ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ เครื่องถ่ายสติกเกอร์

Photo Sticker

ชื่อนักศึกษา

1. นายรัฐศักดิ์	ตั้งพิทักษ์ไกร	รหัสประจำตัว	41031417
2. ว่าที่ ร.ต.วัชรพล	แซ่ตั้น	รหัสประจำตัว	41031419
3. นายสมเพ็ชร	เครือทองศรี	รหัสประจำตัว	41031425
4. นายอัคร	เพชรมณี	รหัสประจำตัว	41031428

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์สุรพงษ์ สิริพงศ์ดี

อาจารย์ที่ปรึกษาร่วม อาจารย์วรวิทย์ สมหา

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์สุรพงษ์	สิริพงศ์ดี	.....
2. อาจารย์วรวิทย์	สมหา	.....
3. อาจารย์อำพล	ทองระอา	.....
4. อาจารย์ไพบูลย์	พวงวงศ์ตระกูล	.....
5. อาจารย์สุระชัย	พิมพ์สาลี	.....

วัน/เดือน/ปีที่สอบ วันศุกร์ที่ 12 พฤษภาคม พ.ศ. 2543 เวลา 12.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 11 เดือน ก.ค. พ.ศ. 2543

ปริญญานิพนธ์

เครื่องถ่ายสติกเกอร์

PHOTO STICKER

นายรัฐศักดิ์ ตั้งพิทักษ์ไกร

ว่าที่ ร.ต.วัชรพล แซ่ตั้น

นายสมเพ็ชร เครือทองศรี

นายอัคร เพชรมณี

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง เครื่องถ่ายสติกเกอร์

PHOTO STICKER

วัตถุประสงค์

1. เพื่อศึกษาหลักการทำงานของโปรแกรม Visual C++
2. เพื่อนำความรู้ที่ได้จากการศึกษาโปรแกรม Visual C++ มาประยุกต์ใช้ในงานอื่นๆได้
3. เพื่อศึกษาหลักการพิมพ์ภาพแบบกราฟฟิกโดยใช้โปรแกรม Visual C++
4. เพื่อศึกษาหลักการใช้กล้อง CCD Camera ในการติดต่อกับโปรแกรม Visual C++
5. เพื่อศึกษาเทคโนโลยีของเครื่องถ่ายสติกเกอร์

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้เกี่ยวกับหลักการเขียนโปรแกรม Visual C++
2. มีความรู้เกี่ยวกับการอินเตอร์เฟสสัญญาณภาพจากกล้อง CCD Camera
3. มีความรู้เกี่ยวกับการจัดการกับภาพกราฟฟิกและการพิมพ์ภาพกราฟฟิกนั้นๆออกมาเป็นภาพสติกเกอร์
4. มีความรู้เกี่ยวกับเทคโนโลยีของเครื่องถ่ายสติกเกอร์ที่มีอยู่ในปัจจุบัน
5. สามารถที่จะนำโปรแกรม Visual C++ ที่ได้ทำการศึกษาไปประยุกต์ใช้ในงานอื่นๆได้
6. สามารถสร้างต้นแบบของเครื่องถ่ายสติกเกอร์

ชื่อหัวข้อ	เครื่องถ่ายสติกเกอร์
นักศึกษา	นายรัฐศักดิ์ ตั้งพิทักษ์ไกร
	ว่าที่ ร.ต.วัชรพล แซ่ตั้น
	นายสมเพ็ชร เครือทองศรี
	นายอัคร เพชรมณี
อาจารย์ที่ปรึกษา	อาจารย์สุรพงษ์ สิริพงศ์ดี
อาจารย์ที่ปรึกษาร่วม	อาจารย์วรวิทย์ สมหา
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอโครงงานเรื่องเครื่องถ่ายสติกเกอร์ โดยการใช้กล้องถ่ายภาพ CCD Camera เป็นตัวกลางในการติดต่อ เพื่อที่จะนำภาพที่ต้องการเข้ามาทำเป็นภาพสติกเกอร์ ซึ่งโปรแกรมที่ใช้เป็นตัวทำงานในเครื่องถ่ายสติกเกอร์นี้ ได้ทำการสร้างจากเทคโนโลยีของโปรแกรม Visual C++ โดยภายในตัวโปรแกรมที่ใช้งานนั้นสามารถที่จะจัดการกับภาพที่รับเข้ามาจากกล้อง CCD Camera ในตอนแรกแล้วทำการนำภาพถ่ายที่ได้มาจัดลงกรอบที่สร้างไว้ หลังจากที่ทำการเลือกกรอบที่เหมาะสมกับภาพแล้วจึงนำภาพที่จัดลงในกรอบนั้นพิมพ์ออกมาเป็นภาพสติกเกอร์ ในการทำภาพ ซึ่งผู้ใช้งานสามารถเลือกกรอบรูปได้ตามความต้องการ

Thesis Title	PHOTO STICKER
Students	Mr.Rattasuk Tungpitukkai
	Mr.Wacharapon Saetun
	Mr.Somphet Khruethongsri
	Mr.Akkhara Petchmanee
Advisor	Mr.Surapong Siripongdee
Co-Advisor	Mr.Worawit Somha
Education Level	Bachelor of Science in Industrial Education
Program in	Electronic and Computer
Academic Year	1999

ABSTRACT

This thesis presents how PHOTO STICKER works. The machine uses a camera called CCD Camera as the communicated tool in door to take a picture into a sticker. The program in the PHOTO STICKER is made from technology of Visual C++ program which put the picture into the chosen frame and becomes the sticker.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปด้วยดี ก็ด้วยความช่วยเหลือในการให้คำแนะนำจากคณาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่าน รวมถึงความเอื้อเฟื้อในเรื่องของวัสดุอุปกรณ์,เครื่องมือ,เครื่องใช้ และสถานที่ต่างๆ ในการปฏิบัติงานเป็นอย่างดี โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษาทั้ง 2 ท่านขอกราบขอบพระคุณไว้ ณ ที่นี้

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา และมารดา ผู้บังเกิดกล้า ผู้เป็นแรงกำลังอันยิ่งใหญ่ทั้งกำลังใจ กำลังทรัพย์ และเป็นผู้ให้ตลอดมา

อนึ่ง ประโยชน์และคุณความดีใดๆก็ตามที่เกิดจากปริญญานิพนธ์ฉบับนี้ ขอมอบให้แด่บิดา มารดา ซึ่งเป็นผู้ให้กำเนิดและครูอาจารย์ที่ได้ประสิทธิ์ประสาทวิชามาตั้งแต่ต้นจนมีวันนี้

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

เนื่องจากปัจจุบันการพัฒนาทางด้านเทคโนโลยีมีความเจริญก้าวหน้าไปอย่างรวดเร็ว จากความเจริญก้าวหน้านี้เองทำให้มนุษย์สามารถที่จะสร้างสรรค์งานต่างๆที่มีคุณภาพและประโยชน์ต่อมนุษย์และสิ่งแวดล้อมขึ้นได้อย่างมาก และที่สำคัญคือความสะดวกสบายในการดำรงชีวิต

ความเจริญก้าวหน้าของเทคโนโลยีสิ่งหนึ่งที่เข้ามามีบทบาทต่อชีวิตของมนุษย์เราอย่างมากคือคอมพิวเตอร์ และคอมพิวเตอร์ได้กลายเป็นแหล่งค้นคว้าข้อมูล ที่เก็บข้อมูล การประมวลผลที่สลับซับซ้อน การสร้างสรรค์และพัฒนางานด้านต่างๆซึ่งคอมพิวเตอร์สามารถสนองความต้องการของมนุษย์ได้อย่างรวดเร็ว

จากความก้าวหน้าทางเทคโนโลยีนี้เองก็ได้มีการผลิตและสร้างเครื่องอำนวยความสะดวกต่างๆขึ้น ซึ่งไม่ใช่เรื่องยากสักเท่าไรนัก แต่เมื่อลองมองย้อนกลับไปอีกทีพบว่า เครื่องมือเครื่องใช้เหล่านี้ล้วนแต่มีราคาแพงเนื่องจากราคาต้นทุนที่สูงขึ้นของวัสดุหรืออาจจะเป็นอุปกรณ์ที่นำเข้าจากต่างประเทศ เราจะทำอย่างไรจึงทำให้เครื่องใช้เหล่านี้มีราคาถูกลง

วัตถุประสงค์ของปริญญานิพนธ์นี้จัดทำขึ้นเพื่อที่จะลดต้นทุนของสินค้าและราคาการนำเข้าของสินค้าที่มีราคาแพงเช่นเครื่องถ่ายสติกเกอร์ โดยได้นำเอา Visual C++ ซึ่งเป็นภาษาหนึ่งที่ใช้สื่อสารกับคอมพิวเตอร์ที่เกิดขึ้นมาพร้อมกับเทคโนโลยี มาใช้สร้างเป็นเครื่องถ่ายสติกเกอร์ โดยการใช้งานร่วมกันระหว่างกล้องถ่ายภาพที่เรียกว่า CCD Camera การ์ดที่ใช้ติดต่อกับเครื่องคอมพิวเตอร์และพร้อมทั้งเครื่องพิมพ์ภาพที่มีความละเอียดสูง ซึ่งขีดความสามารถของเครื่องถ่ายสติกเกอร์ที่สร้างขึ้นนี้มีคุณภาพไม่ยิ่งหย่อนไปกว่าของต่างประเทศเท่าไรนัก และจุดประสงค์อีกข้อก็คือทำให้ได้ศึกษาการทำงานของโปรแกรม Visual C++ ซึ่งนับว่ามีประโยชน์อย่างมากในการที่จะใช้ควบคุมสั่งการให้คอมพิวเตอร์หรืออุปกรณ์ต่างๆ ทำงานได้ตามที่เราต้องการ

1.2 ขอบเขตของปริญญานิพนธ์

1. สามารถถ่ายภาพโดยใช้กล้องรับภาพเพื่อนำภาพมาทำเป็นสติกเกอร์ได้
2. สามารถเลือกเฟรมได้หลายแบบโดยสามารถสร้างรูปแบบเฟรมได้ตามต้องการ
3. สามารถเปิดภาพสติกเกอร์ที่ถ่ายไว้แล้วมาทำการพิมพ์บนกระดาษสติกเกอร์ได้ใหม่

4. สามารถยกเลิกการทำงานในขั้นตอนปัจจุบันเพื่อย้อนกลับไปทำขั้นตอนที่ผ่านมาได้
5. สามารถพิมพ์ภาพสติกเกอร์ที่มีความสมจริงกับภาพต้นฉบับที่ต้องการได้

1.2 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้ได้แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษาและทำความเข้าใจ ในแต่ละบทจะประกอบไปด้วยเนื้อหาที่สำคัญดังนี้

บทที่ 2 กล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับโครงงาน โดยจะใช้โปรแกรม Visual C++ 6.0 เป็นหลักในการสร้างโปรแกรมที่ทำเป็นโปรแกรมสติกเกอร์

บทที่ 3 การออกแบบ การสร้างและการทำงาน โดยกล่าวอธิบายถึงขั้นตอนการออกแบบทางฮาร์ดแวร์และการออกแบบทางด้านซอฟแวร์พร้อมทั้งบล็อกไดอะแกรมการทำงานของโปรแกรมที่สร้างขึ้นเป็นส่วนๆ

บทที่ 4 การทดลองและผลการทดลอง เป็นการทดลองใช้ซอฟแวร์แต่ละส่วนและผลการทดสอบประสิทธิภาพของเครื่องถ่ายสติกเกอร์

บทที่ 5 กล่าวสรุปการทำปริญญานิพนธ์ ปัญหาที่เกิดขึ้นตั้งแต่เริ่มต้นทำปริญญานิพนธ์ วิธีการแก้ปัญหาที่เกิดขึ้นข้อเสนอแนะและแนวทางในการทำปริญญานิพนธ์ไปพัฒนาต่อไป

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข ผังการทำงานและโปรแกรม

ภาคผนวก ค รายละเอียดและคุณสมบัติของอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

เนื้อหาในปริญญานิพนธ์ในบทนี้เป็นทฤษฎีและหลักการที่นำมาใช้ประกอบการสร้างโครงงาน โดยประกอบด้วยประวัติของ Visual C++ และ MFC (Microsoft Foundation Class) หลักการและรายละเอียดการใช้งานในแต่ละส่วนของโปรแกรม Visual C++ Version 6.0 ซึ่ง เนื้อหาในบทนี้จะใช้เป็นพื้นฐานในการนำไปสร้างและออกแบบซอฟต์แวร์เครื่องถ่ายสติกเกอร์

2.2 ประวัติของ Visual C++ และ MFC

Visual C++ ได้รับการพัฒนาขึ้นมาจาก Microsoft C/C++ ให้เป็น IDE ที่ทำงานบนระบบปฏิบัติการวินโดวส์ได้อย่างเต็มที่ รองรับการพัฒนาโปรแกรมบนวินโดวส์โดยมี MFC (Microsoft Foundation Class) เป็นไลบรารี่ที่จะช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดวส์

ในยุคที่ MFC ยังไม่ถือกำเนิด เมื่อเราต้องการจะพัฒนาโปรแกรมบนวินโดวส์เราจะต้องใช้ SDK (Software Development Kit) และคอมไพลเลอร์ภาษา C เช่น Microsoft C++ , Borland C++ ช่วยในการเขียนโปรแกรม โค้ดโปรแกรมที่เขียนขึ้นด้วย SDK นี้จะค่อนข้างซับซ้อนและสร้างความลำบากในการศึกษาทำความเข้าใจ จึงได้มีการสร้างคลาสขึ้นชุดหนึ่ง เขียนขึ้นโดยใช้โครงสร้างแบบ OOP (Objected Oriented Programming) ด้วยภาษา C++ ชื่อว่า MFC (Microsoft Foundation Class) ใช้สำหรับอำนวยความสะดวกในการเขียนโปรแกรมบนวินโดวส์โดยเฉพาะ โค้ดโปรแกรมที่เขียนขึ้นโดยใช้ MFC นั้น จะมีขนาดเล็กและไม่ซับซ้อน ทำให้การพัฒนาโปรแกรมบน วินโดวส์สามารถทำได้ง่ายกว่าการใช้ SDK

OOP คือ Object Oriented Programming เป็นวิธีการเขียนโปรแกรม โดยอาศัยแนวคิดของวัตถุชิ้นหนึ่ง ลักษณะการเขียนโปรแกรมจะเป็นแบบโครงสร้าง (Structure) การเขียนโปรแกรมแบบ OOP มีความสามารถในการปกป้องข้อมูลและการสืบทอดคุณสมบัติ ซึ่งทำให้แนวโน้มของ OOP ได้รับการยอมรับและพัฒนามาใช้ในระบบต่างๆ มากมาย เช่น ระบบปฏิบัติการวินโดวส์ 3.11, หรือวินโดวส์ 9X เป็นต้น

ข้อดีของการเขียนโปรแกรมด้วย Visual C++ และ MFC อีกข้อหนึ่งก็คือ ความสามารถในการพอร์ตเทเบิล (Portability) หมายความว่าหากเรามีซอร์สโค้ดที่เขียนด้วย Visual C++ ในเวอร์ชั่นที่ต่ำกว่า เราสามารถนำมาคอมไพล์และลิงค์ใหม่ได้ โดยใช้ Visual C++ 6

2.3 สภาพแวดล้อมใน Visual C++ 6.0

2.3.1 โปรเจ็กต์เวิร์กสเปซ (Project Workspace)

โปรเจ็กต์ไฟล์ของ Visual C++ 6 เราจะเรียกว่า "เวิร์กสเปซ" โดยจะใช้นามสกุล .dsw เป็นไฟล์ที่เก็บตัวเลือกต่างๆ ของโปรเจ็กต์และเราก็สามารถโหลดโปรเจ็กต์ที่เขียนด้วย Visual C++ เวอร์ชั่นที่ต่ำกว่านี้ได้

ประโยชน์ของโปรเจ็กต์ไฟล์

1) โปรเจ็กต์ไฟล์จะเก็บรายชื่อของไฟล์ที่เป็นซอร์สโค้ดโปรแกรมทั้งหมด ที่ใช้ร่วมกันในโปรเจ็กต์ เช่น ซอร์สโปรแกรม .h, .cpp รวมทั้งไฟล์ฐานข้อมูลโปรแกรมที่ใช้ใน ClassWizard เป็นต้น

2) โปรเจ็กต์ไฟล์จะเก็บค่าตัวเลือกสำหรับการคอมไพล์และการลิงค์เอาไว้ เพื่อบอกให้รู้ว่าโปรเจ็กต์นี้จะทำการคอมไพล์และลิงค์กับไลบรารี่ใด หรือมีการสร้างส่วนประกอบอื่นๆ อีกหรือไม่ เช่น ส่วนประกอบในแก้ไขข้อมูล

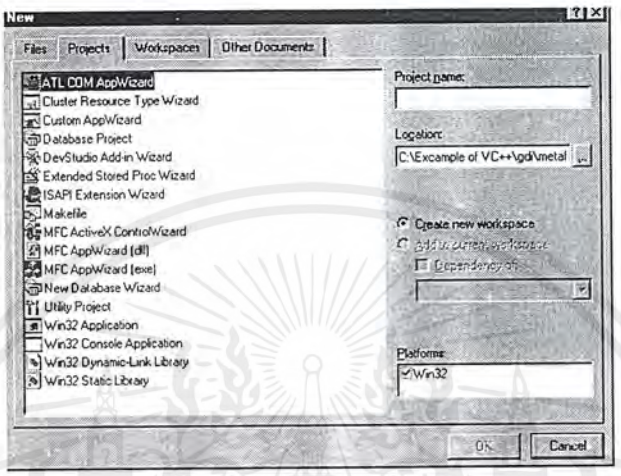
3) โปรเจ็กต์ไฟล์จะเก็บค่าตัวเลือกที่แสดงว่าโปรเจ็กต์นี้เป็นโปรเจ็กต์แบบใดเมื่อทำการคอมไพล์ เช่น Windows Application (.EXE) หรือ Dynamic-Link Library(.DLL) เป็นต้น

2.3.2 การจัดการโปรเจ็กต์เวิร์กสเปซ

ในการเริ่มต้นเขียนโปรแกรมด้วย Visual C++ ทุกครั้งเราจะต้องทำการสร้างโปรเจ็กต์เวิร์กสเปซ (Project Workspaec) ขึ้นมาก่อน โดยเราสามารถเลือกได้ว่าจะสร้างโปรเจ็กต์แบบใด ซึ่งมีให้เลือกหลายแบบ ในการสร้างโปรเจ็กต์เวิร์กสเปซสำหรับโปรแกรมใหม่ เราจะเลือกคำสั่ง File>New ไดอะล็อกซ์จะปรากฏดังรูป New ดังรูปที่ 2.1

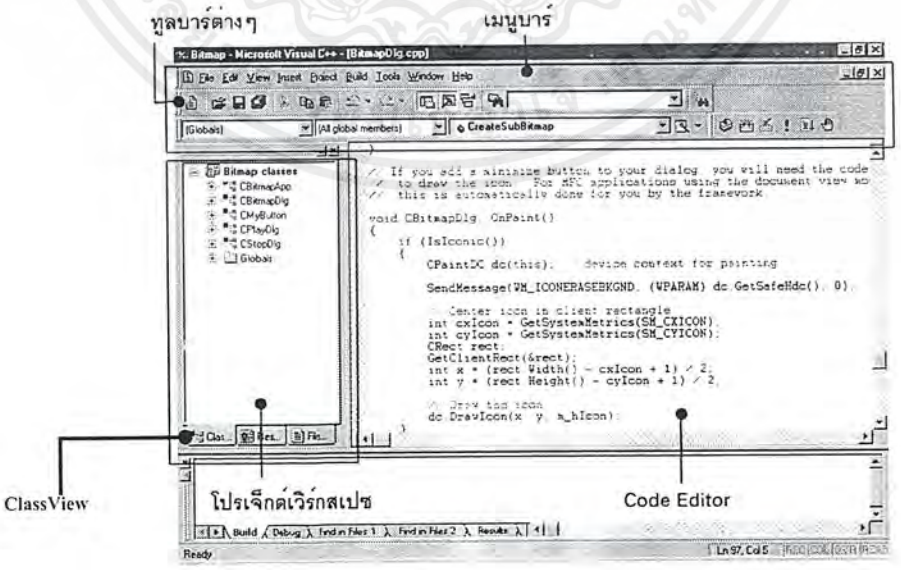
รูปที่ 2.1 การสร้างโปรเจ็กต์เวิร์กสเปซ

จากไดอะล็อกซ์ New ดังรูปที่ 2.2 เราจะพบกับตัวเลือกของโปรเจ็กต์แบบต่างๆ ซึ่งเราสามารถเลือกได้ว่า เราจะสร้างโปรเจ็กต์แบบใด ซึ่งจะมีอยู่ด้วยกัน 14 ตัวเลือก โดยเราจะใช้ MFC AppWizard และ Win32 Application



รูปที่ 2.2 ส่วนแสดงClassView, Resource View และ FileView

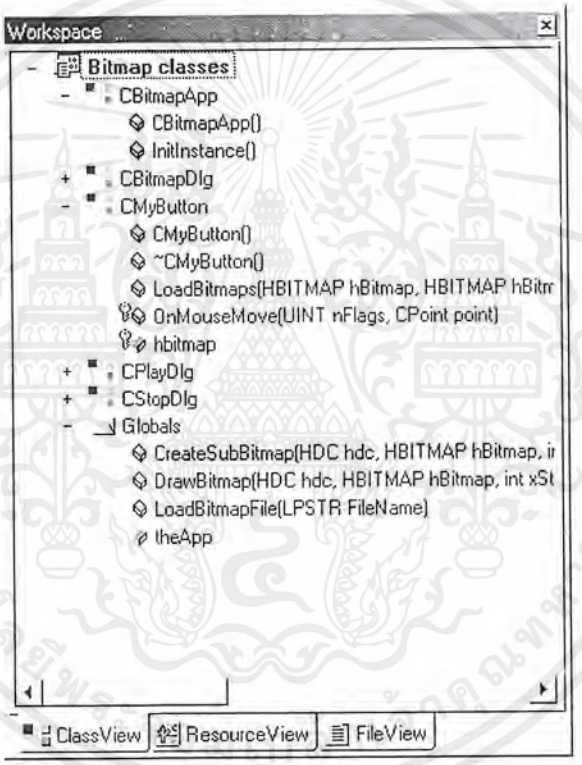
เมื่อเราได้สร้างโปรเจ็กต์เวิร์กสเปซใหม่แล้ว เราจะพบว่าภายในโปรเจ็กต์เวิร์กสเปซประกอบไปด้วยแท็บซ้อนๆกัน 3 แท็บด้วยกันคือ ClassView, Resource View และ FileView ซึ่งโดยปกติแล้ว ถ้ายังไม่มีการสร้างและแก้ไขโปรเจ็กต์ใดๆ จะไม่มีแท็บใดปรากฏเลย ดังรูปที่ 2.3



รูปที่ 2.3 ส่วนแสดงClassView, Resource View และ FileView

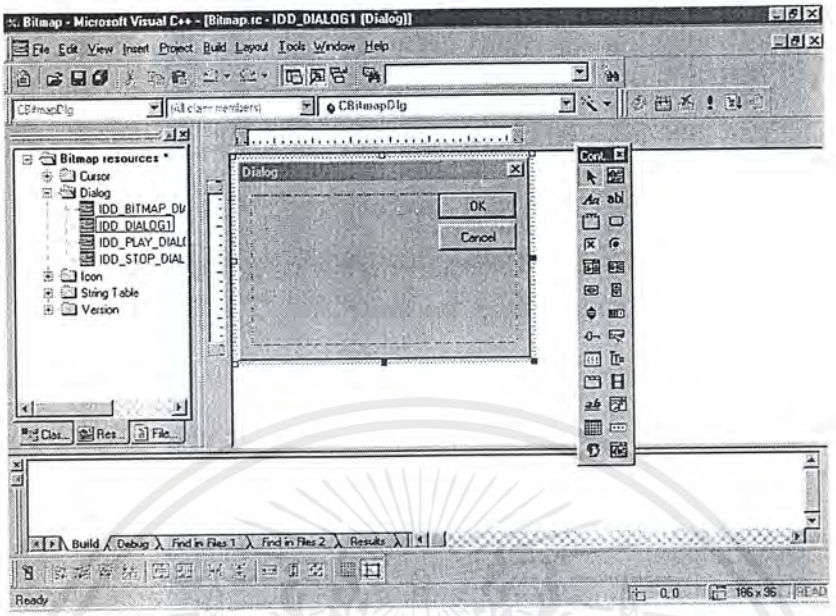
1) ClassView จะใช้สำหรับแสดงรายละเอียดภายในของคลาสต่างๆ ที่อยู่ในโปรเจ็กต์นี้ เราสามารถทราบได้ว่ามีคลาสอะไรบ้างในโปรเจ็กต์ และสมาชิก คลาสแต่ละตัวเป็นแบบ Protect หรือแบบ Public และตัวแปรโกลบอลมีอะไรบ้าง โดยจะใช้สัญลักษณ์ต่อดังไปนี้ ในการบอกระดับของสมาชิก ดังรูปที่ 2.4

- รูปแม่กุญแจ หมายถึง สมาชิกคลาสที่เป็นแบบ private
- รูปดอกกุญแจ หมายถึง สมาชิกคลาสที่เป็นแบบ protect
- รูปกล่อง หมายถึง สมาชิกคลาสเป็นแบบ public



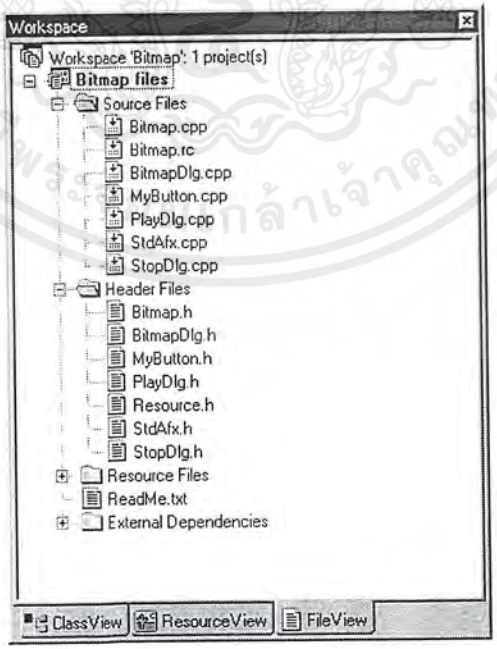
รูปที่ 2.4 รายละเอียดภายในของคลาสต่างๆ

2) Resource View ใช้สำหรับแสดงถึงรีซอร์ส ที่มีในโปรเจ็กต์ เช่น รูปไอคอนของโปรแกรม ไดอะล็อกซ์ ทูลบาร์ เมนู ซึ่งสิ่งเหล่านี้เราสามารถสร้างขึ้นโดยใช้รีซอร์สอิดิตเตอร์ ดังรูปที่ 2.5



รูปที่ 2.5 ไอคอนของโปรแกรม ไดอะล็อกซ์ ทูลบาร์ เมนูของ Resource View

3) **FileView** ใช้สำหรับแสดงรายชื่อไฟล์ที่อยู่ในโปรเจ็กต์ขณะนั้น โดย FileView จะแยกไฟล์ส่วนหัว (.H) ไฟล์โปรแกรม (.CPP) และไฟล์รีซอร์ส (.ICO, RC2) เอาไว้ต่างหากด้วย ดังรูปที่ 2.6



รูปที่ 2.6 รายชื่อไฟล์ที่อยู่ในโปรเจ็กต์โดย FileView

2.3.3 คอมไพเลอร์และลิงค์เกอร์

ในการพัฒนาโปรแกรมบนวินโดวส์นั้นจะต่างจากการพัฒนาโปรแกรมในระบบดอส เช่น การคอมไพล์ซอร์สโค้ด และการคอมไพล์รีซอร์ส เพราะฉะนั้นเครื่องมือหลักๆ ที่ใช้ในกระบวนการ Build ของ Visual C++ มีดังนี้

1) C/C++ Compiler เป็นคอมไพล์เลอร์ภาษา C/C++ ของ Visual C++ ซึ่งจะเป็นโปรแกรมตรวจสอบข้อผิดพลาดของซอร์สโค้ด หากซอร์สโค้ดโปรแกรมมีข้อผิดพลาด คอมไพเลอร์จะแสดงข้อผิดพลาดนั้นออกมาและให้เราแก้ไขให้ถูกต้องก่อนนำไปคอมไพล์ใหม่ ผลที่ได้จากการคอมไพล์นั้นจะอยู่ในรูปของ ออบเจ็กส์ไฟล์ (Object File)

2) Resource Compiler เป็นตัวแปลภาษาของรีซอร์สสคริปต์ (Resource Scripts) เพราะในการพัฒนาโปรแกรมบนวิสโดวส์ จะต้องมีการการสร้างรีซอร์ส เช่น ไอคอน รูปภาพ หรือไดอะล็อกซ์ รีซอร์สคอมไพล์เลอร์ตัวนี้จะเป็นโปรแกรมที่ตรวจเช็คความถูกต้องของรีซอร์ส ผลที่ได้จากการคอมไพล์ก็คือ ออบเจ็กต์ไฟล์ของรีซอร์ส (.res)

3) Linker ลิงค์เกอร์ของ Visual C++ จะเป็นโปรแกรมที่ทำหน้าที่รวบรวมออบเจ็กต์ไฟล์ที่ได้จากการคอมไพล์ ออบเจ็กส์ของรีซอร์สไฟล์และไลบรารี่ไฟล์ มาสร้างเป็นไฟล์ Execute ในที่สุด

ขั้นตอนการ Build โปรแกรมของ Visual C++ นั้นจะเริ่มต้นที่การคอมไพล์รีซอร์สเสียก่อน จากนั้นคอมไพล์จะทำการแปลภาษา (ตรวจสอบซอร์สโค้ด) และสร้างออบเจ็กต์ไฟล์ขึ้นมาจากนั้นลิงเกอร์ก็จะทำการเชื่อมโยงไฟล์เหล่านั้นกับไลบรารี่ไฟล์ให้เป็นไฟล์ของนามสกุล Execute

2.3.4 ทูลบาร์

ทูลบาร์ของโปรแกรม Visual C++ มีมากมายและหลายหน้าที่การใช้งาน ซึ่งครอบคลุมการพัฒนาโปรแกรมและกระบวนการต่างๆ อย่างพร้อมสรรพ ทูลบาร์หลักๆ ของโปรแกรมจะมีอยู่ 8 ทูลบาร์ด้วยกัน ดังนี้

1) Standard เป็นทูลบาร์ที่ประกอบด้วยคำสั่งพื้นฐาน เช่น New, Open, Save, Copy, Paste,Undo,Redo เช่นเดียวกับทูลบาร์ของโปรแกรมไมโครซอฟต์เวิร์ด ดังรูปที่ 2.7

รูปที่ 2.7 ทูลบาร์ของโปรแกรม Visual C++

2) **Build** เป็นทูลบาร์ที่ใช้ในการกำหนดตัวเลือกต่างๆของโปรเจ็กต์และคอมไพล์โปรเจ็กต์ ดังรูปที่ 2.8

รูปที่ 2.8 ทูลบาร์ Build ใช้ในการกำหนดตัวเลือกต่างๆของโปรเจ็กต์

2.3.5 สแตติก ฟอนต์ และปุ่มกด

หลักในการสร้างคอนโทรลมีอยู่ 4 ขั้นตอนด้วยกัน ซึ่งเราสามารถยึดหลักทั้ง 4 นี้ในการสร้างคอนโทรลตัวใดๆ ก็ได้ไม่ว่าเราจะสร้างคอนโทรลสแตติก ปุ่มกด สกอร์ลบาร์ หรืออิดิตบ็อกซ์

ขั้นตอนทั้ง 4 มีดังนี้

ขั้นตอนที่ 1 Declaration คือการประกาศออบเจ็กต์ของคอนโทรลคลาส

ขั้นตอนที่ 2 Allocation คือการจองหน่วยความจำให้กับคอนโทรล

ขั้นตอนที่ 3 Creation คือการสร้างคอนโทรล(และการทำอัพเดตสำหรับคอนโทรลบางตัว)

ขั้นตอนที่ 4 Destroy คือการทำลายหรือคืนหน่วยความจำให้กับระบบเมื่อจบการทำงาน

2.3.6 การสร้างคอนโทรลสแตติก

คอนโทรลสแตติกเป็นคอนโทรลประเภทหนึ่งที่ไม่มีการเปลี่ยนแปลงตัวมันเอง เช่น รูปภาพ, ข้อความ, ตัวอักษร หรือกรอบสี่เหลี่ยม เป็นต้น เราจะต้องใช้คลาส CStatic ในการสร้างและควบคุมการแสดงผลของคอนโทรลสแตติกนี้

ขั้นตอนที่ 1 Declaration

อันดับแรกของการสร้างคอนโทรลนั่นก็คือ การประกาศออบเจกต์ของคอนโทรลคลาสที่ต้องการสร้าง ดังบรรทัดต่อไปนี้

```
CStatic *st;
```

คอนโทรลที่เราต้องการสร้างคือคอนโทรลสแตติก คลาสที่ใช้สำหรับสร้างคอนโทรลสแตติกนี้ก็คือคลาส CStatic เพราะฉะนั้นเราจึงต้องประกาศออบเจ็กต์ของคลาสนี้เพื่อใช้ในการสร้างคอนโทรล โดยกำหนดให้ st เป็นออบเจ็กต์ของคลาส CStatic และได้ประกาศให้อยู่ในรูปของพอยน์เตอร์ เพื่อที่จะได้ใช้เราสมบัติของพอยน์เตอร์ในการจองหน่วยความจำให้กับคอนโทรล

ขั้นตอนที่ 2 Allocation

เมื่อเราได้ประกาศออบเจ็กต์ของคอนโทรลคลาสเป็นที่เรียบร้อย ขั้นตอนต่อไปก็คือการจองหน่วยความจำ เราจะใช้ออบเจ็กต์ของคลาส st และคำสั่ง new ดังบรรทัดต่อไปนี้

st=new CStatic();

st ได้ทำการจองหน่วยความจำให้กับคอนโทรล โดยทำการ new ไปยังคอนสตรัคเตอร์ของคลาส CStatic ในบรรทัดนี้หน่วยความจำจะถูกจองเอาไว้โดยออบเจ็กต์ st เรียบร้อยแล้วเพราะฉะนั้นเมื่อโปรแกรมนี้สิ้นสุดการทำงาน เราจะต้องเขียนโค้ดโปรแกรมสำหรับคืนหน่วยความจำให้กับระบบด้วย โดยใช้คำสั่ง delete

