแกรมย่อย(พารามิเตอร์)> ตัวอย่างดังรูปที่ 6-4



รูปที่ 6-4 แสดงตัวอย่างการใช้โปรแกรมย่อย Sub แบบกำหนดพารามิเตอร์

โปรแกรมย่อย Function

โปรแกรมย่อย Function หรือเรียกสั้นๆ ว่า ฟังก์ชัน คือโปรแกรมย่อยที่สร้างขึ้นเพื่อทำงานบางอย่างแล้วคืนค่ากลับไปยังโปรแกรมย่อยเหตุการณ์ที่เรียกใช้ รูปแบบคือ

[Private | Public] Function <ชื่อของโปรแกรมย่อย>(พารามิเตอร์ที่ส่งมาด้วย) (As Type)

ชุดคำสั่ง

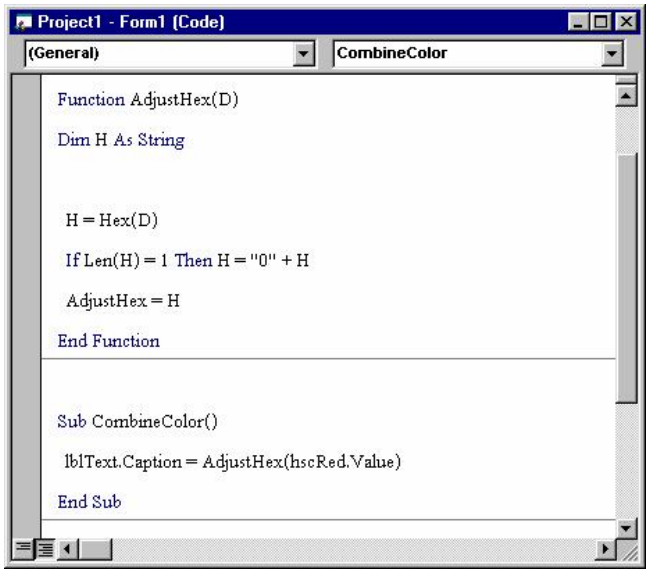
End Function

โดยที่

Private และ Public มีความหมายเดียวกันกับโปรแกรมย่อย Sub

As Type หมายถึง ชนิดของข้อมูลที่ฟังก์ชันนี้ส่งกลับมา (ซึ่งในบางกรณี จะไม่ใส่ก็ได้)

ตัวอย่างการกำหนดและเรียกใช้ฟังก์ชัน แสดงดังรูปที่ 6-5



รูปที่ 6-5 แสดงตัวอย่าง Function

สังเกตว่า การเรียกฟังก์ชัน จะต่างจากการเรียกโปรแกรมย่อย Sub คือ ฟังก์ชันจะเป็นค่าของคุณสมบัติหรือตัวแปรตัวใดตัวหนึ่ง แต่โปรแกรมย่อย Sub จะเป็นชุดคำสังทำงานบางอย่าง ซึ่งเรียกใช้โดยการกำหนดชื่อของโปรแกรมย่อยนั้นโดยตรง

นอกจากโปรแกรมย่อย Sub และฟังก์ชัน ที่เราสามารถสร้างขึ้นมาใช้งานแล้ว วิชวลเบสิกยังมีคำสั่งและฟังก์ชันอยู่มากมายให้เรียกใช้งานได้ ในเบื้องต้นนี้ คำสั่งที่ควรทราบคือ MsgBox ใช้สำหรับแสดงข้อความใน Dialog Box มีรูปแบบดังนี้

```
Msgbox prompt [, buttons] [,title]
```

โดยที่

prompt หมายถึง ข้อความที่กำหนดให้แสดง

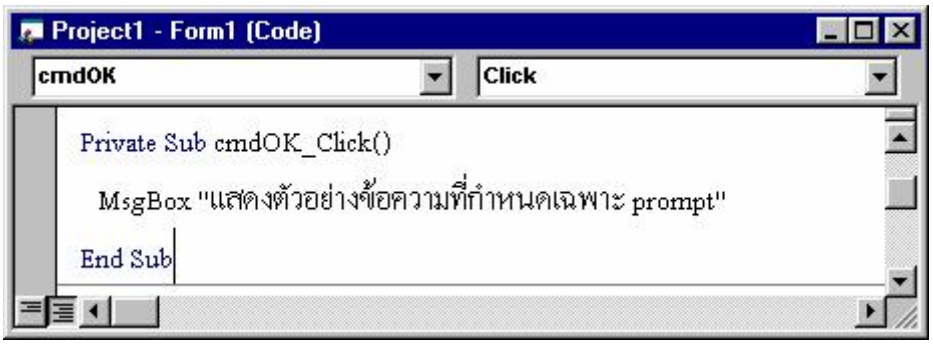
buttons หมายถึง ตัวเลือกที่แสดงปุ่มและ/หรือรูปภาพ (หรือใส่เป็นค่าคงที่ก็ได้) ซึ่งมีตัวเลือกที่ใช้บ่อย ดังตารางที่ 6-1

ตารางที่ 6-1 ตัวเลือกของ buttons ที่ใช้บ่อย

ค่าคงที่	ค่าตัวเลข	รายละเอียด
vbOKOnly	0	แสดงปุ่ม OK
vbCritical	16	แสดงไอคอน “Critical Message”

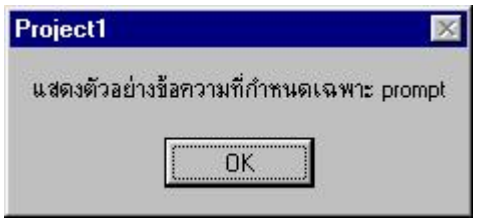
title หมายถึง ข้อความที่กำหนดให้แสดงบนแถบไตเติลของ Dialog Box

ตัวอย่างการเขียนคำสั่งแบบไม่ใส่ buttons และ title แสดงดังรูปที่ 6-6



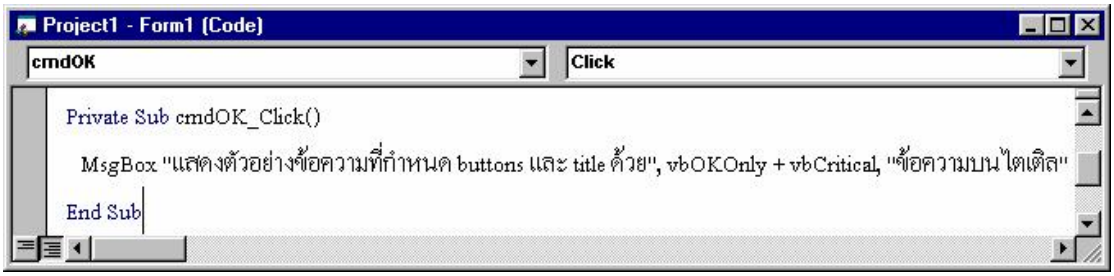
รูปที่ 6-6 ตัวอย่างการเขียนคำสั่ง MsgBox แบบไม่ใส่ buttons และ title

ซึ่งคำสั่งดังรูปที่ 6-6 เมื่อแสดงผล จะแสดงได้ดังรูปที่ 6-7



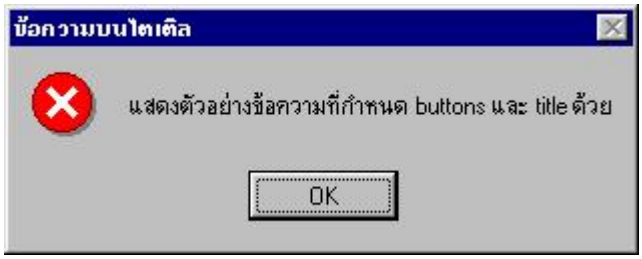
รูปที่ 6-7 การแสดงผลของคำสั่งดังรูปที่ 6-6

ตัวอย่างการเขียนคำสั่งแบบใส่ buttons และ title แสดงดังรูปที่ 6-8



รูปที่ 6-8 ตัวอย่างการเขียนคำสั่ง MsgBox แบบใส่ buttons และ title

ซึ่งคำสั่งดังรูปที่ 6-8 เมื่อแสดงผล จะแสดงได้ดังรูปที่ 6-9



รูปที่ 6-9 การแสดงผลของคำสั่งดังรูปที่ 6-8

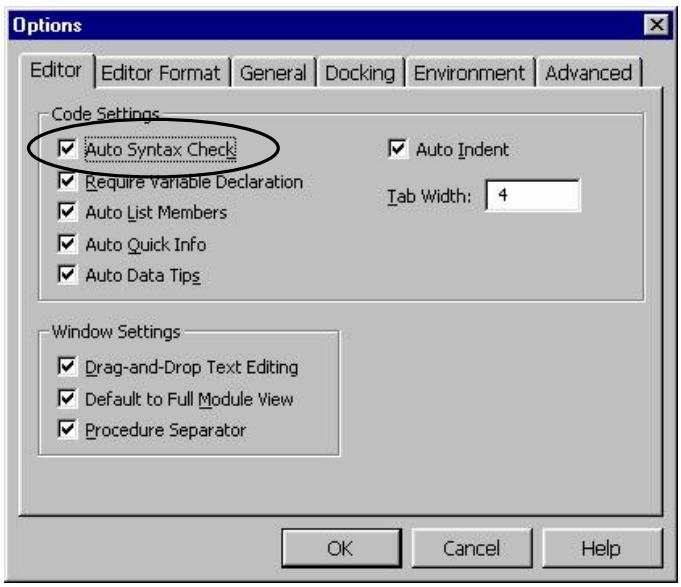
บทที่ 7 การจัดการข้อผิดพลาดเบื้องต้น

ในการเขียนโปรแกรมส่วนใหญ่หรือเรียกได้ว่าทุกโปรเจ็กต์ จะต้องพบกับข้อผิดพลาดในการเขียนโปรแกรมไม่มากก็น้อย สิ่งที่ดีที่สุดคือ การแก้ไขและป้องกันข้อผิดพลาดเหล่านั้นให้หมดไปก่อนที่จะทำเป็นโปรแกรมสำเร็จ (package) ซึ่งวิชวลเบสิกได้มีคำสั่งในการจัดการกับข้อผิดพลาด (error handler) นี้แล้ว โดยสิ่งที่ควรทราบในเบื้องต้นมีดังนี้

ประเภทของข้อผิดพลาด

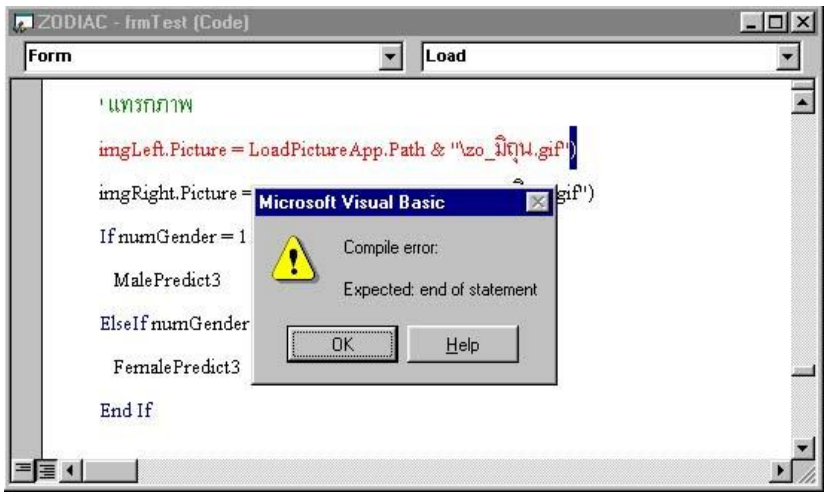
แบ่งเป็น 3 แบบคือ

1. ข้อผิดพลาดที่เกิดจากการเขียนโปรแกรมที่ผิดรูปแบบของภาษา (Syntax Error) ซึ่งสามารถให้วิชวลเบสิก แจ้งเตือนในขณะเขียนโปรแกรมได้ โดยไปที่เมนู Tools/Options แล้วคลิกที่ช่อง Auto Syntax Check ดังรูปที่ 7-1



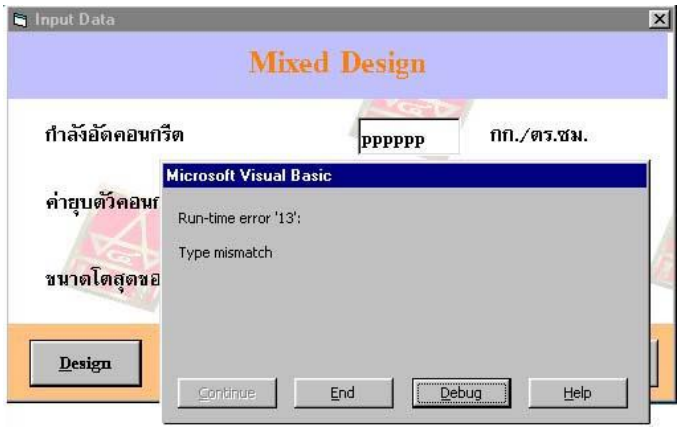
รูปที่ 7-1 การเลือก Auto Syntax Check

หลังจากนี้ เมื่อเราเขียนโปรแกรมผิดไวยากรณ์ของภาษาโปรแกรมเมื่อไร จะปรากฏข้อความแจ้งเตือนทุกครั้ง ตัวอย่างดังรูปที่ 7-2



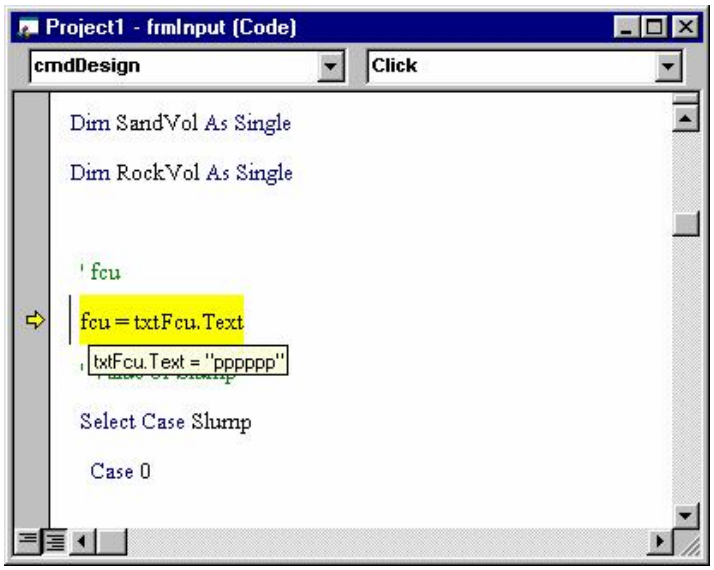
รูปที่ 7-2 ตัวอย่างข้อความแจ้งเตือนข้อผิดพลาด

2. ข้อผิดพลาดที่เกิดขึ้นในขณะรันโปรแกรม (Run-time Error) เกิดขึ้นในขณะที่โปรแกรมทำงานอยู่ ซึ่งมักจะเป็นการป้อนข้อมูลที่ผิดชนิดของตัวแปร เช่น ตัวแปรเป็นแบบตัวเลข แต่ป้อนเป็นตัวอักษร เป็นต้น หรือ มีการคำนวณบางอย่างที่ทำให้เกิดการหารด้วยศูนย์ ซึ่งข้อผิดพลาดนี้ หากเกิดขึ้นขณะที่เรารันโปรแกรมอยู่ในสภาพแวดล้อมของวิชวลเบสิก (คือยังไม่ได้ทำให้เป็นโปรแกรมสำเร็จ) วิชวลเบสิกจะแจ้งข้อผิดพลาดให้ ตัวอย่างดังรูปที่ 7-3



รูปที่ 7-3 ข้อผิดพลาดที่เกิดขึ้นในขณะรันโปรแกรม

และเมื่อคลิกปุ่ม Debug วิชวลเบสิกจะไปที่หน้าต่าง Code ณ ตำแหน่งคำสั่งที่ผิดพลาดให้โดยอัตโนมัติ ตัวอย่างดังรูปที่ 7-4



รูปที่ 7-4 วิชวลเบสิกแสดงตำแหน่งของคำสั่งที่ผิดพลาดให้โดยอัตโนมัติ

ข้อผิดพลาดแบบนี้ เป็นสิ่งสำคัญของโปรแกรม เพราะหากเราไม่ได้แก้ไขแล้วทำเป็นโปรแกรมสำเร็จ เมื่อเกิดข้อผิดพลาด โปรแกรมจะหยุดการทำงานทันที ซึ่งอาจทำให้เกิดความเสียหายได้มาก เช่น หากมีการป้อนข้อมูลแล้วยังไม่ได้บันทึกไฟล์ ก็ต้องกลับมาป้อนข้อมูลใหม่เป็นต้น

3. ข้อผิดพลาดที่เกิดจากแนวคิดที่ไม่ถูกต้องของการออกแบบ (Logic Error) ซึ่งข้อผิดพลาดแบบนี้ วิชวลเบสิกไม่สามารถตรวจสอบได้ เพราะเกิดจากแนวคิดที่ผิดพลาดของผู้เขียนโปรแกรมเอง การตรวจสอบทำได้โดย การตรวจทานอัลกอริทึมที่เขียนไว้หรือการเปรียบเทียบผลที่ได้จากการทำงานของโปรแกรมกับการทำงานตามปกติ เป็นต้น