ขั้นตอนที่ 3 Creation

สำหรับวิธีการสร้างคอนโทรลนั้น เราจะต้องเรียกเมมเบอร์ฟังก์ชั่นที่ใช้สำหรับสร้างคอนโทรลภายในคลาสนั้นๆ โดยเฉพาะเมมเบอร์ฟังก์ชั่นที่ใช้ในการสร้างคอนโทรลของคลาสแต่ละคลาสนั้นจะใช้ที่ชื่อ Create()

การเรียกใช้ฟังก์ชั่น Create() นั้นเราจะเรียกชื่อฟังก์ชั่นขึ้นมาลอยๆไม่ได้ เราจะต้องผ่านค่าพารามิเตอร์ไปให้กับฟังก์ชั่นด้วย พารามิเตอร์แต่ละตัวจะระบุถึงรูปแบบของการแสดงผลขนาดของคอนโทรลและอื่นๆ ซึ่งเราสามารถกำหนดได้ด้วยตนเองว่าจะให้คอนโทรลมีการแสดงผลเป็นอย่างไร บรรทัดต่อไปนี้แสดงการเรียกฟังก์ชั่น CStatic...Create() เพื่อสร้างคอนโทรลสแตติก st->Create "Easy Visual C ++"

```
WS_CHILD¦WS_VISIBLE¦SS_CENTER,
CRect(50,80,150,150),this);
```

พารามิเตอร์ที่ผ่านไปยังฟังก์ชั่น CStatic::Create() นั้นมีด้วยกัน 4 ตัวคือ

1. "Easy Visual C++" ก็คือข้อความที่ต้องการแสดง
2. WS_CHILDIWS_VISIBLEISS_CENTER ก็คือรูปแบบของข้อความ
3. Crect (50,80,150,150) ก็คือจุดเริ่มต้นและขนาดของกรอบโดยอ้างอิงกับรูปสี่เหลี่ยมโดยใช้คลาส Crect เช่นเดียวกับการกำหนดจุดเริ่มต้นและขนาดของดวินโดวส์
4. this ความหมายของพารามิเตอร์ this ในที่นี้คือ "วินโดวส์หลัก"

ในการกำหนดค่าให้กับพารามิเตอร์แต่ละตัวของฟังก์ชั่น Create() นั้น เราจำเป็นจะต้องรู้เสียก่อนว่าพารามิเตอร์แต่ละตัวมีหน้าที่อย่างไร และสามารถเปลี่ยนแปลงเป็นค่าใดๆ ได้บ้างเราอาจจะพบกับการกำหนดค่าพารามิเตอร์ด้วยค่าคงที่ที่เราไม่เคยรู้จักมาก่อน หรือการผ่านค่า

พารามิเตอร์ด้วยการเรียกฟังก์ชั่นอีกตัวหนึ่ง เมื่อเราต้องการการค้นหาหรือศึกษารายละเอียดของพารามิเตอร์และฟังก์ชั่นต่างๆ ของคลาสให้เราเปิด MSDN Library และค้นหาฟังก์ชั่นที่ชื่อ Create() ของคลาส CStatic เราจะพบกับรายละเอียดของฟังก์ชั่น CStatic::Create ดังนี้

```
CStatic:: Create
        BOOL Create (LPCSTR lpText,DWORD dwStyle,
                Const RECT&rect,CWnd pParentWnd  pParentWnd,UINT nID=oxffff)
```

รายละเอียดของแต่ละพารามิเตอร์ของ CStatic::Create() แต่ละตัวมีดังนี้

lpText	ข้อความที่จะแสดงหากไม่ต้องการแสดงข้อความ ให้ใส่ค่า NULL
dwStyle	รูปแบบของการแสดงผลของคอนโทรลที่ MFC ได้กำหนดเอาไว้
rect	คือจุดเริ่มต้นและขนาดของคอนโทรล อ้างอิงกับรูปสี่เหลี่ยมโดยใช้คลาส Crect
pPaentWnd	คือ Parent Window (วินโดวส์หลัก)
nID	ค่าคงที่ Control ID ใช้สำหรับอ้างถึงตัวคอนโทรลตัวที่เราสร้างขึ้น แต่ถ้าโปรแกรมไม่มีการเปลี่ยนแปลงข้อความในคอนโทรล เราก็ไม่จำเป็นต้องใส่พารามิเตอร์เพื่ออ้างอิงถึงคอนโทรลตัวนี้

2.3.7 รูปแบบการแสดงผลของคอนโทรลสแตติก (dwStyle)

รูปแบบของสแตติกคอนโทรลนั้นสมารถเปลี่ยนแปลงได้หลายๆ รูปแบบด้วยกันโดยใช้มาโครที่ทาง MFC ได้กำหนดไว้ให้ใส่ลงในพารามิเตอร์ตัวที่ 2 ของฟังก์ชั่น Create() (ซึ่งก็คือ dwStyle)

```
WS_CHILD¦WS_VISIBLE¦SS_CENTER
```

พารามิเตอร์ในข้างต้น มีค่าคงที่หรือมาโคร WS_CHLID และ WS_VISIBLE แทรกเอาไว้เป็นรูปแบบการแสดงผลทั่วๆไป ซึ่งในการสร้างคอนโทรลตัวหนึ่งๆ นั้น อย่างน้อยเราจะต้องใส่

ค่าคงที่ 2 ตัวนี้เอาไว้ด้วย ซึ่งค่าคงที่สองตัวนี้สืบทอดมาจากคลาส CWnd ทำหน้าที่ควบคุมการแสดงผลของคอนโทรลนอกเหนือจากการควบคุมของคลาส CStatic

2.3.8 มาโครที่สืบทอดมาจากคลาส CWnd ที่ใช้ควบคุมคอนโทรล

WS_CHILD	วินโดวส์ลูก (Child Window)
WS_VISIBLE	แสดงคอนโทรลออกทางจอภาพ (Enable)
WS_DISABLED	ไม่แสดงคอนโทรลออกทางจอภาพ(Disable)
WS_BORDER	แสดงกรอบล้อมรอบคอนโทรลตามขนาด Crect

วิธีการกำหนดรูปแบบของคอนโทรลนั้น เราจะต้องการให้มาโครเหล่านี้มารวมกันโดยใช้เครื่องหมาย OR (<Shift+¦>) คั่นระหว่างกัน เช่น หากเราต้องการให้คอนโทรลสแตติกแสดงข้อความอยู่กึ่งกลางและมีกรอบล้อมรอบเราก็จะต้องแทรกมาโครลงในพารามิเตอร์ตัวที่สองของฟังก์ ชั่น Create() ซึ่งมีลักษณะดังนี้

```
WS_CHILD¦WS_VISIBLE¦WS_BORDER¦SS_CENTER
```

เมื่อเราใส่พารามิเตอร์ตามข้างต้นแล้ว เมื่อกดปุ่มเพื่อดูผลการทำงานรอบๆ ข้อความจะมีกรอบล้อมรอบไปด้วยนั่น ก็เป็นเพราะว่าเราได้แทรกมาโคร WS_BORDER รวมเข้าไปด้วยรูปแสดงผลการรันโปรแกรมเมื่อแทรกมาโคร WS_BORDER

รูปแบบการแสดงผล SS_CENTER เป็นตัวกำหนดตำแหน่งการวางตัวอักษร โดย SS_CENTER จะวางตัวอักษรให้อยู่กึ่งกลางของขนาดคอนโทรล (Crect) รูปแบบการวางตัวอักษรนี้เราสามารถเปลี่ยนไปเป็นรูปแบบอื่นๆได้อีก เช่น แทรกมาโคร SS_LEFT หรือ SS_RIGHT เพื่อจัดให้ข้อความอยู่ในตำแหน่งชิดซ้ายหรือชิดขวาตามต้องการ

ตัวอย่างของการใช้มาโคร SS_LEFT ก็คือ

```
WS_CHILD¦WS_VISIBLE¦WS_BORDERD¦SS_LEFT
```

รูปแบบการแสดงผลข้อความของคลาส CStatic มีมากมายนอกเหนือจากมาโคร SS_LEFT,SS_RIGHT หรือ SS_CENTER แล้วเราสามารถเลือกใช้ได้อีกดังนี้ (SS ย่อมาจาก "Static Style")

SS_BLACKFRAME	SS_BLACKRECT	SS_CENTER
SS_GRAYFRAME	SS_GRAYRECT	SS_LEFT
SS_LEFTNOWORDWSRAP	SS_NOPREFIX	SS_RIGHT
SS_SIMPLE	SS_WHITEFRAME	SS_WHITERECT

2.3.9 การเปลี่ยนตัวอักษรโดยใช้คลาส Cfont

คลาส CStatic สามารถแสดงข้อความที่เป็นแบบดีฟอลต์ของระบบ และยังสามารถใส่กรอบล้อมรอบข้อความได้ รวมทั้งเอาลักษณะอื่นๆ เช่นการควบคุมการตัดคำเมื่อข้อความยาวล้นขนาดของคอนโทรล แต่คลาส CStatic ไม่มีความสามารถในการที่จะขยายตัวอักษรหรือเปลี่ยนรูปแบบของตัวอักษรให้เป็นตัวอักษรแบบอื่นได้ก็เพราะอักษรที่ปรากฏจากการสร้างข้อความโดยใช้คลาส CStatic สามารถเป็นได้แค่เพียงตัวอักษรแบบดีฟอลต์เท่านั้น เพราะฉะนั้นในการเปลี่ยนรูปแบบของตัวอักษรเราจึงต้องใช้คลาส Cfont เข้ามาช่วย ซึ่งคลาส Cfont นั้นสามารถที่จะโหลดเอาฟอนต์ที่มีอยู่ในระบบขณะนั้นมาแสดงผลได้ อีกทั้งเรายังสามารถกำหนดขนาด ตัวหนา หรือตัวเอียงได้อีกด้วย

วิธีการเปลี่ยนรูปแบบของตัวอักษรโดยใช้คลาส Cfont และคลาส CStatic ซึ่งขั้นตอนมีดังนี้

1. สร้างออบเจ็กต์ st จากคลาส CStatic และทำการสร้างข้อความไปตามปกติ
2. สร้างออบเจ็กต์ font จากคลาส Cfont
3. จองหน่วยความจำให้กับคอนโทรล
4. เรียกฟังก์ชั่น CreateFont() เพื่อกำหนดขนาด, รูปแบบและชื่อของฟอนต์ในระบบที่ต้องการแสดงเก็บไว้ในออบเจ็กต์ font
5. เรียกฟังก์ชั่น SetFont() เพื่อกำหนดฟอนต์ให้กับออบเจ็กต์ st จากนั้นก็ทำการ delete คอนโทรลทั้งสองเพื่อคืนหน่วยความจำให้กับระบบ

nHeight	พารามิเตอร์ตัวแรกคือขนาดของฟอนต์ จากตัวอย่างกำหนดให้มีขนาดเป็น 36			
nWeight	พารามิเตอร์ตัวที่ 5 คือน้ำหนักหรือความหนา ซึ่งได้กำหนดค่าเอาไว้ดังนี้			
	FW_DONTCARE	0	FW_THIN	100
	FW_EXTRALIGHT	200	FW_ULTRALGHT	200
	FW_LIGHT	300	FW_NORMAL	400
	FW_REGULAR	400	FW_MEDIUM	500
	FW_SEMIBOLD	600	FW_DEMIBOLD	600
	FW_BOLD	700	FW_EXTRABOLD	800
	FW_ULTRABOLD	800	FW_BLACK	900
	FW_HEAVY	900		
bItalic	กำหนดรูปแบบของฟอนต์ให้เป็นตัวเอียง (TRUE หรือ FALSE)			
bUnderline	กำหนดรูปแบบของฟอนต์ให้ขีดเส้นใต้ (TRUE หรือ FALSE)			

2.3.10 การสร้างปุ่มกดโดยใช้คลาส CButton

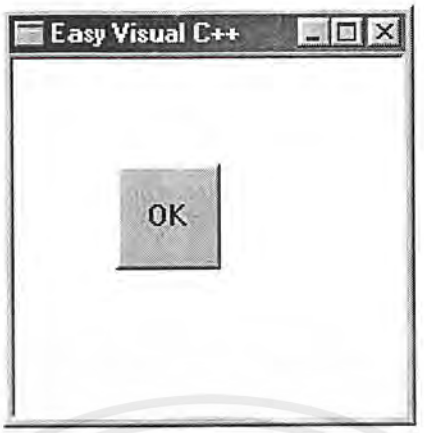
คลาส CButton เป็นคลาสที่ใช้สำหรับสร้างคอนโทรลปุ่มกด โดยวิธีในการสร้างจะมีลักษณะเหมือนกันกับคอนโทรลสแตติก แต่การสร้างปุ่มกดนั้น เราจะต้องกำหนดค่าคงที่ Control ID ให้กับปุ่มกดแต่ละปุ่มด้วย โปรแกรมต่อไปนี้เป็นการสร้างคอนโทรลปุ่มกดในเฟรมวินโดวส์

```
# include "afxwin.h"
#define IDC_BUTTON1 100
class CApp:public CWinApp{
        public:
                virtual BOOL InitInstance();
};
CApp app;
class CWin:public CFrameWnd{
        CButton *button;
        public:
                CWin();
```

```
                ~CWin();
};
BOOL  CApp:: InitInstance(){
        m_pMainWnd=new  CWin();
        m_pMainWnd->ShowWindow(m_nCmdShow);
        m_pMainWnd->UpdateWindow();
        return   TRUE;
};
CWin:: CWin(){
        Create(NULL,"Easy Visual C++,
                WS_OVERLAPPEDWINDOW,CRect( 0,0,200,200));
        button=new CButton();
        button->Create("OK",
                WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
                Crect(50,50,100,100),this,IDC_BUTTON1);
}
CWin :: ~CWin(){
        delete  button;
}
```

ขั้นตอนการสร้างปุ่มกด

1. มีการประกาศค่าคงที่ IDC_BUTTON1 โดยเราจะต้องประกาศให้มีค่ามากกว่า 99 เพราะค่าที่น้อยกว่านั้นทางระบบจะสงวนเอาไว้ใช้
2. สร้างออบเจ็กต์ button จากคลาส CButton
3. จองหน่วยความจำให้กับคอนโทรล และสร้างปุ่มกดโดยเรียกฟังก์ชั่น CButton::Create()
4. delete ออบเจ็กต์ button เพื่อคืนหน่วยความจำ



รูปที่ 2.9 การสร้างปุ่มกดของ Visual C++

2.3.11 การจัดการแมสเสจในวินโดวส์ (Message Map)

ขั้นตอนการแมปแมสเสจ

คอนโทรลปุ่มกดที่เราได้สร้างนั้นจะยังไม่สามารถใช้งานได้ หากเรายังไม่ได้จัดการกับแมสเสจของปุ่มกด กระบวนการจัดการกับแมสเสจนี้เราจะเรียกว่าการแมป ซึ่งกระบวนการแมปแมสเสจนี้จะต้องดำเนินการอยู่ภายในแมสเสจลูป (Message Loop) เพราะฉะนั้นเมื่อเราต้องการแมปปุ่มกดในโปรแกรม เราจะต้องประกาศมาโคร DECLARE _MESSAGE_MAP() ไว้ในส่วนหนึ่งของการประกาศวินโดวส์คลาสดังนี้

```
class CWin::CFrameWnd{
CButon *button;
public;
        CWin();
        ~CWin();
        DECLARE_MESSAGE_MAP();
};
```

มาโคร DECLARE_MESSAGE_MAP เป็นมาโครที่บอกให้คอมไพล์เลอร์ทราบว่า ภายในวินโดวส์นี้จะมีกระบวนการแมปแมสเสจรวมอยู่ด้วย เมื่อเราได้ประกาศมาโครนี้เสร็จแล้วเราจะ

ต้องสร้างลูปของการแมป (Message Loop) ขึ้นในโปรแกรมของเรา ลูปของการแมปมีลักษณะดังนี้

```
BEGIN_MESSAGE_MAP (วินโดวส์คลาส, คลาสแม่ของวินโดวส์คลาส)
        ส่วนที่ใช้ในการ MAP
END_MESSAGE_MAP()
```

บรรทัด BEGIN_MESSAGE_MAP() เป็นมาโครที่แสดงถึงจุดเริ่มต้นของลูป เราจะต้องใส่ค่าพารามิเตอร์ให้กับมาโครนี้สองตัวด้วยกัน พารามิเตอร์ตัวที่สองก็คือคลาสแม่ของคลาสในพารามิเตอร์ตัวแรก และจุดสิ้นสุดของลูปจะถูกปิดท้ายด้วยมาโคร END_MESSAGE_MAP เพราะฉะนั้น เราสามารถเขียนลูปของการแมปแมสเสจในโปรแกรม Button.cpp ได้ดังนี้

```
BEGIN_MESSAGE_MAP(CWin,   CFrameWnd)
        ……..
END_MESSAGE_MAP()
```

ยังมีกฎในการเขียนโค้ดของแมสเสจลูป นั่นก็คือ เราจะต้องเขียนด้วยตัวอักษรพิมพ์ใหญ่ทั้งหมดยกเว้นชื่อคลาสกับชื่อฟังก์ชั่น และในแต่ละบรรทัดของการแมปแมสเสจจะต้องไม่มีเครื่องหมายเซมิโคลอนต่อท้าย เมื่อเราเข้าใจในวิธีการสร้างแมสเสจลูปแล้ว ก่อนที่เราจะแมปแมสเสจให้กับคอนโทรลหรือสิ่งใดๆก็ตาม เราจะต้องรู้เสียก่อนว่าคอนโทรลตัวนั้นสามารถส่งแมสเสจใดกลับมาได้บ้าง ยกตัวอย่างแมสเสจของปุ่มกดที่สามารถส่งกลับมาได้จะมีอยู่ด้วยกัน 2 แมสเสจคือ

BN_CLICKED	จะถูกส่งกลับมาเมื่อปุ่มถูกกด
BN_DOUBLECLICKED	จะถูกส่งมาเมื่อปุ่มถูก Double Click

เมื่อเราต้องการตอบสนองต่อการ Click ปุ่มกดของเรา เราก็จะต้องแมปแมสเสจที่ชื่อ BN_CLICKED โดยวิธีที่ใช้ในการแมปแมสเสจ เราจะต้องเขียนให้อยู่ในรูปต่อไปนี้

ON_ชื่อแมสเสจที่ต้องการแมป (ค่า Control ID, ซึ่งฟังก์ชั่นที่ใช้รองรับการแมป)

ค่า Control ID นี้ก็คือค่าคงที่ที่เราได้ประกาศเอาไว้ในส่วนต้นโปรแกรม ค่า Control ID ของปุ่มกดก็คือ IDC_BUTTON1 นั่นเอง หน้าที่ของค่า Control ID นี้จะใช้สำหรับอ้างอิงถึงคอนโทรลตัวที่เราต้องการจะแมป

สำหรับฟังก์ชั่นที่ใช้รองรับการแมปก็คือฟังก์ชั่นที่บรรจุคำสั่งต่างๆ ที่ต้องการให้ปฏิบัติเมื่อมีการกระทำต่อคอนโทรลตัวนั้น ซึ่งฟังก์ชั่นที่ใช้รองรับการแมปคอนโทรลเหล่านี้ เราจะต้องเขียนขึ้นมาด้วยตนเอง และมีการประกาศ afx_msg นำหน้า Prototype ของฟังก์ชั่นในส่วนของการประกาศวินโดวส์คลาส

เมื่อเรารันโปรแกรม เราก็จะพบกับวินโดวส์ที่แสดงปุ่มกดเช่นเดิม แต่ในคราวนี้ เมื่อเรากดปุ่ม EXIT ก็จะจบโปรแกรมทันที นั่นก็เพราะว่าปุ่ม EXIT เมื่อถูกกดจะเรียกฟังก์ชั่น DestroyWindow() ที่อยู่ในฟังก์ชั่น OnClick() ซึ่งฟังก์ชั่น DestroyWindow() นี้จะใช้เรียกเมื่อเราต้องการจบโปรแกรม ฟังก์ชั่นจะทำลายวินโดวส์หลักและคืนหน่วยความจำของวินโดวส์ให้กับระบบ

2.4 โครงสร้างของ MFC

MFC เป็นไลบรารีที่บรรจุคลาสต่างๆ ไว้มากมาย โดยจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

1. คลาสที่มีต้นกำเนิดมาจากคลาสแม่ คือ CObject
2. คลาสที่เป็นคลาสโดด คือ คลาสที่ถูกสร้างขึ้นมาโดยไม่มีการสืบทอดมาจากคลาสใดๆ

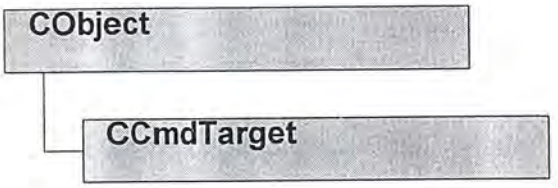
ใน MFC จะมีคลาสที่ชื่อ CObject เป็นคลาสแม่ โดยคลาสที่ใช้ในการเขียนโปรแกรมในระบบ เช่น การสร้างวินโดวส์, การสร้างคอนโทรล จะสืบทอดมาจากคลาส CObject นี้ทั้งนั้น

2.4.1 ภาพรวมของ MFC

1) คลาส CObject

คลาส CObject เป็นคลาสแม่ของทุกๆ คลาสที่ใช้ในการจัดการกับโปรแกรม เราเรียกว่าคลาส CObject เป็นรูท (Root) ซึ่งหน้าที่ของคลาสนี้จะเตรียมกระบวนการต่างๆ ที่ใช้ในโปรแกรม เช่น กระบวนการ Serialization ในการเขียนข้อมูลและอ่านข้อมูลจากดิสก์ให้กับโปรแกรมหรือการจัดการ Runtime Library ในการรันโปรแกรม

2) คลาส CCmdTarget



รูปที่ 2.10 คลาส CCmdTarget ที่สืบที่สืบทอดมาจาก CObject

คลาส CCmdTarget เป็นคลาสที่สืบที่สืบทอดมาจาก CObject ใช้สำหรับสร้างและจัดการกับกระบวนการแมสเสจแมป ที่ใช้สำหรับจัดการกับเหตุการณ์ต่างๆ ที่เราได้ศึกษากันมาแล้วนั้นเอง คลาส CCmdTarget นี้จะทำหน้าที่เป็นคลาสแม่ของคลาสต่างๆ ดังนี้

1. คลาส CWinApp ใช้สำหรับจัดการแอปพลิเคชั่น เช่น กระบวนการเริ่มต้น การรันโปรแกรม, Instance ของโปรแกรม

2. คลาส CWnd เป็นคลาสแม่ของคลาสวินโดวส์ทั้งหมด รับผิดชอบสร้างวินโดวส์ตลอดจนการควบคุมการทำงานของคอนโทรล

3. คลาส CFrameWnd ใช้สำหรับสร้างวินโดวส์แบบเฟรม (สืบทอดมาจากคลาส CWnd)

4. คลาส CView และ CDocument ใช้ในการจัดการ View ในโปรแกรมและการจัดการกับ Document (เอกสาร)

3) คลาส CWnd

รูปที่ 2.11 คลาส CWnd ควบคุมการทำงานของวินโดวส์และคอนโทรลของโปรแกรม

คลาส CWnd เป็นคลาสหนึ่งที่เราจะต้องให้ความสนใจ เพราะมีความสำคัญมากในการเขียนโปรแกรม คลาสนี้จะเป็นคลาสที่ควบคุมการทำงานของวินโดวส์และคอนโทรลทั้งหมดในโปรแกรม เป็นคลาสแม่ให้กับคลาสต่อไปนี้

1. คลาสวินโดวส์ CFrameWnd ใช้ในการควบคุมการทำงานและการแสดงผลของวินโดวส์

2. คลาสวินโดวส์ CSplitterWnd ใช้ในการแสดงผลวินโดวส์แบบแบ่งเฟรม (Splitter Window)

3. คลาสคอนโทรลบาร์ CControlBar ใช้จัดการกับทูลบาร์, ไดอะล็อกซ์บาร์, สเตตัสบาร์ ซึ่งเป็นส่วนหนึ่งของวินโดวส์ในโปรแกรม

4. คลาส CPropertySheet ใช้ในการแสดงวินโดวส์แบบ Property (วินโดวส์ที่มีแท็ปซ้อนๆกัน)

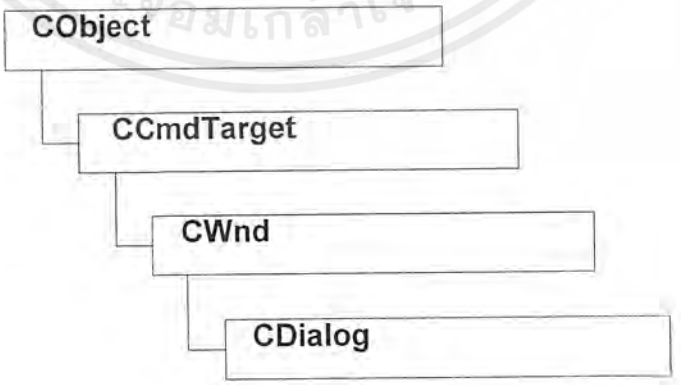
5. คลาส CDialog ใช้ในการควบคุมการแสดงผลของไดอะล็อกซ์ ซึ่งคลาส CDialog นี้ยังสืบทอดไปเป็นคลาสย่อยๆ อีกหลายคลาสเช่น CCommonDialog เป็นต้น

6. คลาส CView ใช้สำหรับแสดงผล View ในโปรแกรม เช่น View ของเอกสารหรือ View ในการวาดภาพของโปรแกรม

7. คลาสคอนโทรล ซึ่งมีอยู่มากมายหลายตัวเช่น CEit, CComboBox เป็นต้น

จากรายชื่อคลาสในข้างต้น ล้วนเป็นคลาสลูกที่สืบทอดมาจาก CWnd ทั้งสิ้น ซึ่งจะสังเกตเห็นได้อย่างหนึ่งว่า หน้าที่ของคลาสแต่ละคลาสล้วนเกี่ยวข้องกับวินโดวส์ และคอนโทรลทั้งนั้น

4) คลาส CDialog



รูปที่ 2.12 คลาส CDialog จะรับผิดชอบในการแสดงผลและควบคุมไดอะล็อกซ์

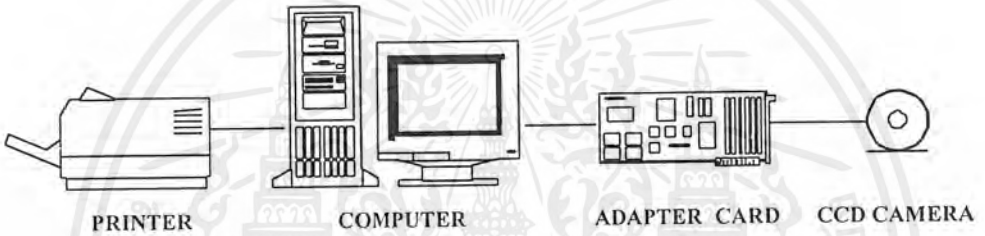
คลาส CDialog เป็นคลาสที่สืบทอดมาจาก CWnd ซึ่งเป็นคลาสที่จะรับผิดชอบในการแสดงผลและควบคุมไดอะล็อกซ์ ไม่ว่าจะเป็นแบบ Modal หรือ Modelless, การกดปุ่ม OK กับ Cancel และการทำงานของฟังก์ชั่น OnlnitDialog

จากคลาส CDialog นี้ยังได้มีการสืบทอดออกไปเป็นคลาสลูกอีก นั่นคือ CCommanDialog ซึ่งเป็นคลาสของคอมมอนไดอะล็อกซ์ (Common Dialog) ใช้ในการสร้าง ไดอะล็อกซ์เปิดไฟล์, บันทึกไฟล์หรือไดอะล็อกซ์แมสเสจบ็อกซ์

บทที่ 3
การออกแบบ การสร้าง และ การทำงาน

3.1 กล่าวนำ

สำหรับการออกแบบการสร้างและการทำงานเครื่องถ่ายสติกเกอร์นั้น จะประกอบด้วยการออกแบบ 2 ส่วนใหญ่ๆที่สำคัญคือ การออกแบบทางด้านฮาร์ดแวร์และการออกแบบทางด้านซอฟต์แวร์ โดยมีรายละเอียดดังต่อไปนี้

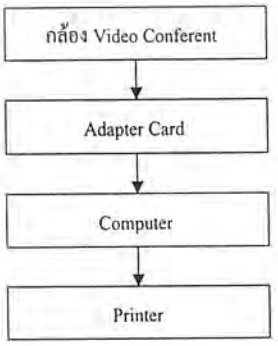


รูปที่ 3.1 การเชื่อมต่ออุปกรณ์ของเครื่องถ่ายสติกเกอร์

3.2 การออกแบบทางด้านฮาร์ดแวร์

3.2.1 การออกแบบส่วนควบคุมการถ่ายสติกเกอร์

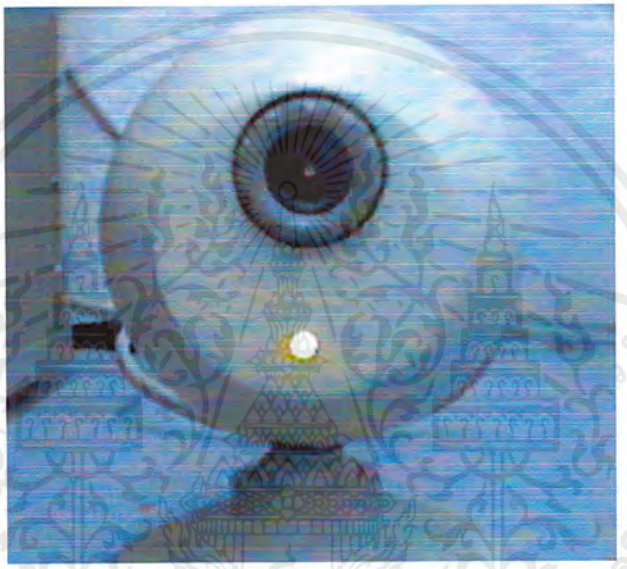
ในการออกแบบส่วน Hardware สำหรับโครงงานนี้ จะประกอบไปด้วยเครื่องคอมพิวเตอร์ซึ่งใช้เมนบอร์ด PC Chip Socket 7 ขนาดของซีพียู Cyrix MII 333 MHz ฮาร์ดดิสเก็บข้อมูล Conner ขนาดความจุ 540 Mbyte โดยมีการเชื่อมต่อและแผนผังการทำงานของเครื่องถ่ายสติกเกอร์ได้ดังรูปที่ 3.1และ3.2



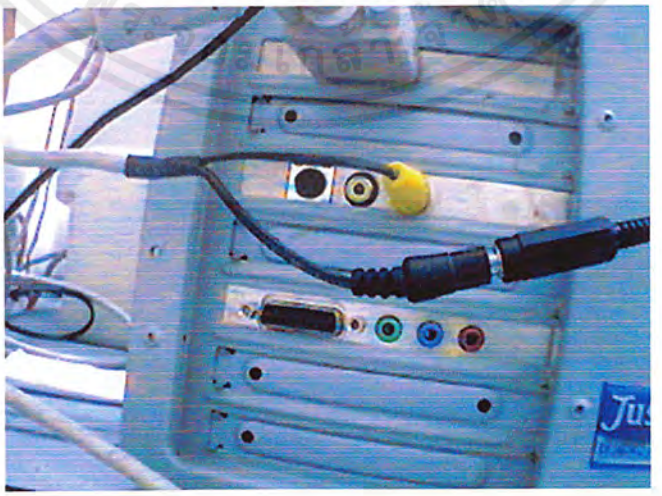
รูปที่ 3.2 ผังการทำงานของเครื่องถ่ายสติกเกอร์

3.2.2 การออกแบบส่วนรับภาพ

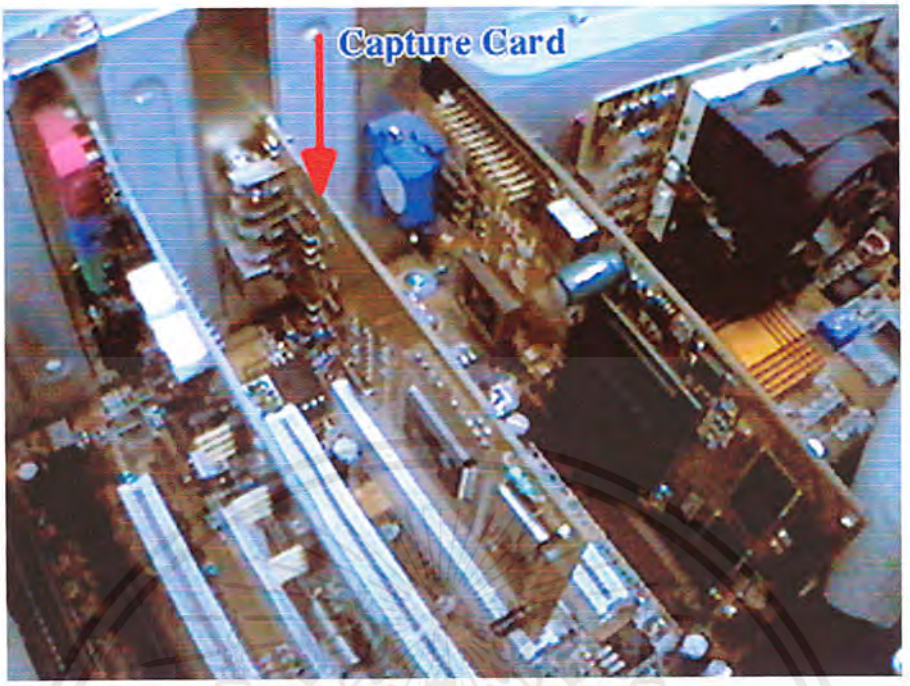
ในการออกแบบส่วนรับภาพสำหรับโครงงานนี้ จะใช้ กล้อง CCD ในการรับภาพซึ่งสามารถปรับจุดโฟกัสที่เหมาะสมในการถ่ายภาพได้ โดยใช้การ์ดอินเตอร์เฟสรับภาพจากกล้องเพื่อติดต่อกับส่วนควบคุมการถ่ายสติกเกอร์ซึ่งเป็นการ์ดยี่ห้อ Life View Fly Video ขนาด 32 บิต รับสถานะของภาพที่เข้ามาเป็นแบบดิจิตอล ซึ่งทำให้ภาพมีลักษณะคมชัด โดยมีลักษณะการต่อการ์ดและกล้องเข้ากับส่วนควบคุมดังรูปที่ 3.3, 3.4 และ 3.5