<u>การแจ้งข้อผิดพลาดกับผู้ใช้อย่างง่าย</u>

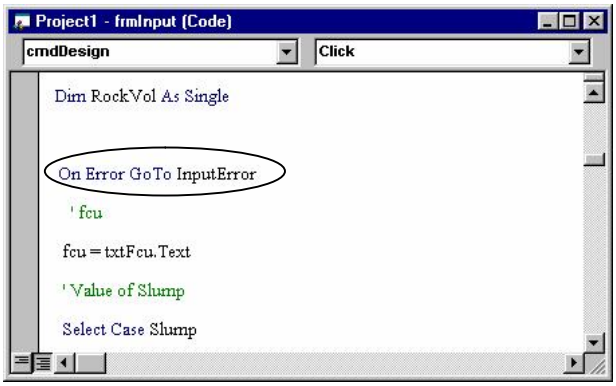
สำหรับข้อผิดพลาดในขณะรันโปรแกรมนั้น ถ้าเป็นการป้อนข้อมูลจากผู้ใช้ แน่นอนว่าเราไม่สามารถบังคับให้ผู้ใช้ป้อนข้อมูลได้ถูกต้องทุกครั้ง แต่เราสามารถเตือนเมื่อผู้ใช้ป้อนข้อมูล

ผิดพลาดได้ พร้อมกับยังสามารถกำหนดให้ผู้ใช้ต้องป้อนข้อมูลให้ถูกต้องก่อน โปรแกรมจึงจะทำงานอย่างอื่นได้ โดยการใช้คำสั่ง

```
On Error Goto[LineNumber]
```

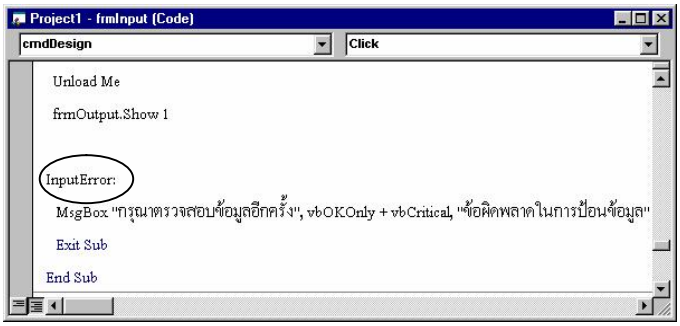
มีขั้นตอนการเขียนดังนี้

1. เขียนคำสั่งนี้ไว้ก่อนการรับข้อมูล ซึ่ง LineNumber หมายถึง ชื่อที่ตั้งขึ้น สำหรับเมื่อเกิดข้อผิดพลาด ให้โปรแกรมกระโดดข้ามคำสั่งอื่น แล้วไปที่บรรทัดชื่อนี้ทันที ตัวอย่างดังรูปที่ 7-5



รูปที่ 7-5 ตัวอย่างการเขียนคำสั่ง On Error Goto

2. เขียน LineNumber: ไว้ในส่วนท้ายของโปรแกรมย่อยเหตุการณ์ (เพื่อยังไม่ให้โปรแกรมทำงานในส่วนอื่นจนกว่าจะป้อนข้อมูลถูกต้อง) เช่นจากตัวอย่าง เราให้ชื่อ LineNumber ว่า InputError ดังนั้น จึงเขียนตามตัวอย่างดังรูปที่ 7-6



รูปที่ 7-6 ตัวอย่างการเขียนชื่อ LineNumber (ซึ่งในที่นี้คือ InputError)

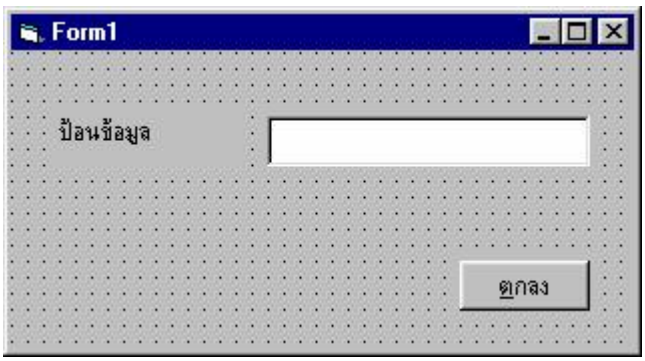
3. เมื่อเกิดข้อผิดพลาดในการป้อนข้อมูล โปรแกรมจะกระโดดมาที่บรรทัด InputError: ซึ่งบรรทัดถัดมาเป็นการสั่งให้โปรแกรมแจ้งข้อผิดพลาด (MsgBox) และออกจาก Sub นี้ด้วยคำสั่ง Exit Sub (ตัวอย่างข้างต้น สามารถละคำสั่ง Exit Sub ได้เพราะบรรทัดถัดไปคือ End Sub หมายถึงออกจาก Sub อยู่แล้ว แต่ในการทำงานจริง บางครั้ง LineNumber ไม่ได้อยู่ท้ายสุดของ Sub เสมอไป จึงต้องออกจากโปรแกรมโดยใช้คำสั่งนี้ เป็นการออกโดยข้ามคำสั่งอื่นๆ ที่อยู่ในบรรทัดถัดไป)

ตัวอย่างการทำงานของโปรแกรมเมื่อใส่คำสั่งจัดการกับข้อผิดพลาดเข้าไป แสดงดังรูปที่ 7-7

รูปที่ 7-7 ตัวอย่างการแสดงผลของคำสั่งแสดงข้อผิดพลาด

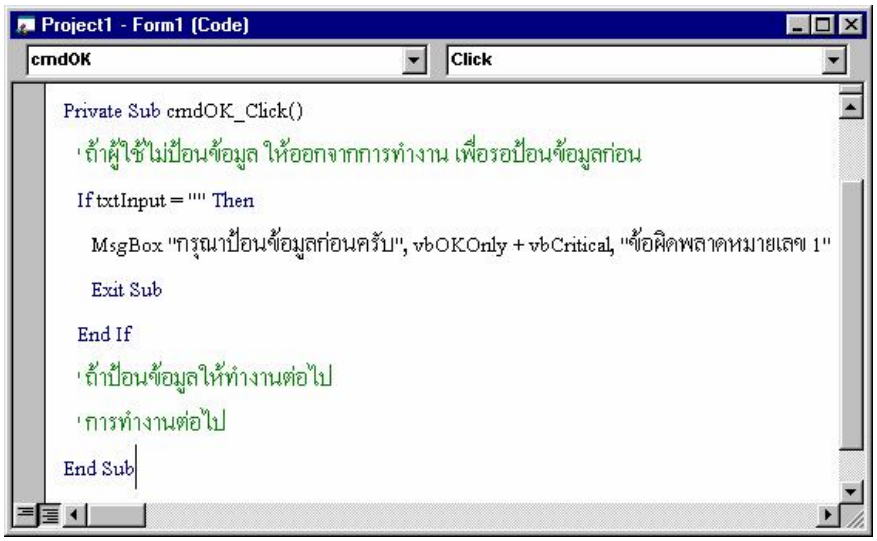
สังเกตว่า ข้อผิดพลาดจากรูปด้านบน เกิดจากการป้อนข้อมูลที่ต้องป้อนเป็นตัวเลข แต่ใส่เป็นตัวอักษร ซึ่งผิดพลาดชัดเจน แต่บางครั้งข้อผิดพลาดไม่ชัดเจน คือ ไม่ถึงกับทำให้โปรแกรมต้องหยุดการทำงาน แต่อาจทำให้การแสดงผลลัพธ์ไม่สมบูรณ์ เช่น กรณีป้อนข้อมูลเป็นข้อความ แต่ผู้ใช้ไม่ใส่ตัวอักษรอะไรเลย กรณีนี้โปรแกรมจะไม่แจ้งข้อผิดพลาด เพราะข้อมูลแบบตัวอักษร คือ String นี้ การไม่ป้อนข้อมูลหมายถึงข้อความว่าง แต่ถ้าโปรแกรมสามารถเตือนได้ว่า ผู้ใช้ไม่ได้ป้อนข้อมูล ก็จะทำให้โปรแกรมนี้สมบูรณ์ขึ้น สามารถทำได้ตามตัวอย่างดังนี้

สมมุติมีช่องรับข้อมูลแบบ TextBox (คุณสมบัติ Name คือ txtInput) รอผู้ใช้ป้อนข้อมูล แล้วคลิกปุ่มตกลง (คุณสมบัติ Name คือ cmdOK) ดังรูปที่ 7-8



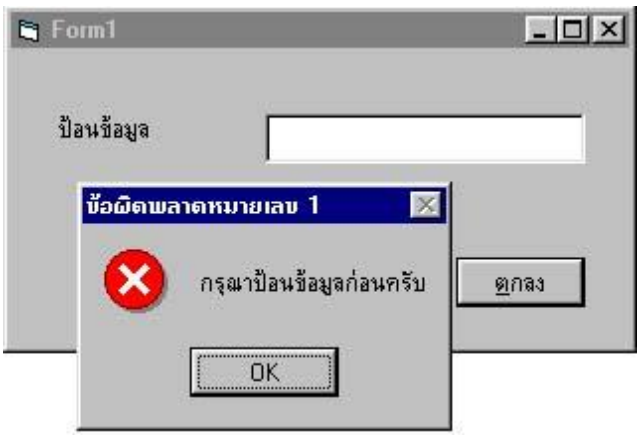
รูปที่ 7-8 ตัวอย่างฟอร์มที่จะแสดงการเขียนคำสั่งแจ้งเตือนการป้อนข้อมูล

กำหนดให้ถ้าผู้ใช้ไม่ป้อนข้อมูลใดๆ ให้แจ้งเตือน แล้วยังไม่ทำงานส่วนอื่น รอจนกว่าผู้ใช้จะป้อนข้อมูล เขียนคำสั่งในโปรแกรมย่อยเหตุการณ์คลิกปุ่มตกลง ดังรูปที่ 7-9



รูปที่ 7-9 เขียนคำสั่งแจ้งเตือนการป้อนข้อมูลในโปรแกรมย่อยเหตุการณ์คลิกปุ่มตกลง

การทำงานของโปรแกรมคือ เมื่อผู้ใช้ไม่ป้อนข้อมูลใดๆ จะเกิดข้อความเตือน แสดงดังรูปที่ 7-10



รูปที่ 7-10 การแสดงผลตามคำสั่งแจ้งเตือนการป้อนข้อมูล

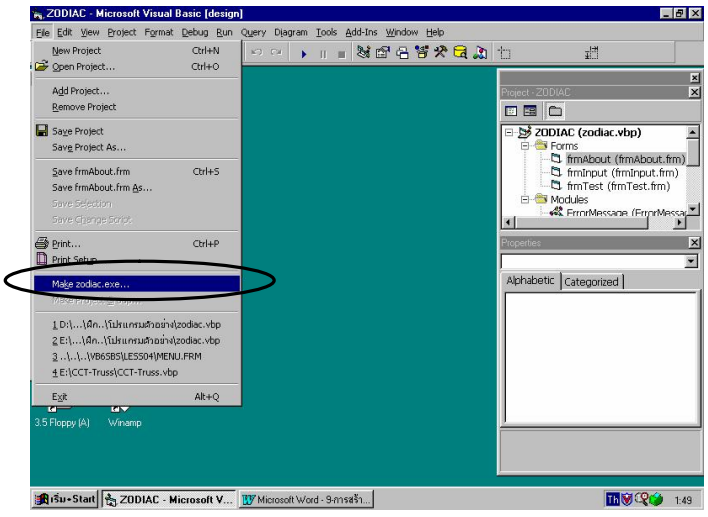
บทที่ 8 การสร้างตัวติดตั้งให้กับโปรแกรม

เมื่อเขียนโปรแกรมเสร็จเรียบร้อยแล้ว ต่อมาคือขั้นตอนการทำให้เป็นโปรแกรมสำเร็จ คือ คอมไพล์ (compile) โปรแกรมให้เป็นแบบ Executable File (นามสกุล .EXE) ทำให้ไม่ต้องทำงานภายใต้สภาพแวดล้อมของวิชวลเบสิก สามารถรันที่ไฟล์นามสกุลนี้ได้เลย แต่หากเราจะทำการเผยแพร่ไฟล์นี้ให้กับผู้ใช้คนอื่น (เครื่องคอมพิวเตอร์เครื่องอื่น) โปรแกรมนี้จะยังไม่สามารถทำงานได้ เนื่องจากจะต้องมีไฟล์ช่วยอื่นๆ เช่น ไฟล์ที่มีนามสกุล .DLL ร่วมไปด้วย จึงจะต้องทำการสร้างตัวติดตั้ง (setup) ให้กับโปรแกรม ซึ่งวิชวลเบสิกจัดเตรียมการทำงานส่วนนี้ไว้แล้ว มีรายละเอียดดังนี้

การคอมไพล์โปรแกรม

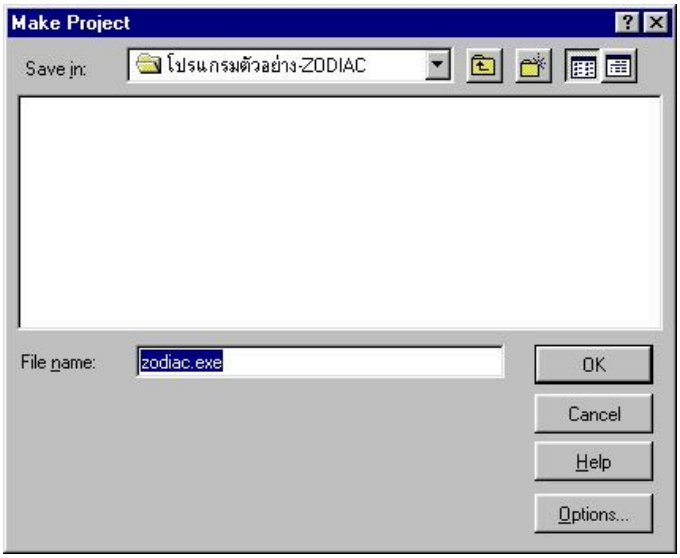
มีขั้นตอนคือ

1. คลิกที่เมนู File แล้วไปที่เมนูย่อย Make <ชื่อโปรเจ็กต์>.exe ตัวอย่างดังรูปที่ 8-1



รูปที่ 8-1 เมนูย่อย Make

2. เมื่อคลิกแล้ว จะปรากฏหน้าจอแสดงชื่อไฟล์ที่จะคอมไพล์เป็นนามสกุล .EXE ซึ่งปกติจะใช้ชื่อเดียวกับไฟล์โปรเจ็กต์ ตัวอย่างดังรูปที่ 8-2

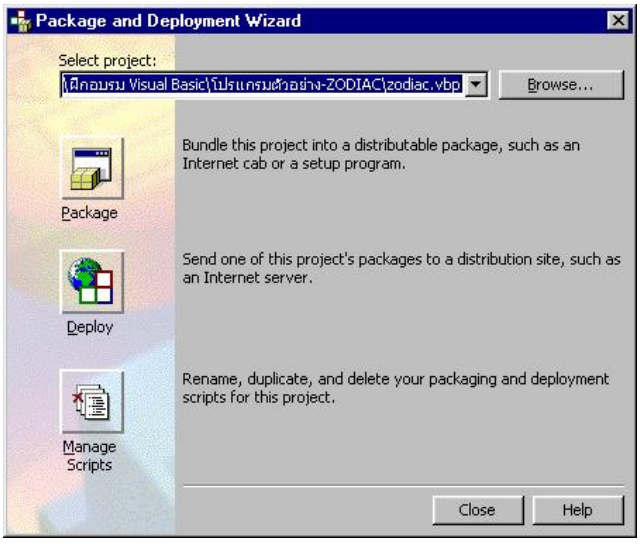


รูปที่ 8-2 กำหนดชื่อไฟล์ที่จะคอมไพล์

3. คลิกที่ปุ่ม OK เป็นการเสร็จขั้นตอนการคอมไพล์โปรแกรม

การสร้างตัวติดตั้ง

1. เรียกใช้ Package & Deployment Wizard
2. เมื่อเข้าไปแล้ว หน้าจอแรกที่ปรากฏ แสดงดังรูปที่ 8-3

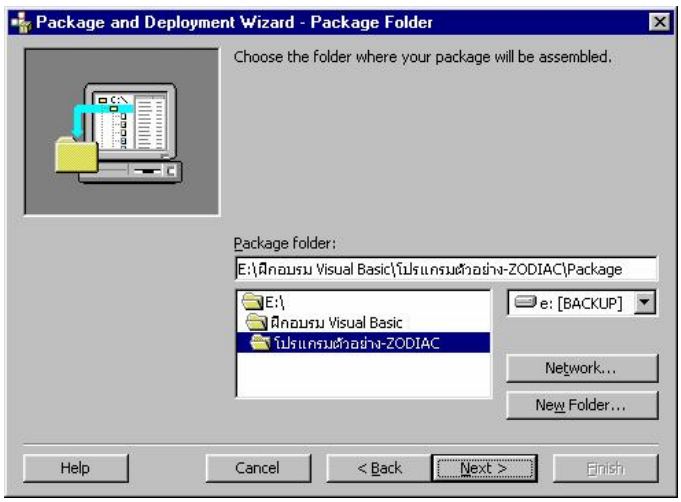


รูปที่ 8-3 หน้าจอแรกที่ปรากฏใน Package & Deployment Wizard

3. คลิกเลือกไฟล์โปรเจ็กต์ที่จะสร้างแผ่น Setup ที่ช่อง Browse หลังจากนั้น คลิกที่ปุ่ม Package จะปรากฏหน้าจอใหม่ดังรูปที่ 8-4