รูปที่ 3.3 ลักษณะของกล้อง CCD



รูปที่ 3.4 ลักษณะการต่อกล้อง CCD เข้ากับส่วนควบคุม



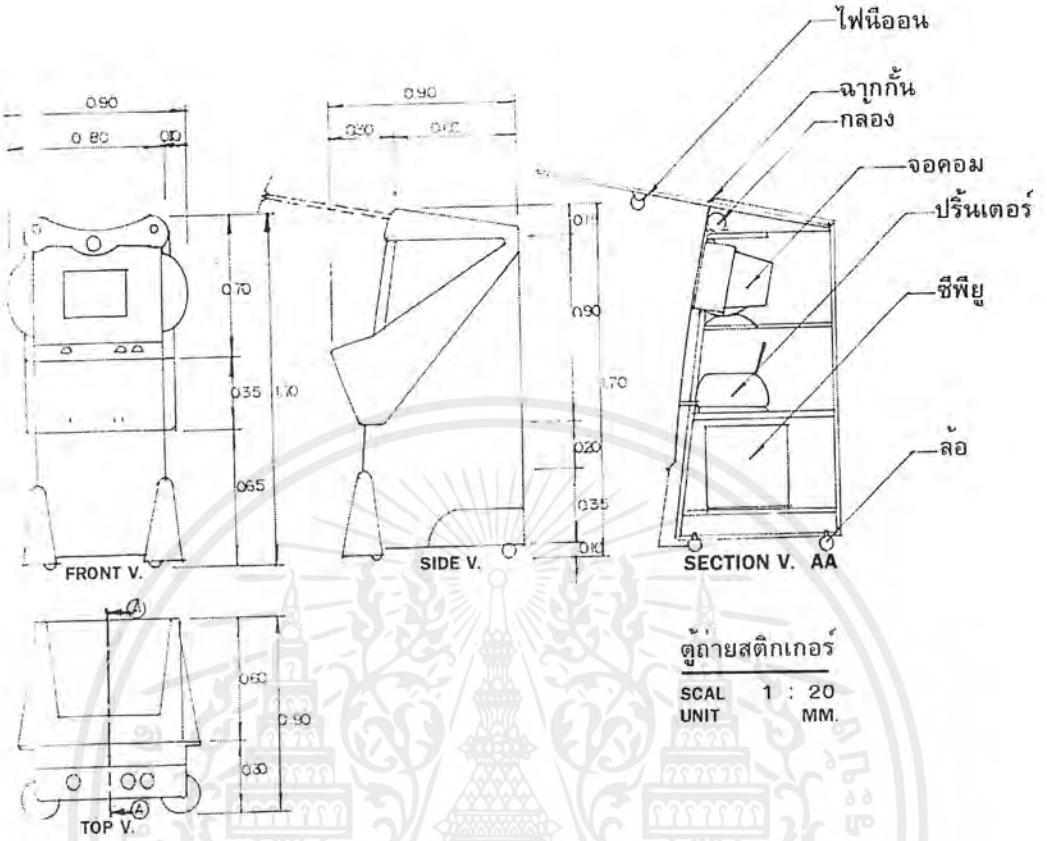
รูปที่ 3.5 ลักษณะการต่อการ์ด Capture เข้ากับส่วนควบคุม

3.2.3 การออกแบบส่วนพิมพ์ภาพสติกเกอร์

ในการออกแบบส่วนพิมพ์ภาพสติกเกอร์สำหรับโครงงานนี้ ใช้ปริ๊นเตอร์ยี่ห้อ Epson Stylus 460 ความละเอียด 720 dpi กระดาษที่ใช้ในการพิมพ์ภาพสติกเกอร์เป็นกระดาษที่ใช้สำหรับถ่ายสติกเกอร์โดยเฉพาะ

3.2.4 การออกแบบส่วนตู้ถ่ายสติกเกอร์

ในการออกแบบส่วนตู้ถ่ายสติกเกอร์ สามารถดูได้ดังรูปที่ 3.6 โดยตัวโครงสร้างข้างในเป็นเหล็กและด้านนอกเป็นไม้ซึ่งมีลักษณะการวางอุปกรณ์แบ่งออกเป็นชั้นๆ ดังรูปที่3.6 นอกจากนั้นได้ติดตั้งล้อเลื่อน 4 ล้อเพื่อให้สามารถเคลื่อนย้ายได้สะดวก โดยได้ออกแบบให้กระดาษสติกเกอร์เลื่อนออกมาเองทางด้านหน้าตรงช่องปริ๊นเตอร์เมื่อพิมพ์ภาพสติกเกอร์เสร็จ ทางด้านหลังเป็นประตูสามารถเปิดปิดและล็อกกุญแจได้เพื่อป้องความเสียหาย และภายในตู้ได้เจาะช่องสำหรับเดินสายไฟต่างๆที่ต่อระหว่างเครื่องให้สามารถเดินสายไฟได้อย่างเป็นระเบียบ ส่วนของฉากกั้นนั้นลักษณะที่สำคัญคือป้องกันไม่ให้แสงจากภายนอกเข้ามามีผลต่อแสงที่เราจัดไว้ภายในตู้ถ่ายสติกเกอร์ โดยแสงที่ได้จากตู้ถ่ายจะมีลักษณะนวลตา ความสว่างที่พอดี ซึ่งจะทำให้ได้ภาพที่สวยงามนอกจากนั้นเรายังสามารถปรับโฟกัสที่ตัวกล้องเพื่อให้ได้ภาพที่กลมกลืนกับแสงได้อีกด้วย



รูปที่ 3.6 โครงสร้างฮาร์ดแวร์ของตู้สติกเกอร์

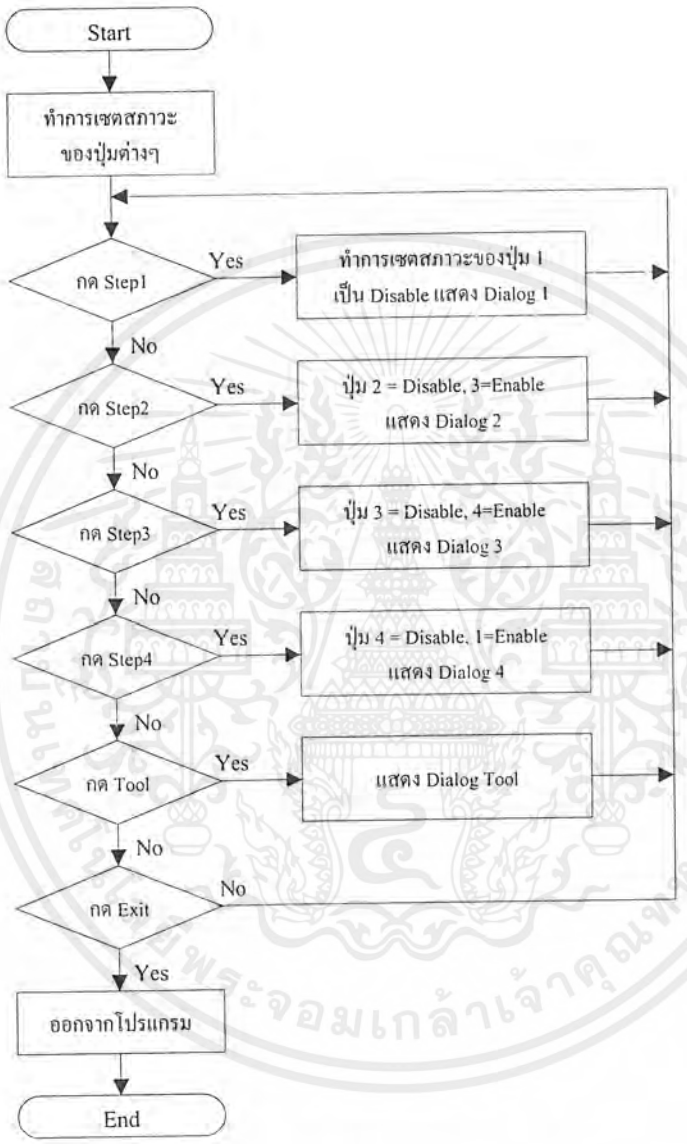
3.3 การออกแบบทางด้านซอฟต์แวร์

ในการออกแบบซอฟต์แวร์นั้น ได้จัดแบ่งการออกแบบเป็น 5 ส่วน โดยส่วนที่ 1 จะเป็นการออกแบบโปรแกรมหลัก ส่วนที่ 2 จะเป็นการออกแบบโปรแกรมเพื่อสร้างไฟล์และเปิดไฟล์ ส่วนที่ 3 เป็นการออกแบบโปรแกรมการเลือกเฟรม ส่วนที่ 4 เป็นการออกแบบโปรแกรมการ Capture ภาพ ส่วนที่ 5 เป็นการออกแบบโปรแกรมเพื่อนำภาพพิมพ์ออกมาเป็นภาพสติกเกอร์

3.3.1 การออกแบบโปรแกรมหลัก

ในส่วนของการออกแบบโปรแกรมเมนโปรแกรมนี้เราจะใช้การติดต่อกับไฟล์ INI เพื่ออ่านค่าสภาวะแวดล้อมต่างๆของโปรแกรม โดยทำการตั้งค่าชื่อไฟล์ที่จะทำาการติดต่อผ่าน member variable m_pszProfileName ในการสร้างไดอะล็อกซ์จะใช้ Class CMyDialog เป็น base Class ซึ่งจะแสดงผลไดอะล็อกซ์ในลักษณะของ bitmap dialog ในไดอะล็อกแรก ใช้ Class CStickerDlg ซึ่งสืบทอดมาจาก Class CMyDialog เป็นตัวควบคุม โดยในส่วนนี้จะมีตัวคอนโทรล

ทั้งหมด 6 ตัว ซึ่งเป็นชนิด Class CMybutton ซึ่งเป็นปุ่มที่แสดงผลในลักษณะ bitmap ซึ่งแต่ละปุ่มได้ทำการ Map message ของการกดปุ่มเพื่อทำหน้าที่เรียกไดอะล็อกซ์ต่างๆซึ่งมีหน้าที่ต่างๆกันดังนี้



รูปที่ 3.7 แผนผังการออกแบบโปรแกรมหลัก

Step 1 เรียกไดอะล็อกซ์การสร้างและเปิดไฟล์

Step 2 เรียกไดอะล็อกซ์การเลือกเฟรม

Step 3 เรียกไดอะล็อกซ์การ Capture ภาพ

Step 4 เรียกไดอะล็อกซ์การพิมพ์ภาพสติกเกอร์

Tool เรียกไดอะล็อกตั้งค่าการบูตเครื่อง

Exit ออกจากโปรแกรม

ซึ่งลักษณะการทำงานของปุ่มตั้งแต่ step ที่ 1 ถึง 4 จะเป็นการทำงานทีละขั้นในขณะที่ปุ่มใดปุ่มหนึ่งมีสถานะ Enable ปุ่มอื่นจะมีสถานะ Disable ซึ่งการตั้งค่าสถานะของปุ่มใช้ฟังก์ชั่น EnableWindow (BOOL bEnable = TRUE) โดยมีผังการออกแบบของโปรแกรมดังรูปที่ 3.7

3.3.2 การออกแบบโปรแกรมการเลือกเฟรม

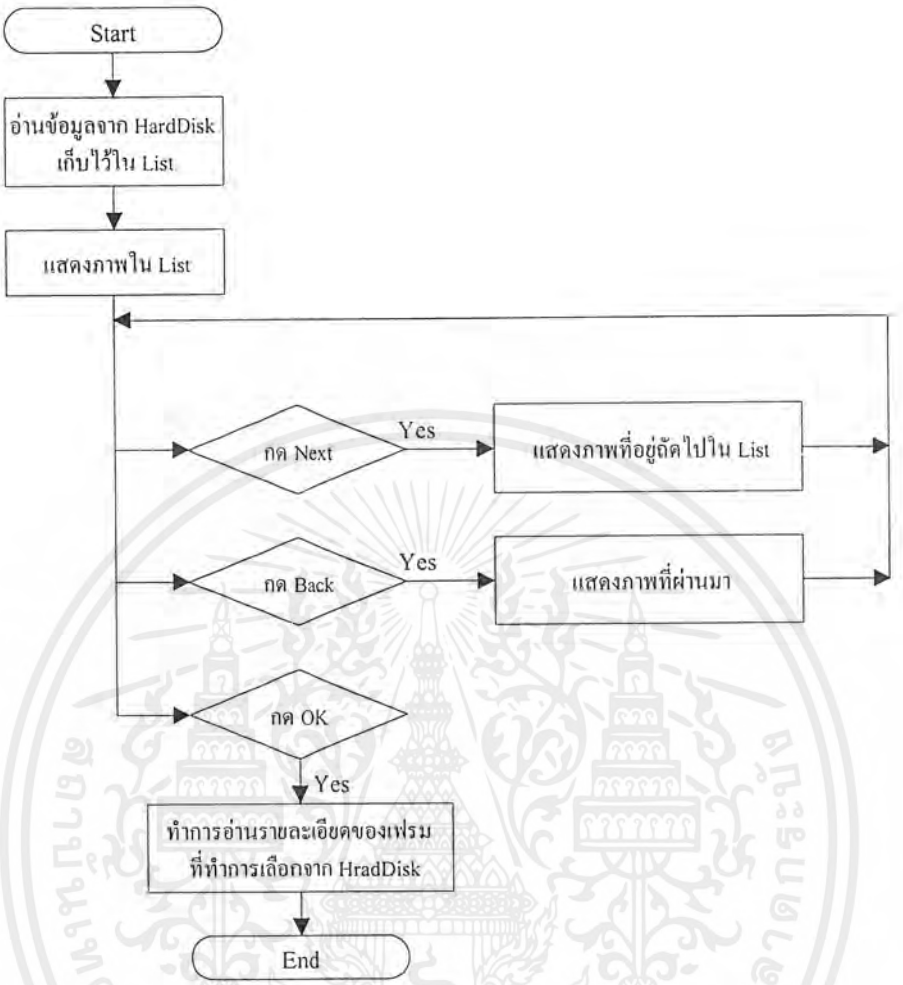
ในส่วนของการออกแบบโปรแกรมการเลือกเฟรมนี้เราจะใช้ Class CSelectFrame เป็นตัวควบคุมไดอะล็อกซ์ ในส่วนนี้จะมี memberVariable ซึ่งใช้เก็บจำนวนรูปแบบและใช้เก็บภาพตัวอย่างรูปแบบ ที่หาได้จาก Folder Frame Stye เนื่องจากรูปแบบของเฟรมสามารถที่เพิ่มหรือลบได้ซึ่งมีจำนวนที่ไม่แน่นอนจึงมีการใช้ Link List ในการเก็บข้อมูลทำให้สามารถเพิ่มเติมจำนวนของรูปแบบเฟรมได้ไม่จำกัดจำนวน โดยจะใช้คอนโทรลชนิด Class CMyButton 3 ตัวในการควบคุมการทำงานดังนี้

Next เลื่อนดูรูปแบบเฟรมต่อไป

OK เลือกรูปแบบที่แสดง

Back เลื่อนดูรูปแบบเฟรมที่ผ่านมา

เมื่อมีการกดปุ่ม OK จะทำการอ่านไฟล์ Script ซึ่งไฟล์นี้จะเก็บรายละเอียดต่างๆของรูปแบบเฟรมที่เลือกไว้ โดยมีผังการออกแบบของโปรแกรมดังรูปที่ 3.8



รูปที่ 3.8 แผนผังการออกแบบโปรแกรมในส่วนของการเลือกเฟรม

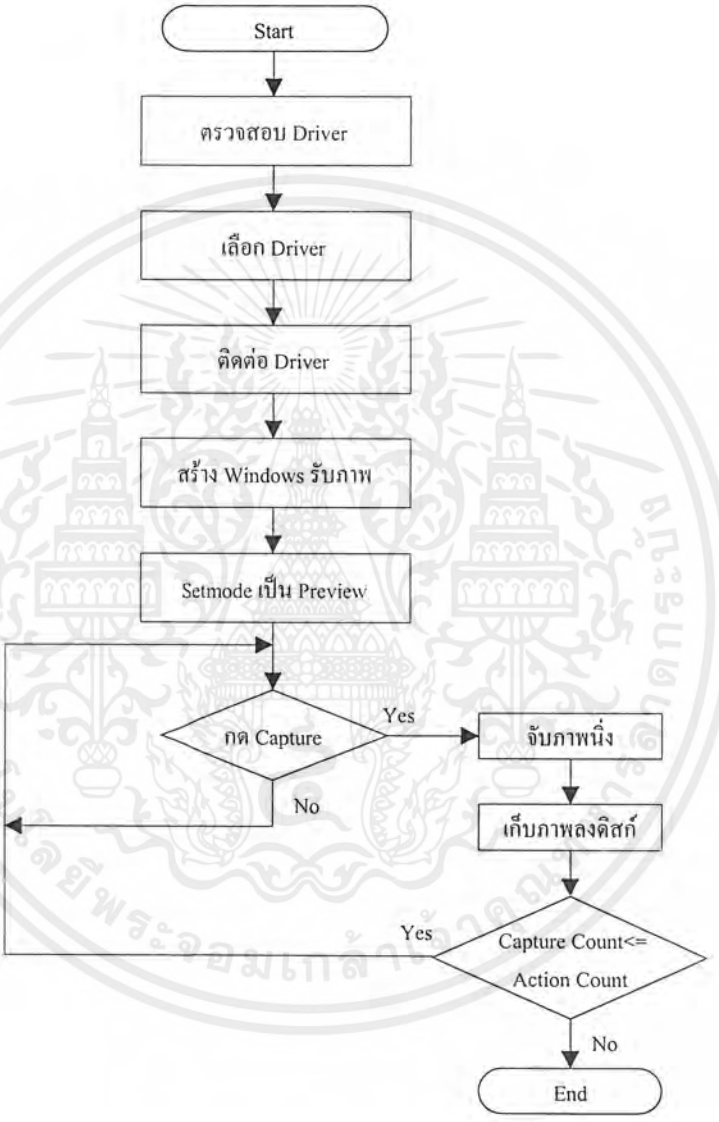
3.3.3 การออกแบบโปรแกรมการ Capture ภาพ

ในส่วนการออกแบบโปรแกรม Capture ภาพนั้นเราจะใช้ Class CCaptureDlg เป็นตัวควบคุมไดอะล็อกซ์ ในส่วนนี้จะมี memberVariable ซึ่งใช้เก็บจำนวนครั้งในการ Capture โดยใช้คอนโทรลชนิด Class CMybutton ในการควบคุมการจับภาพนิ่ง ซึ่งสามารถแบ่งขั้นตอนของการรับภาพและCapture ภาพดังนี้

1. ตรวจสอบ Driver ภายในเครื่อง เพื่อดูรายละเอียดของ Driver
2. เลือก Driver ที่ต้องการติดต่อ โดยโครงงานนี้ใช้ Driver Life view Fly Video EZ
3. ทำการ Connect กับ Driver ที่เลือก
4. สร้าง Windows ขึ้นมา เพื่อใช้แสดงภาพจาก Capture Card
5. Set mode ในการแสดงภาพเป็น Preview ซึ่งเป็นการนำภาพจาก Video Buffer มาแสดง

6. ใช้ Macro capGrabFrameNoStop เพื่อนำภาพนิ่งที่ได้จากการ Capture ไปเก็บในหน่วยความจำ

7. ทำการเก็บภาพที่ได้ลงใน หน่วยความจำ โดยใช้ Macro capSaveDIB แล้วนำภาพที่ได้ไปจัดลงเฟรมต่อไปซึ่งสามารถดูแผนผังการรับภาพจากกล้องได้ดังรูปที่ 3.9



รูปที่ 3.9 แผนผังการทำงานส่วนของการ Capture ภาพ

3.3.4 การออกแบบโปรแกรมเพื่อนำภาพพิมพ์ออกมาเป็นภาพสติกเกอร์

เมื่อได้ภาพจากการ Capture จะนำภาพไปเก็บไว้ในหน่วยความจำเมื่อได้ภาพครบตามจำนวนที่ต้องการก็จะนำมารวมกับรูปแบบเฟรมที่เลือกไว้เป็นภาพเดียวกันจากนั้นก็จะนำภาพที่ได้มาพิมพ์ภาพสติกเกอร์ โดยมีผังการออกแบบของโปรแกรมดังรูปที่ 3.10

ในส่วนนี้ใช้ Class CPrinPreviewDlg เป็นตัวควบคุมไดอะล็อกซ์ ซึ่งจะมีคอนโทรลชนิด Class Cmybutton 2 ตัวเป็นตัวควบคุม คือ

1. Preview Button ใช้แสดงภาพตัวอย่างก่อนสั่งพิมพ์ โดยภาพที่ได้มาจาก Class CPaper ซึ่งเป็นตัวสร้างและแสดงภาพ

2. Print Button สั่งพิมพ์ภาพปริ๊นเตอร์ โดยการวาดภาพจาก Class CPaper ลงบน Device Context ของปริ๊นเตอร์

รูปที่ 3.10 แผนผังการทำงานส่วนของการพิมพ์สติกเกอร์

3.4 การออกแบบภาพกราฟฟิก

ในส่วนของการออกแบบภาพกราฟฟิกนี้ เราใช้โปรแกรม Adobe Photo Shop Version 5.5, Ulead Photo Impact Version 4, Ulead Cool 3D Version 3.5 ในการออกแบบ สามารถแบ่งออกเป็นส่วนๆได้ 5 ส่วนดังนี้ ส่วนที่ 1 เป็นส่วนโปรแกรมหลักดังรูปที่ 3.11 ส่วนที่ 2 เป็นส่วนของการสร้างไฟล์และเปิดไฟล์ดังรูปที่ 3.12 ส่วนที่ 3 เป็นส่วนของการเลือกรูปแบบเฟรมดังรูปที่3.13 ส่วนที่ 4 เป็นส่วนของการ Capture ภาพดังภาพที่ 3.14 และ ส่วนที่ 5 เป็นส่วนของการพิมพ์ภาพสติกเกอร์ดังรูปที่ 3.15

3.4.1 โปรแกรมหลัก

รูปที่ 3.11 ภาพกราฟฟิกส่วนโปรแกรมหลัก

3.4.2 การสร้างไฟล์และเปิดไฟล์

รูปที่ 3.12 ภาพกราฟฟิกส่วนการสร้างไฟล์และเปิดไฟล์

3.4.3 การเลือกรูปแบบเฟรม

รูปที่ 3.13 ภาพกราฟฟิกส่วนการเลือกรูปแบบเฟรม

3.4.4 การ Capture ภาพ

รูปที่ 3.14 ภาพกราฟฟิกส่วนการ Capture ภาพ

3.4.5 การพิมพ์ภาพสติกเกอร์

รูปที่ 3.15 ภาพกราฟฟิกส่วนการพิมพ์ภาพสติกเกอร์

บทที่ 4

การทดลองและผลการทดลอง

4.1 กล่าวนำ

ในบทนี้กล่าวถึงวิธีการทดสอบชุดโปรแกรม ในการทำภาพถ่ายสติกเกอร์ โดยแบ่งการทดสอบออกเป็นส่วนๆ ตามขั้นตอนการทำภาพสติกเกอร์ คือ การสร้างภาพสติกเกอร์ขึ้นใหม่, การเลือกรูปแบบของสติกเกอร์, การทดลองการจับภาพนิ่ง หรือการ Capture ภาพ, การทดลองการจัดภาพลงเฟรมและการทดลองการพิมพ์ภาพสติกเกอร์

4.2 การทดลองการทำงานของเครื่องถ่ายสติกเกอร์

เครื่องถ่ายสติกเกอร์ จะมีลำดับขั้นทดลองการทำงานดังนี้คือ

1. เข้าสู่โปรแกรมถ่ายสติกเกอร์
2. คลิกเลือกที่ปุ่มขั้นตอนที่ 1 ดังรูปที่ 4.1

รูปที่ 4.1 การเลือกขั้นตอนที่ 1

3. เมื่อเข้ามาที่ขั้นตอนที่ 1 ก็จะพบปุ่มให้เลือก 2 ปุ่มคือ NEW และ OPEN ให้เลือกไปที่ปุ่ม NEW เพื่อสร้างสติกเกอร์ใหม่ ดังรูปที่ 4.2

รูปที่ 4.2 การเลือกเมนู NEW เพื่อสร้างสติกเกอร์ใหม่

4. จากนั้นโปรแกรมก็จะกลับมาที่โปรแกรมหลัก จากนั้นเลือกไปที่ขั้นตอนที่ 2

5. เมื่อเข้ามาที่ขั้นตอนที่ 2 ก็จะเป็นการเลือกรูปแบบเฟรมที่ต้องการ โดยกดที่ลูกศรดังรูปที่ 4.3 โดยรูปแบบของเฟรมมีให้เลือกหลายแบบ เช่น 2 แอคชั่น 4 ภาพดังรูปที่ 4.4 10 แอคชั่น 10 ภาพ ดังรูปที่ 4.5 และ 4 แอคชั่น 13 ภาพ ดังรูปที่ 4.6 เป็นต้น

รูปที่ 4.3 การเลือกรูปแบบของเฟรมภาพ

รูปที่ 4.4 การเลือกรูปแบบของเฟรมภาพ 2 แอคชั่น 4 ภาพ

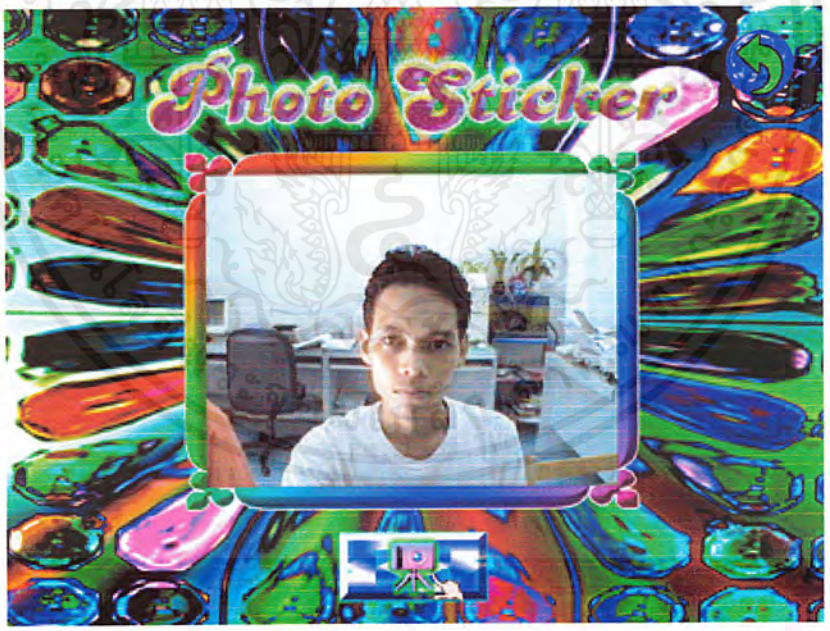
รูปที่ 4.5 การเลือกรูปแบบของเฟรมภาพ 10 แอคชั่น 10 ภาพ

รูปที่ 4.6 การเลือกรูปแบบของเฟรมภาพ 4 แอคชั่น 13 ภาพ

6. เมื่อเลือกรูปแบบที่ต้องการแล้วก็กดปุ่ม OK
7. จากนั้นโปรแกรมก็จะกลับมาที่โปรแกรมหลัก จากนั้นคลิกไปที่ขั้นตอนที่ 3
8. เมื่อเข้ามาที่ขั้นตอนที่ 3 ก็จะเป็นการ Capture ภาพ โดยจะมีภาพคนที่จะถ่ายสติกเกอร์ปรากฏที่จอภาพ ซึ่งภาพที่ปรากฏนั้นจะสวยงามหรือไม่นั้นขึ้นอยู่กับปริมาณแสงขณะนั้นด้วย โดยจะมีปริมาณแสงมากเกินไป น้อยเกินไป และแสงที่พอเหมาะดังแสดงในรูปที่ 4.7 ,4.8, 4.9 ตามลำดับ

รูปที่ 4.7 การ Capture ภาพที่มีแสงสว่างมากเกินไป

รูปที่ 4.8 การ Capture ภาพที่มีแสงสว่างน้อยเกินไป



รูปที่ 4.9 การ Capture ภาพที่มีแสงสว่างพอเหมาะ

9. เมื่อได้ภาพที่ต้องการแล้วก็ให้กดปุ่ม Capture
10. จากนั้นโปรแกรมก็จะกลับมาที่โปรแกรมหลักแล้วคลิกไปที่ขั้นตอนที่ 4
11. เมื่อเข้ามาที่ขั้นตอนที่ 4 ให้เลือกไปที่ปุ่ม Preview ซึ่งจะเป็นการดูภาพ Preview ก่อนทำการพิมพ์ภาพสติกเกอร์ ดังรูปที่ 4.10 ,4.11, 4.12, 4.13

รูปที่ 4.10 การ Preview ก่อนทำการพิมพ์ภาพขนาด 1 ภาพ

รูปที่ 4.11 การ Preview ก่อนทำการพิมพ์ภาพขนาด 4 ภาพ

รูปที่ 4.12 การ Preview ก่อนทำการพิมพ์ภาพขนาด 10 ภาพ

รูปที่ 4.13 การ Preview ก่อนทำการพิมพ์ภาพขนาด 13 ภาพ

12. จากนั้นก็กดที่ปุ่มพิมพ์ภาพ โดยโปรแกรมก็จะทำการพิมพ์ภาพที่ได้ออกมาเป็นภาพสติกเกอร์ ดังรูปที่ 4.14, 4.15, 4.16 และ 4.17

รูปที่ 4.14 ภาพที่ได้จากการพิมพ์ขนาด 1 ภาพ

รูปที่ 4.15 ภาพที่ได้จากการพิมพ์ขนาด 4 ภาพ

รูปที่ 4.16 ภาพที่ได้จากการพิมพ์ขนาด 10 ภาพ

รูปที่ 4.17 ภาพที่ได้จากการพิมพ์ขนาก 13 ภาพ

บทที่ 5
บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา

5.1 บทสรุป

ก่อนที่โครงงานนี้จะสำเร็จได้นั้นจำเป็นอย่างยิ่งที่จะต้องได้รับความช่วยเหลือจากท่านอาจารย์ในภาควิชา และบุคคลอื่นๆ ที่ผู้จัดทำได้รับความช่วยเหลือและข้อแนะนำต่างๆ ที่เป็นประโยชน์ต่อผู้จัดทำโครงงานนี้ขึ้นมา นอกจากนั้นการศึกษาค้นคว้าในหลักการและทฤษฎีต่างๆที่เกี่ยวข้องกับการทำโครงงานนี้ทำให้ผู้จัดทำได้ทราบถึงความรู้ต่างๆ อาจจะหาไม่ได้จากการเรียนในห้องเรียนและจากการทำโครงงานเองทำให้ผู้จัดทำมีความเห็นว่าการเรียนนั้นนอกจากที่เราจะเรียนจากภายในห้องเรียนเท่านั้นแล้วเรายังต้องศึกษาค้นคว้า จากภายนอกอีกด้วยและอีกสิ่งหนึ่งที่มีความสำคัญก็คือการที่ได้เห็นว่าสิ่งที่เราได้เรียนไปนั้นสามารถที่จะนำไปใช้ประโยชน์ได้จริง

จากโครงงานเครื่องถ่ายสติกเกอร์นี้ จะเห็นได้ว่าเครื่องถ่ายสติกเกอร์นี้สามารถนำภาพจากกล้องเข้ามาทำเป็นรูปถ่ายสติกเกอร์ได้ โดยใช้ส่วนควบคุมการรับภาพซึ่งเป็นเครื่องไมโครคอมพิวเตอร์รับภาพจากกล้องโดยผ่านการ์ดอินเตอร์เฟส จากนั้นก็ทำการ Capture ภาพที่ต้องการแล้ว นำภาพที่ได้มาลงเฟรมที่ได้ออกแบบไว้ โดยสามารถแก้ไขเพิ่มเติมรูปแบบของเฟรมได้จากการแก้ไขตัวโปรแกรม จากนั้นนำภาพที่ได้พิมพ์เป็นภาพลงบนกระดาษสติกเกอร์ โดยใช้เครื่องปริ๊นเตอร์สี ซึ่งจะได้รูปถ่ายสติกเกอร์ตามรูปแบบที่เลือกไว้

5.2 ปัญหาในการทำงาน

ในการจัดทำโครงงานนี้มีปัญหาต่างๆ ซึ่งเป็นอุปสรรคในการออกแบบและการทำงานทั้งทางฮาร์ดแวร์และซอร์ฟแวร์ ทำให้การทำงานดังกล่าวไม่เป็นไปตามวัตถุประสงค์ที่ได้วางไว้ซึ่งคณะผู้จัดทำได้ทำการแก้ไขเรียบร้อยแล้ว ปัญหาดังกล่าวมีดังต่อไปนี้

5.2.1 ปัญหาในส่วนของฮาร์ดแวร์

1. ปัญหาที่พบ ในส่วนของการจัดทำตู้ถ่ายสติกเกอร์ซึ่งผู้จัดทำไม่มีความรู้ในการออกแบบ และจัดสร้าง ตู้ที่มีกราฟฟิก ทำให้เกิดความล่าช้าในการออกแบบและการสร้าง