รูปที่ 8-4 หน้าจอที่ปรากฏ หลังจากคลิกปุ่ม Package

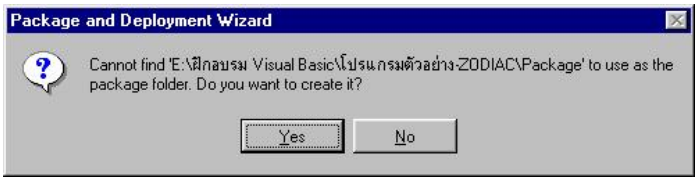
4. คลิกที่ปุ่ม Next จะปรากฏหน้าจอใหม่ ดังรูปที่ 8-5



รูปที่ 8-5 หน้าจอที่ปรากฏ หลังจากคลิกปุ่ม Next

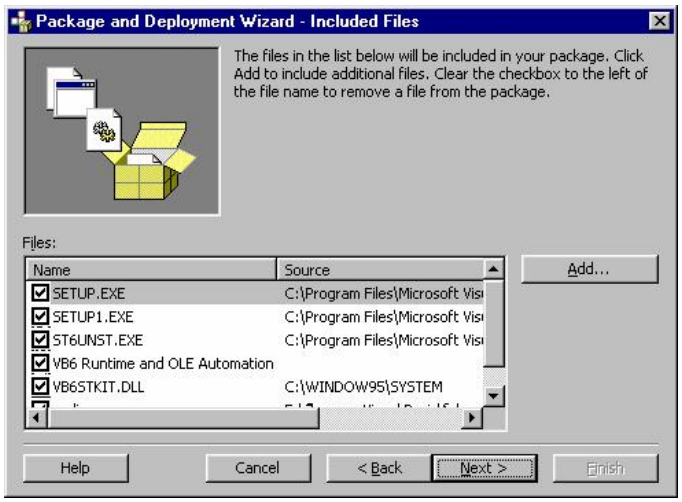
หน้าจอนี้ เป็นการเลือกเก็บกลุ่มไฟล์ Setup ที่ Wizard จะทำให้ไว้ที่โฟลเดอร์ใด ซึ่งปกติจะเก็บไว้ที่โฟลเดอร์ย่อยชื่อว่า Package ของโฟลเดอร์ที่เราสร้างโปรเจ็กต์นั้นขึ้นมา แต่เราสามารถ

เปลี่ยนแปลงได้ในหน้าจอนี้ เมื่อเลือกได้แล้ว คลิกที่ปุ่ม Next ซึ่งถ้าเป็นการเลือกโฟลเดอร์ตามที่ Wizard กำหนด จะปรากฏข้อความดังรูปที่ 8-6



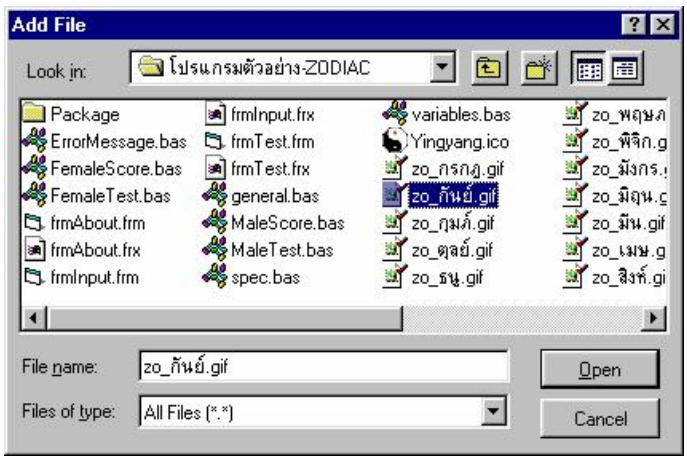
รูปที่ 8-6 ข้อความที่ปรากฏ หลังจากคลิกปุ่ม Next

ข้อความนี้เกิดขึ้นเพราะโฟลเดอร์ชื่อ Package นี้ ยังไม่มีอยู่ จึงถามเราว่าจะสร้างขึ้นมาหรือไม่ คลิกที่ปุ่ม Yes จะปรากฏหน้าจอใหม่ ดังรูปที่ 8-7



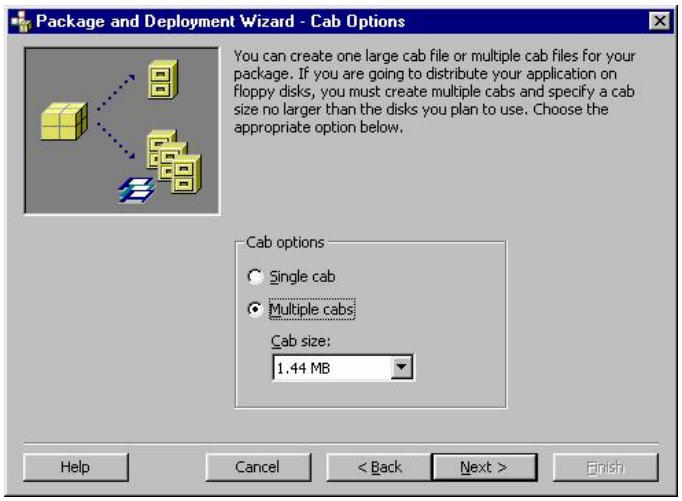
รูปที่ 8-7 หน้าจอที่ปรากฏ หลังจากคลิกปุ่ม Yes

5. หน้าจอนี้ จะแสดงกลุ่มไฟล์ Setup ทั้งหมดที่ใช้ในการทำแผ่น Setup ซึ่งแบ่งเป็น 2 กลุ่มคือ ไฟล์ Setup ใช้ในการติดตั้งโปรแกรมไปที่เครื่องอื่น กับ ไฟล์โปรแกรม ซึ่งจะประกอบด้วยไฟล์นามสกุล .EXE และไฟล์ไลบรารีต่างๆ ที่จำเป็น (ดังนั้น โปรแกรมที่รันบนวินโดวส์จึงเรียกว่า “แอพพลิเคชัน” เพราะเกิดจากการนำไฟล์มากกว่าหนึ่งไฟล์มาทำงานร่วมกัน) ซึ่งในทางปฏิบัติแล้ว อาจมีบางไฟล์ที่เราต้อง Add เข้าไปเพิ่มเติมเอง ในกรณีที่โปรแกรมที่เราสร้างขึ้นจำเป็นต้องมีไฟล์รูปแบบอื่น เช่น ไฟล์ข้อมูล ไฟล์ภาพ เป็นต้น ทำได้โดยคลิกที่ปุ่ม Add แล้วเลือกไฟล์นั้น ตัวอย่างดังรูปที่ 8-8



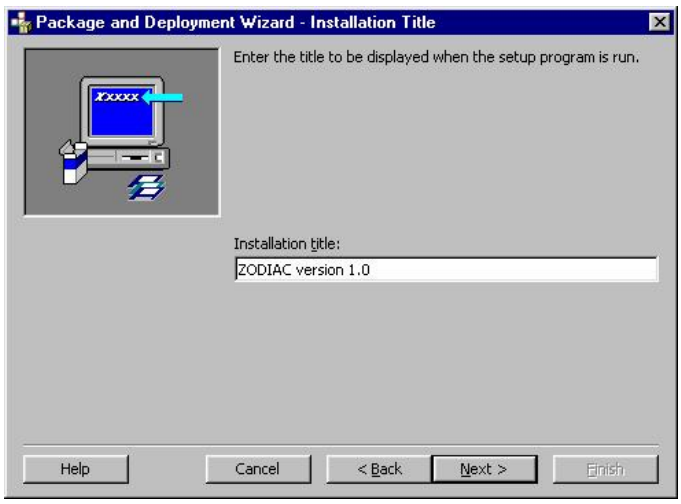
รูปที่ 8-8 การเลือกไฟล์เพิ่มเติม

6. หลังจาก Add เสร็จแล้ว จึงคลิกปุ่ม Next ซึ่งจะปรากฏหน้าจอใหม่ ดังรูปที่ 8-9



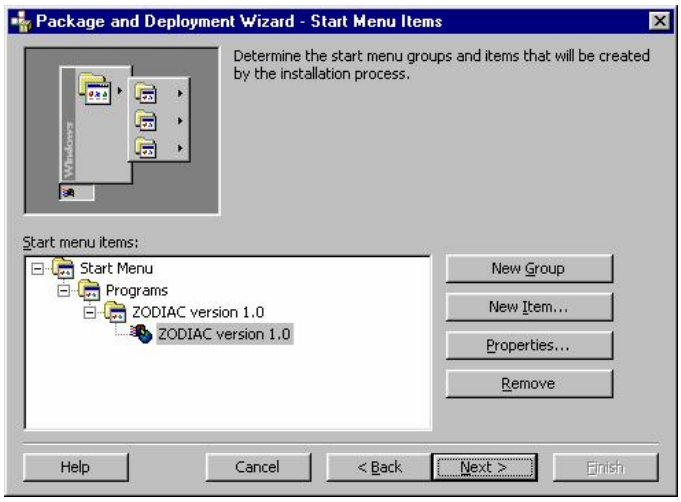
รูปที่ 8-9 หน้าจอที่ปรากฏ หลังจากเพิ่มเติมไฟล์ (ถ้ามี) แล้วคลิกปุ่ม Next

หน้าจอนี้เป็นทางเลือกว่าจะทำไฟล์ Setup แบบใส่แผ่นซีดีรอม คือใช้ Single Cab หรือแบบใส่แผ่นฟล็อปปีดิสก์ คือใช้ Multiple Cab เมื่อเลือกแล้วจึงคลิกปุ่ม Next จะปรากฏหน้าจอใหม่ ดังรูปที่ 8-10



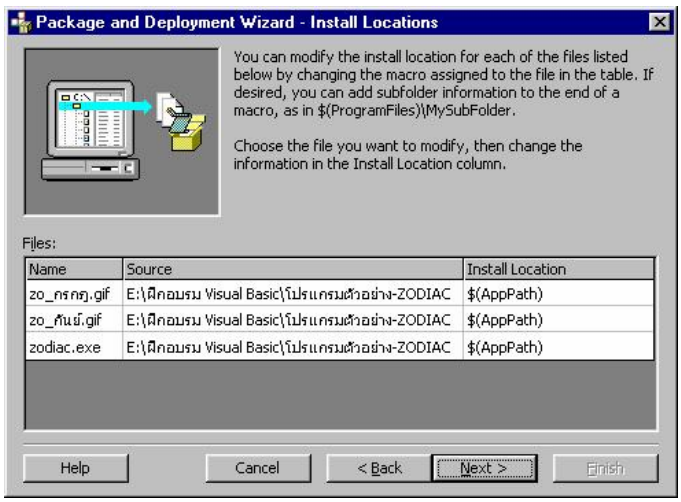
รูปที่ 8-10 หน้าจอที่ปรากฏ หลังจากเลือกรูปแบบการทำแผ่น Setup แล้วคลิกปุ่ม Next

7. หน้าจอนี้เป็นการกำหนดข้อความ ที่จะปรากฏขณะทำการติดตั้งโปรแกรม เมื่อกำหนดแล้ว คลิกที่ปุ่ม Next จะปรากฏหน้าจอใหม่ดังรูปที่ 8-11



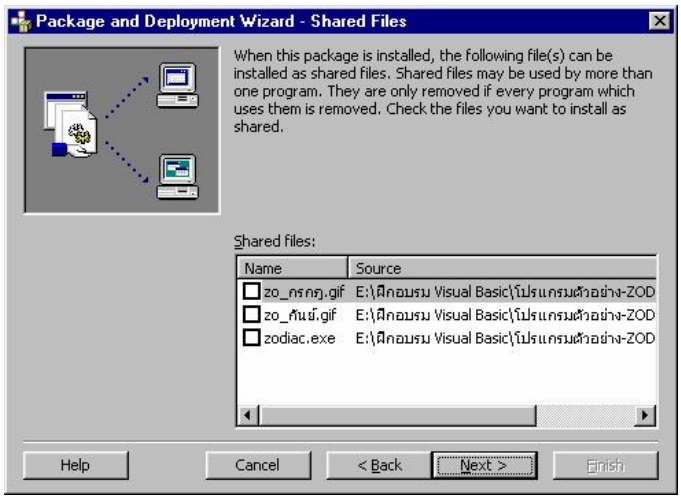
รูปที่ 8-11 หน้าจอที่ปรากฏ หลังจากกำหนดข้อความแล้วคลิกปุ่ม Next

8. หน้าจอนี้คือการกำหนดตำแหน่งเมนูย่อยของโปรแกรมที่จะแสดงในเมนู Start ของวินโดวส์ ปกติเราก็จะไม่แก้ไขตรงส่วนนี้ ต่อไปคลิกที่ปุ่ม Next จะปรากฏหน้าจอใหม่ดังรูปที่ 8-12



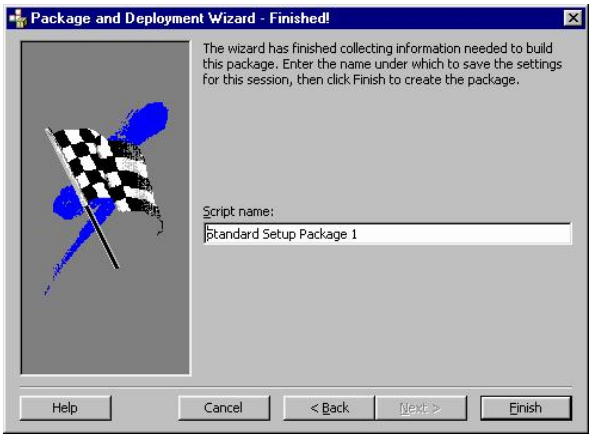
รูปที่ 8-12 หน้าจอที่ปรากฏ หลังจากแก้ไขตำแหน่ง (หรือไม่แก้ไขก็ได้) แล้วคลิกปุ่ม Next

9. หน้าจอนี้แสดงตำแหน่งของไฟล์เมื่อติดตั้งลงในเครื่อง (Install Location) ซึ่งปกติเราก็จะไม่แก้ไขตรงส่วนนี้ ต่อไปคลิกที่ปุ่ม Next จะปรากฏหน้าจอใหม่ดังรูปที่ 8-13



รูปที่ 8-13 หน้าจอต่อมา หลังจากคลิกปุ่ม Next

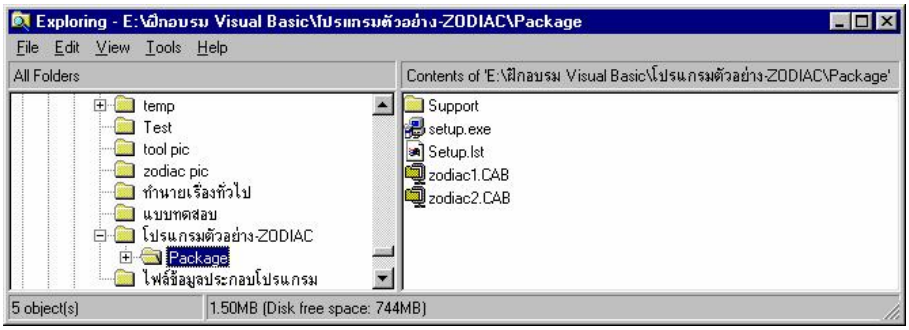
10. หน้าจอนี้สำหรับกำหนด (หรือไม่กำหนดก็ได้) ไฟล์ที่ให้โปรแกรมอื่นสามารถเรียกใช้ได้ด้วย (Shared Files) ต่อไปคลิกที่ปุ่ม Next จะปรากฏหน้าจอใหม่ดังรูปที่ 8-14



รูปที่ 8-14 หน้าจอที่ปรากฏ หลังจากเลือกหรือไม่เลือก Shared Files แล้วคลิกปุ่ม Next

11. หน้าจอนี้แสดงว่าการทำตัวติดตั้งเสร็จเรียบร้อย คลิกปุ่ม Finish เพื่อออกจาก Wizard

หลังจากนั้น ถ้าเราเข้าไปที่โฟลเดอร์ย่อย Package จะเห็นว่ามีไฟล์ชื่อ Setup 2 นามสกุล และไฟล์ (ที่ถูกบีบอัด) นามสกุล .CAB ซึ่งจะเป็นไฟล์ทั้งหมดที่เป็นไฟล์ส่วนตัวติดตั้งโปรแกรม ส่วนโฟลเดอร์ย่อย Support เป็นไฟล์ทั้งหมดที่นำมาบีบอัด คล้ายกับเป็นการสำรองไฟล์ ซึ่งไม่ใช่ส่วนตัวติดตั้ง จึงไม่ต้องนำไปไว้ในแผ่นซีดีหรือแผ่นดิสก์ที่ทำเป็นแผ่น Setup ดังรูปที่ 8-15