   แนวทางการแก้ไข คือ ขอคำแนะนำจากผู้ที่มีความเชี่ยวชาญในด้านการออกแบบ

2. ระบบแสงสว่างภายในตู้ถ่ายสติกเกอร์ไม่พอเหมาะ

   แนวทางการแก้ไข คือ ทำการเพิ่มแสงไฟในส่วนของด้านหน้าเข้าไปอีกจากที่ใช้หลอด

ฟลูออเรสท์เซ็นต์ขนาด 18 วัตต์ 2 หลอดก็เพิ่มอีก 2 หลอดเป็น 4 หลอด นอกจากนั้นได้ทำฉากกั้นเพื่อป้องกันแสงรบกวนจากภายนอก และปรับโฟกัสของกล้องให้ได้ภาพที่กลมกลืนกับแสงและสวยงาม

5.2.2 ปัญหาในส่วนของซอฟแวร์

1. เอกสารเกี่ยวกับการเขียนโปรแกรมภาษา Visual C++ มีน้อย ทำให้เกิดความล่าช้าในการศึกษาค้นคว้าและการออกแบบตัวโปรแกรม

แนวทางในการแก้ไข คือ ค้นคว้าหาข้อมูลเพิ่มเติมจากเว็บไซต์ ที่มาพร้อมกับหนังสือ Visual C++ เวอร์ชัน 6.0

2. ในการ Capture ภาพ การ์ดมีความเร็วในการประมวลผลค่อนข้างช้าจึงทำให้ภาพที่ได้ไม่เป็น Real Time

แนวทางในการแก้ไข คือ ใช้การแสดงภาพแบบ Preview

5.3 ประโยชน์ที่ได้รับจากการทำโครงงาน

1. มีความชำนาญในการใช้โปรแกรมคอมพิวเตอร์ต่างๆมากขึ้น
2. มีความรู้ในการออกแบบและการสร้างตู้และอุปกรณ์ฮาร์ดแวร์
3. มีความรู้ความเข้าใจเทคโนโลยีการทำภาพสติกเกอร์มากขึ้น
4. ลดต้นทุนของเครื่องถ่ายสติกเกอร์ทำให้ไม่ต้องนำเข้าจากต่างประเทศ
5. เป็นแนวทางในการพัฒนาเทคโนโลยีการถ่ายภาพสติกเกอร์ต่อไป

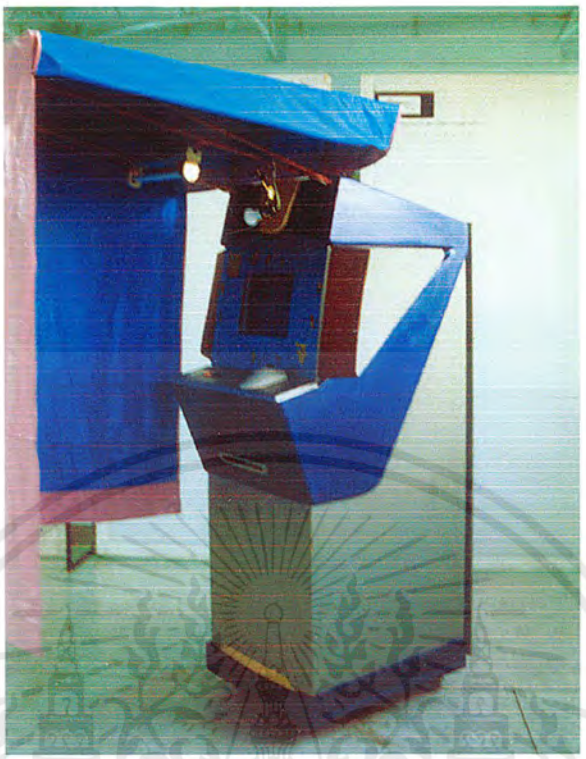
5.4 แนวทางในการพัฒนา

1. ออกแบบตู้ถ่ายสติกเกอร์ให้มีขนาดเล็กและสวยงามกว่านี้
2. ออกแบบโปรแกรมให้สามารถเก็บภาพลงในหน่วยความจำได้
3. ออกแบบโปรแกรมให้สามารถเก็บภาพสติกเกอร์ที่ได้ถ่ายไว้แล้วลงในหน่วยความจำได้และสามารถนำภาพเดิมมาทำเป็นรูปสติกเกอร์ได้อีก
4. ออกแบบโปรแกรมให้สามารถนำภาพถ่ายที่ถ่ายจากกล้องทั่วไปมาทำเป็นสติกเกอร์ได้
5. ออกแบบโปรแกรมให้เครื่องถ่ายสติกเกอร์สามารถทำงานเองอัตโนมัติได้
6. ออกแบบโปรแกรมให้สามารถ Capture ภาพที่เป็น Real Time ได้

ภาคผนวก ก
เครื่องต้นแบบ

รูปที่ ก.1 ส่วนแสดงด้านหน้าของเครื่องถ่ายสติกเกอร์

รูปที่ ก.2 ส่วนแสดงหน้าจอของเครื่องถ่ายสติกเกอร์

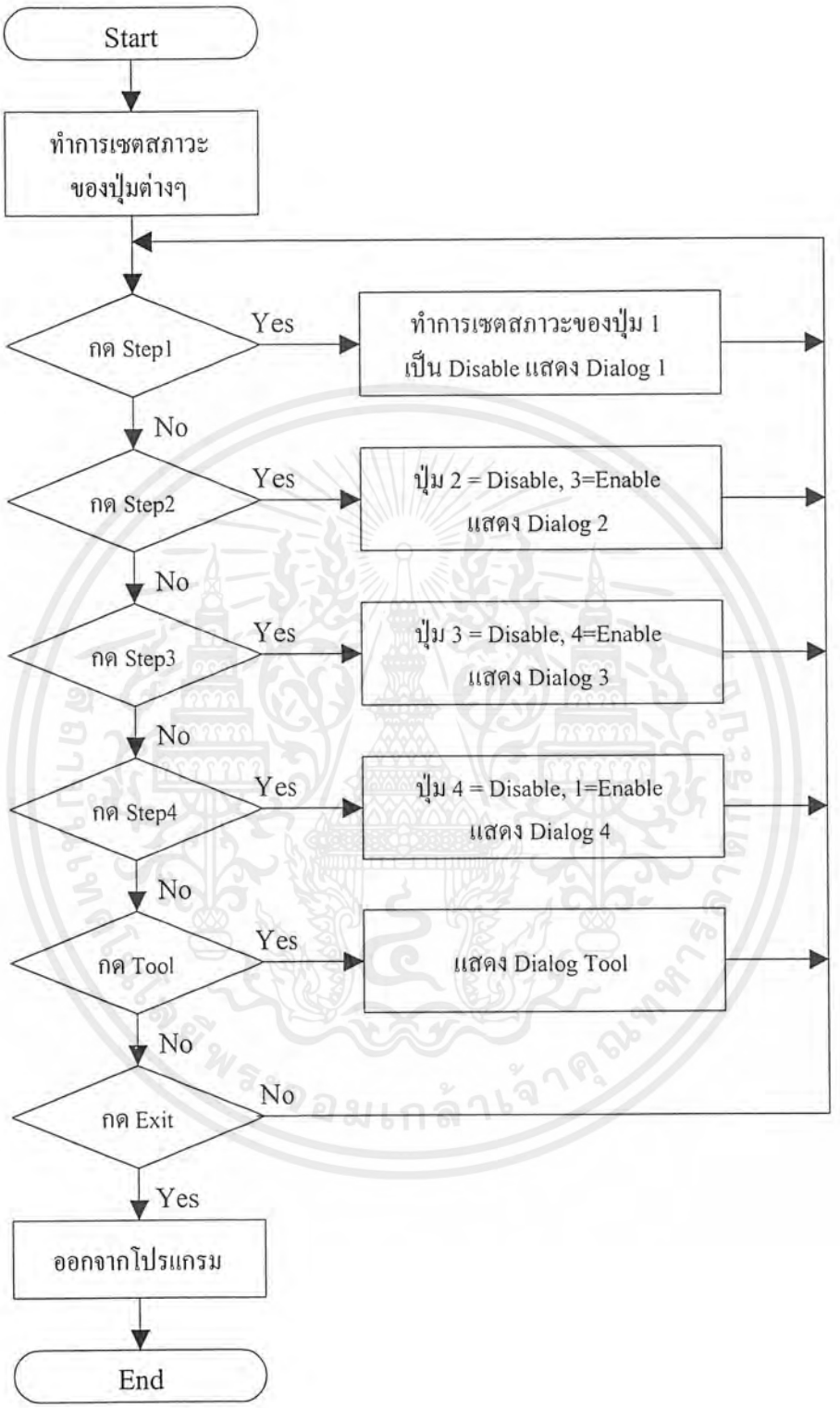


รูปที่ ก.3 ส่วนแสดงด้านข้างของเครื่องถ่ายสติกเกอร์

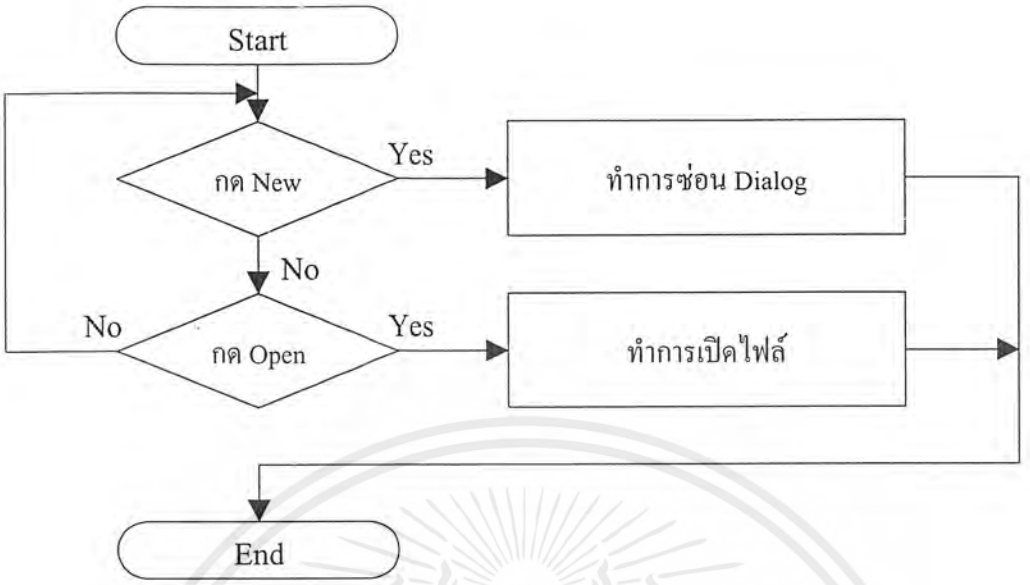
รูปที่ ก.4 ส่วนแสดงด้านหลังของเครื่องถ่ายสติกเกอร์

ภาคผนวก ข

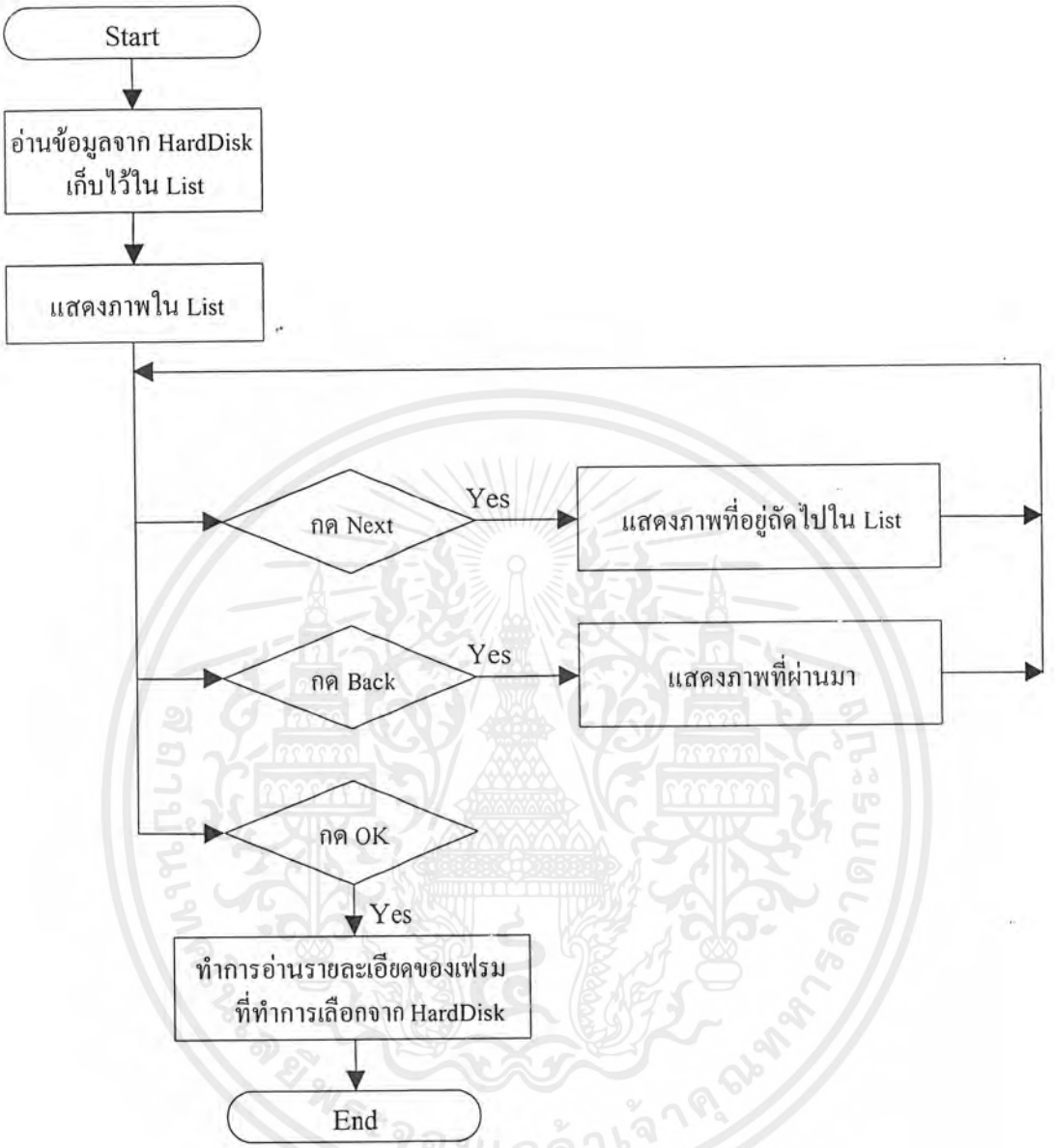
ผังการทำงานและโปรแกรม



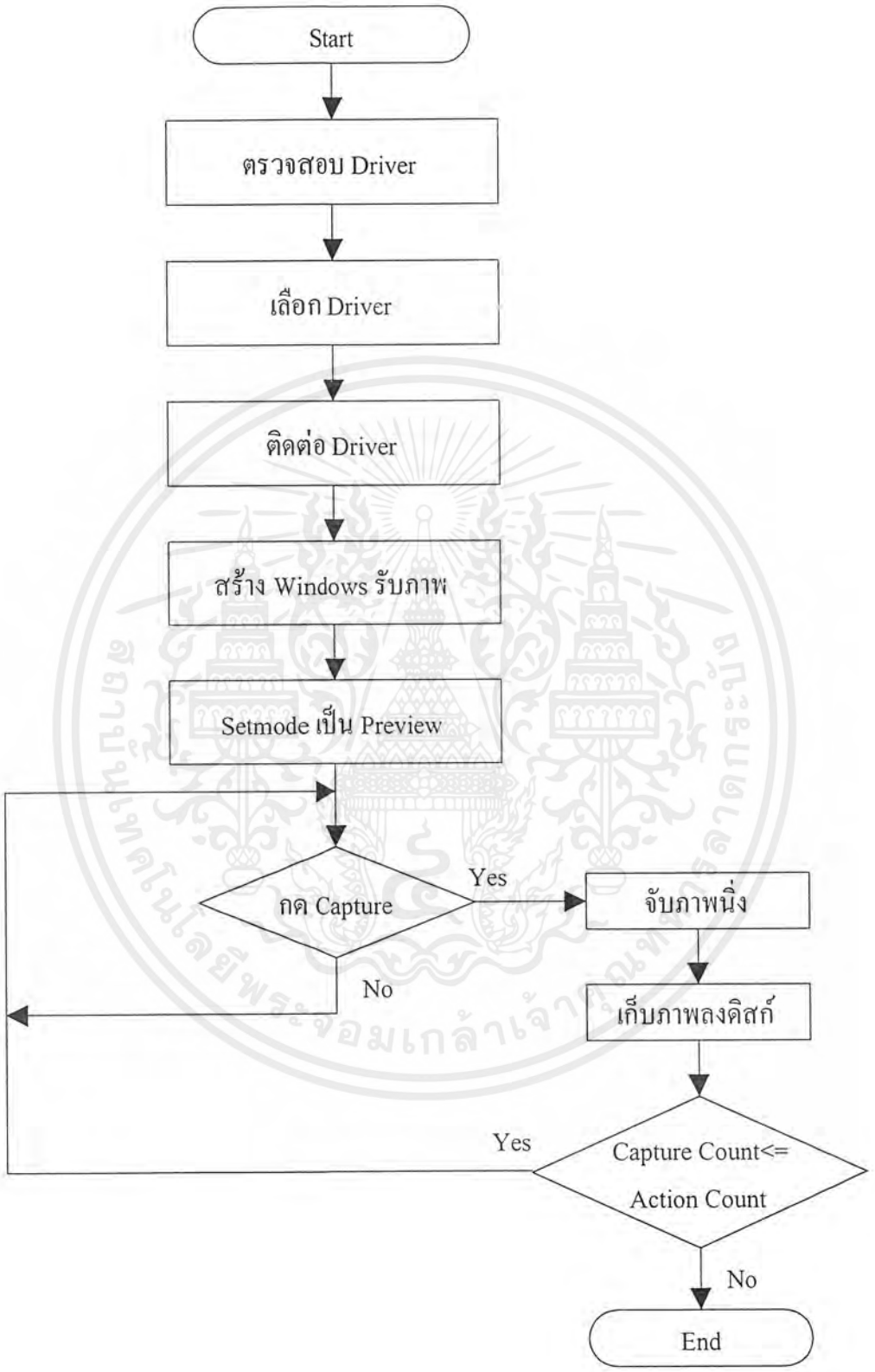
รูปที่ ข.1 ผังการทำงานของโปรแกรมควบคุมส่วนโปรแกรมหลัก



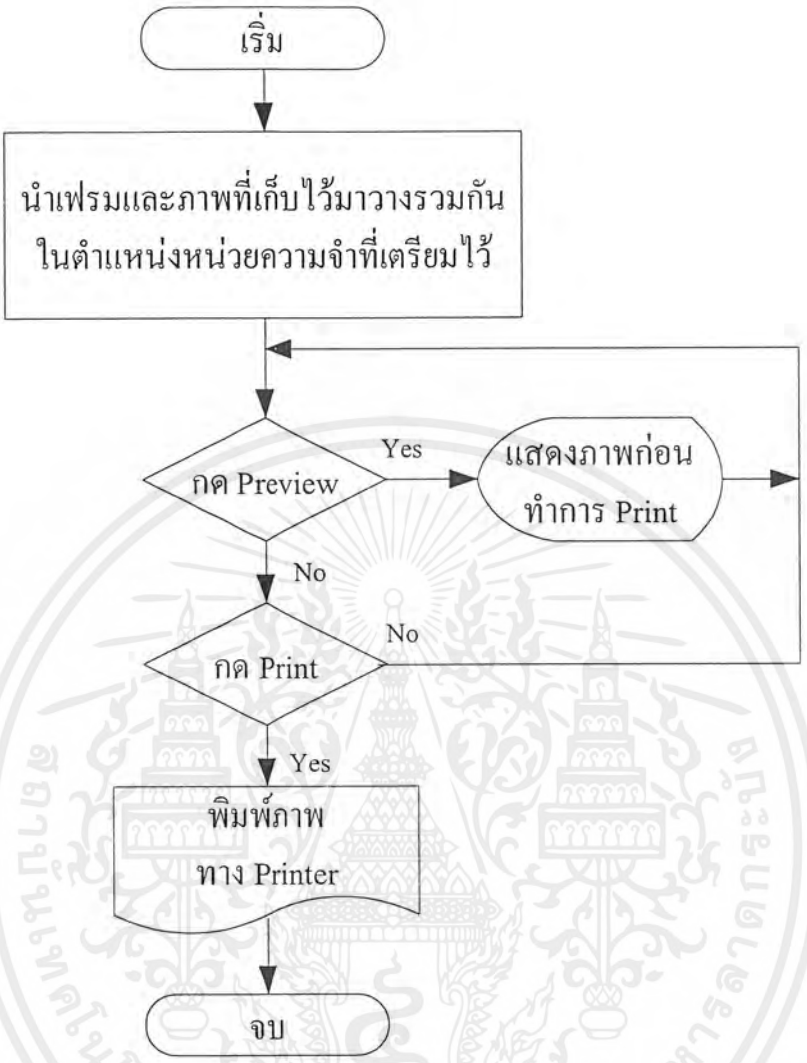
รูปที่ ข.2 ผังการทำงานของโปรแกรมควบคุมส่วนของการสร้างและเปิดไฟล์



รูปที่ ข.3 ผังการทำงานของโปรแกรมควบคุมส่วนของการเลือกเฟรม



รูปที่ ข.4 ผังการทำงานของโปรแกรมควบคุมส่วนของการ Capture ภาพ



รูปที่ ข.5 ผังการทำงานของโปรแกรมควบคุมส่วนของการ Print ภาพ

ไฟล์ต่างๆ ในโปรเจ็กต์

stdafx.h,stdafx.cpp :	ทำการ include ไฟล์ระบบมาตรฐานต่างๆ
sticker.h,sticker.cpp :	ไฟล์หลักสำหรับควบคุม Sticker Appliction
stickerdlg.h,stickerdlg.cpp :	ไฟล์สำหรับเก็บคลาส CStickerDlg
newopendlg.h,newopendlg.cpp :	ไฟล์สำหรับเก็บคลาส CNewOpenDlg
opendlg.h,opendlg.cpp :	ไฟล์สำหรับเก็บคลาส COpenDlg
selectframe.h,selectframe.cpp :	ไฟล์สำหรับเก็บคลาส CSelectFrame
capturedlg.h,capturedlg.cpp :	ไฟล์สำหรับเก็บคลาส CCaptureDlg
printpreviewdlg.h,printpreview.cpp :	ไฟล์สำหรับเก็บคลาส CPrintPreviewDlg
tooldlg.h,tooldlg.cpp :	ไฟล์สำหรับเก็บคลาส CToolDlg
paper.h,paper.cpp :	ไฟล์สำหรับเก็บคลาส CPaper
frame.h,frame.cpp :	ไฟล์สำหรับเก็บคลาส CFrame
mybutton.h,button.cpp :	ไฟล์สำหรับเก็บคลาส CMyButton
mydialog.h,mydialog.cpp :	ไฟล์สำหรับเก็บคลาส CMyDialog
midi.h,midi.cpp :	ไฟล์สำหรับเก็บคลาส CMidi
directsound.h,directsound.cpp :	ไฟล์สำหรับเก็บคลาส CDirectSound
dibsectionlite.h,dibsectionlite.cpp :	ไฟล์สำหรับเก็บคลาส CDIBSectionLite

โปรแกรมควบคุมการทำงานของเครื่องถ่ายสติกเกอร์

StdAfx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently

#if
!defined(AFX_STDAFX_H_C3043CA9_D150_11D3_8AFC_E03A49C10001_INCLUDED)
#define
AFX_STDAFX_H__C3043CA9_D150_11D3_8AFC_E03A49C10001__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN        // Exclude rarely-used stuff from
Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>       // MFC support for Internet Explorer 4
Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // MFC support for Windows
Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_STDAFX_H_C3043CA9_D150_11D3_8AFC_E03A49C10001_INCLUDED)
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//    Sticker.pch will be the pre-compiled header
//    stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

Sticker.h

```
// Sticker.h : main header file for the STICKER application
//

#if
!defined(AFXSTICKER_H_C3043CA5_D150_11D38AFC_E03A49C10001_INCLUDED_)
```

```
#define
AFX_STICKER_H__C3043CA5_D150_11D3_8AFC_E03A49C10001__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
      #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"         // main symbols
#include "Paper.h"
#include "vfw.h"

/////////////////////////////////////////////////////////////////////////////
// CStickerApp:
// See Sticker.cpp for the implementation of this class
//

class CStickerApp : public CWinApp
{
protected:
      CString     SkinsPath;
      CString ApplicationPath;
      CString FrameStylePath;
      CString FrameFileName;
      CString OpenFileName;
      CString ScriptFileName;
      CString OpenStickerPath;
      int FrameCount;
      int OpenCount;
      int FrameSelect;
public:
      CPaper *pPaper;
      BOOL m_New;

public:
      CStickerApp();
      CString GetSkinsPath();
      CString GetAppPath();
      CString GetFrameStylePath();
      CString GetOpenStickerPath();
      CString     GetFrameFileName(UINT FrameID=0);
      CString GetOpenFileName(UINT OpenID=0);
      SetFrameSelect(int FrameSelect);
      SetScriptFileName(CString FileName);
      CString GetScriptFileName();
      int GetFrameSelect();
      int GetFrameCount();
      int GetOpenCount();
      CreatePaper();
      DestroyPaper();
      HBITMAP GetFrameAt(UINT FrameID);
      COLORREF GetColorRefAt(UINT FrameID);
      FindSticker();
      CList<HBITMAP,HBITMAP> m_StickerList;
      ClearStickerList();
```

```
//Sticker Dialog
CString Sticker_Dialog_Background;
CString Sticker_Dialog_Step1_Button;
CString Sticker_Dialog_Step2_Button;
CString Sticker_Dialog_Step3_Button;
CString Sticker_Dialog_Step4_Button;
CString Sticker_Dialog_Exit_Button;
CString     Sticker_Dialog_Tool_Button;
CString Sticker_Dialog_Cur;
CString Sticker_Dialog_Step1_Cur;
CString Sticker_Dialog_Step2_Cur;
CString Sticker_Dialog_Step3_Cur;
CString Sticker_Dialog_Step4_Cur;
CString Sticker_Dialog_Exit_Cur;
CString Sticker_Dialog_Tool_Cur;
CString Sticker_Dialog_INI_Path;
int         Sticker_Dialog_Step1_X;
int         Sticker_Dialog_Step1_Y;
int         Sticker_Dialog_Step1_Width;
int         Sticker_Dialog_Step1_Height;
int         Sticker_Dialog_Step2_X;
int         Sticker_Dialog_Step2_Y;
int         Sticker_Dialog_Step2_Width;
int         Sticker_Dialog_Step2_Height;
int         Sticker_Dialog_Step3_X;
int         Sticker_Dialog_Step3_Y;
int         Sticker_Dialog_Step3_Width;
int         Sticker_Dialog_Step3_Height;
int         Sticker_Dialog_Step4_X;
int         Sticker_Dialog_Step4_Y;
int         Sticker_Dialog_Step4_Width;
int         Sticker_Dialog_Step4_Height;
int         Sticker_Dialog_Exit_X;
int         Sticker_Dialog_Exit_Y;
int         Sticker_Dialog_Exit_Width;
int         Sticker_Dialog_Exit_Height;
int         Sticker_Dialog_Tool_X;
int         Sticker_Dialog_Tool_Y;
int         Sticker_Dialog_Tool_Width;
int         Sticker_Dialog_Tool_Height;
int         Sticker_Dialog_Background_Width;
int         Sticker_Dialog_Background_Height;

//New Open Dialog
CString New_Open_Dialog_Background;
CString New_Open_Dialog_New_Button;
CString New_Open_Dialog_Open_Button;
CString New_Open_Dialog_New_Cur;
CString New_Open_Dialog_Open_Cur;
CString New_Open_Dialog_Cur;
CString New_Open_Dialog_INI_Path;
int         New_Open_Dialog_New_X;
int         New_Open_Dialog_New_Y;
int         New_Open_Dialog_New_Width;
int         New_Open_Dialog_New_Height;
int         New_Open_Dialog_Open_X;
int         New_Open_Dialog_Open_Y;
int         New_Open_Dialog_Open_Width;
```

```
int         New_Open_Dialog_Open_Height;

int         New_Open_Dialog_Background_Width;
int         New_Open_Dialog_Background_Height;

//Open Dialog
CString Open_Dialog_Background;
CString Open_Dialog_Back_Button;
CString Open_Dialog_Next_Button;
CString Open_Dialog_Done_Button;
CString Open_Dialog_Back_Cur;
CString Open_Dialog_Next_Cur;
CString Open_Dialog_Done_Cur;
CString Open_Dialog_Cur;
CString Open_Dialog_INI_Path;

int         Open_Dialog_Next_X;
int         Open_Dialog_Next_Y;
int         Open_Dialog_Next_Width;
int         Open_Dialog_Next_Height;
int         Open_Dialog_Back_X;
int         Open_Dialog_Back_Y;
int         Open_Dialog_Back_Width;
int         Open_Dialog_Back_Height;
int         Open_Dialog_Done_X;
int         Open_Dialog_Done_Y;
int         Open_Dialog_Done_Width;
int         Open_Dialog_Done_Height;
int         Open_Dialog_Preview_X;
int         Open_Dialog_Preview_Y;
int         Open_Dialog_Preview_Width;
int         Open_Dialog_Preview_Height;
int         Open_Dialog_Background_Width;
int         Open_Dialog_Background_Height;
//...

//Select Frame Dialog
CString Select_Frame_Dialog_Background;
CString Select_Frame_Dialog_Back_Button;
CString Select_Frame_Dialog_Next_Button;
CString Select_Frame_Dialog_Done_Button;
CString Select_Frame_Dialog_Back_Cur;
CString Select_Frame_Dialog_Next_Cur;
CString Select_Frame_Dialog_Done_Cur;
CString Select_Frame_Dialog_Cur;
CString Select_Frame_Dialog_INI_Path;

int         Select_Frame_Dialog_Next_X;
int         Select_Frame_Dialog_Next_Y;
int         Select_Frame_Dialog_Next_Width;
int         Select_Frame_Dialog_Next_Height;
int         Select_Frame_Dialog_Back_X;
int         Select_Frame_Dialog_Back_Y;
int         Select_Frame_Dialog_Back_Width;
int         Select_Frame_Dialog_Back_Height;
int         Select_Frame_Dialog_Done_X;
int         Select_Frame_Dialog_Done_Y;
int         Select_Frame_Dialog_Done_Width;
```

```
	int		Select_Frame_Dialog_Done_Height;
	int		Select_Frame_Dialog_Preview_X;
	int		Select_Frame_Dialog_Preview_Y;
	int		Select_Frame_Dialog_Preview_Width;
	int		Select_Frame_Dialog_Preview_Height;
	int		Select_Frame_Dialog_Background_Width;
	int		Select_Frame_Dialog_Background_Height;
	//...

	//Capture Dialog
	CString Capture_Dialog_Background;
	CString Capture_Dialog_Capture_Button;
	CString Capture_Dialog_Capture_Cur;
	CString Capture_Dialog_Cur;
	CString Capture_Dialog_INI_Path;

	int		Capture_Dialog_Capture_X;
	int		Capture_Dialog_Capture_Y;
	int		Capture_Dialog_Capture_Width;
	int		Capture_Dialog_Capture_Height;
	int		Capture_Dialog_Preview_X;
	int		Capture_Dialog_Preview_Y;
	int		Capture_Dialog_Preview_Width;
	int		Capture_Dialog_Preview_Height;
	int		Capture_Dialog_Background_Width;
	int		Capture_Dialog_Background_Height;
	//...

	//Print Preview Dialog
	CString PrintPreview_Dialog_Background;
	CString PrintPreview_Dialog_Print_Button;
	CString PrintPreview_Dialog_PrintPreview_Button;
	CString PrintPreview_Dialog_Print_Cur;
	CString PrintPreview_Dialog_PrintPreview_Cur;
	CString PrintPreview_Dialog_Cur;
	CString PrintPreview_Dialog_INI_Path;

	int		PrintPreview_Dialog_Print_X;
	int		PrintPreview_Dialog_Print_Y;
	int		PrintPreview_Dialog_Print_Width;
	int		PrintPreview_Dialog_Print_Height;
	int		PrintPreview_Dialog_PrintPreview_X;
	int		PrintPreview_Dialog_PrintPreview_Y;
	int		PrintPreview_Dialog_PrintPreview_Width;
	int		PrintPreview_Dialog_PrintPreview_Height;
	int		PrintPreview_Dialog_Preview_X;
	int		PrintPreview_Dialog_Preview_Y;
	int		PrintPreview_Dialog_Preview_Width;
	int		PrintPreview_Dialog_Preview_Height;
	int		PrintPreview_Dialog_Background_Width;
	int		PrintPreview_Dialog_Background_Height;
	//...

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CStickerApp)
	public:
	virtual BOOL InitInstance();
```

```
	//}}AFX_VIRTUAL

// Implementation

	//{{AFX_MSG(CStickerApp)
		// NOTE - the ClassWizard will add and remove member
functions here.
		//    DO NOT EDIT what you see in these blocks of
generated code !
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

void DrawBitmap(HDC hdc, HBITMAP hBitmap, int xStart, int yStart,
                        int nWidth=0, int nHeight=0);
void DrawStretchBitmap(HDC hdc, HBITMAP hBitmap, int xStart, int
yStart,
                        int nWidth=0, int nHeight=0, DWORD
dwStyle=SRCCOPY);
HBITMAP CreateSubBitmap(HDC hdc, HBITMAP hBitmap, int xStart,
                                int yStart, int nWidth=0, int
nHeight=0);
HBITMAP LoadBitmapFile(LPSTR FileName);

void DrawTransparentBitmap(HDC hdc, HBITMAP hBitmap, int xStart,
                        int yStart, int Width, int Height,
COLORREF cTransparentColor);
HBITMAP CopyScreenToBitmap(LPRECT lpRect, HDC hScr);
void DrawDIBSection( HDC hDC, HBITMAP hBitmap, int xDest, int yDest
);
BOOL SetBitmapPixels(CDC*     pDC, HBITMAP  hBmp, COLORREF
rgbTransparent,
                        int     nXOffset, int      nYOffset);
HBITMAP GetInvertedBitmap( HBITMAP hBitmap, BOOL bLateral );
BOOL WriteDIB( LPTSTR szFile, HANDLE hDIB);
HANDLE DDBToDIB( CBitmap& bitmap, DWORD dwCompression, CPalette*
pPal );
BOOL WriteWindowToDIB( LPTSTR szFile, CWnd *pWnd );
#endif //
!defined(AFX_STICKER_H__C3043CA5_D150_11D3_8AFC_E03A49C10001__INCLUD
ED_)
```

Sticker.cpp

```
// Sticker.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Sticker.h"
#include "StickerDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
CStickerApp

BEGIN_MESSAGE_MAP(CStickerApp, CWinApp)
	//{{AFX_MSG_MAP(CStickerApp)
		// NOTE - the ClassWizard will add and remove mapping
macros here.
		//    DO NOT EDIT what you see in these blocks of
generated code!
	//}}AFX_MSG
	ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CStickerApp construction

CStickerApp::CStickerApp()
{
	// TODO: add construction code here,
	free((void*)m_pszProfileName);//Change the name of the .INI
file.
	// Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CStickerApp object

CStickerApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CStickerApp initialization

BOOL CStickerApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the
size
	//  of your final executable, you should remove from the
following
	//  the specific initialization routines you do not need.
```

```
#ifdef _AFXDLL
	Enable3dControls();			// Call this when using
MFC in a shared DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC
statically
#endif

	m_pszProfileName=_tcsdup(_T("MySticker.INI"));
	SkinsPath = GetProfileString("Environtment", "SkinsPath");
	ApplicationPath = GetProfileString("Environtment",
"ApplicationPath");
	FrameStylePath = GetProfileString("Environtment",
"FrameStylePath");
	OpenStickerPath = GetProfileString("Environtment",
"OpenStickerPath");

	//Sticker Dialog
	Sticker_Dialog_Background.Format(SkinsPath+"/Main/Background.B
MP");
	Sticker_Dialog_Step1_Button.Format(SkinsPath+"/Main/Step1.BMP"
);
	Sticker_Dialog_Step2_Button.Format(SkinsPath+"/Main/Step2.BMP"
);
	Sticker_Dialog_Step3_Button.Format(SkinsPath+"/Main/Step3.BMP"
);
	Sticker_Dialog_Step4_Button.Format(SkinsPath+"/Main/Step4.BMP"
);
	Sticker_Dialog_Exit_Button.Format(SkinsPath+"/Main/Exit.BMP");
	Sticker_Dialog_Tool_Button.Format(SkinsPath+"/Main/Tool.BMP");
	Sticker_Dialog_Cur.Format(SkinsPath+"/Main/Dialog.CUR");
	Sticker_Dialog_Step1_Cur.Format(SkinsPath+"/Main/Step1.CUR");
	Sticker_Dialog_Step2_Cur.Format(SkinsPath+"/Main/Step2.CUR");
	Sticker_Dialog_Step3_Cur.Format(SkinsPath+"/Main/Step3.CUR");
	Sticker_Dialog_Step4_Cur.Format(SkinsPath+"/Main/Step4.CUR");
	Sticker_Dialog_Exit_Cur.Format(SkinsPath+"/Main/Exit.CUR");
	Sticker_Dialog_Tool_Cur.Format(SkinsPath+"/Main/Tool.CUR");
	Sticker_Dialog_INI_Path.Format(SkinsPath+"/Main/Main.INI");

	m_pszProfileName=_tcsdup(_T(Sticker_Dialog_INI_Path.GetBuffer
(0)));
	Sticker_Dialog_Background_Width = GetProfileInt
("Background","Width",0);
	Sticker_Dialog_Background_Height = GetProfileInt
("Background","Height",0);
	Sticker_Dialog_Step1_Width = GetProfileInt("Step1","Width",0);
	Sticker_Dialog_Step1_Height =
GetProfileInt("Step1","Height",0);
	Sticker_Dialog_Step1_X = GetProfileInt("Step1","PositionX",0);
	Sticker_Dialog_Step1_Y = GetProfileInt("Step1","PositionY",0);
	Sticker_Dialog_Step2_Width = GetProfileInt("Step2","Width",0);
	Sticker_Dialog_Step2_Height =
GetProfileInt("Step2","Height",0);
	Sticker_Dialog_Step2_X = GetProfileInt("Step2","PositionX",0);
	Sticker_Dialog_Step2_Y = GetProfileInt("Step2","PositionY",0);
	Sticker_Dialog_Step3_Width = GetProfileInt("Step3","Width",0);
	Sticker_Dialog_Step3_Height =
GetProfileInt("Step3","Height",0);
```

```
	Sticker_Dialog_Step3_X = GetProfileInt("Step3","PositionX",0);
	Sticker_Dialog_Step3_Y = GetProfileInt("Step3","PositionY",0);
	Sticker_Dialog_Step4_Width = GetProfileInt("Step4","Width",0);
	Sticker_Dialog_Step4_Height =
GetProfileInt("Step4","Height",0);
	Sticker_Dialog_Step4_X = GetProfileInt("Step4","PositionX",0);
	Sticker_Dialog_Step4_Y = GetProfileInt("Step4","PositionY",0);
	Sticker_Dialog_Exit_Width = GetProfileInt("Exit","Width",0);
	Sticker_Dialog_Exit_Height = GetProfileInt("Exit","Height",0);
	Sticker_Dialog_Exit_X = GetProfileInt("Exit","PositionX",0);
	Sticker_Dialog_Exit_Y = GetProfileInt("Exit","PositionY",0);
	Sticker_Dialog_Tool_Width = GetProfileInt("Tool","Width",0);
	Sticker_Dialog_Tool_Height = GetProfileInt("Tool","Height",0);
	Sticker_Dialog_Tool_X = GetProfileInt("Tool","PositionX",0);
	Sticker_Dialog_Tool_Y = GetProfileInt("Tool","PositionY",0);
	//...

	//New Open Dialog
	New_Open_Dialog_Background.Format(SkinsPath+"/NewOpenDialog/Ba
ckground.BMP");
	New_Open_Dialog_New_Button.Format(SkinsPath+"/NewOpenDialog/Ne
wButton.BMP");
	New_Open_Dialog_Open_Button.Format(SkinsPath+"/NewOpenDialog/O
penButton.BMP");
	New_Open_Dialog_Cur.Format(SkinsPath+"/NewOpenDialog/Dialog.CU
R");
	New_Open_Dialog_New_Cur.Format(SkinsPath+"/NewOpenDialog/New.C
UR");
	New_Open_Dialog_Open_Cur.Format(SkinsPath+"/NewOpenDialog/Open
.CUR");
	New_Open_Dialog_INI_Path.Format(SkinsPath+"/NewOpenDialog/NewO
pen.INI");


	m_pszProfileName=_tcsdup(_T(New_Open_Dialog_INI_Path.GetBuffer
(0)));
	New_Open_Dialog_Background_Width = GetProfileInt
("BackGround","Width",0);
	New_Open_Dialog_Background_Height = GetProfileInt
("BackGround","Height",0);
	New_Open_Dialog_New_Width =
GetProfileInt("NewButton","Width",0);
	New_Open_Dialog_New_Height =
GetProfileInt("NewButton","Height",0);
	New_Open_Dialog_New_X =
GetProfileInt("NewButton","PositionX",0);
	New_Open_Dialog_New_Y =
GetProfileInt("NewButton","PositionY",0);
	New_Open_Dialog_Open_Width =
GetProfileInt("OpenButton","Width",0);
	New_Open_Dialog_Open_Height =
GetProfileInt("OpenButton","Height",0);
	New_Open_Dialog_Open_X =
GetProfileInt("OpenButton","PositionX",0);
	New_Open_Dialog_Open_Y =
GetProfileInt("OpenButton","PositionY",0);
	//...
```

```
	//Open Dialog
	Open_Dialog_Background.Format(SkinsPath+"/OpenDialog/Backgroun
d.BMP");
	Open_Dialog_Next_Button.Format(SkinsPath+"/OpenDialog/Next.BMP
");
	Open_Dialog_Back_Button.Format(SkinsPath+"/OpenDialog/Back.BMP
");
	Open_Dialog_Done_Button.Format(SkinsPath+"/OpenDialog/Done.BMP
");
	Open_Dialog_Cur.Format(SkinsPath+"/OpenDialog/Dialog.CUR");
	Open_Dialog_Next_Cur.Format(SkinsPath+"/OpenDialog/Next.CUR");
	Open_Dialog_Back_Cur.Format(SkinsPath+"/OpenDialog/Back.CUR");
	Open_Dialog_Done_Cur.Format(SkinsPath+"/OpenDialog/Done.CUR");
	Open_Dialog_INI_Path.Format(SkinsPath+"/OpenDialog/Open.INI");


	m_pszProfileName=_tcsdup(_T(Open_Dialog_INI_Path.GetBuffer
(0)));
	Open_Dialog_Background_Width =
GetProfileInt("BackGround","Width",0);
	Open_Dialog_Background_Height =
GetProfileInt("BackGround","Height",0);
	Open_Dialog_Next_Width = GetProfileInt("Next","Width",0);
	Open_Dialog_Next_Height = GetProfileInt("Next","Height",0);
	Open_Dialog_Next_X = GetProfileInt("Next","PositionX",0);
	Open_Dialog_Next_Y = GetProfileInt("Next","PositionY",0);
	Open_Dialog_Done_Width = GetProfileInt("Done","Width",0);
	Open_Dialog_Done_Height = GetProfileInt("Done","Height",0);
	Open_Dialog_Done_X = GetProfileInt("Done","PositionX",0);
	Open_Dialog_Done_Y = GetProfileInt("Done","PositionY",0);
	Open_Dialog_Back_Width = GetProfileInt("Back","Width",0);
	Open_Dialog_Back_Height = GetProfileInt("Back","Height",0);
	Open_Dialog_Back_X = GetProfileInt("Back","PositionX",0);
	Open_Dialog_Back_Y = GetProfileInt("Back","PositionY",0);
	Open_Dialog_Preview_Width =
GetProfileInt("Preview","Width",0);
	Open_Dialog_Preview_Height =
GetProfileInt("Preview","Height",0);
	Open_Dialog_Preview_X =
GetProfileInt("Preview","PositionX",0);
	Open_Dialog_Preview_Y =
GetProfileInt("Preview","PositionY",0);

	//...

	//Select Frame Dialog
	Select_Frame_Dialog_Background.Format(SkinsPath+"/SelectFrameD
ialog/Background.BMP");
	Select_Frame_Dialog_Next_Button.Format(SkinsPath+"/SelectFrame
Dialog/Next.BMP");
	Select_Frame_Dialog_Back_Button.Format(SkinsPath+"/SelectFrame
Dialog/Back.BMP");
	Select_Frame_Dialog_Done_Button.Format(SkinsPath+"/SelectFrame
Dialog/Done.BMP");
	Select_Frame_Dialog_Cur.Format(SkinsPath+"/SelectFrameDialog/D
ialog.CUR");
	Select_Frame_Dialog_Next_Cur.Format(SkinsPath+"/SelectFrameDia
log/Next.CUR");
```

```
	Select_Frame_Dialog_Back_Cur.Format(SkinsPath+"/SelectFrameDia
log/Back.CUR");
	Select_Frame_Dialog_Done_Cur.Format(SkinsPath+"/SelectFrameDia
log/Done.CUR");
	Select_Frame_Dialog_INI_Path.Format(SkinsPath+"/SelectFrameDia
log/SelectFrame.INI");


	m_pszProfileName=_tcsdup(_T(Select_Frame_Dialog_INI_Path.GetBu
ffer(0)));
	Select_Frame_Dialog_Background_Width = GetProfileInt
("BackGround","Width",0);
	Select_Frame_Dialog_Background_Height = GetProfileInt
("BackGround","Height",0);
	Select_Frame_Dialog_Next_Width =
GetProfileInt("Next","Width",0);
	Select_Frame_Dialog_Next_Height = GetProfileInt
("Next","Height",0);
	Select_Frame_Dialog_Next_X =
GetProfileInt("Next","PositionX",0);
	Select_Frame_Dialog_Next_Y =
GetProfileInt("Next","PositionY",0);
	Select_Frame_Dialog_Done_Width =
GetProfileInt("Done","Width",0);
	Select_Frame_Dialog_Done_Height = GetProfileInt
("Done","Height",0);
	Select_Frame_Dialog_Done_X =
GetProfileInt("Done","PositionX",0);
	Select_Frame_Dialog_Done_Y =
GetProfileInt("Done","PositionY",0);
	Select_Frame_Dialog_Back_Width =
GetProfileInt("Back","Width",0);
	Select_Frame_Dialog_Back_Height = GetProfileInt
("Back","Height",0);
	Select_Frame_Dialog_Back_X =
GetProfileInt("Back","PositionX",0);
	Select_Frame_Dialog_Back_Y =
GetProfileInt("Back","PositionY",0);
	Select_Frame_Dialog_Preview_Width = GetProfileInt
("Preview","Width",0);
	Select_Frame_Dialog_Preview_Height = GetProfileInt
("Preview","Height",0);
	Select_Frame_Dialog_Preview_X =
GetProfileInt("Preview","PositionX",0);
	Select_Frame_Dialog_Preview_Y =
GetProfileInt("Preview","PositionY",0);

	//...

	//Select Frame Dialog
	Capture_Dialog_Background.Format(SkinsPath+"/CaptureDialog/Bac
kground.BMP");
	Capture_Dialog_Capture_Button.Format(SkinsPath+"/CaptureDialog
/Capture.BMP");
	Capture_Dialog_Cur.Format(SkinsPath+"/CaptureDialog/Dialog.CUR
");
	Capture_Dialog_Capture_Cur.Format(SkinsPath+"/CaptureDialog/Ca
pture.CUR");
```

```
      Capture_Dialog_INI_Path.Format(SkinsPath+"/CaptureDialog/Captu
re.INI");


      m_pszProfileName=_tcsdup(_T(Capture_Dialog_INI_Path.GetBuffer
(0)));
      Capture_Dialog_Background_Width = GetProfileInt
("BackGround","Width",0);
      Capture_Dialog_Background_Height = GetProfileInt
("BackGround","Height",0);
      Capture_Dialog_Capture_Width =
GetProfileInt("Capture","Width",0);
      Capture_Dialog_Capture_Height =
GetProfileInt("Capture","Height",0);
      Capture_Dialog_Capture_X =
GetProfileInt("Capture","PositionX",0);
      Capture_Dialog_Capture_Y =
GetProfileInt("Capture","PositionY",0);
      Capture_Dialog_Preview_Width =
GetProfileInt("Preview","Width",0);
      Capture_Dialog_Preview_Height =
GetProfileInt("Preview","Height",0);
      Capture_Dialog_Preview_X =
GetProfileInt("Preview","PositionX",0);
      Capture_Dialog_Preview_Y =
GetProfileInt("Preview","PositionY",0);

      //...

      //Select Frame Dialog
      PrintPreview_Dialog_Background.Format(SkinsPath+"/PrintPreview
Dialog/Background.BMP");
      PrintPreview_Dialog_Print_Button.Format(SkinsPath+"/PrintPrevi
ewDialog/Print.BMP");
      PrintPreview_Dialog_PrintPreview_Button.Format(SkinsPath+"/Pri
ntPreviewDialog/PrintPreview.BMP");
      PrintPreview_Dialog_Cur.Format(SkinsPath+"/PrintPreviewDialog/
Dialog.CUR");
      PrintPreview_Dialog_Print_Cur.Format(SkinsPath+"/PrintPreviewD
ialog/Print.CUR");
      PrintPreview_Dialog_PrintPreview_Cur.Format(SkinsPath+"/PrintP
reviewDialog/PrintPreview.CUR");
      PrintPreview_Dialog_INI_Path.Format(SkinsPath+"/PrintPreviewDi
alog/PrintPreview.INI");


      m_pszProfileName=_tcsdup(_T(PrintPreview_Dialog_INI_Path.GetBu
ffer(0)));
      PrintPreview_Dialog_Background_Width = GetProfileInt
("BackGround","Width",0);
      PrintPreview_Dialog_Background_Height = GetProfileInt
("BackGround","Height",0);
      PrintPreview_Dialog_Print_Width = GetProfileInt
("Print","Width",0);
      PrintPreview_Dialog_Print_Height = GetProfileInt
("Print","Height",0);
      PrintPreview_Dialog_Print_X =
GetProfileInt("Print","PositionX",0);
```

```
      PrintPreview_Dialog_Print_Y =
GetProfileInt("Print","PositionY",0);
      PrintPreview_Dialog_PrintPreview_Width = GetProfileInt
("PrintPreview","Width",0);
      PrintPreview_Dialog_PrintPreview_Height = GetProfileInt
("PrintPreview","Height",0);
      PrintPreview_Dialog_PrintPreview_X = GetProfileInt
("PrintPreview","PositionX",0);
      PrintPreview_Dialog_PrintPreview_Y = GetProfileInt
("PrintPreview","PositionY",0);
      PrintPreview_Dialog_Preview_Width = GetProfileInt
("Preview","Width",0);
      PrintPreview_Dialog_Preview_Height = GetProfileInt
("Preview","Height",0);
      PrintPreview_Dialog_Preview_X =
GetProfileInt("Preview","PositionX",0);
      PrintPreview_Dialog_Preview_Y =
GetProfileInt("Preview","PositionY",0);

      //...

      //for startup
      CString
SubKey("Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce");
      CString Data;
      Data.Format(GetAppPath()+"\\Sticker.EXE");

      HKEY hKey;
      RegOpenKey( HKEY_LOCAL_MACHINE, LPCTSTR(SubKey), &hKey);

      m_pszProfileName=_tcsdup(_T("MySticker.INI"));

      if (GetProfileInt("Environtment","StartupType",0)==0)
            RegDeleteValue( hKey, "TheSticker");
      else
            RegSetValueEx( hKey, "TheSticker",0, REG_SZ,
            (const unsigned char *)LPCSTR(Data),(DWORD)
(Data.GetLength()+1)*sizeof(TCHAR));

      RegCloseKey(hKey);

      //...

      //for scan frame
      FrameFileName.Empty();
      CFileFind file;
      FrameCount = 0;
      CString Path;
      Path.Format("%s%s",LPCTSTR(GetFrameStylePath()),"\\*.*");
      int result;
      if (result=file.FindFile(LPCTSTR(Path)))
      {
            while (file.FindNextFile())
            {
                  if (file.IsDirectory()&&
                        (file.GetFileName()!=".")&&
                        (file.GetFileName()!=".."))
                  {
```

```
                        FrameFileName.Insert(FrameFileName.GetLength
(),LPCTSTR(file.GetFileName()));
                        FrameFileName.Insert(FrameFileName.GetLength
(),",");
                        FrameCount++;
                }
            }
            if (file.IsDirectory()&&
                        (file.GetFileName()!=".")&&
                        (file.GetFileName()!=".."))
            {
                FrameFileName.Insert(FrameFileName.GetLength
(),LPCTSTR(file.GetFileName()));
                FrameFileName.Insert(FrameFileName.GetLength
(),",");
                FrameCount++;
            }
      }
      //...
      file.Close();
      CStickerDlg dlg;
      m_pMainWnd = &dlg;

      int nResponse = dlg.DoModal();
      if (nResponse == IDOK)
      {
            // TODO: Place code here to handle when the dialog is
            //  dismissed with OK
      }
      else if (nResponse == IDCANCEL)
      {
            // TODO: Place code here to handle when the dialog is
            //  dismissed with Cancel
      }

      // Since the dialog has been closed, return FALSE so that we
exit the
      //  application, rather than start the application's message
pump.
            // Initialize DirectSound Object
      return FALSE;
}

HBITMAP CStickerApp::GetFrameAt(UINT FrameID)
{
      POSITION pos;
      CFrame *frame;
      pos = pPaper->m_FrameList.GetHeadPosition();
      for (int i=0;i<FrameID;i++)
            pPaper->m_FrameList.GetNext(pos);
      frame = pPaper->m_FrameList.GetAt(pos);
      return frame->GetFrame();
}

COLORREF StickerApp::GetColorRefAt(UINT FrameID)
{
      POSITION pos;
      CFrame *frame;
```

```
	pos = pPaper->m_FrameList.GetHeadPosition();
	for (int i=0;i<FrameID;i++)
		pPaper->m_FrameList.GetNext(pos);
	frame = pPaper->m_FrameList.GetAt(pos);
	return frame->GetColorRef();
}

CStickerApp::FindSticker()
{
	POSITION pos;
	pos = m_StickerList.GetHeadPosition();
	//for scan frame
	OpenFileName.Empty();
	CFileFind file;
	OpenCount = 0;
	CString Path;
	Path.Format("%s%s",LPCTSTR(GetOpenStickerPath()),"\\*.BMP");
	int result;
	if (result=file.FindFile(LPCTSTR(Path)))
	{
		while (file.FindNextFile())
		{
			if ((file.GetFileName()!=".")&&
				(file.GetFileName()!=".."))
			{
				OpenFileName.Insert(OpenFileName.GetLength
(),LPCTSTR(file.GetFileName()));
				OpenFileName.Insert(OpenFileName.GetLength
(),",");
				OpenCount++;
			}
		}
		if ((file.GetFileName()!=".")&&
			(file.GetFileName()!=".."))
		{
			OpenFileName.Insert(OpenFileName.GetLength
(),LPCTSTR(file.GetFileName()));
			OpenFileName.Insert(OpenFileName.GetLength(),",");
			OpenCount++;
		}
	}
	//...
	file.Close();

	for (int i=0;i<OpenCount;i++)
	{
		Path.Format("%s%s%s",GetOpenStickerPath(),"\\",
			GetOpenFileName(i));
		m_StickerList.AddTail(::LoadBitmapFile((char*)LPCSTR
(Path)));
		Path.Empty();
	}
}

CStickerApp::ClearStickerList()
{
	POSITION pos;
	pos = m_StickerList.GetHeadPosition();
```

```
	HBITMAP bmp;
	while (!m_StickerList.IsEmpty())
	{
		bmp = m_StickerList.GetHead();
		DeleteObject(bmp);
		m_StickerList.RemoveHead();
	}
}

CString CStickerApp::GetSkinsPath()
{
	return SkinsPath;
}

int CStickerApp::GetFrameCount()
{
	return FrameCount;
}

int CStickerApp::GetOpenCount()
{
	return OpenCount;
}

int CStickerApp::GetFrameSelect()
{
	return FrameSelect;
}

CStickerApp::SetFrameSelect(int _FrameSelect)
{
	FrameSelect = _FrameSelect;
}

CStickerApp::SetScriptFileName(CString FileName)
{
	ScriptFileName = FileName;
}

CString CStickerApp::GetScriptFileName()
{
	return ScriptFileName;
}

CString CStickerApp::GetAppPath()
{
	return ApplicationPath;
}

CString CStickerApp::GetFrameStylePath()
{
	return FrameStylePath;
}

CString CStickerApp::GetOpenStickerPath()
{
	return OpenStickerPath;
}
```

```
CString     CStickerApp::GetFrameFileName(UINT FrameID/*=0*/)
{
      CString name;
      name.Insert(0,LPCTSTR(FrameFileName));
      int index;
      for (int i=0;i<FrameID;i++)
      {
            index = name.Find(",");
            name.Delete(0,index+1);
      }
      index = name.Find(",");
      name.SetAt(index,NULL);
      return name;
}

CString     CStickerApp::GetOpenFileName(UINT FrameID/*=0*/)
{
      CString name;
      name.Insert(0,LPCTSTR(OpenFileName));
      int index;
      for (int i=0;i<FrameID;i++)
      {
            index = name.Find(",");
            name.Delete(0,index+1);
      }
      index = name.Find(",");
      name.SetAt(index,NULL);
      return name;
}

CStickerApp::CreatePaper()
{
      pPaper = new CPaper(ScriptFileName);
}

CStickerApp::DestroyPaper()
{
      pPaper->DestroyFrame();
      delete pPaper;
}
```

StickerDlg.h

```
// StickerDlg.h : header file
//

#if
!defined(AFX_STICKERDLG_H__C3043CA7_D150_11D3_8AFC_E03A49C10001__INC
LUDED_)
#define
AFX_STICKERDLG_H__C3043CA7_D150_11D3_8AFC_E03A49C10001__INCLUDED_

//#include "DirectSound.h"
//#include "MIDI.h"
#include "MyButton.h"
#include "NewOpenDlg.h"
#include "MyDialog.h"
#include "SelectFrame.h"
```

```
#include "ToolDlg.h"
#include "CaptureDlg.h"
#include "PrintPreviewDlg.h"
#include "OpenDlg.h"

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CStickerDlg dialog

class CStickerDlg : public CMyDialog
{
// Construction
public:
	CStickerDlg(CWnd* pParent = NULL);  // standard constructor
	CNewOpenDlg m_NewOpenDlg;
	CSelectFrame m_SelectFrameDlg;
	CToolDlg m_ToolDlg;
	CCaptureDlg m_CaptureDlg;
	CPrintPreviewDlg m_PrintPreviewDlg;
	COpenDlg m_OpenDlg;

// Dialog Data
	//{{AFX_DATA(CStickerDlg)
	enum { IDD = IDD_STICKER_DIALOG };
	CMyButton	m_ToolButton;
	CMyButton	m_Step4Button;
	CMyButton	m_Step3Button;
	CMyButton	m_Step2Button;
	CMyButton	m_Step1Button;
	CMyButton	m_ExitButton;
	CButton	m_DlgArea;
	CStatic	m_text;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CStickerDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	//
DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON		m_hIcon;
//	CDirectSound	m_Sound;
//	CMIDI		m_Midi;

	// Generated message map functions
	//{{AFX_MSG(CStickerDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnExitButton();
	afx_msg void OnStep1();
	afx_msg void OnTool();
```

```
	afx_msg void OnStep2();
	afx_msg void OnStep3();
	afx_msg void OnStep4();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_STICKERDLG_H__C3043CA7_D150_11D3_8AFC_E03A49C10001__INC
LUDED_)
```