รูปที่ 8-15 ไฟล์และโฟลเดอร์ ในโฟลเดอร์ Package

ในกรณีของการทำแผ่น Setup ไว้ในแผ่นดิสก์ ไฟล์ setup.exe และ setup.lst จะต้องอยู่ที่ดิสก์แผ่นที่ 1 ส่วนไฟล์นามสกุล .CAB จะเป็นสิ่งที่บ่งบอกจำนวนแผ่น Setup ทั้งหมด โดยไฟล์ชื่อที่ลงท้ายด้วย 1 ก็จะอยู่ที่แผ่นที่ 1 ลงท้ายด้วยตัวเลขอื่นก็อยู่ที่แผ่นดิสก์ที่เท่านั้น (สังเกตว่า ไฟล์ที่บรรจุในดิสก์แต่ละแผ่น จะมีความจุไม่เกิน 1.38 MB) เช่นตัวอย่างดังรูปที่ 8-15 แสดงว่ามีแผ่น Setup จำนวน 2 แผ่น เป็นต้น

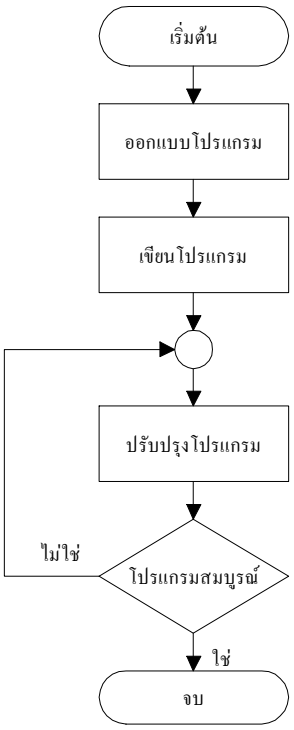
ภาคที่ 2
ปฏิบัติการสร้างโปรแกรม

บทที่ 9 ตัวอย่างการสร้างโปรแกรมออกแบบส่วนผสมคอนกรีต

ในบทนี้ จะแสดงขั้นตอนการสร้างโปรแกรมตั้งแต่เริ่มต้นออกแบบ จนถึงโปรแกรมเสร็จสมบูรณ์ พร้อมทั้งแสดงการสร้างตัวติดตั้งให้กับโปรแกรมด้วย โดยใช้ตัวอย่างคือ สร้างโปรแกรมออกแบบส่วนผสมคอนกรีต และให้ชื่อโปรแกรมว่า MixedDesign ซึ่งก่อนที่จะเข้าสู่ตัวอย่าง จะอธิบายถึงขั้นตอนการสร้างโปรแกรมเบื้องต้น ดังนี้

สรุปขั้นตอนการสร้างโปรแกรม

การสร้างโปรแกรม มีขั้นตอนหลัก ดังแผนภูมิที่ 9-1



แผนภูมิที่ 9-1 ขั้นตอนหลักการสร้างโปรแกรม

สรุปขั้นตอนการออกแบบโปรแกรม

สามารถใช้ขั้นตอนดังนี้

- กำหนดวัตถุประสงค์

  กำหนดวัตถุประสงค์ของโปรแกรม ซึ่งควรมีลักษณะ

  1. ประโยคกระทัดรัดได้ใจความ

2. กำหนดลักษณะที่จะทำ และเจาะจง
3. ไม่ควรกำหนดในลักษณะกว้างมากเกินไป
4. ไม่ควรกำหนดรายละเอียดมากเกินไป

- ข้อมูลนำเข้าและผลลัพธ์

  มีข้อมูลนำเข้า หรือ อินพุต (input) และผลลัพธ์ หรือ เอาต์พุต (output) อะไรบ้าง

- ออกแบบส่วนติดต่อกับผู้ใช้

  ส่วนผู้ใช้ป้อนข้อมูล ส่วนแสดงผลลัพธ์ และส่วนแสดงรายละเอียดอื่นๆ รวมทั้งลูกเล่นต่างๆ ของโปรแกรม (ถ้ามี)

- พัฒนาอัลกอริทึม

  พัฒนาอัลกอริทึมในส่วนของขั้นตอนการประมวลผลสำหรับงานนั้น โดยอาจเขียนอยู่ในรูปลำดับขั้นตอน หรือผังงาน

สรุปขั้นตอนการเขียนโปรแกรม

มีขั้นตอนหลักดังนี้

- สร้างส่วนติดต่อกับผู้ใช้
- เขียนรหัสคำสั่ง

ออกแบบโปรแกรม *MixedDesign*

- วัตถุประสงค์ของโปรแกรม

  ออกแบบส่วนผสมผสมคอนกรีต คือ ปูนซีเมนต์ น้ำ ทราย และหิน ในหนึ่งลูกบาศก์เมตร ตามข้อมูลที่ผู้ใช้กำหนด โดยใช้วิธีของมาตรฐานว.ส.ท. 1014-40 “ข้อกำหนดมาตรฐานวัสดุและการก่อสร้าง สำหรับโครงสร้างคอนกรีต” ผู้อ่านสามารถดูรายละเอียดเพิ่มเติมได้ในบทที่ 3 ส่วนผสมคอนกรีต

- ข้อมูลนำเข้าและผลลัพธ์

  ข้อมูลนำเข้า

  กำลังอัดทรงกระบอกของคอนกรีต (f_c')

  ค่ายุบตัวของคอนกรีต

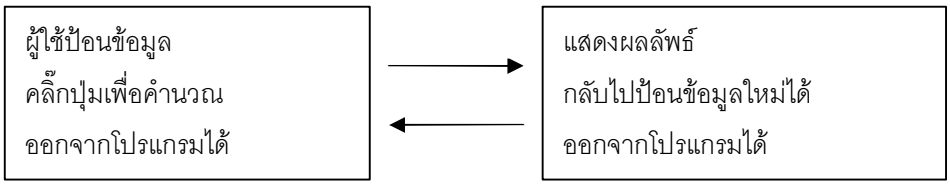
  ขนาดใหญ่สุดของหิน

  ผลลัพธ์

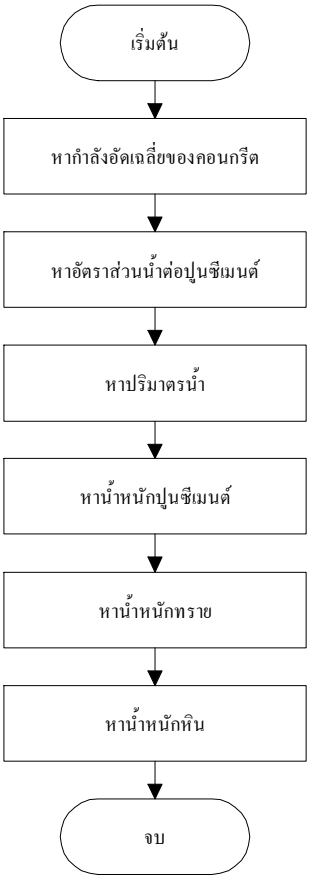
  น้ำหนักปูนซีเมนต์

ปริมาตรน้ำ
น้ำหนักทราย
ขนาดใหญ่สุดของหิน และน้ำหนักหิน

- ออกแบบส่วนติดต่อกับผู้ใช้



- ผังงานการคำนวณหาส่วนผสมคอนกรีต แสดงดังแผนภูมิที่ 9-2



แผนภูมิที่ 9-2 ผังงานการคำนวณหาส่วนผสมคอนกรีต

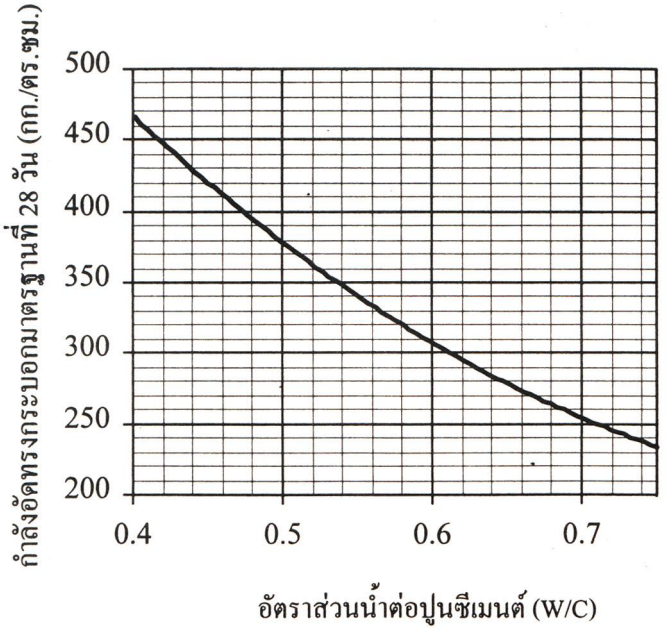
คำอธิบายเพิ่มเติม

- หากำลังอัดเฉลี่ยคอนกรีต (fcr) ได้จากตารางที่ 9-1

ตารางที่ 9-1 กำลังเฉลี่ยคอนกรีต กรณีที่ไม่มีข้อมูลกำลังอัดคอนกรีต

fc' (ksc)	fcr (ksc)
น้อยกว่า 210	$fc' + 70$
210 – ต่ำกว่า 350	$fc' + 85$
350 – 450	$fc' + 100$

- หาอัตราส่วนน้ำต่อปูนซีเมนต์ ตามมาตรฐานแล้ว หาได้จากกราฟ ดังรูปที่ 9-1



รูปที่ 9-1 ความสัมพันธ์ของอัตราส่วนน้ำต่อปูนซีเมนต์และกำลังอัดทรงกระบอกมาตรฐานที่ 28 วัน

แต่ในกรณีของการเขียนโปรแกรมแล้ว เราจะต้องแปลงกราฟให้เป็นรูปสมการ เพื่อให้คอมพิวเตอร์สามารถคำนวณได้ ซึ่งอาศัยวิธีการสร้างสมการเส้นถดถอย (regression equation) สามารถจัดรูปสมการได้ดังนี้

$$wc = a(fcr)^3 + b(fcr)^2 + c(fcr) + d$$

โดยที่ wc = อัตราส่วนน้ำต่อปูนซีเมนต์

fcr = กำลังอัดเฉลี่ยคอนกรีต (ksc)

a = -2×10^{-8}

b = 2.3571×10^{-5}

c = -0.0104

d = 2.14914286

- หาปริมาณน้ำ ได้จากตารางที่ 9-2

ตารางที่ 9-2 ปริมาณน้ำที่เหมาะสมในส่วนผสมคอนกรีต

ค่ายุบตัว (cm)	**ปริมาณน้ำต่อคอนกรีตหนึ่งลูกบาศก์เมตร (ลิตร)**	
	หินขนาดใหญ่สุด 25 mm	**หินขนาดใหญ่สุด** 20 mm
7.5	170	180
10.0	180	190
12.5	190	200
15.0	200	210

- หาน้ำหนักปูนซีเมนต์

น้ำหนักปูนซีเมนต์ (kg) = น้ำหนักน้ำ/อัตราส่วนน้ำต่อปูนซีเมนต์

หมายเหตุ : น้ำหนักน้ำ = ปริมาตรน้ำ

- หาน้ำหนักทราย

น้ำหนักทราย (kg) = ปริมาตรทราย x ความถ่วงจำเพาะของทราย (ใช้ค่า 2.65)

โดยที่

ปริมาตรทราย = ปริมาณส่วนละเอียด – น้ำหนักปูนซีเมนต์/ความถ่วงจำเพาะของปูนซีเมนต์

= ปริมาณส่วนละเอียด – น้ำหนักปูนซีเมนต์/3.15

ปริมาณส่วนละเอียด หาได้จากตารางที่ 9-3

ตารางที่ 9-3 ปริมาณส่วนละเอียดในคอนกรีตหนึ่งลูกบาศก์เมตร

ขนาดโตสุดของหิน (mm)	ปริมาณส่วนละเอียด (ลิตร)
25	380
20	400

- หาน้ำหนักหิน

  น้ำหนักหิน (kg) = ปริมาตรหิน x ความถ่วงจำเพาะ (ใช้ค่า 2.70)

  โดยที่

  ปริมาตรหิน = ปริมาตรคอนกรีต – ปริมาตรน้ำ – ปริมาตรปูนซีเมนต์ – ปริมาตรทราย – ปริมาตรอากาศ

  = 1000 – ปริมาตรน้ำ – ปริมาณส่วนละเอียด – 10

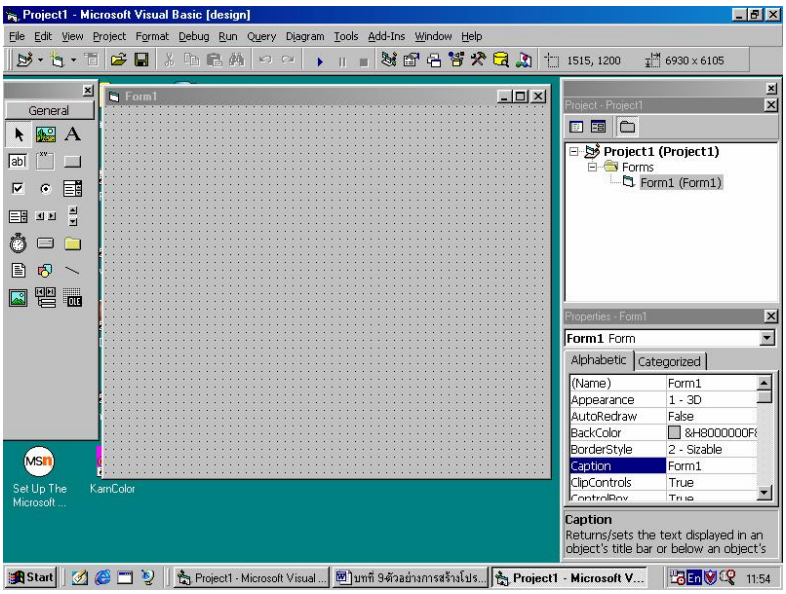
  = 990 – ปริมาตรน้ำ – ปริมาณส่วนละเอียด

เขียนโปรแกรม MixedDesign

❑ สร้างส่วนติดต่อกับผู้ใช้

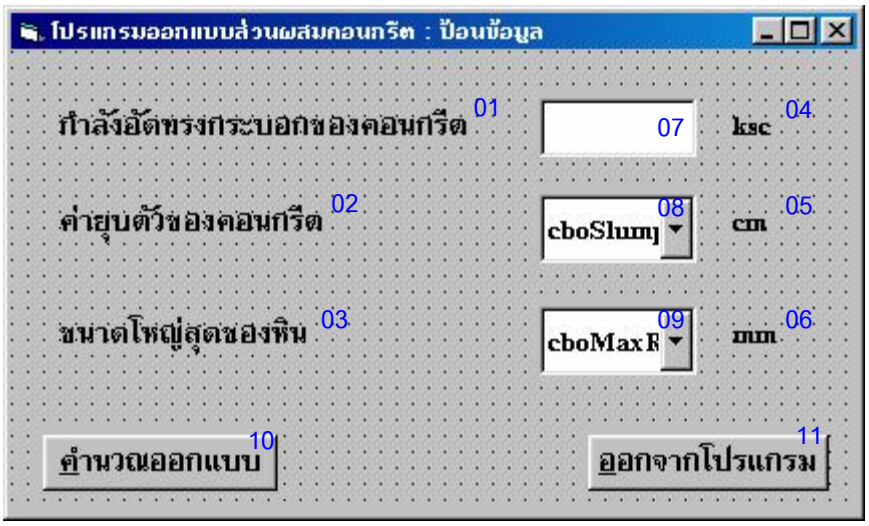
มีขั้นตอนดังนี้

1. เมื่อเปิดโปรแกรมวิชวลเบสิก แล้วสร้างโปรเจ็กต์ใหม่แบบ Standard EXE หน้าจอจะแสดงผลดังรูปที่ 9-2



รูปที่ 9-2 สร้างโปรเจ็กต์ใหม่แบบ Standard EXE

2. ใส่คอนโทรลลงไปในฟอร์มเริ่มต้นนี้ ซึ่งเป็นส่วนผู้ใช้ป้อนข้อมูล ดังรูปที่ 9-3



รูปที่ 9-3 รายละเอียดของฟอร์มส่วนผู้ใช้ป้อนข้อมูล

3. จากรูปที่ 9-3 มีการกำหนดคุณสมบัติของฟอร์ม และแต่ละคอนโทรล ดังตารางที่ 9-4 และ 9-5

ตารางที่ 9-4 กำหนดคุณสมบัติของฟอร์มส่วนผู้ใช้ป้อนข้อมูล

คุณสมบัติ	**กำหนดค่า**
Name	frmInput
BorderStyle	1 – Fixed Single
Caption	โปรแกรมออกแบบส่วนผสมคอนกรีต : ป้อนข้อมูล
Height	3870
MaxButton	False
Width	6450