StickerDlg.cpp

```
// StickerDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Sticker.h"
#include "StickerDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CStickerDlg dialog

CStickerDlg::CStickerDlg(CWnd* pParent /*=NULL*/)
	: CMyDialog(CStickerDlg::IDD, pParent),
	m_NewOpenDlg(NULL),
	m_SelectFrameDlg(NULL),
	m_ToolDlg(NULL),
	m_CaptureDlg(NULL),
	m_PrintPreviewDlg(NULL),
	m_OpenDlg(NULL)
{
	//{{AFX_DATA_INIT(CStickerDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent
DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CStickerDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CStickerDlg)
	DDX_Control(pDX, IDC_TOOL, m_ToolButton);
	DDX_Control(pDX, IDC_STEP4, m_Step4Button);
	DDX_Control(pDX, IDC_STEP3, m_Step3Button);
	DDX_Control(pDX, IDC_STEP2, m_Step2Button);
```

```
	DDX_Control(pDX, IDC_STEP1, m_Step1Button);
	DDX_Control(pDX, IDC_EXIT_BUTTON, m_ExitButton);
	DDX_Control(pDX, IDC_DLG_AREA, m_DlgArea);
	DDX_Control(pDX, IDC_TEXT1, m_text);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CStickerDlg, CMyDialog)
	//{{AFX_MSG_MAP(CStickerDlg)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_EXIT_BUTTON, OnExitButton)
	ON_BN_CLICKED(IDC_STEP1, OnStep1)
	ON_BN_CLICKED(IDC_TOOL, OnTool)
	ON_BN_CLICKED(IDC_STEP2, OnStep2)
	ON_BN_CLICKED(IDC_STEP3, OnStep3)
	ON_BN_CLICKED(IDC_STEP4, OnStep4)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CStickerDlg message handlers

BOOL CStickerDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this
automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	CStickerApp *pApp = (CStickerApp*) AfxGetApp();

	HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;

	hButtonAll = ::LoadBitmapFile(pApp->
Sticker_Dialog_Step1_Button.GetBuffer(0));

	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Sticker_Dialog_Step1_Width, pApp->
Sticker_Dialog_Step1_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Sticker_Dialog_Step1_Height, pApp->
Sticker_Dialog_Step1_Width, pApp->Sticker_Dialog_Step1_Height*2-1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Sticker_Dialog_Step1_Height*2, pApp->
Sticker_Dialog_Step1_Width, pApp->Sticker_Dialog_Step1_Height*3-1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
		0, pApp->Sticker_Dialog_Step1_Height*3, pApp->
Sticker_Dialog_Step1_Width, pApp->Sticker_Dialog_Step1_Height*4-1);
```

```
      m_Step1Button.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_Step1Button.SizeToContent();

      hButtonAll = ::LoadBitmapFile(pApp->
Sticker_Dialog_Step2_Button.GetBuffer(0));

      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Sticker_Dialog_Step2_Width, pApp->
Sticker_Dialog_Step2_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step2_Height, pApp->
Sticker_Dialog_Step2_Width, pApp->Sticker_Dialog_Step2_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step2_Height*2, pApp->
Sticker_Dialog_Step2_Width, pApp->Sticker_Dialog_Step2_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Sticker_Dialog_Step2_Height*3, pApp->
Sticker_Dialog_Step2_Width, pApp->Sticker_Dialog_Step2_Height*4-1);

      m_Step2Button.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_Step2Button.SizeToContent();

      hButtonAll = ::LoadBitmapFile(pApp-
>Sticker_Dialog_Step3_Button.GetBuffer(0));

      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Sticker_Dialog_Step3_Width, pApp->
Sticker_Dialog_Step3_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step3_Height, pApp->
Sticker_Dialog_Step3_Width, pApp->Sticker_Dialog_Step3_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step3_Height*2, pApp->
Sticker_Dialog_Step3_Width, pApp->Sticker_Dialog_Step3_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Sticker_Dialog_Step3_Height*3, pApp->
Sticker_Dialog_Step3_Width, pApp->Sticker_Dialog_Step3_Height*4-1);

      m_Step3Button.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_Step3Button.SizeToContent();

      hButtonAll = ::LoadBitmapFile(pApp-
>Sticker_Dialog_Step4_Button.GetBuffer(0));

      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Sticker_Dialog_Step4_Width, pApp->
Sticker_Dialog_Step4_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step4_Height, pApp->
Sticker_Dialog_Step4_Width, pApp->Sticker_Dialog_Step4_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Step4_Height*2, pApp->
Sticker_Dialog_Step4_Width, pApp->Sticker_Dialog_Step4_Height*3-1);
```

```
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Sticker_Dialog_Step4_Height*3, pApp->
Sticker_Dialog_Step4_Width, pApp->Sticker_Dialog_Step4_Height*4-1);

      m_Step4Button.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_Step4Button.SizeToContent();

      hButtonAll = ::LoadBitmapFile(pApp-
>Sticker_Dialog_Exit_Button.GetBuffer(0));

      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Sticker_Dialog_Exit_Width, pApp->
Sticker_Dialog_Exit_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Exit_Height, pApp->
Sticker_Dialog_Exit_Width, pApp->Sticker_Dialog_Exit_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Exit_Height*2, pApp->
Sticker_Dialog_Exit_Width, pApp->Sticker_Dialog_Exit_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Sticker_Dialog_Exit_Height*3, pApp->
Sticker_Dialog_Exit_Width, pApp->Sticker_Dialog_Exit_Height*4-1);

      m_ExitButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_ExitButton.SizeToContent();

      hButtonAll = ::LoadBitmapFile(pApp-
>Sticker_Dialog_Tool_Button.GetBuffer(0));

      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Sticker_Dialog_Tool_Width, pApp->
Sticker_Dialog_Tool_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Tool_Height, pApp->
Sticker_Dialog_Tool_Width, pApp->Sticker_Dialog_Tool_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Sticker_Dialog_Tool_Height*2, pApp->
Sticker_Dialog_Tool_Width, pApp->Sticker_Dialog_Tool_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Sticker_Dialog_Tool_Height*3, pApp->
Sticker_Dialog_Tool_Width, pApp->Sticker_Dialog_Tool_Height*4-1);

      m_ToolButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_ToolButton.SizeToContent();

      //for cursor
      SetCursor(::LoadCursorFromFile(pApp-
>Sticker_Dialog_Cur.GetBuffer(0)));
      m_Step1Button.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Step1_Cur.GetBuffer(0)));
      m_Step2Button.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Step2_Cur.GetBuffer(0)));
```

```
      m_Step3Button.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Step3_Cur.GetBuffer(0)));
      m_Step4Button.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Step4_Cur.GetBuffer(0)));
      m_ExitButton.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Exit_Cur.GetBuffer(0)));
      m_ToolButton.SetCursor(::LoadCursorFromFile(pApp->
Sticker_Dialog_Tool_Cur.GetBuffer(0)));

      //for sound
      m_Step1Button.SetSound(IDR_DEFAULT_SOUND);
      m_Step2Button.SetSound(IDR_DEFAULT_SOUND);
      m_Step3Button.SetSound(IDR_DEFAULT_SOUND);
      m_Step4Button.SetSound(IDR_DEFAULT_SOUND);
      m_ExitButton.SetSound(IDR_DEFAULT_SOUND);
      m_ToolButton.SetSound(IDR_DEFAULT_SOUND);


      m_Step1Button.MoveWindow(pApp->Sticker_Dialog_Step1_X,
                               pApp->Sticker_Dialog_Step1_Y,
                               pApp-
>Sticker_Dialog_Step1_Width,
                               pApp-
>Sticker_Dialog_Step1_Height);

      m_Step2Button.MoveWindow(pApp->Sticker_Dialog_Step2_X,
                               pApp->Sticker_Dialog_Step2_Y,
                               pApp-
>Sticker_Dialog_Step2_Width,
                               pApp-
>Sticker_Dialog_Step2_Height);

      m_Step3Button.MoveWindow(pApp->Sticker_Dialog_Step3_X,
                               pApp->Sticker_Dialog_Step3_Y,
                               pApp-
>Sticker_Dialog_Step3_Width,
                               pApp-
>Sticker_Dialog_Step3_Height);

      m_Step4Button.MoveWindow(pApp->Sticker_Dialog_Step4_X,
                               pApp->Sticker_Dialog_Step4_Y,
                               pApp-
>Sticker_Dialog_Step4_Width,
                               pApp-
>Sticker_Dialog_Step4_Height);

      m_ExitButton.MoveWindow(pApp->Sticker_Dialog_Exit_X,
                               pApp->Sticker_Dialog_Exit_Y,
                               pApp->Sticker_Dialog_Exit_Width,
                               pApp-
>Sticker_Dialog_Exit_Height);

      m_ToolButton.MoveWindow(pApp->Sticker_Dialog_Tool_X,
                               pApp->Sticker_Dialog_Tool_Y,
                               pApp->Sticker_Dialog_Tool_Width,
                               pApp-
>Sticker_Dialog_Tool_Height);
```

```
	HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>Sticker_Dialog_Background.GetBuffer(0));

	SetBitmap(hBitmap);
	SetSound(IDR_START_SOUND);
	//SetMidi(IDR_DEFAULT_MIDI);
	ShowWindow(TRUE);
	BringWindowToTop();

	WINDOWPLACEMENT wp;
	GetWindowPlacement(&wp);
	wp.showCmd = SW_SHOWMAXIMIZED;
	SetWindowPlacement(&wp);
	m_text.MoveWindow(700,560,100,100);
	MoveWindow(0,0,pApp->Sticker_Dialog_Background_Width,pApp->
Sticker_Dialog_Background_Height);
	CenterWindow();

	m_NewOpenDlg.Create(IDD_NEW_OPEN_DIALOG, this);
	m_ToolDlg.Create(IDD_TOOL_DIALOG, this);
	m_SelectFrameDlg.Create(IDD_SELECT_FRAME_DIALOG, this);
	m_CaptureDlg.Create(IDD_CAPTURE_DIALOG, this);
	m_PrintPreviewDlg.Create(IDD_PRINT_PREVIEW_DIALOG, this);
	m_OpenDlg.Create(IDD_OPEN_DIALOG, this);

	m_Step2Button.EnableWindow(FALSE);
	m_Step3Button.EnableWindow(FALSE);
	m_Step4Button.EnableWindow(FALSE);
	m_ToolButton.EnableWindow(TRUE);

	return TRUE;  // return TRUE  unless you set the focus to a
control
}

// If you add a minimize button to your dialog, you will need the
code below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CStickerDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
```

```
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            //::DrawBitmap(::GetWindowDC(m_hWnd), m_hBackground, 0,
0, 0, 0);
            CDialog::OnPaint();
      }
}

// The system calls this to obtain the cursor to display while the
user drags
//  the minimized window.
HCURSOR CStickerDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

HBITMAP LoadBitmapFile(LPSTR FileName)
{
      return (HBITMAP) ::LoadImage(AfxGetApp()->m_hInstance,
                        FileName, IMAGE_BITMAP,0,0,

LR_LOADFROMFILE|LR_DEFAULTCOLOR);//|LR_LOADTRANSPARENT);
}

void DrawBitmap(HDC hdc, HBITMAP hBitmap, int xStart, int yStart,
                        int nWidth, int nHeight)
{
      BITMAP      bm;
      HDC         hdcMem;
      POINT ptSize, ptOrg;

      //Create memory device context
      hdcMem = CreateCompatibleDC(hdc);
      //Set hdcMem to same type hBitmap
      SelectObject(hdcMem, hBitmap);
      SetMapMode(hdcMem, GetMapMode(hdc));
      GetObject(hBitmap, sizeof(BITMAP), (LPVOID) &bm);
      if (!(nWidth && nHeight))
      {
            ptSize.x = bm.bmWidth;
            ptSize.y = bm.bmHeight;
      }
      else
      {
            ptSize.x = nWidth;
            ptSize.y = nHeight;
      }
      DPtoLP(hdc, &ptSize, 1);

      ptOrg.x = 0;
      ptOrg.y = 0;
      DPtoLP(hdcMem, &ptOrg, 1);

      BitBlt(hdc, xStart, yStart, ptSize.x, ptSize.y,
            hdcMem, ptOrg.x, ptOrg.y, SRCCOPY);
      DeleteDC(hdcMem);
```

```
}

void DrawStretchBitmap(HDC hdc, HBITMAP hBitmap, int xStart, int
yStart,
                       int nWidth, int nHeight, DWORD dwStyle)
{
      BITMAP      bm;
      HDC         hdcMem;
      POINT ptSize, ptOrg;

      //Create memory device context
      hdcMem = CreateCompatibleDC(hdc);
      //Set hdcMem to same type hBitmap
      SelectObject(hdcMem, hBitmap);
      SetMapMode(hdcMem, GetMapMode(hdc));
      GetObject(hBitmap, sizeof(BITMAP), (LPVOID) &bm);
      if (!(nWidth && nHeight))
      {
            ptSize.x = bm.bmWidth;
            ptSize.y = bm.bmHeight;
      }
      else
      {
            ptSize.x = bm.bmWidth;//nWidth;
            ptSize.y = bm.bmHeight;//nHeight;
      }
      DPtoLP(hdc, &ptSize, 1);

      ptOrg.x = 0;
      ptOrg.y = 0;
      DPtoLP(hdcMem, &ptOrg, 1);

      //BitBlt(hdc, xStart, yStart, ptSize.x, ptSize.y,
      //       hdcMem, ptOrg.x, ptOrg.y, SRCCOPY);
      SetStretchBltMode(hdc, COLORONCOLOR);
      StretchBlt( hdc, xStart, yStart, nWidth, nHeight,
                  hdcMem,     ptOrg.x, ptOrg.y, ptSize.x, ptSize.y,
dwStyle);
      DeleteDC(hdcMem);
}

HBITMAP CreateSubBitmap(HDC hdc, HBITMAP hBitmap, int xStart,
                                    int yStart, int nWidth, int
nHeight)
{
      BITMAP      bm;
      HDC         hdcSource,hdcDest;
      POINT ptSize, ptOrg;
      HBITMAP hDest;

      hdcSource = CreateCompatibleDC(hdc);
      SelectObject(hdcSource, hBitmap);
      SetMapMode(hdcSource, GetMapMode(hdc));

/*    hdcDest = CreateCompatibleDC(hdc);
      hDest = ::CreateCompatibleBitmap(hdcSource, nWidth, nHeight);
      //hDest = ::CreateBitmap(nWidth, nHeight, 256, 1, NULL);
      SelectObject(hdcDest, hDest);
```

```
      SetMapMode(hdcDest, GetMapMode(hdc));*/

      GetObject(hBitmap, sizeof(BITMAP), (LPVOID) &bm);
      //Set size of bitmap to create
      if (!(nWidth && nHeight))
      {
            ptSize.x = bm.bmWidth - xStart;
            ptSize.y = bm.bmHeight - yStart;
      }
      else
      {
            ptSize.x = nWidth;
            ptSize.y = nHeight;
      }
      DPtoLP(hdc, &ptSize, 1);

      hdcDest = CreateCompatibleDC(hdc);
      hDest = ::CreateCompatibleBitmap(hdcSource, ptSize.x,
ptSize.y);
      //hDest = ::CreateBitmap(nWidth, nHeight, 256, 1, NULL);
      SelectObject(hdcDest, hDest);
      SetMapMode(hdcDest, GetMapMode(hdc));

      ptOrg.x = xStart;
      ptOrg.y = yStart;
      DPtoLP(hdcSource, &ptOrg, 1);

      BitBlt(hdcDest, 0, 0, ptSize.x, ptSize.y,
            hdcSource, ptOrg.x, ptOrg.y, SRCCOPY);

      DeleteDC(hdcSource);
      DeleteDC(hdcDest);
      return hDest;
}

HBITMAP CreateStretchBitmap(HDC hdc, HBITMAP hBitmap, int xStart,
                                    int yStart, int nWidth, int
nHeight)
{
      BITMAP      bm;
      HDC         hdcSource,hdcDest;
      POINT ptSize, ptOrg;
      HBITMAP hDest;

      hdcSource = CreateCompatibleDC(hdc);
      SelectObject(hdcSource, hBitmap);
      SetMapMode(hdcSource, GetMapMode(hdc));

/*    hdcDest = CreateCompatibleDC(hdc);
      hDest = ::CreateCompatibleBitmap(hdcSource, nWidth, nHeight);
      //hDest = ::CreateBitmap(nWidth, nHeight, 256, 1, NULL);
      SelectObject(hdcDest, hDest);
      SetMapMode(hdcDest, GetMapMode(hdc));*/

      GetObject(hBitmap, sizeof(BITMAP), (LPVOID) &bm);
      //Set size of bitmap to create
      if (!(nWidth && nHeight))
      {
```

```
            ptSize.x = bm.bmWidth - xStart;
            ptSize.y = bm.bmHeight - yStart;
      }
      else
      {
            ptSize.x = nWidth;
            ptSize.y = nHeight;
      }
      DPtoLP(hdc, &ptSize, 1);

      hdcDest = CreateCompatibleDC(hdc);
      hDest = ::CreateCompatibleBitmap(hdcSource, nWidth, nHeight);
      //hDest = ::CreateBitmap(nWidth, nHeight, 256, 1, NULL);
      SelectObject(hdcDest, hDest);
      SetMapMode(hdcDest, GetMapMode(hdc));

      ptOrg.x = xStart;
      ptOrg.y = yStart;
      DPtoLP(hdcSource, &ptOrg, 1);

      SetStretchBltMode(hdcDest, COLORONCOLOR);
      StretchBlt(hdcDest, 0, 0, nWidth, nHeight,
            hdcSource, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY);

      DeleteDC(hdcSource);
      DeleteDC(hdcDest);
      return hDest;
}

void DrawTransparentBitmap(HDC hdc, HBITMAP hBitmap, int xStart,
                           int yStart, int Width, int Height,
COLORREF cTransparentColor)
{
   BITMAP     bm;   COLORREF   cColor;
   HBITMAP    bmAndBack, bmAndObject, bmAndMem, bmSave;
   HBITMAP    bmBackOld, bmObjectOld, bmMemOld, bmSaveOld;
   HDC        hdcMem, hdcBack, hdcObject, hdcTemp, hdcSave;
   POINT      ptSize;
   HBITMAP    hBM;

   hBM = CreateStretchBitmap(hdc, hBitmap, xStart, yStart, Width,
Height);

   hdcTemp = CreateCompatibleDC(hdc);
   SelectObject(hdcTemp, /*hBitmap*/hBM);   // Select the bitmap
   GetObject(/*hBitmap*/hBM, sizeof(BITMAP), (LPSTR)&bm);
   ptSize.x = bm.bmWidth;            // Get width of bitmap
   ptSize.y = bm.bmHeight;           // Get height of bitmap
   DPtoLP(hdcTemp, &ptSize, 1);      // Convert from device
                                     // to logical points
   // Create some DCs to hold temporary data.
   hdcBack   = CreateCompatibleDC(hdc);   hdcObject =
CreateCompatibleDC(hdc);
   hdcMem    = CreateCompatibleDC(hdc);   hdcSave   =
CreateCompatibleDC(hdc);
   // Create a bitmap for each DC. DCs are required for a number of
   // GDI functions.   // Monochrome DC
   bmAndBack   = CreateBitmap(ptSize.x, ptSize.y, 1, 1, NULL);
```

```
   // Monochrome DC
   bmAndObject = CreateBitmap(ptSize.x, ptSize.y, 1, 1, NULL);
   bmAndMem    = CreateCompatibleBitmap(hdc, ptSize.x, ptSize.y);
   bmSave      = CreateCompatibleBitmap(hdc, ptSize.x, ptSize.y);
   // Each DC must select a bitmap object to store pixel data.
   bmBackOld   = HBITMAP(SelectObject(hdcBack, bmAndBack));
   bmObjectOld = HBITMAP(SelectObject(hdcObject, bmAndObject));
   bmMemOld    = HBITMAP(SelectObject(hdcMem, bmAndMem));
   bmSaveOld   = HBITMAP(SelectObject(hdcSave, bmSave));    // Set
proper mapping mode.
   SetMapMode(hdcTemp, GetMapMode(hdc));
   // Save the bitmap sent here, because it will be overwritten.
   SetStretchBltMode(hdcSave, COLORONCOLOR);
   StretchBlt(hdcSave, 0, 0, ptSize.x, ptSize.y, hdcTemp, 0, 0,
ptSize.x, ptSize.y, SRCCOPY);
   // Set the background color of the source DC to the color.
   // contained in the parts of the bitmap that should be
transparent
   cColor = SetBkColor(hdcTemp, cTransparentColor);
   // Create the object mask for the bitmap by performing a BitBlt
   // from the source bitmap to a monochrome bitmap.
   SetStretchBltMode(hdcObject, COLORONCOLOR);
   StretchBlt(hdcObject, 0, 0, ptSize.x, ptSize.y, hdcTemp, 0, 0,
ptSize.x, ptSize.y, SRCCOPY);
   // Set the background color of the source DC back to the original
// color.
   SetBkColor(hdcTemp, cColor);    // Create the inverse of the
object mask.
   SetStretchBltMode(hdcBack, COLORONCOLOR);
   StretchBlt(hdcBack, 0, 0, ptSize.x, ptSize.y, hdcObject, 0, 0,
ptSize.x, ptSize.y,
          NOTSRCCOPY);
   // Copy the background of the main DC to the destination.
   SetStretchBltMode(hdcMem, COLORONCOLOR);
   StretchBlt(hdcMem, 0, 0, ptSize.x, ptSize.y, hdc, xStart, yStart,
ptSize.x, ptSize.y,
          SRCCOPY);   // Mask out the places where the bitmap will
be placed.
   StretchBlt(hdcMem, 0, 0, ptSize.x, ptSize.y, hdcObject, 0, 0,
ptSize.x, ptSize.y, SRCAND);
   // Mask out the transparent colored pixels on the bitmap.
   SetStretchBltMode(hdcTemp, COLORONCOLOR);
   StretchBlt(hdcTemp, 0, 0, ptSize.x, ptSize.y, hdcBack, 0, 0,
ptSize.x, ptSize.y, SRCAND);
   // XOR the bitmap with the background on the destination DC.
   StretchBlt(hdcMem, 0, 0, ptSize.x, ptSize.y, hdcTemp, 0, 0,
ptSize.x, ptSize.y, SRCPAINT);
   // Copy the destination to the screen.
   SetStretchBltMode(hdc, COLORONCOLOR);
   StretchBlt(hdc, xStart, yStart, ptSize.x, ptSize.y, hdcMem, 0, 0,
ptSize.x, ptSize.y,
          SRCCOPY);
   // Place the original bitmap back into the bitmap sent here.
   StretchBlt(hdcTemp, 0, 0, Width, Height, hdcSave, 0, 0, ptSize.x,
ptSize.y, SRCCOPY);

/*   COLORREF color = ::GetPixel(hdcTemp,Width/2,Height/2);
   CString txt;
```

```
   txt.Format("%d",color);
   MessageBox(NULL,LPCTSTR(txt),NULL,NULL);*/
   // Delete the memory bitmaps.
   DeleteObject(SelectObject(hdcBack, bmBackOld));
   DeleteObject(SelectObject(hdcObject, bmObjectOld));
   DeleteObject(SelectObject(hdcMem, bmMemOld));
   DeleteObject(SelectObject(hdcSave, bmSaveOld));   // Delete the
memory DCs.
   DeleteDC(hdcMem);   DeleteDC(hdcBack);   DeleteDC(hdcObject);
   DeleteDC(hdcSave);   DeleteDC(hdcTemp);
}

void CStickerDlg::OnExitButton()
{
      // TODO: Add your control notification handler code here
      DestroyWindow();
      //ExitWindowsEx(EWX_SHUTDOWN, NULL);
}

void CStickerDlg::OnStep1()
{
      // TODO: Add your control notification handler code here

      m_NewOpenDlg.ShowWindow(TRUE);
      m_Step1Button.EnableWindow(FALSE);
      m_Step2Button.EnableWindow(TRUE);
}

void CStickerDlg::OnTool()
{
      // TODO: Add your control notification handler code here
      m_ToolDlg.ShowWindow(TRUE);
}

void CStickerDlg::OnStep2()
{
      // TODO: Add your control notification handler code here
      m_Step2Button.EnableWindow(FALSE);
      m_Step3Button.EnableWindow(TRUE);
      m_SelectFrameDlg.ShowWindow(TRUE);
}

void CStickerDlg::OnStep3()
{
      // TODO: Add your control notification handler code here
      CStickerApp *pApp;
      pApp = (CStickerApp*) AfxGetApp();
      m_Step3Button.EnableWindow(FALSE);
      m_Step4Button.EnableWindow(TRUE);
      //m_CaptureDlg.Frame.SetFrame(pApp->GetFrameAt(0));
      //m_CaptureDlg.Frame.SetColorRef(pApp->GetColorRefAt(0));
      //m_CaptureDlg.Frame.DrawFrame(::GetWindowDC(m_CaptureDlg.m_hW
nd));
      m_CaptureDlg.ShowWindow(TRUE);
      //m_CaptureDlg.Frame.DrawFrame(::GetWindowDC(m_CaptureDlg.m_hW
nd));
}
```

```
/*************************************************************************
 *
 * CopyScreenToBitmap()
 *
 * Parameter:
 *
 * LPRECT lpRect     - specifies the window
 *
 * Return Value:
 *
 * HDIB              - identifies the device-dependent bitmap
 *
 * Description:
 *
 * This function copies the specified part of the screen to a
device-
 * dependent bitmap.
 *
 *
 ************************************************************************
****/

HBITMAP CopyScreenToBitmap(LPRECT lpRect, HDC hScr)
{
    HDC         hScrDC, hMemDC;         // screen DC and memory DC
    HBITMAP     hBitmap, hOldBitmap;    // handles to deice-
dependent bitmaps
    int         nX, nY, nX2, nY2;       // coordinates of rectangle
to grab
    int         nWidth, nHeight;        // DIB width and height
    int         xScrn, yScrn;           // screen resolution

    // check for an empty rectangle

    if (IsRectEmpty(lpRect))
      return NULL;

    // create a DC for the screen and create
    // a memory DC compatible to screen DC

    hScrDC = hScr;//CreateDC("FlyVideo'98 Capture Driver", NULL,
NULL, NULL);//("DISPLAY", NULL, NULL, NULL);
    hMemDC = CreateCompatibleDC(hScrDC);

    // get points of rectangle to grab

    nX = lpRect->left;
    nY = lpRect->top;
    nX2 = lpRect->right;
    nY2 = lpRect->bottom;

    // get screen resolution

    xScrn = GetDeviceCaps(hScrDC, HORZRES);
    yScrn = GetDeviceCaps(hScrDC, VERTRES);

    //make sure bitmap rectangle is visible
```

```
    if (nX < 0)
        nX = 0;
    if (nY < 0)
        nY = 0;
    if (nX2 > xScrn)
        nX2 = xScrn;
    if (nY2 > yScrn)
        nY2 = yScrn;

    nWidth = nX2 - nX;
    nHeight = nY2 - nY;

    // create a bitmap compatible with the screen DC
    hBitmap = CreateCompatibleBitmap(hScrDC, nWidth, nHeight);

    // select new bitmap into memory DC
    hOldBitmap = (HBITMAP)SelectObject(hMemDC, hBitmap);

    // bitblt screen DC to memory DC
    BitBlt(hMemDC, 0, 0, nWidth, nHeight, hScrDC, nX, nY, SRCCOPY);

    // select old bitmap back into memory DC and get handle to
    // bitmap of the screen

    hBitmap = (HBITMAP)SelectObject(hMemDC, hOldBitmap);

    // clean up

    DeleteDC(hScrDC);
    DeleteDC(hMemDC);

    // return handle to the bitmap

    return hBitmap;
}

void CStickerDlg::OnStep4()
{
      // TODO: Add your control notification handler code here
      m_Step4Button.EnableWindow(FALSE);
      m_Step1Button.EnableWindow(TRUE);
      m_PrintPreviewDlg.ShowWindow(TRUE);
}

// DrawDIBSection - Draws a DIB section onto a device
// hDC            - Handle to a device context
// hBitmap  - Handle of the DIB Section
// xDest    - x-coordinate of the upper-left corner of the
destination rect
// yDest    - y-coordinate of the upper-left corner of the
destination rect
void DrawDIBSection( HDC hDC, HBITMAP hBitmap, int xDest, int yDest
)
{
      HPALETTE hPal;

      HDC hDCMem = ::CreateCompatibleDC( hDC );
```

```
      // Create a logical palette for the bitmap
      DIBSECTION ds;
      BITMAPINFOHEADER &bmInfo = ds.dsBmih;
      if( ::GetObject(hBitmap, sizeof(ds), &ds ) == 0 )
            return;                    // Not a DIB Section

      HGDIOBJ hBmpOld =  ::SelectObject(hDCMem, hBitmap);

      int nColors = bmInfo.biClrUsed ? bmInfo.biClrUsed : 1 <<
ds.dsBm.bmBitsPixel;

      if( ::GetDeviceCaps(hDC, RASTERCAPS) & RC_PALETTE )
      {
            // Create a halftone palette if colors > 256.
            if( nColors > 256 )
                  hPal = ::CreateHalftonePalette(hDC);
            else
            {
                  // Create the palette
                  RGBQUAD *pRGB = new RGBQUAD[nColors];

                  ::GetDIBColorTable( hDCMem, 0, nColors, pRGB );

                  UINT nSize = sizeof(LOGPALETTE) + (sizeof
(PALETTEENTRY) * nColors);
                  LOGPALETTE *pLP = (LOGPALETTE *) new BYTE[nSize];

                  pLP->palVersion = 0x300;
                  pLP->palNumEntries = nColors;

                  for( int i=0; i < nColors; i++)
                  {
                        pLP->palPalEntry[i].peRed = pRGB[i].rgbRed;
                        pLP->palPalEntry[i].peGreen = pRGB
[i].rgbGreen;
                        pLP->palPalEntry[i].peBlue = pRGB
[i].rgbBlue;
                        pLP->palPalEntry[i].peFlags = 0;
                  }

                  hPal = ::CreatePalette( pLP );

                  delete[] pLP;
                  delete[] pRGB;
            }

            HPALETTE hPalOld = ::SelectPalette(hDC,hPal,FALSE);
            ::RealizePalette(hDC);

      BitBlt(hDC,xDest,yDest,bmInfo.biWidth,bmInfo.biHeight,hDCMem,0
,0,SRCCOPY);

            ::SelectPalette(hDC,hPalOld,FALSE);
            // delete GDI objects
            ::DeleteObject(hPal);
      }
      else
```

```
      BitBlt(hDC,xDest,yDest,bmInfo.biWidth,bmInfo.biHeight,hDCMem,0
,0,SRCCOPY);

      ::SelectObject(hDCMem, hBmpOld);
      ::DeleteDC(hDCMem);
}

BOOL SetBitmapPixels(CDC*     pDC,
                     HBITMAP  hBmp,
                     COLORREF rgbTransparent,
                              int      nXOffset,
                              int      nYOffset)
{
      //Sanity..
      if (hBmp    == NULL || pDC     == NULL)
            return FALSE;

      // Create a memory DC inside which we will scan the bitmap
conten
      HDC hMemDC = CreateCompatibleDC(NULL);
      ASSERT(hMemDC);

      if (hMemDC)
      {
            // Get bitmap siz
            BITMAP bm;
            GetObject(hBmp, sizeof(bm), &bm);

            //Make sure x and Y offset are relativ
            //to center of the bitmap; thi
            //way the bitmap will appear centere
            //on (nXOffset,nYOffset
            nXOffset -= bm.bmWidth /2;
            nYOffset -= bm.bmHeight/2;

            // Create a 32 bits depth bitmap and select it into the
memory DC
            BITMAPINFOHEADER RGB32BITSBITMAPINFO =
            {
                  sizeof(BITMAPINFOHEADER),// biSize
                  bm.bmWidth,                    // biWidth;
           -bm.bmHeight,                  // biHeight
                  1,                                    // biPlanes;
                  32,                                   // biBitCount
                  BI_RGB,                               //
biCompression;
                  0,                                    //
biSizeImage;
                  0,                                    //
biXPelsPerMeter;
                  0,                                    //
biYPelsPerMeter;
                  0,                                    // biClrUsed;
                  0                                     //
biClrImportant;
            };

            VOID * pbits32;
```

```
            HBITMAP hbm32 = CreateDIBSection(
                              hMemDC,
                              (BITMAPINFO *)&RGB32BITSBITMAPINFO,
                                            DIB_RGB_COLORS,
                                            &pbits32,
                                            NULL,
                                            0);

            if (hbm32)
            {
                  HBITMAP holdBmp = (HBITMAP)SelectObject(hMemDC,
hbm32);

                  // Create a DC just to copy the bitmap into the
memory D
                  HDC hDC = CreateCompatibleDC(hMemDC);

                  if (hDC)
                  {
                        // Get how many bytes per row w
                        //have for the bitmap bit
                        //(rounded up to 32 bits
                        BITMAP bm32;
                        VERIFY(GetObject(hbm32, sizeof(bm32),
&bm32));

                        while (bm32.bmWidthBytes % 4)
                              bm32.bmWidthBytes++;

                        // Copy the bitmap into the memory D
                        HBITMAP holdBmp = (HBITMAP)SelectObject(hDC,
hBmp);
                        VERIFY(BitBlt(hMemDC,
                                      0,
                                                  0,
                                                  bm.bmWidth,
                                                  bm.bmHeight,
                                                  hDC,
                                                  0,
                                                  0,
                                                  SRCCOPY));

                        //Scan each bitmap row from bottom to to
                        //(the bitmap is inverted vertically
                        BYTE *p32 = (BYTE *)bm32.bmBits +
                                    (bm32.bmHeight - 1) *
bm32.bmWidthBytes;

                        //Scan each bitmap pixel from left to right
                        //Search for a continuous lin
                        //of non transparent pixel
                        for (int y = 0; y < bm.bmHeight; y++)
                        {
                              for (int x = 0; x < bm.bmWidth; x++)
                              {
                                    int  x0 = x; //Ptr to start of
region to fil

                                    LONG *p = (LONG *)p32 + x;
```

```
                                    COLORREF
                                    rgbBase =RGB(GetBValue((DWORD)
(*p)),
                                                 GetGValue
((DWORD)(*p)),
GetRValue((DWORD)(*p)));

                                    COLORREF rgbCurr = rgbBase;

                                    if  rgbBase != rgbTransparent)
                                    {
                                         while(rgbCurr == rgbBase
&&
                                               x        <
bm.bmWidth)
                                         {
                                               p++;
                                               x++;
                                               rgbCurr = RGB
(GetBValue((DWORD)(*p)),
GetGValue((DWORD)(*p)),
        GetRValue((DWORD)(*p)));
                                         }

                                         CRgn rgnTemp;

                                         rgnTemp.CreateRectRgn(
                                                    x0 +
nXOffset,
                                                    y  +
nYOffset,
                                                    x  +
nXOffset,
                                                    y+1+
nYOffset);

                                         CBrush  brTemp(rgbBase);
                                         pDC->FillRgn
(&rgnTemp,&brTemp);

                                         x--; //Decrement to
account for x for loo
                                    }    //end if not transparent
                               }    //end for

                               // Go to next ro
                               //(remember, the bitmap is inverted
vertically
                               p32 -= bm32.bmWidthBytes;
                          }    //end for

                          // Clean u
                          SelectObject(hDC, holdBmp);
                          DeleteDC(hDC);
                     }
```

```
                DeleteObject(SelectObject(hMemDC, holdBmp));
            }

            DeleteDC(hMemDC);
        }

        return TRUE;
}
//////////////////////////////

// GetInvertedBitmap    - Creates a new bitmap with the inverted
image
// Returns          - Handle to a new bitmap with inverted image
// hBitmap          - Bitmap to invert
// bLateral         - Flag to indicate whether to invert laterally or
vertically
HBITMAP GetInvertedBitmap( HBITMAP hBitmap, BOOL bLateral )
{
        // Create a memory DC compatible with the display
        CDC sourceDC, destDC;
        sourceDC.CreateCompatibleDC( NULL );
        destDC.CreateCompatibleDC( NULL );

        // Get logical coordinates
        BITMAP bm;
        ::GetObject( hBitmap, sizeof( bm ), &bm );

        // Create a bitmap to hold the result
        HBITMAP hbmResult = ::CreateCompatibleBitmap(CClientDC(NULL),
                                bm.bmWidth, bm.bmHeight);

        // Select bitmaps into the DCs
        HBITMAP hbmOldSource = (HBITMAP)::SelectObject(
sourceDC.m_hDC, hBitmap );
        HBITMAP hbmOldDest = (HBITMAP)::SelectObject( destDC.m_hDC,
hbmResult );

        if( bLateral )
                destDC.StretchBlt( 0, 0, bm.bmWidth, bm.bmHeight,
&sourceDC,
                                bm.bmWidth-1, 0, -bm.bmWidth, bm.bmHeight,
SRCCOPY );
        else
                destDC.StretchBlt( 0, 0, bm.bmWidth, bm.bmHeight,
&sourceDC,
                                0, bm.bmHeight-1, bm.bmWidth, -bm.bmHeight,
SRCCOPY );

        // Reselect the old bitmaps
        ::SelectObject( sourceDC.m_hDC, hbmOldSource );
        ::SelectObject( destDC.m_hDC, hbmOldDest );

        return hbmResult;
}

// WriteDIB         - Writes a DIB to file
// Returns          - TRUE on success
// szFile           - Name of file to write to
```

```
// hDIB                 - Handle of the DIB
BOOL WriteDIB( LPTSTR szFile, HANDLE hDIB)
{
      BITMAPFILEHEADER  hdr;
      LPBITMAPINFOHEADER      lpbi;

      if (!hDIB)
            return FALSE;

      CFile file;
      if( !file.Open( szFile, CFile::modeWrite|CFile::modeCreate) )
            return FALSE;

      lpbi = (LPBITMAPINFOHEADER)hDIB;

      int nColors = 1 << lpbi->biBitCount;

      // Fill in the fields of the file header
      hdr.bfType        = ((WORD) ('M' << 8) | 'B');  // is always
"BM"
      hdr.bfSize        = GlobalSize (hDIB) + sizeof( hdr );
      hdr.bfReserved1   = 0;
      hdr.bfReserved2   = 0;
      hdr.bfOffBits             = (DWORD) (sizeof( hdr ) + lpbi->
biSize +
                                    nColors * sizeof(RGBQUAD));

      // Write the file header
      file.Write( &hdr, sizeof(hdr) );

      // Write the DIB header and the bits
      file.Write( lpbi, GlobalSize(hDIB) );

      return TRUE;
}

// DDBToDIB       - Creates a DIB from a DDB
// bitmap         - Device dependent bitmap
// dwCompression  - Type of compression - see BITMAPINFOHEADER
// pPal                 - Logical palette

HANDLE DDBToDIB( CBitmap& bitmap, DWORD dwCompression, CPalette*
pPal )
{
      BITMAP                  bm;
      BITMAPINFOHEADER  bi;
      LPBITMAPINFOHEADER      lpbi;
      DWORD             dwLen;
      HANDLE                  hDIB;
      HANDLE                  handle;
      HDC               hDC;
      HPALETTE          hPal;

      ASSERT( bitmap.GetSafeHandle() );

      // The function has no arg for bitfields

      if( dwCompression == BI_BITFIELDS )
```

```
            return NULL;

      // If a palette has not been supplied use defaul palette

      hPal = (HPALETTE) pPal->GetSafeHandle();
      if (hPal==NULL)
            hPal = (HPALETTE) GetStockObject(DEFAULT_PALETTE);

      // Get bitmap information

      bitmap.GetObject(sizeof(bm),(LPSTR)&bm);

      // Initialize the bitmapinfoheader

      bi.biSize            = sizeof(BITMAPINFOHEADER);
      bi.biWidth           = bm.bmWidth;
      bi.biHeight                = bm.bmHeight;
      bi.biPlanes                = 1;
      bi.biBitCount              = bm.bmPlanes * bm.bmBitsPixel;
      bi.biCompression   = dwCompression;
      bi.biSizeImage             = 0;
      bi.biXPelsPerMeter         = 0;
      bi.biYPelsPerMeter         = 0;
      bi.biClrUsed               = 0;
      bi.biClrImportant = 0;

      // Compute the size of the  infoheader and the color table

      int nColors = (1 << bi.biBitCount);
      if( nColors > 256 )
            nColors = 0;
      dwLen  = bi.biSize + nColors * sizeof(RGBQUAD);

      // We need a device context to get the DIB from

      hDC = GetDC(NULL);
      hPal = SelectPalette(hDC,hPal,FALSE);
      RealizePalette(hDC);

      // Allocate enough memory to hold bitmapinfoheader and color
table

      hDIB = GlobalAlloc(GMEM_FIXED,dwLen);

      if (!hDIB){
            SelectPalette(hDC,hPal,FALSE);
            ReleaseDC(NULL,hDC);
            return NULL;
      }

      lpbi = (LPBITMAPINFOHEADER)hDIB;
      *lpbi = bi;

      // Call GetDIBits with a NULL lpBits param, so the device
driver
      // will calculate the biSizeImage field
```

```
      GetDIBits(hDC, (HBITMAP)bitmap.GetSafeHandle(), 0L, (DWORD)
bi.biHeight,
                  (LPBYTE)NULL, (LPBITMAPINFO)lpbi, (DWORD)
DIB_RGB_COLORS);

      bi = *lpbi;

      // If the driver did not fill in the biSizeImage field, then
compute it
      // Each scan line of the image is aligned on a DWORD (32bit)
boundary
      if (bi.biSizeImage == 0){
            bi.biSizeImage = ((((bi.biWidth * bi.biBitCount) + 31) &
~31) / 8)
                                    * bi.biHeight;

            // If a compression scheme is used the result may infact
be larger
            // Increase the size to account for this.
            if (dwCompression != BI_RGB)
                  bi.biSizeImage = (bi.biSizeImage * 3) / 2;
      }

      // Realloc the buffer so that it can hold all the bits
      dwLen += bi.biSizeImage;
      if (handle = GlobalReAlloc(hDIB, dwLen, GMEM_MOVEABLE))
            hDIB = handle;
      else{
            GlobalFree(hDIB);

            // Reselect the original palette

            SelectPalette(hDC,hPal,FALSE);
            ReleaseDC(NULL,hDC);
            return NULL;
      }

      // Get the bitmap bits

      lpbi = (LPBITMAPINFOHEADER)hDIB;

      // FINALLY get the DIB

      BOOL bGotBits = GetDIBits( hDC, (HBITMAP)bitmap.GetSafeHandle
(),
                        0L,                     // Start scan line
                        (DWORD)bi.biHeight,           // # of scan
lines
                        (LPBYTE)lpbi                  // address for
bitmap bits
                        + (bi.biSize + nColors * sizeof(RGBQUAD)),
                        (LPBITMAPINFO)lpbi,           // address of
bitmapinfo
                        (DWORD)DIB_RGB_COLORS);       // Use RGB for
color table

      if( !bGotBits )
      {
```

```
            GlobalFree(hDIB);
            SelectPalette(hDC,hPal,FALSE);
            ReleaseDC(NULL,hDC);
            return NULL;
      }

      SelectPalette(hDC,hPal,FALSE);
      ReleaseDC(NULL,hDC);
      return hDIB;
}

BOOL WriteWindowToDIB( LPTSTR szFile, CWnd *pWnd )
{
      CBitmap     bitmap;
      CWindowDC   dc(pWnd);
      CDC         memDC;
      CRect       rect;

      memDC.CreateCompatibleDC(&dc);

      pWnd->GetWindowRect(rect);

      bitmap.CreateCompatibleBitmap(&dc, rect.Width(),rect.Height()
);

      CBitmap* pOldBitmap = memDC.SelectObject(&bitmap);

      memDC.BitBlt(0, 0, rect.Width(),rect.Height(), &dc, 0, 0,
SRCCOPY);

      // Create logical palette if device support a palette

      CPalette pal;

      if( dc.GetDeviceCaps(RASTERCAPS) & RC_PALETTE )
      {
            UINT nSize = sizeof(LOGPALETTE) + (sizeof(PALETTEENTRY)
* 256);
            LOGPALETTE *pLP = (LOGPALETTE *) new BYTE[nSize];
            pLP->palVersion = 0x300;

            pLP->palNumEntries =
                  GetSystemPaletteEntries( dc, 0, 255, pLP->
palPalEntry );

            // Create the palette

            pal.CreatePalette( pLP );

            delete[] pLP;
      }

      memDC.SelectObject(pOldBitmap);

      // Convert the bitmap to a DIB

      HANDLE hDIB = DDBToDIB( bitmap, BI_RGB, &pal );
```

```
      if( hDIB == NULL )
            return FALSE;

      // Write it to file

      WriteDIB( szFile, hDIB );

      // Free the memory allocated by DDBToDIB for the DIB

      GlobalFree( hDIB );
      return TRUE;
}
```

MyDialog.h

```
#if
!defined(AFX_MYDIALOG_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLU
DED_)
#define
AFX_MYDIALOG_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// bmdlg.h : header file
//
#include "DirectSound.h"
#include "Midi.h"
/////////////////////////////////////////////////////////////////////////
// CBitmapDialog dialog

#define BITMAP_TILE     0
#define BITMAP_CENTER   1

class CMyDialog : public CDialog
{
// Construction
public:
      CMyDialog (UINT TemplateID, CWnd* pParent = NULL);
      CMyDialog (LPCSTR TemplateName, CWnd* pParent = NULL);

      void SetBitmap (HBITMAP hBitmap, int Type = BITMAP_TILE);
      void SetBitmap (UINT ResID, int Type = BITMAP_TILE);
      void SetBitmap (LPCSTR ResName, int Type = BITMAP_TILE);
      void SetSound (UINT uResourceID = NULL);
      void SetMidi (UINT uResourceID = NULL);
      void SetCursor(HCURSOR hCursor=NULL);

// Attributes
private:
      CBitmap                 mBitmap;
      CBrush                  mHollowBrush;
      int                     mType;
      UINT              m_SoundName;
      CDirectSound      m_Sound;
      UINT              m_MidiName;
      CMIDI             m_Midi;
      HCURSOR                 m_hCursor;
```

```
// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(CMyDialog)
	afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT
nCtlColor);
	afx_msg BOOL OnEraseBkgnd(CDC* pDC);
	afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
	afx_msg void OnMouseMove(UINT nFlags, CPoint point);
	afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_MYDIALOG_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLU
DED_)
```

MyDialog.cpp

```
// bmdlg.cpp : implementation file
//

#include "stdafx.h"

#include "MyDialog.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CBitmapDialog dialog


CMyDialog::CMyDialog(UINT TemplateID, CWnd* pParent /*=NULL*/)
	: CDialog(TemplateID, pParent)
{
	mHollowBrush.CreateStockObject (HOLLOW_BRUSH);
}

CMyDialog::CMyDialog(LPCSTR TemplateName, CWnd* pParent /*=NULL*/)
	: CDialog(TemplateName, pParent)
{
	mHollowBrush.CreateStockObject (HOLLOW_BRUSH);
}

BEGIN_MESSAGE_MAP(CMyDialog, CDialog)
	//{{AFX_MSG_MAP(CMyDialog)
	ON_WM_CTLCOLOR()
```

```
      ON_WM_ERASEBKGND()
      ON_WM_SHOWWINDOW()
      ON_WM_MOUSEMOVE()
      ON_WM_SETCURSOR()
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CMyDialog::SetBitmap (HBITMAP hBitmap, int Type)
{
      mBitmap.Attach(hBitmap);
      mType = Type;
}

void CMyDialog::SetBitmap (UINT ResID, int Type)
{
      mBitmap.LoadBitmap (ResID);
      mType = Type;
}

void CMyDialog::SetBitmap (LPCSTR ResName, int Type)
{
      mBitmap.LoadBitmap (ResName);
      mType = Type;
}

void CMyDialog::SetSound (UINT uResource /*= NULL*/)
{
      m_SoundName = uResource;
}

void CMyDialog::SetCursor(HCURSOR hCursor /*=NULL*/)
{
      if (hCursor!=NULL)
            m_hCursor = hCursor;
      else
            m_hCursor = NULL;//AfxGetApp()->LoadCursor(IDC_POINTER);
}

void CMyDialog::SetMidi (UINT uResource /*= NULL*/)
{
      m_MidiName = uResource;
}

/////////////////////////////////////////////////////////////////////////////
// CMyDialog message handlers

HBRUSH CMyDialog::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
      // Return a hollow brush for a static text control. All other
controls
      // return the default brush so they are displayed correctly.
      // Feel free to experiment with returning different brushes
for
      // different controls.
      if (mBitmap.GetSafeHandle () != NULL){
            switch (nCtlColor){
                  case CTLCOLOR_STATIC:
                        pDC->SetBkMode (TRANSPARENT);
```

```
                        return (HBRUSH) mHollowBrush.m_hObject;
                        break;
                  case CTLCOLOR_BTN:
                  case CTLCOLOR_DLG:
                  case CTLCOLOR_EDIT:
                  case CTLCOLOR_LISTBOX:
                  case CTLCOLOR_MSGBOX:
                  case CTLCOLOR_SCROLLBAR:
                  default:
                        return CDialog::OnCtlColor (pDC, pWnd,
nCtlColor);
                        break;
                  }
            }
      else
            return CDialog::OnCtlColor (pDC, pWnd, nCtlColor);
}

BOOL CMyDialog::OnEraseBkgnd(CDC* pDC)
{
      // Only execute this code if a bitmap was loaded. Otherwise we
      // will just call the base class function.
      if (mBitmap.GetSafeHandle () != NULL){
            CDC MemDC;
            BITMAP bm;
            CRect Rect;
            int x = 0, y = 0;

            // Get the dimension of the dialogs client area and the
bitmap.
            GetClientRect (Rect);
            mBitmap.GetObject (sizeof (BITMAP), &bm);
            MemDC.CreateCompatibleDC (pDC);
            CBitmap *pOldBitmap = MemDC.SelectObject (&mBitmap);
            // Display the bitmap in the center of the dialog.
            if (mType == BITMAP_CENTER){
                  CDialog::OnEraseBkgnd(pDC);
                  x = (Rect.Width () - bm.bmWidth) / 2;
                  y = (Rect.Height () - bm.bmHeight) / 2;
                  pDC->BitBlt (x, y, bm.bmWidth, bm.bmHeight,
&MemDC, 0, 0, SRCCOPY);
                  }
            // Tile the bitmap within the dialogs client area.
            else{
                  while (y < Rect.Height ()){
                        while (x < Rect.Width ()){
                              pDC->BitBlt (x, y, bm.bmWidth,
bm.bmHeight, &MemDC, 0, 0, SRCCOPY);
                              x += bm.bmWidth;
                              }
                        x = 0;
                        y += bm.bmHeight;
                        }
                  }
            // Clean up.
            MemDC.SelectObject (pOldBitmap);
            return TRUE;
            }
```

```
	else
		return CDialog::OnEraseBkgnd(pDC);
}

void CMyDialog::OnShowWindow(BOOL bShow, UINT nStatus)
{
	CDialog::OnShowWindow(bShow, nStatus);

	// TODO: Add your message handler code here
	if (bShow)
	{
		m_Midi.Create(m_MidiName);
		m_Midi.Play();
		m_Sound.Create(m_SoundName);
		m_Sound.Play();
	}
	else
	{
		m_Midi.Stop();
		m_Sound.Stop();
	}
}

void CMyDialog::OnMouseMove(UINT nFlags, CPoint point)
{
	// TODO: Add your message handler code here and/or call
default
	if (GetCursor()!=m_hCursor)
		::SetCursor(m_hCursor);
	CDialog::OnMouseMove(nFlags, point);
}

BOOL CMyDialog::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
	// TODO: Add your message handler code here and/or call
default
	if ((message==WM_MOUSEMOVE)||
		(message==WM_LBUTTONDOWN)||
		(message==WM_LBUTTONUP)||
		(message==WM_MBUTTONDOWN)||
		(message==WM_MBUTTONUP)||
		(message==WM_RBUTTONDOWN)||
		(message==WM_RBUTTONUP))
	{
		if (GetCursor()!=m_hCursor)
			SetCursor(m_hCursor);
		return TRUE;
	}
	return CDialog::OnSetCursor(pWnd, nHitTest, message);
}
```

MyButton.h

```
#if
!defined(AFX_MYBUTTON_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLU
DED_)
#define
AFX_MYBUTTON_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLUDED_
```

```
#include "DirectSound.h"

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// MyButton.h : header file
//

/////////////////////////////////////////////////////////////////////////
// CMyButton window

class CMyButton : public CBitmapButton
{
// Construction
public:
      CMyButton();

// Attributes
public:

// Operations
public:

// Overrides
      LoadBitmaps(HBITMAP hBitmap=NULL, HBITMAP hBitmapSel=NULL,
                  HBITMAP hBitmapFocus=NULL, HBITMAP
hBitmapDisabled=NULL);
      SetCursor(HCURSOR hCursor);
      SetSound(UINT uResourceID = NULL);
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CMyButton)
      //}}AFX_VIRTUAL

// Implementation
public:
      virtual ~CMyButton();

      // Generated message map functions
protected:
      HCURSOR m_hCursor;
      UINT  m_SoundName;
      CDirectSound m_Sound;
      //{{AFX_MSG(CMyButton)
      afx_msg void OnMouseMove(UINT nFlags, CPoint point);
      afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT
message);
      afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
      //}}AFX_MSG

      DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
```

```
#endif //
!defined(AFX_MYBUTTON_H__316CDACF_29A0_11D3_9224_80E7CA730C3F__INCLU
DED_)
```

MyButton.cpp

```
// MyButton.cpp : implementation file
//

#include "stdafx.h"
#include "Sticker.h"
#include "MyButton.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
// CMyButton

CMyButton::CMyButton()
{
      m_SoundName = IDR_DEFAULT_SOUND;
}

CMyButton::~CMyButton()
{
}

CMyButton::LoadBitmaps(HBITMAP hBitmap, HBITMAP hBitmapSel,
                              HBITMAP hBitmapFocus, HBITMAP
hBitmapDisabled)
{
      if (hBitmap)
            m_bitmap.Attach(hBitmap);
      if (hBitmapSel)
            m_bitmapSel.Attach(hBitmapSel);
      if (hBitmapFocus)
            m_bitmapFocus.Attach(hBitmapFocus);
      if (hBitmapDisabled)
            m_bitmapDisabled.Attach(hBitmapDisabled);
      return 0;
}

CMyButton::SetCursor(HCURSOR hCursor=NULL)
{
      if (hCursor!=NULL)
            m_hCursor = hCursor;
      else
            m_hCursor = NULL;//AfxGetApp()->LoadCursor(IDC_POINTER);
}

CMyButton::SetSound(UINT uResourceID)
{
      m_SoundName = uResourceID;
}
```

```
BEGIN_MESSAGE_MAP(CMyButton, CBitmapButton)
	//{{AFX_MSG_MAP(CMyButton)
	ON_WM_MOUSEMOVE()
	ON_WM_SETCURSOR()
	ON_WM_LBUTTONUP()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMyButton message handlers


void CMyButton::OnMouseMove(UINT nFlags, CPoint point)
{
	// TODO: Add your message handler code here and/or call
default

	if (!(m_hWnd==::GetFocus()))
	{
		::SetFocus(m_hWnd);
	} else
	{
		if (GetCursor()!=m_hCursor)
			::SetCursor(m_hCursor);
	}
	CBitmapButton::OnMouseMove(nFlags, point);
}

BOOL CMyButton::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
	// TODO: Add your message handler code here and/or call
default
	if ((message==WM_MOUSEMOVE)||
		(message==WM_LBUTTONDOWN)||
		(message==WM_LBUTTONUP)||
		(message==WM_MBUTTONDOWN)||
		(message==WM_MBUTTONUP)||
		(message==WM_RBUTTONDOWN)||
		(message==WM_RBUTTONUP))
	{
		if (GetCursor()!=m_hCursor)
			SetCursor(m_hCursor);
		return TRUE;
	}
	return CBitmapButton::OnSetCursor(pWnd, nHitTest, message);
}

void CMyButton::OnLButtonUp(UINT nFlags, CPoint point)
{
	// TODO: Add your message handler code here and/or call
default
	RECT r;
	CRect cr;

	::GetWindowRect(m_hWnd, &r);
	cr.SetRect(0,0,r.right-r.left,r.bottom-r.top);
	if (cr.PtInRect(point))
	{
```

```
            m_Sound.Create(m_SoundName);
            m_Sound.Play(0);
      }
      CBitmapButton::OnLButtonUp(nFlags, point);
}
```

ToolDlg.h

```
#if
!defined(AFX_TOOLDLG_H__A79612C4_D260_11D3_8AFC_80624AC10801__INCLUD
ED_)
#define
AFX_TOOLDLG_H__A79612C4_D260_11D3_8AFC_80624AC10801__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ToolDlg.h : header file
//////////////////////////////////////////////////////////////////////
// CToolDlg dialog

class CToolDlg : public CDialog
{
// Construction
public:
      CToolDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
      //{{AFX_DATA(CToolDlg)
      enum { IDD = IDD_TOOL_DIALOG };
      CButton     m_Done;
      //}}AFX_DATA
// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CToolDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
      //}}AFX_VIRTUAL

// Implementation
protected:
      // Generated message map functions
      //{{AFX_MSG(CToolDlg)
      afx_msg void OnToolDoneButton();
      virtual BOOL OnInitDialog();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_TOOLDLG_H__A79612C4_D260_11D3_8AFC_80624AC10801__INCLUD
ED_)
```

ToolDlg.cpp

```
// ToolDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Sticker.h"
#include "ToolDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CToolDlg dialog


CToolDlg::CToolDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CToolDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CToolDlg)
	//}}AFX_DATA_INIT
}


void CToolDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CToolDlg)
	DDX_Control(pDX, IDC_TOOL_DONE_BUTTON, m_Done);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CToolDlg, CDialog)
	//{{AFX_MSG_MAP(CToolDlg)
	ON_BN_CLICKED(IDC_TOOL_DONE_BUTTON, OnToolDoneButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CToolDlg message handlers

void CToolDlg::OnToolDoneButton()
{
	// TODO: Add your control notification handler code here
	CButton *StartWindow,*StartSticker;
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	CString
SubKey("Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce");
	CString Data;
	Data.Format(pApp->GetAppPath()+"\\Sticker.EXE");

	HKEY hKey;
	RegOpenKey( HKEY_LOCAL_MACHINE, LPCTSTR(SubKey), &hKey);
```

```
	pApp->m_pszProfileName=_tcsdup(_T("MySticker.INI"));
	StartWindow = (CButton*) GetDlgItem(IDC_START_WINDOW);
	StartSticker = (CButton*) GetDlgItem(IDC_START_STICKER);
	if (StartWindow->GetCheck() == 1)
	{
		pApp->WriteProfileInt("Environtment","StartupType",0);
		RegDeleteValue( hKey, "TheSticker");
	}
	if (StartSticker->GetCheck() == 1)
	{
		pApp->WriteProfileInt("Environtment","StartupType",1);
		RegSetValueEx( hKey, "TheSticker",0, REG_SZ,
		(const unsigned char *)LPCSTR(Data),(DWORD)
(Data.GetLength()+1)*sizeof(TCHAR));
	}
	RegCloseKey(hKey);
	ShowWindow(FALSE);
}

BOOL CToolDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// TODO: Add extra initialization here
	CStickerApp *pApp;
	CButton *StartWindow,*StartSticker;
	StartWindow = (CButton*) GetDlgItem(IDC_START_WINDOW);
	StartSticker = (CButton*) GetDlgItem(IDC_START_STICKER);
	pApp = (CStickerApp*) AfxGetApp();
	pApp->m_pszProfileName=_tcsdup(_T("MySticker.INI"));
	if (pApp->GetProfileInt("Environtment","StartupType",0)==0)
		StartWindow->SetCheck(1);
	else
		StartSticker->SetCheck(1);

	return TRUE;  // return TRUE unless you set the focus to a
control
		      // EXCEPTION: OCX Property Pages should return
FALSE
}
```

MIDI.h

```
////////////////////////////////////////////////////////////////////////
// Copyright (C) 1998 by J rg K nig
// All rights reserved
//
// This file is part of the completely free tetris clone "CGTetris".
//
// This is free software.
// You may redistribute it by any means providing it is not sold for
profit
// without the authors written consent.
//
// No warrantee of any kind, expressed or implied, is included with
this
// software; use at your own risk, responsibility for damages (if
any) to
```

```
// anyone resulting from the use of this software rests entirely
with the
// user.
//
// Send bug reports, bug fixes, enhancements, requests, flames,
etc., and
// I'll try to keep a version up to date.  I can be reached as
follows:
//    J.Koenig@adg.de                 (company site)
//    Joerg.Koenig@rhein-neckar.de    (private site)
//////////////////////////////////////////////////////////////////////
// Midi.h : main header file for the MIDI application
//
// This class is based on the DirectX sample "mstream".
#ifndef MIDI_h
#define MIDI_h
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#include <mmsystem.h>
#pragma message("linking with multimedia library")
#pragma comment(lib, "winmm.lib")
#include <vector>
using namespace std;
// This message is sent to the controlling window, if the volume
changes in
// another way than explicitly set by the owner of the CMIDI object.
// WPARAM   the pointer to the MIDI object
// LPARAM   lo-word: the number of the channel that changed volume
//                hi-word: the new volume in percent
#define     WM_MIDI_VOLUMECHANGED   WM_USER+23
#define MIDI_CTRLCHANGE             ((BYTE)0xB0)            // +
ctrlr + value
#define MIDI_PRGMCHANGE             ((BYTE)0xC0)            // + new
patch
#define MIDI_CHANPRESS              ((BYTE)0xD0)            // +
pressure (1 byte)
#define MIDICTRL_VOLUME             ((BYTE)0x07)
#define MIDIEVENT_CHANNEL(dw) (dw & 0x0000000F)
#define MIDIEVENT_TYPE(dw)          (dw & 0x000000F0)
#define MIDIEVENT_DATA1(dw)         ((dw & 0x0000FF00) >> 8)
#define MIDIEVENT_VOLUME(dw)  ((dw & 0x007F0000) >> 16)
#define MIDI_SYSEX                          ((BYTE)0xF0)            //
SysEx begin
#define MIDI_SYSEXEND               ((BYTE)0xF7)            // SysEx
end
#define MIDI_META                   ((BYTE)0xFF)            // Meta
event begin
#define MIDI_META_TEMPO             ((BYTE)0x51)            // Tempo
change
#define MIDI_META_EOT               ((BYTE)0x2F)            // End-
of-track
// flags for the ConvertToBuffer() method
#define CONVERTF_RESET              0x00000001
#define CONVERTF_STATUS_DONE        0x00000001
#define CONVERTF_STATUS_STUCK       0x00000002
#define CONVERTF_STATUS_GOTEVENT    0x00000004
```

```
// Return values from the ConvertToBuffer() method
#define CONVERTERR_NOERROR          0           // No error occured
#define CONVERTERR_CORRUPT          -101  // The input file is
corrupt
// The converter has already encountered a corrupt file and cannot
convert any
// more of this file -- must reset the converter
#define CONVERTERR_STUCK            -102
#define CONVERTERR_DONE             -103  // Converter is done
#define CONVERTERR_BUFFERFULL -104  // The buffer is full
#define CONVERTERR_METASKIP         -105  // Skipping unknown meta
event
#define STATUS_KILLCALLBACK         100         // Signals that the
callback should die
#define STATUS_CALLBACKDEAD         200         // Signals callback
is done processing
#define STATUS_WAITINGFOREND  300       // Callback's waiting for
buffers to play
// Description of a track
//
struct TRACK
{
      DWORD fdwTrack;           // Track's flags
      DWORD dwTrackLength;    // Total bytes in track
      LPBYTE      pTrackStart;      // -> start of track data buffer
      LPBYTE      pTrackCurrent;    // -> next byte to read in
buffer
      DWORD tkNextEventDue;   // Absolute time of next event in
track
      BYTE  byRunningStatus;// Running status from last channel msg

      TRACK()
            : fdwTrack(0)
            , dwTrackLength(0)
            , pTrackStart(0)
            , pTrackCurrent(0)
            , tkNextEventDue(0)
            , byRunningStatus(0)
      {
      }
};

#define ITS_F_ENDOFTRK        0x00000001


// This structure is used to pass information to the ConvertToBuffer
()
// system and then internally by that function to send information
about the
// target stream buffer and current state of the conversion process
to internal
// lower level conversion routines.
struct CONVERTINFO
{
    MIDIHDR mhBuffer;               // Standard Windows stream
buffer header
    DWORD   dwStartOffset;          // Start offset from
mhStreamBuffer.lpStart
```

```
    DWORD   dwMaxLength;            // Max length to convert on this
pass
    DWORD   dwBytesRecorded;
    DWORD   tkStart;
    BOOL    bTimesUp;

      CONVERTINFO()
            : dwStartOffset(0)
            , dwMaxLength(0)
            , dwBytesRecorded(0)
            , tkStart(0)
            , bTimesUp(FALSE)
      {
            memset(&mhBuffer, 0, sizeof(MIDIHDR));
      }
};

// Temporary event structure which stores event data until we're
ready to
// dump it into a stream buffer
struct TEMPEVENT
{
      DWORD tkEvent;            // Absolute time of event
      BYTE  byShortData[4];     // Event type and parameters if
channel msg
      DWORD dwEventLength;      // Length of data which follows if
meta or sysex
      LPBYTE      pLongData;          // -> Event data if applicable
};

class CMIDI
{
protected:
      typedef vector<TRACK>           TrackArray_t;
      typedef vector<DWORD>           VolumeArray_t;
      typedef vector<CONVERTINFO>     ConvertArray_t;

      enum {
            NUM_CHANNELS = 16,              // 16 volume channels
            VOLUME_INIT = 100,              // 100% volume by default
            NUM_STREAM_BUFFERS = 2,
            OUT_BUFFER_SIZE = 1024, // Max stream buffer size in
bytes
            DEBUG_CALLBACK_TIMEOUT = 2000,
            VOLUME_MIN = 0,
            VOLUME_MAX = 127        // == 100%
      };

public:
      CMIDI();
      virtual ~CMIDI();

      BOOL  Create(LPVOID pSoundData, DWORD dwSize, CWnd * pParent =
0);
      BOOL  Create(LPCTSTR pszResID, CWnd * pParent = 0);
      BOOL  Create(UINT uResID, CWnd * pParent = 0);

      BOOL  Play(BOOL bInfinite = FALSE);
```

```
      BOOL  Stop(BOOL bReOpen = TRUE);
      BOOL  IsPlaying() const { return m_bPlaying; }

      BOOL  Pause();
      BOOL  Continue();
      BOOL  IsPaused() const { return m_bPaused; }

            // Set playback position back to the start
      BOOL  Rewind();

            // Get the number of volume channels
      DWORD GetChannelCount() const;

            // Set the volume of a channel in percent. Channels are
from 0 to (GetChannelCount()-1)
      void  SetChannelVolume(DWORD channel, DWORD percent);

            // Get the volume of a channel in percent
      DWORD GetChannelVolume(DWORD channel) const;

            // Set the volume for all channels in percent
      void  SetVolume(DWORD percent);

            // Get the average volume for all channels
      DWORD GetVolume() const;

            // Set the tempo of the playback. Default: 100%
      void  SetTempo(DWORD percent);

            // Get the current tempo in percent (usually 100)
      DWORD GetTempo() const;

            // You can (un)set an infinite loop during playback.
            // Note that "Play()" resets this setting!
      void  SetInfinitePlay(BOOL bSet = TRUE);

protected:  // implementation
            // This function converts MIDI data from the track
buffers.
      int         ConvertToBuffer(DWORD dwFlags, CONVERTINFO *
lpciInfo);

            // Fills in the event struct with the next event from
the track
      BOOL  GetTrackEvent(TRACK * ptsTrack, TEMPEVENT * pteTemp);

            // Retrieve the next byte from the track buffer,
refilling the buffer from
            // disk if necessary.
      BOOL  GetTrackByte(TRACK * ptsTrack, LPBYTE lpbyByte) {
                        if( DWORD(ptsTrack->pTrackCurrent -
ptsTrack->pTrackStart) == ptsTrack->dwTrackLength )
                              return FALSE;
                        *lpbyByte = *ptsTrack->pTrackCurrent++;
                        return TRUE;
                  }
```

```
            // Attempts to parse a variable length DWORD from the
given track.
      BOOL  GetTrackVDWord(TRACK * ptsTrack, LPDWORD lpdw);

            // Put the given event into the given stream buffer at
the given location.
      int         AddEventToStreamBuffer( TEMPEVENT * pteTemp,
CONVERTINFO * lpciInfo );

            // Opens a MIDI stream. Then it goes about converting
the data into a midiStream buffer for playback.
      BOOL  StreamBufferSetup();

      void  FreeBuffers();


protected:  // error handling
            // The default implementation writes the error message
in the
            // debuggers output window. Override if you want a
different
            // behavior.
      virtual void      MidiError(MMRESULT Result);

            // Failure in converting track into stream.
            // The default implementation displays the offset and
the total
            // number of bytes of the failed track and the error
message in
            // the debuggers output window.
      virtual void      TrackError(TRACK *, LPSTR ErrMsg);


protected:  // overridables
            // NOTE THAT, IF YOU OVERRIDE ONE OF THESE METHODS, YOU
MUST CALL
            // THE BASE CLASS IMPLEMENTATION TOO!

            // called when a MIDI output device is opened
      virtual void      OnMidiOutOpen();

            // called when the MIDI output device is closed
      virtual void      OnMidiOutClose();

            // called when the specified system-exclusive or stream
buffer
            // has been played and is being returned to the
application
      virtual void      OnMidiOutDone(MIDIHDR &);

            // called when a MEVT_F_CALLBACK event is reached in the
MIDI output stream
      virtual void      OnMidiOutPositionCB(MIDIHDR &, MIDIEVENT &);


private:    // callback procedure
            // This procedure calls the overridables above.
```

```
      static void CALLBACK    MidiProc(HMIDIOUT, UINT, DWORD, DWORD,
DWORD);


protected:  // data members
      DWORD             m_dwSoundSize;
      LPVOID            m_pSoundData;
      DWORD             m_dwFormat;
      DWORD             m_dwTrackCount;
      DWORD             m_dwTimeDivision;
      BOOL              m_bPlaying;
      HMIDISTRM         m_hStream;
      DWORD             m_dwProgressBytes;
      BOOL              m_bLooped;
      DWORD             m_tkCurrentTime;
      DWORD             m_dwBufferTickLength;
      DWORD             m_dwCurrentTempo;
      DWORD             m_dwTempoMultiplier;
      BOOL              m_bInsertTempo;
      BOOL              m_bBuffersPrepared;
      int               m_nCurrentBuffer;
      UINT              m_uMIDIDeviceID;
      int               m_nEmptyBuffers;
      BOOL              m_bPaused;
      UINT              m_uCallbackStatus;
      HANDLE            m_hBufferReturnEvent;
      CWnd *            m_pWndParent;
      TrackArray_t      m_Tracks;
      VolumeArray_t     m_Volumes;
      ConvertArray_t    m_StreamBuffers;

      // data members especially for ConvertToBuffer()
    TRACK *             m_ptsTrack;
      TRACK *           m_ptsFound;
    DWORD               m_dwStatus;
    DWORD               m_tkNext;
      DWORD             m_dwMallocBlocks;
    TEMPEVENT           m_teTemp;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations
immediately before the previous line.

#endif // MIDI_h
```