ตารางที่ 9-5 กำหนดคุณสมบัติของคอนโทรลในฟอร์ม frmInput

เลขที่	**คอนโทรล**	**คุณสมบัติ**	**กำหนดค่า**
01	Label	Caption Font	กำลังอัดทรงกระบอกของคอนกรีต EucrosiaUPC (Font), Bold (Font style), 16 (size)

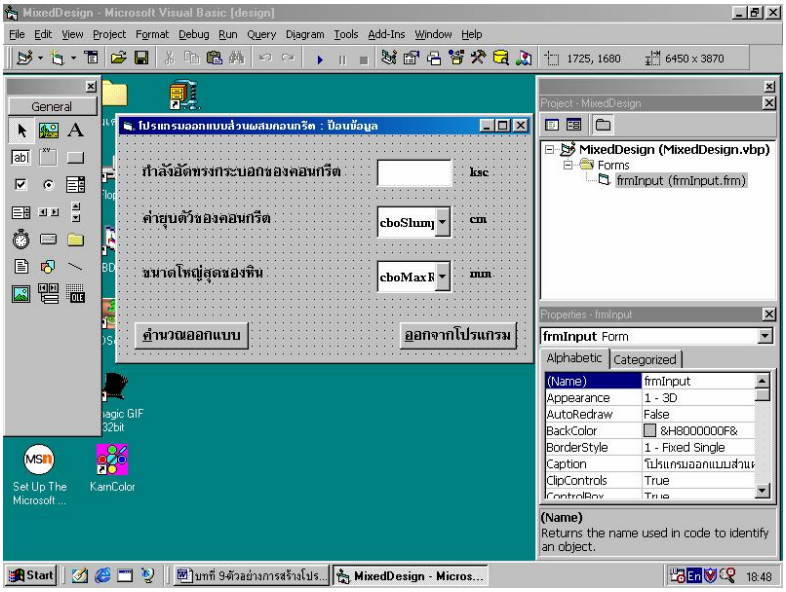
ตารางที่ 9-5 (ต่อ)

เลขที่	คอนโทรล	คุณสมบัติ	กำหนดค่า
02	Label	Caption	ค่ายุบตัวของคอนกรีต
		Font	EucrosiaUPC, Bold, 16
03	Label	Caption	ขนาดใหญ่สุดของหิน
		Font	EucrosiaUPC, Bold, 16
04	Label	Caption	ksc
		Font	EucrosiaUPC, Bold, 16
05	Label	Caption	cm
		Font	EucrosiaUPC, Bold, 16
06	Label	Caption	mm
		Font	EucrosiaUPC, Bold, 16
07	TextBox	Name	txtFcu
		Font	EucrosiaUPC, Bold, 16
		Text	——(ไม่ใส่ข้อความ)
08	ComboBox	Name	cboSlump
		Font	EucrosiaUPC, Bold, 16
		List	7.5, 10.0, 12.5, 15.0
		Style	2 – Dropdown List
09	ComboBox	Name	cboMaxRock
		Font	EucrosiaUPC, Bold, 16
		List	20, 25
		Style	2 – Dropdown List
10	CommandButton	Name	cmdDesign
		Caption	&คำนวณออกแบบ
		Font	EucrosiaUPC, Bold, 16

ตารางที่ 9-5 (ต่อ)

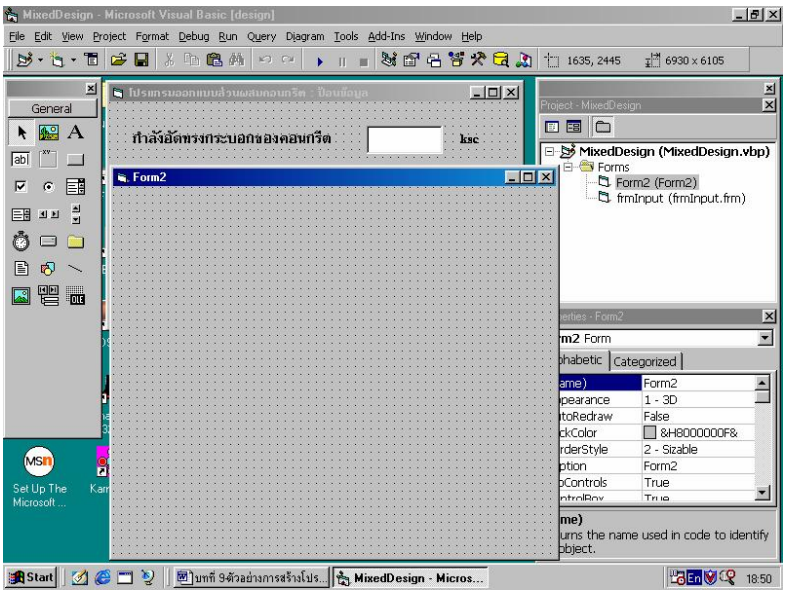
เลขที่	คอนโทรล	คุณสมบัติ	กำหนดค่า
11	CommandButton	Name	cmdExit
		Caption	&ออกจากโปรแกรม
		Font	EucrosiaUPC, Bold, 16

4. เซฟไฟล์ฟอร์มชื่อว่า frmInput กำหนด Name ของโปรเจ็กต์และเซฟไฟล์ในชื่อเดียวกันว่า MixedDesign ซึ่งได้ดังรูปที่ 9-4 (แนะนำให้สร้างโฟลเดอร์เพื่อเก็บไฟล์ต่างๆ เพราะในการเขียนโปรแกรมด้วยวิชวลเบสิก จะมีไฟล์มากกว่าหนึ่งไฟล์)



รูปที่ 9-4 เซฟไฟล์ฟอร์ม frmInput และโปรเจ็คต์ MixedDesign

5. เพิ่มฟอร์มใหม่โดยไปที่เมนู Project\Add Form เสร็จแล้วจะได้ฟอร์มใหม่ดังรูปที่ 9-5



รูปที่ 9-5 เพิ่มฟอร์มใหม่

6. ใส่คอนโทรลลงไปในฟอร์มใหม่นี้ ซึ่งเป็นส่วนแสดงผลลัพธ์ ดังรูปที่ 9-6

รูปที่ 9-6 รายละเอียดของฟอร์มส่วนแสดงผลลัพธ์

7. จากรูปที่ 9-6 มีการกำหนดคุณสมบัติของฟอร์ม และแต่ละคอนโทรล ดังตารางที่ 9-6 และ 9-7

ตารางที่ 9-6 กำหนดคุณสมบัติของฟอร์มส่วนแสดงผลลัพธ์

คุณสมบัติ	กำหนดค่า
Name	frmOutput
BorderStyle	1 – Fixed Single
Caption	โปรแกรมออกแบบส่วนผสมคอนกรีต : ผลลัพธ์
Height	4320
MaxButton	False
Width	6135

ตารางที่ 9-7 กำหนดคุณสมบัติของคอนโทรลในฟอร์ม frmOutput

เลขที่	คอนโทรล	คุณสมบัติ	กำหนดค่า
01	Label	Caption	ส่วนผสมในหนึ่งลูกบาศก์เมตร
		Font	EucrosiaUPC, Bold, 16
02	Label	Caption	ปูนซีเมนต์
		Font	EucrosiaUPC, Bold, 16
03	Label	Caption	น้ำ
		Font	EucrosiaUPC, Bold, 16
04	Label	Caption	ทราย
		Font	EucrosiaUPC, Bold, 16
05	Label	Caption	หิน (ขนาดใหญ่สุด
		Font	EucrosiaUPC, Bold, 16
06	Label	Caption	mm)
		Font	EucrosiaUPC, Bold, 16
07	Label	Caption	kg
		Font	EucrosiaUPC, Bold, 16

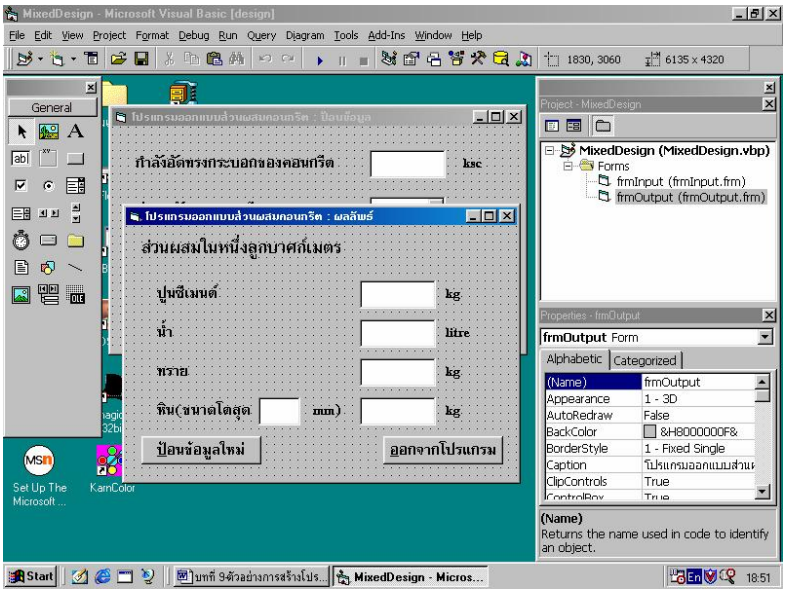
ตารางที่ 9-7 (ต่อ)

เลขที่	คอนโทรล	คุณสมบัติ	กำหนดค่า
08	Label	Caption	litre
		Font	EucrosiaUPC, Bold, 16
09	Label	Caption	kg
		Font	EucrosiaUPC, Bold, 16
10	Label	Caption	kg
		Font	EucrosiaUPC, Bold, 16
11	Label	Name	lblCement
		BackColor	&H00FFFFFF& (สีขาว)
		BorderStyle	1 – Fixed Single
		Caption	—— (ยังไม่ใส่ข้อความ)
		Font	EucrosiaUPC, Bold, 16
12	Label	Name	lblWater
		BackColor	&H00FFFFFF& (สีขาว)
		BorderStyle	1 – Fixed Single
		Caption	—— (ยังไม่ใส่ข้อความ)
		Font	EucrosiaUPC, Bold, 16
13	Label	Name	lblSand
		BackColor	&H00FFFFFF& (สีขาว)
		BorderStyle	1 – Fixed Single
		Caption	—— (ยังไม่ใส่ข้อความ)
		Font	EucrosiaUPC, Bold, 16
14	Label	Name	lblMaxRock
		BackColor	&H00FFFFFF& (สีขาว)
		BorderStyle	1 – Fixed Single
		Caption	—— (ยังไม่ใส่ข้อความ)
		Font	EucrosiaUPC, Bold, 16

ตารางที่ 9-7 (ต่อ)

เลขที่	คอนโทรล	คุณสมบัติ	กำหนดค่า
15	Label	Name	lblRock
		BackColor	&H00FFFFFF& (สีขาว)
		BorderStyle	1 – Fixed Single
		Caption	—— (ยังไม่ใส่ข้อความ)
		Font	EucrosiaUPC, Bold, 16
16	CommandButton	Name	cmdInput
		Caption	&ป้อนข้อมูลใหม่
		Font	EucrosiaUPC, Bold, 16
17	CommandButton	Name	cmdExit
		Caption	&ออกจากโปรแกรม
		Font	EucrosiaUPC, Bold, 16

8. เซฟไฟล์ฟอร์มนี้ชื่อว่า frmOutput ซึ่งเมื่อเสร็จถึงขั้นตอนนี้แล้ว จะได้ส่วนติดต่อกับผู้ใช้ทั้งหมดของโปรแกรมนี้ ดังรูปที่ 9-7



รูปที่ 9-7 ส่วนติดต่อกับผู้ใช้ทั้งหมดของโปรแกรม

- เขียนรหัสคำสั่ง

เขียนรหัสคำสั่งในโปรแกรมย่อยเหตุการณ์ต่างๆ ของฟอร์มหรือคอนโทรล โดยเราต้องกำหนดเหตุการณ์ที่จะตอบสนองในแต่ละฟอร์ม ก่อนที่จะเขียนรหัสคำสั่งลงไป ซึ่งกำหนดเหตุการณ์ในคอนโทรลของฟอร์ม frmInput ตามหมายเลขในรูปที่ 9-8

รูปที่ 9-8 กำหนดเหตุการณ์ในฟอร์ม frmInput

เหตุการณ์ของฟอร์ม frmInput ที่จะตอบสนองต่อผู้ใช้ แสดงดังตารางที่ 9-8

ตารางที่ 9-8 เหตุการณ์ของฟอร์ม frmInput ที่จะตอบสนองต่อผู้ใช้

เหตุการณ์	**การตอบสนอง**
Load	แสดงค่าเริ่มต้นของกำลังอัดทรงกระบอกของคอนกรีต, ค่ายุบตัวของคอนกรีต และขนาดใหญ่สุดของหิน
Resize	ปรับขนาดของฟอร์มให้อยู่กึ่งกลางจอภาพเมื่อแสดงผลทุกครั้ง

จากตารางที่ 9-8 เขียนรหัสคำสั่งได้ดังนี้

เหตุการณ์ Load

```
Private Sub Form_Load()
  txtFcu.Text = fcu
```

```
  cboSlump.Text = cboSlump.List(Slump)
  cboMaxRock.Text = cboMaxRock.List(MaxRock)
End Sub
```

เหตุการณ์ Resize

```
Private Sub Form_Resize()
  If WindowState = 0 Then
    Move (Screen.Width - Me.Width) \ 2, (Screen.Height - Me.Height) \ 2
  End If
End Sub
```

สำหรับเหตุการณ์ที่จะตอบสนองต่อผู้ใช้ของคอนโทรลต่างๆ ในฟอร์ม frmInput แสดงดังตารางที่ 9-9

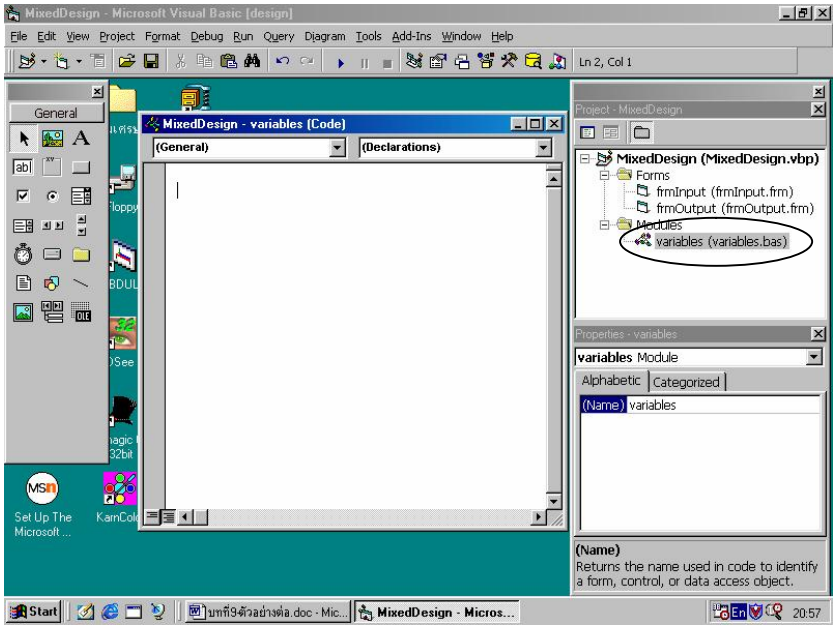
ตารางที่ 9-9 เหตุการณ์ที่จะตอบสนองต่อผู้ใช้ของคอนโทรลต่างๆ ในฟอร์ม frmInput

หมายเลข	Name	เหตุการณ์	การตอบสนอง
01	cboSlump	Cilck	รับข้อมูลใน ComboBox ให้กับตัวแปรของค่ายุบตัวของคอนกรีต
02	cboMaxRock	Cilck	รับข้อมูลใน ComboBox ให้กับตัวแปรของขนาดใหญ่สุดของหิน
03	cmdDesign	Cilck	คำนวณออกแบบส่วนผสมของคอนกรีต
04	cmdExit	Cilck	ออกจากโปรแกรม

ถึงขั้นตอนนี้ จะต้องมีการประกาศตัวแปรเพื่อรับข้อมูล คำนวณออกแบบ และแสดงผลลัพธ์ ซึ่งเราจะต้องพิจารณาเองว่า จะประกาศตัวแปรให้อยู่ในขอบเขตใด (คือประกาศแบบ Dim หรือ Global) ซึ่งจะยกตัวอย่างข้อพิจารณาตัวแปรในโปรแกรม ดังนี้

- ตัวแปรรับข้อมูล จะรับข้อมูลที่ฟอร์ม frmInput แล้วบางตัวแปรจะนำไปแสดงผลที่ฟอร์ม frmOutput ด้วย แสดงว่าบางตัวแปร ต้องใช้ในฟอร์มมากกว่าหนึ่งฟอร์ม ดังนั้น ในเบื้องต้นเพื่อให้ง่าย จึงกำหนดตัวแปรรับข้อมูลทุกตัวแปรให้เป็นแบบ Global

- ตัวแปรผลลัพธ์ จะคำนวณที่ฟอร์ม frmInput แล้วไปแสดงผลที่ฟอร์ม frmOutput แสดงว่าตัวแปร ต้องใช้ในฟอร์มมากกว่าหนึ่งฟอร์มเช่นกัน ดังนั้น กำหนดตัวแปรเป็นแบบ Global ดังนั้น โปรแกรมนี้จึงต้องมีการเพิ่มโมดูล ซึ่งกำหนดคุณสมบัติ Name และเซฟชื่อไฟล์เป็นชื่อเดียวกันว่า variables ดังรูปที่ 9-10

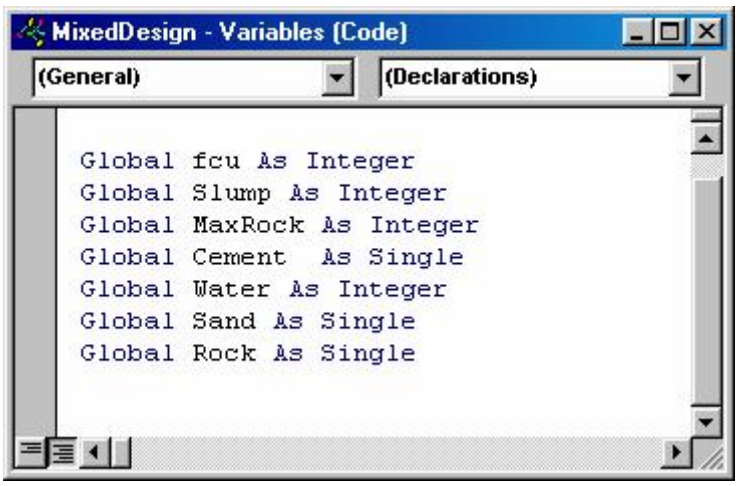


รูปที่ 9-9 เพิ่มโมดูล มีคุณสมบัติ Name และชื่อไฟล์ว่า variables

กำหนดตัวแปรในโมดูล variabls ดังนี้

```
Global fcu As Integer          (รับข้อมูลกำลังอัดคอนกรีต)
Global Slump As Integer        (รับข้อมูลในรายการค่ายุบตัวคอนกรีต)
Global MaxRock As Integer      (รับข้อมูลในรายการขนาดใหญ่สุดของหิน)
Global Cement  As Single       (ผลลัพธ์น้ำหนักปูนซีเมนต์)
Global Water As Integer        (ผลลัพธ์ปริมาตรน้ำ)
Global Sand As Single          (ผลลัพธ์น้ำหนักทราย)
Global Rock As Single          (ผลลัพธ์น้ำหนักหิน)
```

ซึ่งการกำหนดตัวแปรแบบ Global จะกำหนดในหน้าต่าง Code ของโมดูล variables แสดงดังรูปที่ 9-10



รูปที่ 9-10 แสดงการกำหนดตัวแปรในหน้าต่าง Code

เพราะฉะนั้น จากตารางที่ 9-9 เขียนรหัสคำสั่งได้ดังนี้
(สำหรับรหัสที่เริ่มต้นด้วยสัญลักษณ์ ' หมายถึง ข้อความถัดมาเป็นการเขียนคำอธิบายคำสั่ง (comment) เพื่อทำให้อ่านโปรแกรมได้ง่ายขึ้น ไม่ใช่รหัสคำสั่ง)

หมายเลข 01 เหตุการณ์คลิกในช่องรับข้อมูลค่ายุบตัวคอนกรีต

```
Private Sub cboSlump_click()
  Slump = cboSlump.ListIndex
End Sub
```

หมายเลข 02 เหตุการณ์คลิกในช่องรับข้อมูลขนาดใหญ่โตสุดของหิน

```
Private Sub cboMaxRock_Click()
  MaxRock = cboMaxRock.ListIndex
End Sub
```

หมายเลข 03 เหตุการณ์คลิกปุ่มคำนวณออกแบบ

(เหตุการณ์นี้ เป็นการคำนวณหาส่วนผสมคอนกรีต ซึ่งจัดเป็นหัวใจของโปรแกรมนี้ สังเกตการกำหนดค่าคงที่และตัวแปรในขอบเขตของโปรแกรมย่อย ซึ่งตัวแปรเหล่านี้ จะไม่มีผลต่อโปรแกรม

ย่อยอื่นๆ และลำดับการเขียนรหัสคำสั่ง ทำตามผังงานในแผนภูมิที่ 9-2 และหลังจากที่โปรแกรมคำนวณออกแบบเสร็จแล้ว จะทำการปิดฟอร์มตัวเองคือ frmInput และเปิดฟอร์ม frmOutput ขึ้นมาแสดงผล)

```
Private Sub cmdDesign_Click()
Const a = -0.00000002
Const b = 0.000023571
Const c = -0.0104
Const d = 2.14914286
Const SGc = 3.15
Const SGs = 2.65
Const SGr = 2.7

Dim S As Single
Dim MaxR As Integer
Dim fcr As Integer
Dim wc As Single
Dim FindVol As Integer
Dim SandVol As Single
Dim RockVol As Single

 ' fcu
 fcu = txtFcu.Text
 ' Value of Slump
 Select Case Slump
   Case 0
     S = 7.5
   Case 1
     S = 10
   Case 2
```

```
    S = 12.5
  Case 3
    S = 15
End Select
' Value of Max. Rock
Select Case MaxRock
  Case 0
    MaxR = 20
  Case 1
    MaxR = 25
End Select
' ********* Solution *********
' fcr
If fcu < 210 Then
  fcr = fcu + 70
ElseIf (fcu >= 210) And (fcu < 350) Then
  fcr = fcu + 85
ElseIf (fcu >= 350) And (fcu <= 450) Then
  fcr = fcu + 100
End If
' Water Cement Ratio
wc = (a * fcr ^ 3) + (b * fcr ^ 2) + (c * fcr) + d
' Find Volumn of Water
If S = 7.5 And MaxR = 25 Then
  Water = 170
ElseIf S = 7.5 And MaxR = 20 Then
  Water = 180
ElseIf S = 10 And MaxR = 25 Then
  Water = 180
ElseIf S = 10 And MaxR = 20 Then
```

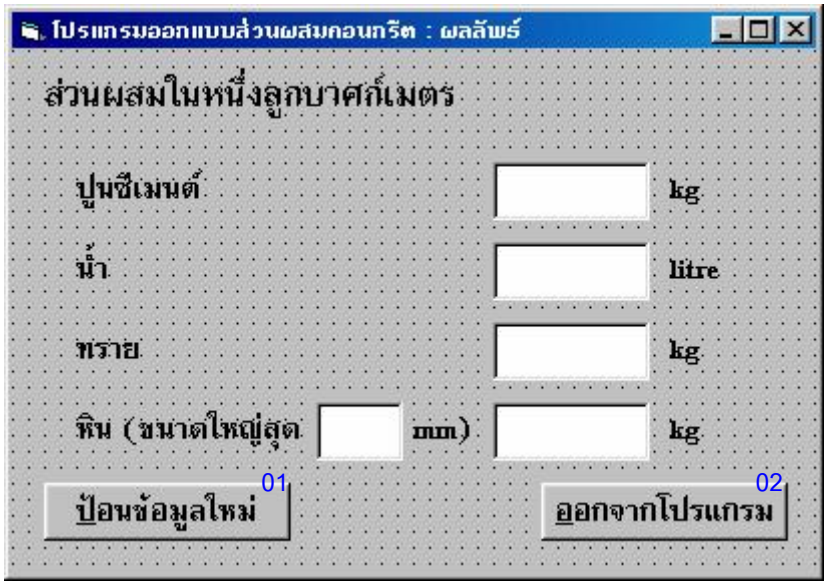
```
  Water = 190
ElseIf S = 12.5 And MaxR = 25 Then
  Water = 190
ElseIf S = 12.5 And MaxR = 20 Then
  Water = 200
ElseIf S = 15 And MaxR = 25 Then
  Water = 200
ElseIf S = 15 And MaxR = 20 Then
  Water = 210
End If
' Find Weight of Cement
Cement = Water / wc
' Find Weight of Sand
If MaxR = 25 Then
  FindVol = 380
ElseIf MaxR = 20 Then
  FindVol = 400
End If
' Find Weight of Sand
SandVol = FindVol - (Cement / SGc)
Sand = SandVol * SGs
' Find Weight of Rock
RockVol = 990 - Water - FindVol
Rock = RockVol * SGr

Unload Me
frmOutput.Show 1
End Sub
```

หมายเลข 04 เหตุการณ์คลิ๊กปุ่ม Exit

```
Private Sub cmdExit_Click()
  End
End Sub
```

สำหรับฟอร์ม frmOutput กำหนดเหตุการณ์ในคอนโทรล ตามหมายเลขในรูปที่ 9-11



รูปที่ 9-11 กำหนดเหตุการณ์ในฟอร์ม frmOutput

สำหรับเหตุการณ์ของฟอร์ม frmOutput ที่จะตอบสนองต่อผู้ใช้ แสดงดังตารางที่ 9-10

ตารางที่ 9-10 เหตุการณ์ของฟอร์ม frmOutput ที่จะตอบสนองต่อผู้ใช้

เหตุการณ์	**การตอบสนอง**
Load	แสดงผลลัพธ์ น้ำหนักปูนซีเมนต์ ปริมาตรน้ำ น้ำหนักทราย น้ำหนักหิน และขนาดใหญ่สุดของหิน (เฉพาะค่านี้ได้จากการรับข้อมูลของผู้ใช้)
Resize	ปรับขนาดของฟอร์มให้อยู่กึ่งกลางจอภาพเมื่อแสดงผลทุกครั้ง

จากตารางที่ 9-10 เขียนรหัสคำสั่งได้ดังนี้

เหตุการณ์ Load

```
Private Sub Form_Load()
  lblCement.Caption = Cement
  lblWater.Caption = Water
  lblSand.Caption = Sand
  If MaxRock = 0 Then
    lblMaxRock.Caption = 20
  ElseIf MaxRock = 1 Then
    lblMaxRock.Caption = 25
  End If
  lblRock.Caption = Rock
End Sub
```

เหตุการณ์ Resize

```
Private Sub Form_Resize()
  If WindowState = 0 Then
    Move (Screen.Width - Me.Width) \ 2, (Screen.Height - Me.Height) \ 2
  End If
End Sub
```

สำหรับเหตุการณ์ที่จะตอบสนองต่อผู้ใช้ของคอนโทรลในฟอร์ม frmOutput แสดงดังตารางที่ 9-11

ตารางที่ 9-11 เหตุการณ์ที่จะตอบสนองต่อผู้ใช้ของคอนโทรลในฟอร์ม frmOutput

หมายเลข	Name	เหตุการณ์	การตอบสนอง
01	cmdInput	Cilck	ปิดฟอร์มตัวเองคือ frmOutput และเปิดฟอร์ม frmOutput
02	cmdExit	Cilck	ออกจากโปรแกรม

จากตารางที่ 9-11 เขียนรหัสคำสั่งได้ดังนี้

หมายเลข 01 เหตุการณ์คลิกปุ่มป้อนข้อมูลใหม่

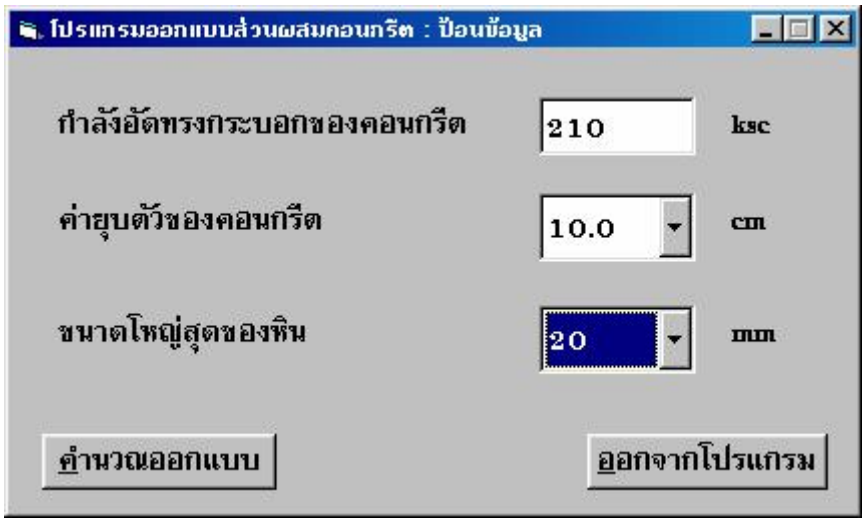
```
Private Sub cmdInput_Click()
  Unload Me
  frmInput.Show
End Sub
```

หมายเลข 02 เหตุการณ์คลิกปุ่มออกจากโปรแกรม

```
Private Sub cmdExit_Click()
  End
End Sub
```

- ตัวอย่างการทำงาน

เมื่อรันโปรแกรม จะแสดงหน้าต่างป้อนข้อมูลให้ผู้ใช้ป้อนข้อมูลทั้ง 3 ข้อมูล เมื่อป้อนข้อมูลแล้ว จะปรากฏตัวอย่างดังรูปที่ 9-12



รูปที่ 9-12 ตัวอย่างการป้อนข้อมูลในโปรแกรม

เมื่อคลิกปุ่มคำนวณออกแบบ โปรแกรมจะทำงานต่อเนื่องกันดังนี้ (ซึ่งเป็นสิ่งที่ผู้ใช้มองไม่เห็น)

- กำหนดตัวแปรเพื่อรับข้อมูลที่ผู้ใช้ป้อน
- คำนวณออกแบบ
- ปิดฟอร์ม frmInput
- เปิดฟอร์ม frmOutput

เมื่อฟอร์ม frmOutput เปิดขึ้น จะแสดงผลดังรูปที่ 9-13

รูปที่ 9-13 การแสดงผลของโปรแกรม

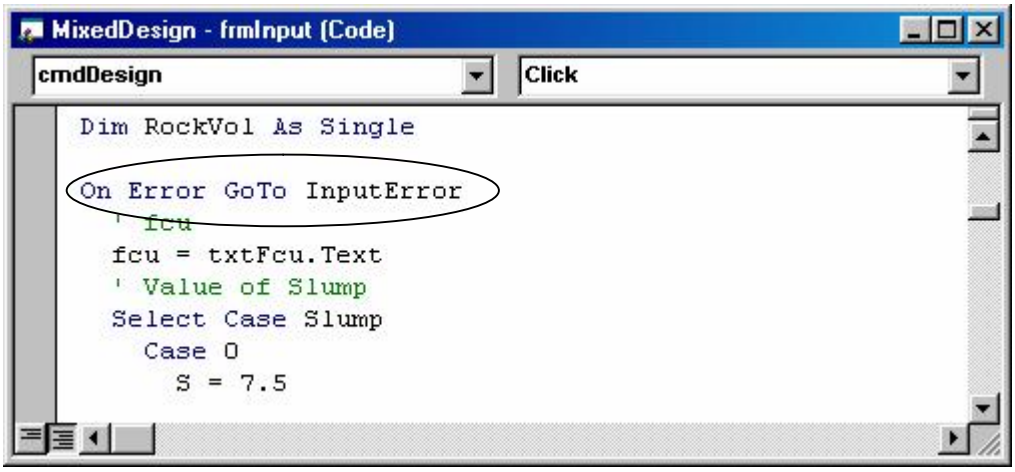
ถ้าผู้ใช้คลิกปุ่มป้อนข้อมูลใหม่ โปรแกรมจะปิดฟอร์มผลลัพธ์นี้ และเปิดฟอร์มป้อนข้อมูลขึ้นมาใหม่ ซึ่งแสดงผลเหมือนเดิมดังรูปที่ 9-12

- ปรับปรุงโปรแกรม

โปรแกรมนี้ สามารถปรับปรุงได้อีกหลายจุดเพื่อให้มีประสิทธิภาพมากขึ้น (ซึ่งนี่คือเหตุผลที่โปรแกรมทั่วไป มีการปรับรุ่น (version) เพิ่มเติมได้อีกหลายรุ่น) ซึ่งจะยกตัวอย่างมาหนึ่งจุดคือการตรวจสอบข้อผิดพลาดในการป้อนข้อมูล มีรายละเอียดดังนี้

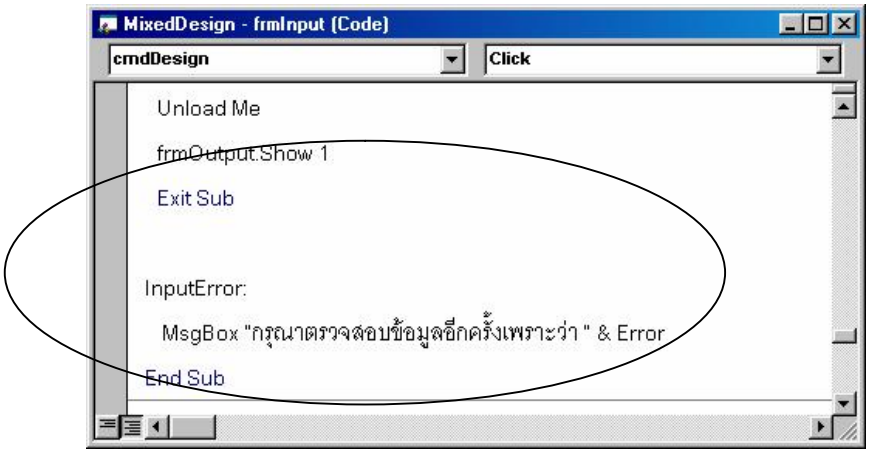
การตรวจสอบข้อผิดพลาดในการป้อนข้อมูล

การป้อนข้อมูลของโปรแกรมนี้ ในส่วนของ ComboBox ทั้งสองช่อง ได้กำหนดไว้แล้วในตอนออกแบบว่าเป็นแบบ Dropdown List ซึ่งหมายความว่าบังคับให้ผู้ใช้เลือกข้อมูลที่มีอยู่เท่านั้น การดักจับข้อผิดพลาดจึงไม่จำเป็น แต่ในช่อง TextBox ซึ่งรับข้อมูลกำลังอัดของคอนกรีต หากผู้ใช้ใส่ข้อมูลที่ไม่ใช่ตัวเลข โปรแกรมจะเกิดข้อผิดพลาดในขณะรันโปรแกรมทันที จึงควรเขียนคำสั่งดักจับข้อผิดพลาดไว้ ซึ่งก็คือเขียนในโปรแกรมย่อยเหตุการณ์ cmdDesign_Click แทรกไว้ก่อนบรรทัดที่จะกำหนดตัวแปร fcu ให้รับค่าจาก txtFcu.Text โดยใช้คำสั่ง On Error Goto ดังรูปที่ 9-14 (ในที่นี้ใช้ LineNumber ชื่อว่า InputError)



รูปที่ 9-14 การแทรกคำสั่ง On Error Goto

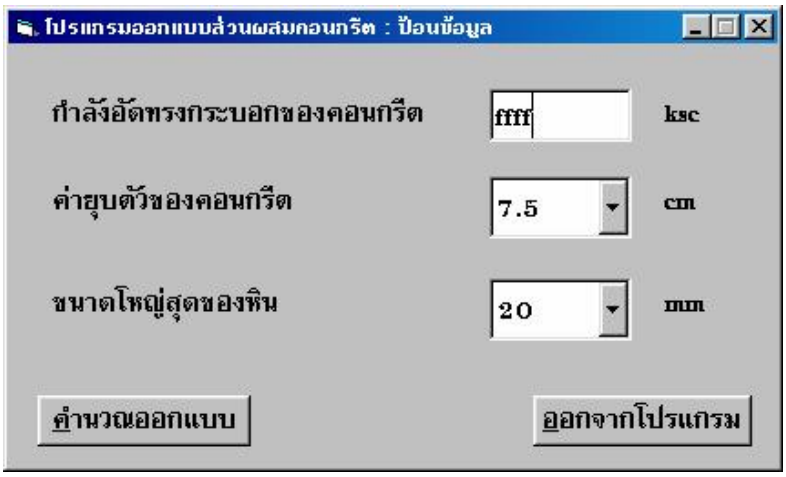
ต่อมา แทรกคำสั่งของ LineNumber ที่ชื่อ InputError ไว้ส่วนท้ายของโปรแกรมย่อยเหตุการณ์ cmdDesign_Click ดังรูปที่ 9-15



รูปที่ 9-15 การแทรกคำสั่งของ LineNumber ที่ชื่อ InputError

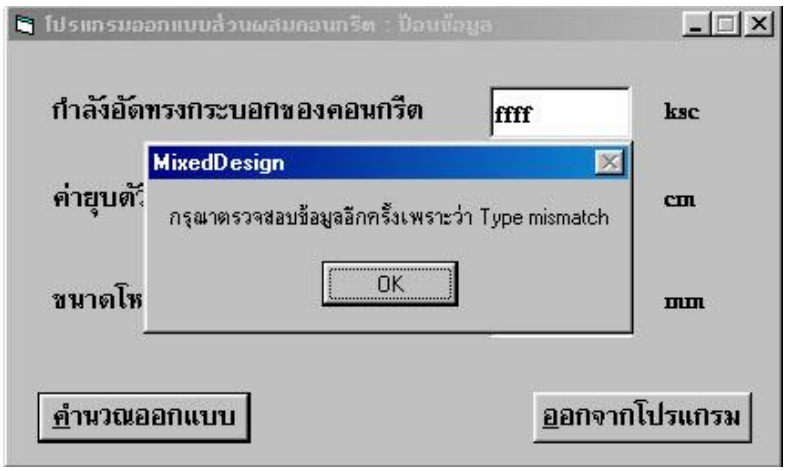
สังเกตว่า ก่อนบรรทัด InputError มีคำสั่ง Exit Sub หมายถึง ถ้าไม่มีข้อผิดพลาดในการป้อนข้อมูล ก็ให้ออกจากโปรแกรมย่อยนี้โดยไม่ต้องทำคำสั่ง InputError แต่ถ้าเกิดข้อผิดพลาด ให้โปรแกรมข้ามคำสั่งต่างๆ แล้วมาที่ InputError :ซึ่งมีคำสั่งต่อมาคือแสดงข้อความแจ้งข้อผิดพลาด โดยฟังก์ชัน Error ท้ายคำสั่ง MsgBox จะแสดงข้อความแจ้งข้อผิดพลาดให้

ตัวอย่างถ้าป้อนข้อมูลกำลังอัดคอนกรีตผิดพลาด ดังรูปที่ 9-16



รูปที่ 9-16 ตัวอย่างการป้อนข้อมูลผิดพลาด

เมื่อคลิกปุ่มคำนวณออกแบบ โปรแกรมจะแสดงข้อความแจ้งข้อผิดพลาด ดังรูปที่ 9-17



รูปที่ 9-17 ตัวอย่างแสดงข้อความแจ้งข้อผิดพลาด

สำหรับการปรับปรุงในด้านความสวยงามนั้น วิชวลเบสิกมีเครื่องมือมาให้อย่างเพียงพอที่จะทำได้โดยไม่ยาก อย่างไรก็ตาม ในงานวิศวกรรมโยธานี้ ความถูกต้องของโปรแกรม จะต้องเป็นสิ่งที่ให้ความสำคัญในลำดับแรก ความสวยงามเป็นเรื่องรองลงไป

เมื่อปรับปรุงโปรแกรมเป็นที่พอใจแล้ว ขั้นตอนต่อไปคือการสร้างตัวติดตั้งให้กับโปรแกรม ซึ่งสามารถทำตามลำดับขั้นตอนที่อธิบายไว้ในบทที่ 8 เป็นอันเสร็จขั้นตอนการสร้างโปรแกรม

หนังสืออ้างอิง

กิตติ ภักดีวัฒนะกุล และจำลอง ครูอุตสาหะ. Visual Basic 6 ฉบับโปรแกรมเมอร์. พิมพ์ครั้งที่ 8. กรุงเทพฯ : บริษัท ดวงกมลสมัย จำกัด, 2543.

จิระ จริงจิตร. เรียนลัด Visual Basic. พิมพ์ครั้งที่ 2. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2538.

ฉัททวุฒิ พีชผล และพิชิต สันติกุลานนท์. คู่มือเรียน Visual Basic 6. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2542.

โชติพันธุ์ หล่อเลิศสุนทร และฐิตะพันธ์ หล่อเลิศสุนทร. สอนเขียน Visual Basic 6.0 .ให้เป็น Project. กรุงเทพฯ : Soft Express&Publishing, 2543.

ธาริน สิทธิธรรมชารี และธนัญชัย จำนงค์ภักดี. Microsoft Visual Basic Version 5.0. กรุงเทพฯ : บริษัท ซัคเซส มีเดีย จำกัด, 2541.

วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์. ข้อกำหนดมาตรฐานวัสดุและการก่อสร้างสำหรับโครงสร้างคอนกรีต. ม.ป.ท., 2540.

สมศักดิ์ อัศวกุลไพบูลย์. (ผู้เรียบเรียง). การเขียนโปรแกรมบนวินโดวส์ด้วย Visual Basic 4.0 ภาคปฏิบัติ. กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2540.

สุทธิศักดิ์ พงศ์ธนาพาณิช. Visual Basic 4.0 Professional. กรุงเทพฯ : บริษัท ซีเอ็ด ยูเคชั่น จำกัด, 2539.

Michael Halvorson. Learn Microsoft Visual Basic 6.0 NOW. USA : Microsoft Press, 1999.

แหล่งข้อมูลอื่นๆ

Mastering Visual Basic 6.0 (CD-ROM)

ภาคผนวก ก

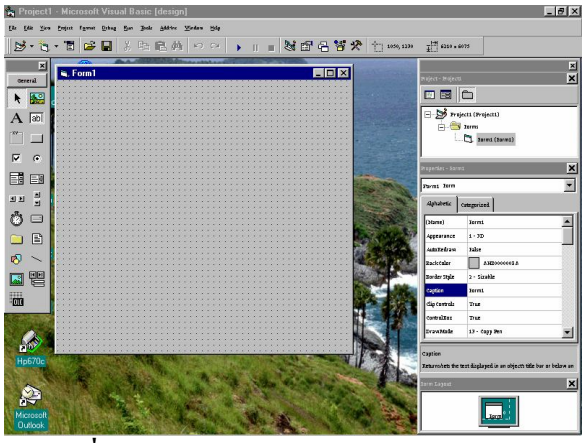
บทความเรื่อง
การปรับเปลี่ยนฟอนต์ในวิชวลเบสิก

ถามมา

ผมลงโปรแกรม VB5 Professional Edition แต่ตรง Project Explorer และ Properties Windows นั้น ฟอนต์ตัวเล็กมากเลยครับ ไม่ทราบว่ามีวิธีไหน ทำให้ฟอนต์ตัวใหญ่ขึ้นได้ไหมครับ?

ตอบไป

ปัญหานี้ เท่าที่พบจะเกิดขึ้นกับระบบวินโดวส์ 95 ไทยเอดิชัน คาดว่าถ้าไม่ใช่ไทยเอดิชันจะไม่เป็นไร ซึ่งผมเองก็เจอ คือเมื่อลงโปรแกรมวิชวลเบสิกเสร็จแล้วลองเปิดดู ก็จะพบว่าฟอนต์มีขนาดเล็กมาก ตามรูปที่ 1



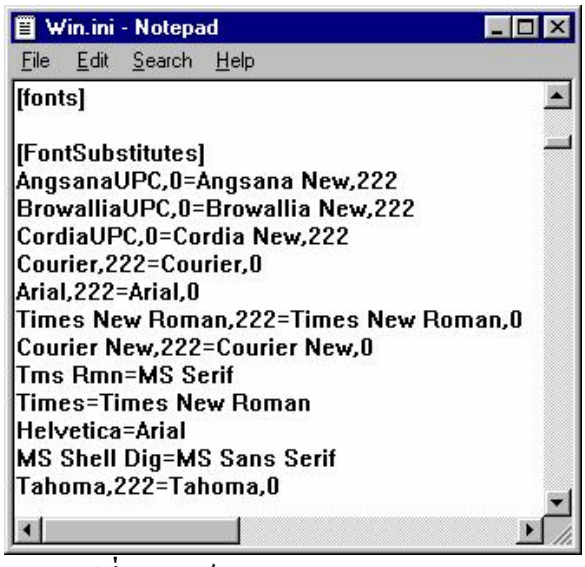
รูปที่ 1 ฟอนต์ในวิชวลเบสิกมีขนาดเล็กมาก

วิธีแก้ไขคือ เข้าไปเพิ่มฟอนต์ในไฟล์ Win.ini โดยทำตามขั้นตอนดังนี้

- ออกจากวิชวลเบสิก เปิดโปรแกรม Notepad ไปที่ File/Open คือเลือกเปิดไฟล์ ซึ่งจะมีหน้าต่างให้เลือกไฟล์ พิมพ์ win.ini ที่ช่อง File name ดังรูปที่ 2 (โดยทั่วไปไฟล์นี้ จะอยู่ที่โฟลเดอร์ Windows)

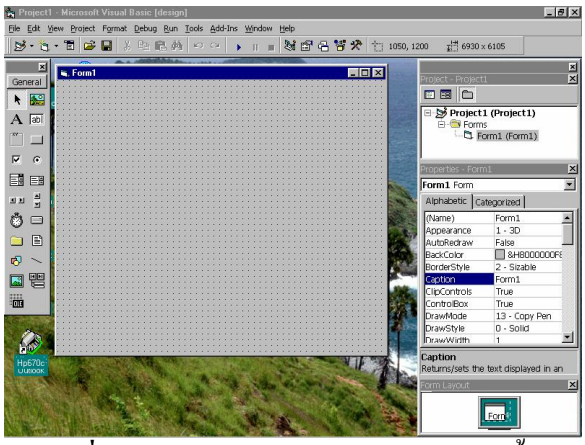
รูปที่ 2 ใช้ Notepad เปิดไฟล์ Win.ini

- ในไฟล์ Win.ini เลื่อนบรรทัดไปที่ส่วน [FontSubstitutes] แล้วพิมพ์ต่อท้ายบรรทัดในส่วนนี้ ว่า Tahoma,222=Tahoma,0 ดังรูปที่ 3



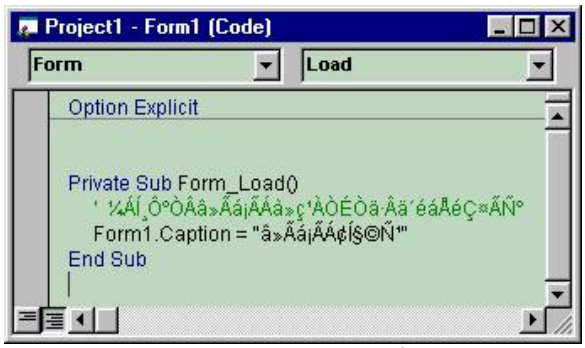
รูปที่ 3 พิมพ์ Tahoma,222=Tahoma,0

- ต่อมาเซฟให้เรียบร้อย ปิดโปรแกรม Notepad รีสตาร์ทวินโดวส์ใหม่ เสร็จแล้วลองเปิดวิชวลเบสิกดูใหม่ จะเห็นว่าฟอนต์มีขนาดใหญ่ขึ้นกว่าเดิม ดังรูปที่ 4



รูปที่ 4 ฟอนต์ในวิชวลเบสิกมีขนาดใหญ่ขึ้น

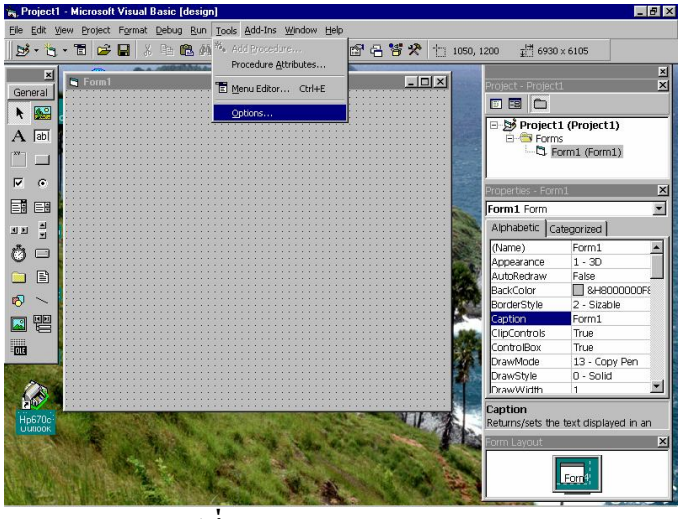
ครับ เป็นอันแก้ปัญหาเรียบร้อยตามคำถาม อ่านฟอนต์ในวิชวลเบสิกสบายตาขึ้นมาก ทีนี้ผมขอเสริมต่อซึ่งเป็นเรื่องใกล้เคียงกัน คือในการเขียนรหัสเพื่อให้แสดงผลเป็นภาษาไทย หรือการใส่คำอธิบายเป็นภาษาไทยลงในหน้าต่าง Code นั้น ปกติจะไม่แสดงเป็นภาษาไทยให้ คือจะออกเป็นอักขระแอสกีส่วนขยายที่เรานำส่วนนี้มาทำเป็นภาษาไทย ตัวอย่างดังรูปที่ 5



รูปที่ 5 การแสดงผลในหน้าต่าง Code เมื่อเขียนเป็นภาษาไทย

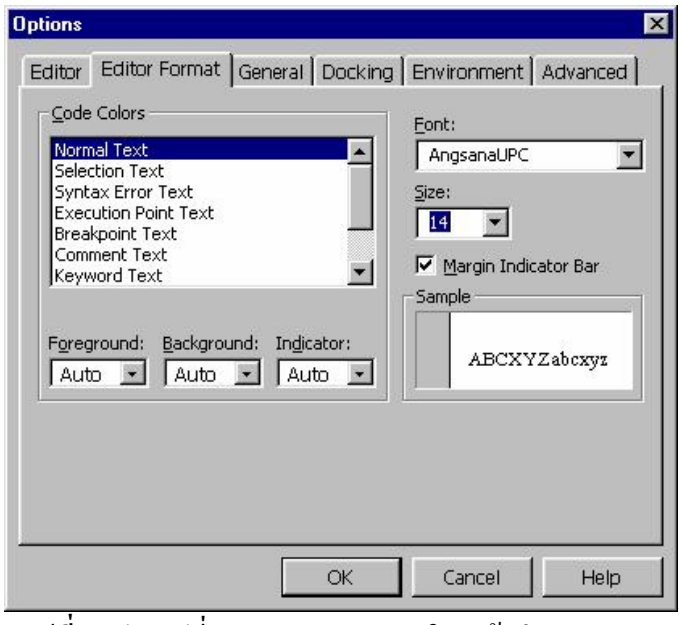
ที่เป็นเช่นนี้ เพราะฟอนต์ปกติที่กำหนดมานั้น เป็นฟอนต์ที่ไม่มีภาษาไทย (ก็แน่นอนเพราะคนทำวิชวลเบสิกไม่ใช่คนไทย) แต่เราสามารถเปลี่ยนฟอนต์ได้ โดยทำตามขั้นตอนดังนี้

- ไปที่เมนู Tools/Options… ของวิชวลเบสิก ดังรูปที่ 6



รูปที่ 6 เมนู Tools/Options…

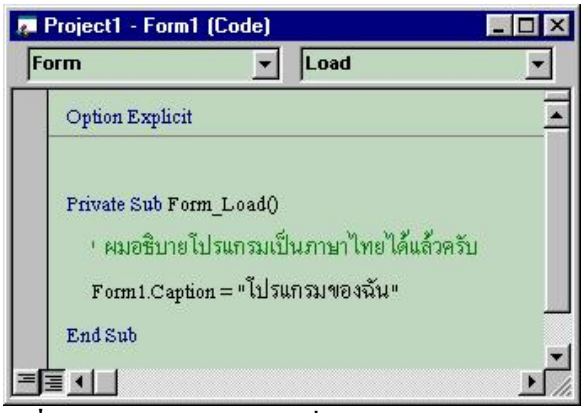
- เมื่อเข้าไปในเมนูย่อย Options… จะมีหน้าต่างตัวเลือกให้ปรับเปลี่ยนค่าต่างๆ ให้ไปที่ Editor Format ในช่อง Font แล้วเลือกฟอนต์ที่แสดงภาษาไทยได้ เช่น AngsanaUPC และช่อง Size คือขนาดของฟอนต์ เลือกที่ 14 (ผมลองดูแล้วคิดว่าขนาดกำลังพอดี) ดังรูปที่ 7



รูปที่ 7 ปรับเปลี่ยน Font และ Size ในหน้าต่าง Options

- ต่อมาคลิ๊กปุ่ม OK ก็เรียบร้อย

ทีนี้ ลองไปดูรหัสในหน้าต่าง Code ใหม่ จะเห็นว่าอักขระประหลาดกลายเป็นอักขระภาษาไทยแล้ว ดังรูปที่ 8



รูปที่ 8 อักขระประหลาดเปลี่ยนเป็นอักขระภาษาไทย

ถึงตอนนี้ เราก็สามารถเขียนโปรแกรมได้คล่องตัวและชัดเจนยิ่งขึ้น เพราะสามารถเขียนคำอธิบายเป็นภาษาไทยได้ ซึ่งเข้าใจดีกว่าเขียนภาษาอังกฤษแปลกๆ และการเขียนรหัสให้แสดงผลเป็นภาษาไทย ก็เห็นได้ในหน้าต่าง Code ว่าเขียนอะไรลงไป ไม่ต้องรอให้แสดงผลก่อนแล้วจึงทราบว่าเขียนผิดหรือถูก พยายามเขียนกันมากๆนะครับ เราเขียนโปรแกรมใช้เองไม่ต้องซื้อโปรแกรมต่างประเทศ ก็ถือว่าช่วยชาติได้ทางหนึ่งแล้ว ติดขัดปัญหาอะไรก็**ถามมา**ได้

- บทความนี้เคยลงตีพิมพ์ในนิตยสาร Computer.Today
ปีที่ 8 ปักษ์แรก ธันวาคม 2541

ภาคผนวก ข

บทความเรื่อง
การสลับเมนูภาษาไทย-อังกฤษ

ถามมา

ผมอยากจะเขียนโปรแกรมสร้างเมนูที่สามารถสลับภาษาไทย-อังกฤษได้ แบบเวิร์ด 6 จะต้องทำอย่างไรบ้างครับ

ตอบไป

ปัญหานี้ง่ายมาก แนวคิดของการสลับเมนูภาษาไทย-อังกฤษคือ ตามปกติการสร้างเมนูในวิชวลเบสิก ทำได้โดยกำหนดในหน้าต่าง Tools/Menu Editor... ซึ่งจะมีการกำหนด Property Caption คือข้อความที่แสดงที่หน้าจอ และ Name คือตัวแปรของเมนูนั้นๆ ในตำราทั่วไปมักจะกล่าวถึงเพียงเท่านี้ แต่ความเป็นจริงคือ **เมื่อเรามี Name ของเมนู เราก็สามารถเขียนโปรแกรมเพื่อแก้ไข Caption ให้เป็นข้อความอื่นหรือแม้แต่ภาษาไทยได้ และก็สลับภาษาไทย-อังกฤษได้**

กำหนดรายละเอียดของเมนู

เราจะลองยกตัวอย่าง โดยกำหนดเมนู 2 เมนู ดังนี้

File (แฟ้ม) Window (หน้าต่าง)

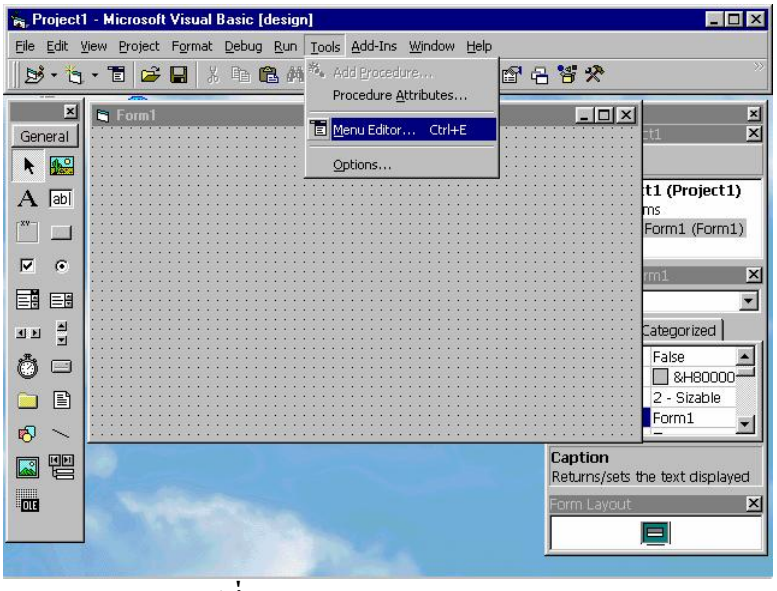
└── Toggle English-Thai (สลับภาษาไทย-อังกฤษ)

กำหนด Property เริ่มต้น ที่จะเขียนไว้ใน Menu Editor ดังนี้

Caption	Name
File	mnuFile
Window	mnuWindow
Toggle English-Thai	winToggle

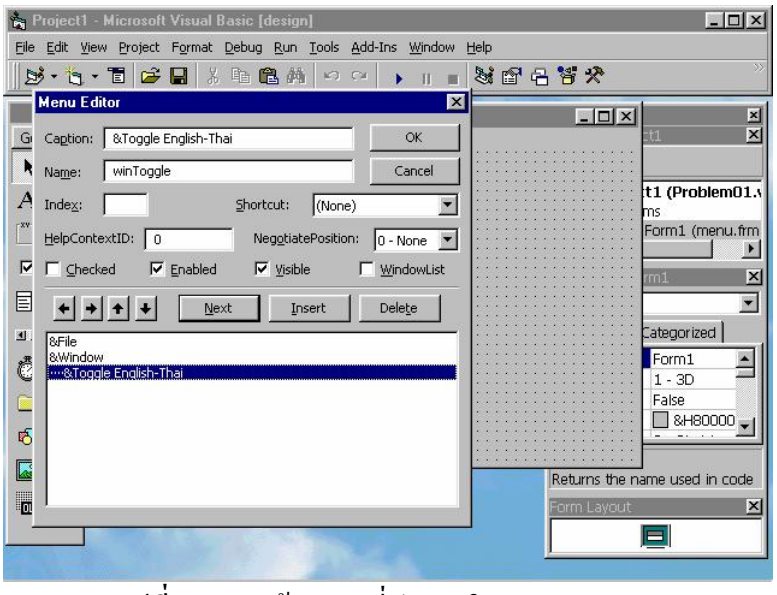
ทำตามขั้นตอนดังนี้

1. เปิดวิชวลเบสิก แล้วไปที่เมนู Tools/Menu Editor… ดังรูปที่ 1



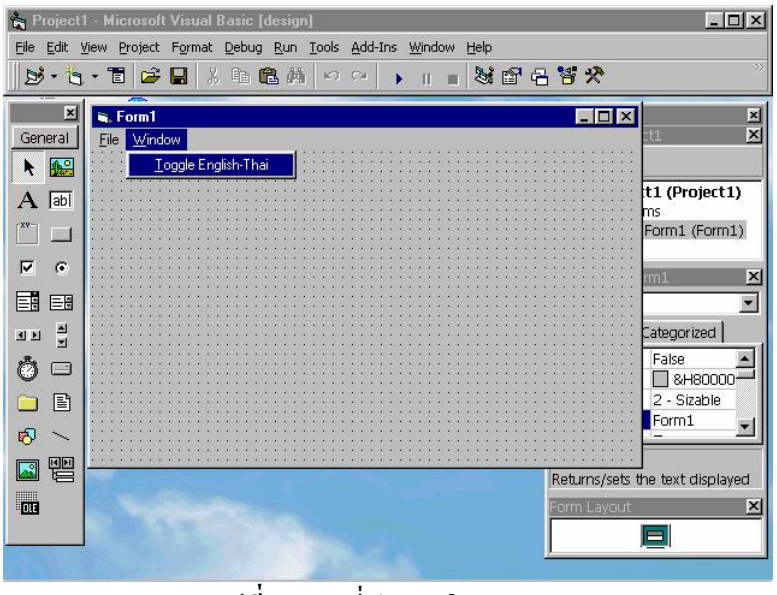
รูปที่ 1 เมนู Tools/Menu Editor…

2. คลิกไปที่ Menu Editor… จะมีหน้าต่างให้ป้อนเมนูตามที่เรากำหนดทั้ง Caption และ Name เมื่อป้อนเสร็จแล้ว จะปรากฏดังรูปที่ 2



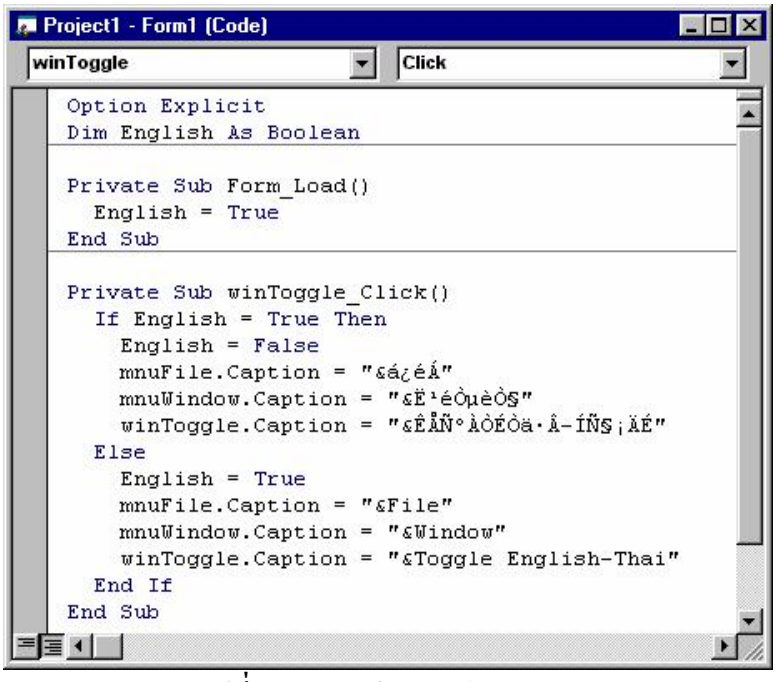
รูปที่ 2 แสดงข้อความที่ปรากฏใน Menu Editor

3. คลิก OK จะปรากฏเมนูที่ Form1 ดังรูปที่ 3



รูปที่ 3 เมนูที่ปรากฏใน Form1

4. เขียนโปรแกรมที่ Form1 (Code) ตามรายละเอียดในรูปที่ 4



รูปที่ 4 รายละเอียดของโปรแกรม

ที่เห็นเป็นอักษรประหลาดในโปรแกรมย่อย winToggle_Click เพราะเป็นการเขียนภาษาไทยเข้าไปในโปรแกรม (แล้วไม่ได้ปรับฟอนต์ให้แสดงภาษาไทย) ดังนี้

mnuFile.Caption = “&แฟ้ม”

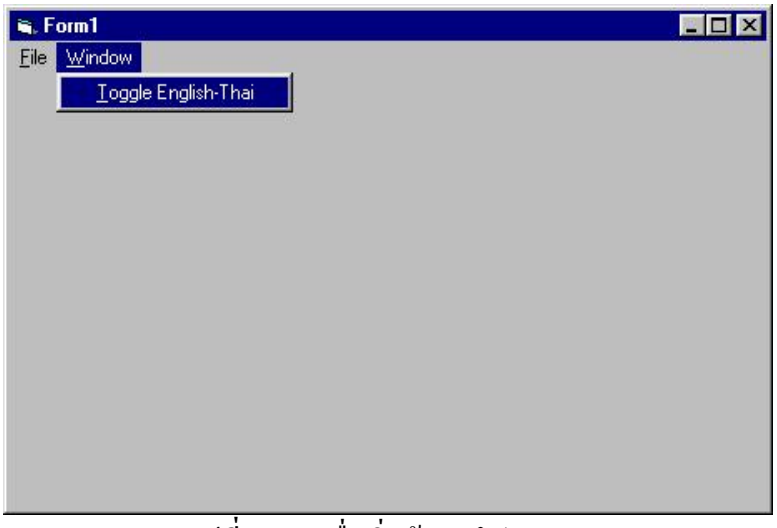
mnuWindow.Caption = “&หน้าต่าง”

winToggle.Caption = “&สลับภาษาไทย-อังกฤษ”

อธิบายโปรแกรมที่เขียนเข้าไปเป็นภาษามนุษย์ได้คือ เราใช้ตัวแปร English เป็นตัวตรวจสอบว่าเมนูตอนนั้นเป็นภาษาอังกฤษ (True) หรือภาษาไทย (False) โดยเริ่มต้นกำหนดให้เป็นภาษาอังกฤษ เมื่อคลิกที่เมนูย่อย Toggle English-Thai (หรือ สลับภาษาไทย-อังกฤษ) ก็ทำการสลับค่าของ English พร้อมกับเปลี่ยน Caption ของแต่ละเมนูย่อยโดยเรียกผ่านตัวแปร Name ที่เรากำหนดไว้เท่านี้เอง

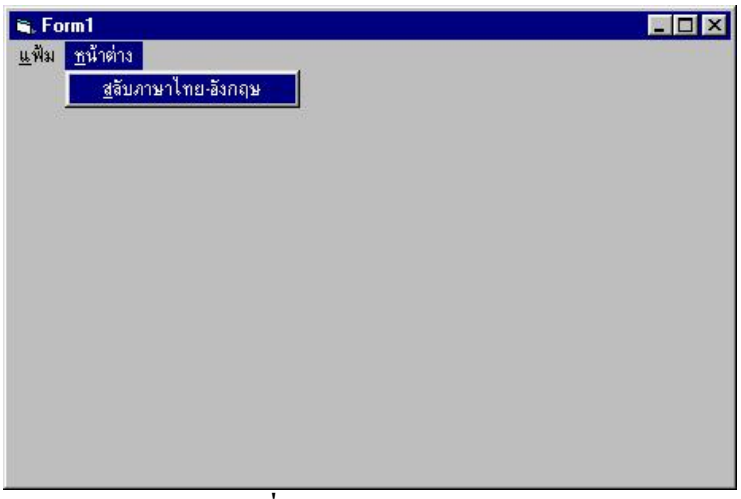
ลองรันโปรแกรมดู

เมื่อรันโปรแกรมจะแสดงเมนูเป็นภาษาอังกฤษดังรูปที่ 5



รูปที่ 5 เมนูเมื่อเริ่มต้นรันโปรแกรม

ถ้าผู้ใช้คลิกที่เมนูย่อย Toggle English-Thai เมนูจะเปลี่ยนเป็นภาษาไทยดังรูปที่ 6



รูปที่ 6 เมนูเป็นภาษาไทย

และถ้าผู้ใช้คลิกที่เมนูย่อย สลับภาษาไทย-อังกฤษ เมนูก็จะกลับเป็นภาษาอังกฤษดังเดิม

ส่งท้าย

- ถ้ามีหลายเมนู เราก็ทำในทำนองนี้ อาจเขียนโปรแกรมยาวหน่อยแต่ก็คุ้ม
- ถึงตอนนี้ก็ขอกล่าวคำว่า สวัสดีครับ

- บทความนี้เคยลงตีพิมพ์ในนิตยสาร Computer.Today
ปีที่ 8 ปักษ์หลัง กันยายน 2541