MIDI.cpp

```
////////////////////////////////////////////////////////////////////////
// Copyright (C) 1998 by J rg K nig
// All rights reserved
//
// This file is part of the completely free tetris clone "CGTetris".
//
// This is free software.
// You may redistribute it by any means providing it is not sold for
profit
// without the authors written consent.
```

```
// No warrantee of any kind, expressed or implied, is included with
this
// software; use at your own risk, responsibility for damages (if
any) to
// anyone resulting from the use of this software rests entirely
with the
// user.
//
// Send bug reports, bug fixes, enhancements, requests, flames,
etc., and
// I'll try to keep a version up to date.  I can be reached as
follows:
//    J.Koenig@adg.de                 (company site)
//    Joerg.Koenig@rhein-neckar.de    (private site)
//////////////////////////////////////////////////////////////////////
// The CMIDI class is based on a sample in the DirectX SDK (mstream)

#include "stdafx.h"
#include "Midi.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#define MThd            0x6468544D      // Start of file
#define MTrk            0x6B72544D      // Start of track
#define BUFFER_TIME_LENGTH          60  // Amount to fill in
milliseconds
// These structures are stored in MIDI files; they need to be byte
aligned.
//
#pragma pack(1)

// Contents of MThd chunk.
struct MIDIFILEHDR
{
    WORD    wFormat;                // Format (hi-lo)
    WORD    wTrackCount;            // # tracks (hi-lo)
    WORD    wTimeDivision;          // Time division (hi-lo)
};

#pragma pack() // End of need for byte-aligned structures


// Macros for swapping hi/lo-endian data
//
#define WORDSWAP(w)             (((w) >> 8) | \
                                 (((w) << 8) & 0xFF00))

#define DWORDSWAP(dw)   (((dw) >> 24) | \
                                 (((dw) >> 8) & 0x0000FF00) | \
                                 (((dw) << 8) & 0x00FF0000) | \
                                 (((dw) << 24) & 0xFF000000))

static char gteBadRunStat[]   = "Reference to missing running
status.";
static char gteRunStatMsgTrunc[]= "Running status message
truncated";
```

```
static char gteChanMsgTrunc[] = "Channel message truncated";
static char gteSysExLenTrunc[]      = "SysEx event truncated
(length)";
static char gteSysExTrunc[]   = "SysEx event truncated";
static char gteMetaNoClass[]  = "Meta event truncated (no class
byte)";
static char gteMetaLenTrunc[] = "Meta event truncated (length)";
static char gteMetaTrunc[]    = "Meta event truncated";
static char gteNoMem[]        = "Out of memory during malloc call";
CMIDI::CMIDI()
      : m_dwSoundSize(0)
      , m_pSoundData(0)
      , m_dwFormat(0)
      , m_dwTrackCount(0)
      , m_dwTimeDivision(0)
      , m_bPlaying(FALSE)
      , m_hStream(0)
      , m_dwProgressBytes(0)
      , m_bLooped(FALSE)
      , m_tkCurrentTime(0)
      , m_dwBufferTickLength(0)
      , m_dwCurrentTempo(0)
      , m_dwTempoMultiplier(100)
      , m_bInsertTempo(FALSE)
      , m_bBuffersPrepared(FALSE)
      , m_nCurrentBuffer(0)
      , m_uMIDIDeviceID(MIDI_MAPPER)
      , m_nEmptyBuffers(0)
      , m_bPaused(FALSE)
      , m_uCallbackStatus(0)
      , m_hBufferReturnEvent(0)
      , m_ptsTrack(0)
      , m_ptsFound(0)
      , m_dwStatus(0)
      , m_tkNext(0)
      , m_dwMallocBlocks(0)
{
      m_hBufferReturnEvent = ::CreateEvent(0, FALSE, FALSE, TEXT
("Wait For Buffer Return"));
      ASSERT(m_hBufferReturnEvent != 0);
}

CMIDI::~CMIDI()
{
      Stop(FALSE);

      if(m_hBufferReturnEvent)
            ::CloseHandle(m_hBufferReturnEvent);
}


BOOL CMIDI::Create(UINT uResID, CWnd * pWndParent /* = NULL */)
{
      return      Create(MAKEINTRESOURCE(uResID), pWndParent);
}


BOOL CMIDI::Create(LPCTSTR pszResID, CWnd * pWndParent /* = NULL */)
```

```
{
	////////////////////////////////////////////////////////////////
	// load resource
	HINSTANCE hApp = ::GetModuleHandle(0);
	ASSERT(hApp);

	HRSRC hResInfo = ::FindResource(hApp, pszResID, TEXT("MIDI"));
	if(hResInfo == 0)
		return FALSE;

	HGLOBAL hRes = ::LoadResource(hApp, hResInfo);
	if(hRes == 0)
		return FALSE;

	LPVOID pTheSound = ::LockResource(hRes);
	if(pTheSound == 0)
		return FALSE;
	DWORD dwTheSound = ::SizeofResource(hApp, hResInfo);
	return Create(pTheSound, dwTheSound, pWndParent);
}

BOOL CMIDI::Create(LPVOID pSoundData, DWORD dwSize, CWnd *
pWndParent /* = NULL */)
{
	if( m_pSoundData ) {
		// already created
		ASSERT(FALSE);
		return FALSE;
	}
	ASSERT(pSoundData != 0);
	ASSERT(dwSize > 0);
	register LPBYTE p = LPBYTE(pSoundData);
	// check header of MIDI
	if(*(DWORD*)p != MThd) {
		ASSERT(FALSE);
		return FALSE;
	}
	p += sizeof(DWORD);

	// check header size
	DWORD dwHeaderSize = DWORDSWAP(*(DWORD*)p);
	if( dwHeaderSize != sizeof(MIDIFILEHDR) ) {
		ASSERT(FALSE);
		return FALSE;
	}
	p += sizeof(DWORD);
	// get header
	MIDIFILEHDR hdr;
	::CopyMemory(&hdr, p, dwHeaderSize);
	m_dwFormat = DWORD(WORDSWAP(hdr.wFormat));
	m_dwTrackCount = DWORD(WORDSWAP(hdr.wTrackCount));
	m_dwTimeDivision = DWORD(WORDSWAP(hdr.wTimeDivision));
	p += dwHeaderSize;
	// create the array of tracks
	m_Tracks.resize(m_dwTrackCount);
	for(register DWORD i = 0; i < m_dwTrackCount; ++i) {
		// check header of track
		if(*(DWORD*)p != MTrk) {
```

```
                    ASSERT(FALSE);
                    return FALSE;
               }
               p += sizeof(DWORD);

               m_Tracks[i].dwTrackLength = DWORDSWAP(*(DWORD*)p);
               p += sizeof(DWORD);

               m_Tracks[i].pTrackStart = m_Tracks[i].pTrackCurrent = p;
                    p += m_Tracks[i].dwTrackLength;
          // Handle bozo MIDI files which contain empty track chunks
               if( !m_Tracks[i].dwTrackLength ) {
                    m_Tracks[i].fdwTrack |= ITS_F_ENDOFTRK;
                    continue;
               }
               // We always preread the time from each track so the
mixer code can
               // determine which track has the next event with a
minimum of work
               if( !GetTrackVDWord( &m_Tracks[i], &m_Tracks
[i].tkNextEventDue )) {
                    TRACE0("Error in MIDI data\n");
                    ASSERT(FALSE);
                    return FALSE;
               }
          }
          m_pSoundData = pSoundData;
          m_dwSoundSize = dwSize;
          m_pWndParent = pWndParent;
          // allocate volume channels and initialise them
          m_Volumes.resize(NUM_CHANNELS, VOLUME_INIT);

          if( ! StreamBufferSetup() ) {
               ASSERT(FALSE);
               return FALSE;
          }

          return TRUE;
}
BOOL CMIDI :: Play(BOOL bInfinite /* = FALSE */) {
          if( IsPaused() ) {
               Continue();
               return TRUE;
          }
          // calling Play() while it is already playing will restart
from scratch
          if( IsPlaying() )
               Stop();
          // Clear the status of our callback so it will handle
          // MOM_DONE callbacks once more
          m_uCallbackStatus = 0;
          if( !m_bLooped )
               m_bInsertTempo = TRUE;
          MMRESULT mmResult;
          if( (mmResult = midiStreamRestart(m_hStream)) !=
MMSYSERR_NOERROR ) {
               MidiError(mmResult);
               return FALSE;
```

```
    }

    m_bPlaying = TRUE;
    m_bLooped = bInfinite;

    return m_bPlaying;
}
BOOL CMIDI :: Stop(BOOL bReOpen /*=TRUE*/) {
    MMRESULT mmrRetVal;
    if( IsPlaying()   || (m_uCallbackStatus !=
STATUS_CALLBACKDEAD) ) {
        m_bPlaying = m_bPaused = FALSE;
        if( m_uCallbackStatus != STATUS_CALLBACKDEAD &&
m_uCallbackStatus != STATUS_WAITINGFOREND )
            m_uCallbackStatus = STATUS_KILLCALLBACK;
        if( (mmrRetVal = midiStreamStop(m_hStream) ) !=
MMSYSERR_NOERROR ) {
            MidiError(mmrRetVal);
            return FALSE;
        }
        if( (mmrRetVal = midiOutReset((HMIDIOUT)m_hStream)) !=
MMSYSERR_NOERROR ) {
            MidiError(mmrRetVal);
            return FALSE;
        }
        // Wait for the callback thread to release this thread,
which it will do by
        // calling SetEvent() once all buffers are returned to
it
        if( WaitForSingleObject( m_hBufferReturnEvent,
DEBUG_CALLBACK_TIMEOUT ) == WAIT_TIMEOUT ) {
            // Note, this is a risky move because the callback
may be genuinely busy, but
            // when we're debugging, it's safer and faster
than freezing the application,
            // which leaves the MIDI device locked up and
forces a system reset...
            TRACE0("Timed out waiting for MIDI callback\n");
            m_uCallbackStatus = STATUS_CALLBACKDEAD;
        }
    }
    if( m_uCallbackStatus == STATUS_CALLBACKDEAD ) {
        m_uCallbackStatus = 0;
        FreeBuffers();
        if( m_hStream ) {
            if( (mmrRetVal = midiStreamClose(m_hStream) ) !=
MMSYSERR_NOERROR ) {
                MidiError(mmrRetVal);
            }
            m_hStream = 0;
        }

        if( bReOpen ) {
            if( !StreamBufferSetup() ) {
                // Error setting up for MIDI file
                // Notification is already taken care of...
                return FALSE;
            }
```

```
                if( ! m_bLooped ) {
                        Rewind();
                        m_dwProgressBytes = 0;
                        m_dwStatus = 0;
                }
        }
    }
    return TRUE;
}
BOOL CMIDI :: Pause() {
    if( ! m_bPaused && m_bPlaying && m_pSoundData && m_hStream ) {
        midiStreamPause(m_hStream);
        m_bPaused = TRUE;
    }
    return FALSE;
}
BOOL CMIDI :: Continue() {
    if( m_bPaused && m_bPlaying && m_pSoundData && m_hStream ) {
        midiStreamRestart(m_hStream);
        m_bPaused = FALSE;
    }
    return FALSE;
}
BOOL CMIDI :: Rewind() {
    if( ! m_pSoundData )
        return FALSE;
    for(register DWORD i = 0; i < m_dwTrackCount; ++i) {
        m_Tracks[i].pTrackCurrent = m_Tracks[i].pTrackStart;
        m_Tracks[i].byRunningStatus = 0;
        m_Tracks[i].tkNextEventDue = 0;
        m_Tracks[i].fdwTrack = 0;
      // Handle bozo MIDI files which contain empty track chunks
        if( !m_Tracks[i].dwTrackLength ) {
            m_Tracks[i].fdwTrack |= ITS_F_ENDOFTRK;
            continue;
        }

        // We always preread the time from each track so the
mixer code can
        // determine which track has the next event with a
minimum of work
        if( !GetTrackVDWord( &m_Tracks[i], &m_Tracks
[i].tkNextEventDue )) {
            TRACE0("Error in MIDI data\n");
            ASSERT(FALSE);
            return FALSE;
        }
    }

    return TRUE;
}
DWORD CMIDI :: GetChannelCount() const {
    return m_Volumes.size();
}
void CMIDI :: SetVolume(DWORD dwPercent) {
    const DWORD dwSize = m_Volumes.size();
    for( register DWORD i = 0; i < dwSize; ++i )
        SetChannelVolume(i, dwPercent);
```

```
}
DWORD CMIDI :: GetVolume() const {
      DWORD dwVolume = 0;
      const DWORD dwSize = m_Volumes.size();
      for( register DWORD i = 0; i < dwSize; ++i )
            dwVolume += GetChannelVolume(i);

      return dwVolume / GetChannelCount();
}
void CMIDI :: SetChannelVolume(DWORD dwChannel, DWORD dwPercent) {
      ASSERT(dwChannel < m_Volumes.size());

      if( !m_bPlaying )
            return;
      m_Volumes[dwChannel] = (dwPercent > 100) ? 100 : dwPercent;
      DWORD dwEvent = MIDI_CTRLCHANGE | dwChannel | ((DWORD)
MIDICTRL_VOLUME << 8) |
((DWORD)(m_Volumes[dwChannel]*VOLUME_MAX/100) << 16);
      MMRESULT mmrRetVal;
      if(( mmrRetVal = midiOutShortMsg((HMIDIOUT)m_hStream,
dwEvent)) != MMSYSERR_NOERROR ) {
            MidiError(mmrRetVal);
            return;
      }
}
DWORD CMIDI :: GetChannelVolume(DWORD dwChannel) const {
      ASSERT(dwChannel < GetChannelCount());
      return m_Volumes[dwChannel];
}
void CMIDI :: SetTempo(DWORD dwPercent) {
      m_dwTempoMultiplier = dwPercent ? dwPercent : 1;
      m_bInsertTempo = TRUE;
}
DWORD CMIDI :: GetTempo() const {
      return m_dwTempoMultiplier;
}
void CMIDI :: SetInfinitePlay(BOOL bSet) {
      m_bLooped = bSet;
}
////////////////////////////////////////////////////////////////////
//This function converts MIDI data from the track buffers setup by a
// previous call to ConverterInit().  It will convert data until an
error is
// encountered or the output buffer has been filled with as much
event data
// as possible, not to exceed dwMaxLength. This function can take a
couple
// bit flags, passed through dwFlags. Information about the
success/failure
// of this operation and the number of output bytes actually
converted will
// be returned in the CONVERTINFO structure pointed at by lpciInfo.
int CMIDI :: ConvertToBuffer(DWORD dwFlags, CONVERTINFO * lpciInfo)
{
      int       nChkErr;
      lpciInfo->dwBytesRecorded = 0;
      if( dwFlags & CONVERTF_RESET ) {
            m_dwProgressBytes = 0;
```

```
            m_dwStatus = 0;
            memset( &m_teTemp, 0, sizeof(TEMPEVENT));
            m_ptsTrack = m_ptsFound = 0;
      }

      // If we were already done, then return with a warning...
      if( m_dwStatus & CONVERTF_STATUS_DONE ) {
            if( m_bLooped ) {
                  Rewind();
                  m_dwProgressBytes = 0;
                  m_dwStatus = 0;
            } else
                  return CONVERTERR_DONE;
      } else if( m_dwStatus & CONVERTF_STATUS_STUCK ) {
            // The caller is asking us to continue, but we're
already hosed because we
            // previously identified something as corrupt, so
complain louder this time.
            return( CONVERTERR_STUCK );
      } else if( m_dwStatus & CONVERTF_STATUS_GOTEVENT ) {
            // Turn off this bit flag
            m_dwStatus ^= CONVERTF_STATUS_GOTEVENT;

            // The following code for this case is duplicated from
below, and is
            // designed to handle a "straggler" event, should we
have one left over
            // from previous processing the last time this function
was called.

            // Don't add end of track event 'til we're done
            if( m_teTemp.byShortData[0] == MIDI_META &&
m_teTemp.byShortData[1] == MIDI_META_EOT ) {
                  if( m_dwMallocBlocks ) {
                        delete [] m_teTemp.pLongData;
                        --m_dwMallocBlocks;
                  }
            } else if(( nChkErr = AddEventToStreamBuffer( &m_teTemp,
lpciInfo )) != CONVERTERR_NOERROR ) {
                  if( nChkErr == CONVERTERR_BUFFERFULL ) {
                        // Do some processing and tell caller that
this buffer's full
                        m_dwStatus |= CONVERTF_STATUS_GOTEVENT;
                        return CONVERTERR_NOERROR;
                  } else if( nChkErr == CONVERTERR_METASKIP ) {
                        // We skip by all meta events that aren't
tempo changes...
                  } else {
                        TRACE0("Unable to add event to stream
buffer.\n");
                        if( m_dwMallocBlocks ) {
                              delete [] m_teTemp.pLongData;
                              m_dwMallocBlocks--;
                        }
                        return( TRUE );
                  }
            }
      }
```

```
      for(;;) {
            m_ptsFound = 0;
            m_tkNext = 0xFFFFFFFFL;
            // Find nearest event due
            for( register DWORD idx = 0; idx < m_Tracks.size();
++idx ) {
                  m_ptsTrack = &m_Tracks[idx];
                  if( !(m_ptsTrack->fdwTrack & ITS_F_ENDOFTRK) &&
(m_ptsTrack->tkNextEventDue < m_tkNext) ) {
                        m_tkNext = m_ptsTrack->tkNextEventDue;
                        m_ptsFound = m_ptsTrack;
                  }
            }

            // None found?  We must be done, so return to the caller
with a smile.
            if( !m_ptsFound ) {
                  m_dwStatus |= CONVERTF_STATUS_DONE;
                  // Need to set return buffer members properly
                  return CONVERTERR_NOERROR;
            }

            // Ok, get the event header from that track
            if( !GetTrackEvent( m_ptsFound, &m_teTemp )) {
                  // Warn future calls that this converter is stuck
at a corrupt spot
                  // and can't continue
                  m_dwStatus |= CONVERTF_STATUS_STUCK;
                  return CONVERTERR_CORRUPT;
            }

            // Don't add end of track event 'til we're done
            if( m_teTemp.byShortData[0] == MIDI_META &&
m_teTemp.byShortData[1] == MIDI_META_EOT ) {
                  if( m_dwMallocBlocks ) {
                        delete [] m_teTemp.pLongData;
                        --m_dwMallocBlocks;
                  }
                  continue;
            }

            if(( nChkErr = AddEventToStreamBuffer( &m_teTemp,
lpciInfo )) != CONVERTERR_NOERROR ) {
                  if( nChkErr == CONVERTERR_BUFFERFULL ) {
                        // Do some processing and tell somebody this
buffer is full...
                        m_dwStatus |= CONVERTF_STATUS_GOTEVENT;
                        return CONVERTERR_NOERROR;
                  } else if( nChkErr == CONVERTERR_METASKIP ) {
                        // We skip by all meta events that aren't
tempo changes...
                  } else {
                        TRACE0("Unable to add event to stream
buffer.\n");
                        if( m_dwMallocBlocks ) {
                              delete [] m_teTemp.pLongData;
                              m_dwMallocBlocks--;
                        }
```

```
                        return TRUE;
                    }
                }
            }

        return CONVERTERR_NOERROR;
}

// GetTrackEvent
//
// Fills in the event struct with the next event from the track
//
// pteTemp->tkEvent will contain the absolute tick time of the event
// pteTemp->byShortData[0] will contain
//  MIDI_META if the event is a meta event;
//   in this case pteTemp->byShortData[1] will contain the meta
class
//  MIDI_SYSEX or MIDI_SYSEXEND if the event is a SysEx event
//  Otherwise, the event is a channel message and pteTemp->
byShortData[1]
//   and pteTemp->byShortData[2] will contain the rest of the event.
//
// pteTemp->dwEventLength will contain
//  The total length of the channel message in pteTemp->byShortData
if
//   the event is a channel message
//  The total length of the paramter data pointed to by
//   pteTemp->pLongData otherwise
//
// pteTemp->pLongData will point at any additional paramters if the
//  event is a SysEx or meta event with non-zero length; else
//  it will contain NULL
//
// Returns TRUE on success or FALSE on any kind of parse error
// Prints its own error message ONLY in the debug version
//
// Maintains the state of the input track (i.e.
// ptsTrack->pTrackPointers, and ptsTrack->byRunningStatus).
//
BOOL CMIDI :: GetTrackEvent(TRACK * ptsTrack, TEMPEVENT * pteTemp) {
      DWORD   idx;
      UINT    dwEventLength;

      // Clear out the temporary event structure to get rid of old
data...
      memset( pteTemp, 0, sizeof(TEMPEVENT));

      // Already at end of track? There's nothing to read.
    if( ptsTrack->fdwTrack & ITS_F_ENDOFTRK )
            return FALSE;

      // Get the first byte, which determines the type of event.
      BYTE byByte;
      if( !GetTrackByte(ptsTrack, &byByte) )
            return FALSE;

      // If the high bit is not set, then this is a channel message
      // which uses the status byte from the last channel message
```

```
      // we saw. NOTE: We do not clear running status across SysEx
or
      // meta events even though the spec says to because there are
      // actually files out there which contain that sequence of
data.
      if( !(byByte & 0x80) ) {
            // No previous status byte? We're hosed.
            if( !ptsTrack->byRunningStatus ) {
                  TrackError(ptsTrack, gteBadRunStat);
                  return FALSE;
            }

            pteTemp->byShortData[0] = ptsTrack->byRunningStatus;
            pteTemp->byShortData[1] = byByte;

            byByte = pteTemp->byShortData[0] & 0xF0;
            pteTemp->dwEventLength = 2;

            // Only program change and channel pressure events are 2
bytes long;
            // the rest are 3 and need another byte
            if(( byByte != MIDI_PRGMCHANGE ) && ( byByte !=
MIDI_CHANPRESS )) {
                  if( !GetTrackByte( ptsTrack, &pteTemp->byShortData
[2] ))
                        return FALSE;
                  ++pteTemp->dwEventLength;
            }
      } else if(( byByte & 0xF0 ) != MIDI_SYSEX ) {
            // Not running status, not in SysEx range - must be
            // normal channel message (0x80-0xEF)
            pteTemp->byShortData[0] = byByte;
            ptsTrack->byRunningStatus = byByte;

            // Strip off channel and just keep message type
            byByte &= 0xF0;

            dwEventLength = ( byByte == MIDI_PRGMCHANGE || byByte ==
MIDI_CHANPRESS ) ? 1 : 2;
            pteTemp->dwEventLength = dwEventLength + 1;

            if( !GetTrackByte( ptsTrack, &pteTemp->byShortData[1] ))
                  return FALSE;
            if( dwEventLength == 2 )
                  if( !GetTrackByte( ptsTrack, &pteTemp->byShortData
[2] ))
                        return FALSE;
      } else if(( byByte == MIDI_SYSEX ) || ( byByte ==
MIDI_SYSEXEND )) {
            // One of the SysEx types. (They are the same as far as
we're concerned;
            // there is only a semantic difference in how the data
would actually
            // get sent when the file is played. We must take care
to put the proper
            // event type back on the output track, however.)
            //
            // Parse the general format of:
```

```
            //  BYTE    bEvent (MIDI_SYSEX or MIDI_SYSEXEND)
            //  VDWORD  cbParms
            //  BYTE    abParms[cbParms]
            pteTemp->byShortData[0] = byByte;
            if( !GetTrackVDWord( ptsTrack, &pteTemp->dwEventLength
)) {
                  TrackError( ptsTrack, gteSysExLenTrunc );
                  return FALSE;
            }

            // Malloc a temporary memory block to hold the parameter
data
            pteTemp->pLongData = new BYTE [pteTemp->dwEventLength];
            if( pteTemp->pLongData == 0 ) {
                  TrackError( ptsTrack, gteNoMem );
                  return FALSE;
            }
            // Increment our counter, which tells the program to
look around for
            // a malloc block to free, should it need to exit or
reset before the
            // block would normally be freed
            ++m_dwMallocBlocks;

            // Copy from the input buffer to the parameter data
buffer
            for( idx = 0; idx < pteTemp->dwEventLength; idx++ )
                  if( !GetTrackByte( ptsTrack, pteTemp->pLongData +
idx )) {
                        TrackError( ptsTrack, gteSysExTrunc );
                        return FALSE;
                  }
      } else if( byByte == MIDI_META ) {
            // It's a meta event. Parse the general form:
            //  BYTE    bEvent      (MIDI_META)
            //  BYTE    bClass
            //  VDWORD  cbParms
            //  BYTE    abParms[cbParms]
            pteTemp->byShortData[0] = byByte;

            if( !GetTrackByte( ptsTrack, &pteTemp->byShortData[1] ))
                  return FALSE;

            if( !GetTrackVDWord( ptsTrack, &pteTemp->dwEventLength
)) {
                  TrackError( ptsTrack, gteMetaLenTrunc );
                  return FALSE;
            }

            // NOTE: It's perfectly valid to have a meta with no
data
            // In this case, dwEventLength == 0 and pLongData ==
NULL
            if( pteTemp->dwEventLength ) {
                  // Malloc a temporary memory block to hold the
parameter data
                  pteTemp->pLongData = new BYTE [pteTemp->
dwEventLength];
```

```
                if( pteTemp->pLongData == 0 ) {
                      TrackError( ptsTrack, gteNoMem );
                      return FALSE;
                }
                // Increment our counter, which tells the program
to look around for
                // a malloc block to free, should it need to exit
or reset before the
                // block would normally be freed
                ++m_dwMallocBlocks;

                // Copy from the input buffer to the parameter
data buffer
                for( idx = 0; idx < pteTemp->dwEventLength; idx++
)
                      if( !GetTrackByte( ptsTrack, pteTemp->
pLongData + idx )) {
                            TrackError( ptsTrack, gteMetaTrunc );
                            return FALSE;
                      }
          }
          if( pteTemp->byShortData[1] == MIDI_META_EOT )
                ptsTrack->fdwTrack |= ITS_F_ENDOFTRK;
     } else {
          // Messages in this range are system messages and aren't
supposed to
          // be in a normal MIDI file. If they are, we've either
misparsed or the
          // authoring software is stupid.
          return FALSE;
     }
     // Event time was already stored as the current track time
     pteTemp->tkEvent = ptsTrack->tkNextEventDue;

     // Now update to the next event time. The code above MUST
properly
     // maintain the end of track flag in case the end of track
meta is
     // missing.  NOTE: This code is a continuation of the track
event
     // time pre-read which is done at the end of track
initialization.
     if( !( ptsTrack->fdwTrack & ITS_F_ENDOFTRK )) {
          DWORD tkDelta;

          if( !GetTrackVDWord( ptsTrack, &tkDelta ))
                return FALSE;

          ptsTrack->tkNextEventDue += tkDelta;
     }

     return TRUE;
}

// GetTrackVDWord
// Attempts to parse a variable length DWORD from the given track. A
VDWord
// in a MIDI file
```

```
//  (a) is in lo-hi format
//  (b) has the high bit set on every byte except the last
//
// Returns the DWORD in *lpdw and TRUE on success; else
// FALSE if we hit end of track first.
BOOL CMIDI :: GetTrackVDWord(TRACK * ptsTrack, LPDWORD lpdw) {
      ASSERT(ptsTrack != 0);
      ASSERT(lpdw != 0);

      if( ptsTrack->fdwTrack & ITS_F_ENDOFTRK )
            return FALSE;

      BYTE  byByte;
      DWORD dw = 0;

      do {
            if( !GetTrackByte( ptsTrack, &byByte ))
                  return FALSE;

            dw = ( dw << 7 ) | ( byByte & 0x7F );
      } while( byByte & 0x80 );

      *lpdw = dw;

      return TRUE;
}
// AddEventToStreamBuffer
// Put the given event into the given stream buffer at the given
location
// pteTemp must point to an event filled out in accordance with the
// description given in GetTrackEvent
//
// Handles its own error notification by displaying to the
appropriate
// output device (either our debugging window, or the screen).
int CMIDI :: AddEventToStreamBuffer( TEMPEVENT * pteTemp,
CONVERTINFO *lpciInfo ) {
      MIDIEVENT * pmeEvent = (MIDIEVENT *)( lpciInfo->
mhBuffer.lpData
                                          + lpciInfo->dwStartOffset
                                          + lpciInfo->
dwBytesRecorded );

      // When we see a new, empty buffer, set the start time on
it...
      if( !lpciInfo->dwBytesRecorded )
            lpciInfo->tkStart = m_tkCurrentTime;

      // Use the above set start time to figure out how much longer
we should fill
      // this buffer before officially declaring it as "full"
      if( m_tkCurrentTime - lpciInfo->tkStart > m_dwBufferTickLength
)
            if( lpciInfo->bTimesUp ) {
                  lpciInfo->bTimesUp = FALSE;
                  return CONVERTERR_BUFFERFULL;
            } else
                  lpciInfo->bTimesUp = TRUE;
```

```
      DWORD tkNow = m_tkCurrentTime;
      // Delta time is absolute event time minus absolute time
      // already gone by on this track
      DWORD tkDelta = pteTemp->tkEvent - m_tkCurrentTime;

      // Event time is now current time on this track
      m_tkCurrentTime = pteTemp->tkEvent;

      if( m_bInsertTempo ) {
            m_bInsertTempo = FALSE;
            if( lpciInfo->dwMaxLength-lpciInfo->dwBytesRecorded <
3*sizeof(DWORD)) {
                  // Cleanup from our write operation
                  return CONVERTERR_BUFFERFULL;
            }
            if( m_dwCurrentTempo ) {
                  pmeEvent->dwDeltaTime = 0;
                  pmeEvent->dwStreamID = 0;
                  pmeEvent->dwEvent = ( m_dwCurrentTempo * 100 ) /
m_dwTempoMultiplier;
                  pmeEvent->dwEvent |= (((DWORD)MEVT_TEMPO ) << 24 )
| MEVT_F_SHORT;
                  lpciInfo->dwBytesRecorded += 3 * sizeof(DWORD);
                  pmeEvent += 3 * sizeof(DWORD);
            }
      }
      if( pteTemp->byShortData[0] < MIDI_SYSEX ) {
            // Channel message. We know how long it is, just copy
it.
            // Need 3 DWORD's: delta-t, stream-ID, event
            if( lpciInfo->dwMaxLength-lpciInfo->dwBytesRecorded <
3*sizeof(DWORD)) {
                  // Cleanup from our write operation
                  return CONVERTERR_BUFFERFULL;
            }
            pmeEvent->dwDeltaTime = tkDelta;
            pmeEvent->dwStreamID = 0;
            pmeEvent->dwEvent = ( pteTemp->byShortData[0] )
                              | (((DWORD)pteTemp->byShortData[1] )
<< 8 )
                              | (((DWORD)pteTemp->byShortData[2] )
<< 16 )
                              | MEVT_F_SHORT;

            if((( pteTemp->byShortData[0] & 0xF0) == MIDI_CTRLCHANGE
) && ( pteTemp->byShortData[1] == MIDICTRL_VOLUME )) {
                  // If this is a volume change, generate a callback
so we can grab
                  // the new volume for our cache
                  pmeEvent->dwEvent |= MEVT_F_CALLBACK;
            }
            lpciInfo->dwBytesRecorded += 3 *sizeof(DWORD);
      } else if(( pteTemp->byShortData[0] == MIDI_SYSEX ) || (
pteTemp->byShortData[0] == MIDI_SYSEXEND )) {
            TRACE0("AddEventToStreamBuffer: Ignoring SysEx
event.\n");
            if( m_dwMallocBlocks ) {
```

```
                delete [] pteTemp->pLongData;
                --m_dwMallocBlocks;
            }
        } else {
            // Better be a meta event.
            //  BYTE    byEvent
            //  BYTE    byEventType
            //  VDWORD  dwEventLength
            //  BYTE    pLongEventData[dwEventLength]
            ASSERT( pteTemp->byShortData[0] == MIDI_META );
            // The only meta-event we care about is change tempo
            if( pteTemp->byShortData[1] != MIDI_META_TEMPO ) {
                if( m_dwMallocBlocks ) {
                    delete [] pteTemp->pLongData;
                    --m_dwMallocBlocks;
                }
                return CONVERTERR_METASKIP;
            }

            // We should have three bytes of parameter data...
            ASSERT(pteTemp->dwEventLength == 3);
            // Need 3 DWORD's: delta-t, stream-ID, event data
            if( lpciInfo->dwMaxLength - lpciInfo->dwBytesRecorded <
3 *sizeof(DWORD)) {
                // Cleanup the temporary event if necessary and
return
                if( m_dwMallocBlocks ) {
                    delete [] pteTemp->pLongData;
                    --m_dwMallocBlocks;
                }
                return CONVERTERR_BUFFERFULL;
            }
            pmeEvent->dwDeltaTime = tkDelta;
            pmeEvent->dwStreamID = 0;
        // Note: this is backwards from above because we're converting
a single
        //       data value from hi-lo to lo-hi format...
            pmeEvent->dwEvent = ( pteTemp->pLongData[2] )
                              | (((DWORD)pteTemp->pLongData[1] ) << 8 )
                              | (((DWORD)pteTemp->pLongData[0] ) << 16 );
            // This next step has absolutely nothing to do with the
conversion of a
            // MIDI file to a stream, it's simply put here to add
the functionality
            // of the tempo slider. If you don't need this, be sure
to remove the
            // next two lines.
            m_dwCurrentTempo = pmeEvent->dwEvent;
            pmeEvent->dwEvent = (pmeEvent->dwEvent * 100 ) /
m_dwTempoMultiplier;
            pmeEvent->dwEvent |= (((DWORD)MEVT_TEMPO ) << 24 ) |
MEVT_F_SHORT;
            m_dwBufferTickLength = (m_dwTimeDivision * 1000 *
BUFFER_TIME_LENGTH) / m_dwCurrentTempo;
            TRACE1("m_dwBufferTickLength = %lu\n",
m_dwBufferTickLength);
            if( m_dwMallocBlocks ) {
                delete [] pteTemp->pLongData;
```

```
                --m_dwMallocBlocks;
            }
            lpciInfo->dwBytesRecorded += 3 *sizeof(DWORD);
      }

      return CONVERTERR_NOERROR;
}
// StreamBufferSetup()
// Opens a MIDI stream. Then it goes about converting the data into
a midiStream buffer for playback.
BOOL CMIDI :: StreamBufferSetup() {
      int         nChkErr;
      BOOL  bFoundEnd = FALSE;
      MMRESULT          mmrRetVal;
      if( !m_hStream )
            if(( mmrRetVal = midiStreamOpen( &m_hStream,
                        &m_uMIDIDeviceID,
                        DWORD(1), DWORD(MidiProc),
                        DWORD(this),
                        CALLBACK_FUNCTION )) !=
MMSYSERR_NOERROR ) {
            MidiError(mmrRetVal);
            return FALSE;
      }
      // allocate stream buffers and initialise them
      m_StreamBuffers.resize(NUM_STREAM_BUFFERS);
      MIDIPROPTIMEDIV mptd;
      mptd.cbStruct = sizeof(mptd);
      mptd.dwTimeDiv = m_dwTimeDivision;
      if(( mmrRetVal = midiStreamProperty( m_hStream, (LPBYTE)&mptd,
                  MIDIPROP_SET | MIDIPROP_TIMEDIV )) !=
MMSYSERR_NOERROR ) {
            MidiError( mmrRetVal );
            return FALSE;
      }
      m_nEmptyBuffers = 0;
      DWORD dwConvertFlag = CONVERTF_RESET;
      for( m_nCurrentBuffer = 0; m_nCurrentBuffer <
NUM_STREAM_BUFFERS; m_nCurrentBuffer++ ) {
      m_StreamBuffers[m_nCurrentBuffer].mhBuffer.dwBufferLength =
OUT_BUFFER_SIZE;
            m_StreamBuffers[m_nCurrentBuffer].mhBuffer.lpData = new
char [OUT_BUFFER_SIZE];
            if( m_StreamBuffers[m_nCurrentBuffer].mhBuffer.lpData ==
0 )
                  return FALSE;
            // Tell the converter to convert up to one entire
buffer's length of output
            // data. Also, set a flag so it knows to reset any saved
state variables it
            // may keep from call to call.
            m_StreamBuffers[m_nCurrentBuffer].dwStartOffset = 0;
            m_StreamBuffers[m_nCurrentBuffer].dwMaxLength =
OUT_BUFFER_SIZE;
            m_StreamBuffers[m_nCurrentBuffer].tkStart = 0;
            m_StreamBuffers[m_nCurrentBuffer].bTimesUp = FALSE;
            if(( nChkErr = ConvertToBuffer( dwConvertFlag,
&m_StreamBuffers[m_nCurrentBuffer] )) != CONVERTERR_NOERROR ) {
```

```
                if( nChkErr == CONVERTERR_DONE ) {
                        bFoundEnd = TRUE;
                } else {
                        TRACE0("Initial conversion pass failed\n");
                        return FALSE;
                }
        }
    m_StreamBuffers[m_nCurrentBuffer].mhBuffer.dwBytesRecorded =
m_StreamBuffers[m_nCurrentBuffer].dwBytesRecorded;
        if( !m_bBuffersPrepared )
                if(( mmrRetVal = midiOutPrepareHeader( (HMIDIOUT)
m_hStream,
        &m_StreamBuffers[m_nCurrentBuffer].mhBuffer,
                                sizeof(MIDIHDR))) !=
MMSYSERR_NOERROR ) {
                        MidiError( mmrRetVal );
                        return FALSE;
                }
        if(( mmrRetVal = midiStreamOut( m_hStream,
    &m_StreamBuffers[m_nCurrentBuffer].mhBuffer,
sizeof(MIDIHDR))) != MMSYSERR_NOERROR ) {
        MidiError(mmrRetVal);
                break;
        }
        dwConvertFlag = 0;
        if( bFoundEnd )
                break;
    }
    m_bBuffersPrepared = TRUE;
    m_nCurrentBuffer = 0;
    return TRUE;
}

// This function unprepares and frees all our buffers -- something
we must
// do to work around a bug in MMYSYSTEM that prevents a device from
playing
// back properly unless it is closed and reopened after each stop.
void CMIDI :: FreeBuffers() {
    DWORD idx;
    MMRESULT    mmrRetVal;
    if( m_bBuffersPrepared ) {
        for( idx = 0; idx < NUM_STREAM_BUFFERS; idx++ )
                if(( mmrRetVal = midiOutUnprepareHeader(
(HMIDIOUT)m_hStream,
                                                &m_StreamBuffers
[idx].mhBuffer,
                                                sizeof(MIDIHDR))) !=
MMSYSERR_NOERROR ) {
                        MidiError(mmrRetVal);
                }
        m_bBuffersPrepared = FALSE;
    }
    // Free our stream buffers...
    for( idx = 0; idx < NUM_STREAM_BUFFERS; idx++ )
    if( m_StreamBuffers[idx].mhBuffer.lpData ) {
        delete [] m_StreamBuffers[idx].mhBuffer.lpData;
        m_StreamBuffers[idx].mhBuffer.lpData = 0;
```

```
	}
}

////////////////////////////////////////////////////////////////////
void CMIDI :: MidiError(MMRESULT mmResult) {
	#ifdef _DEBUG
		char chText[512];
		midiOutGetErrorText(mmResult, chText, sizeof(chText));
		TRACE1("Midi error: %hs\n", chText);
	#endif
}
void CMIDI :: TrackError(TRACK * ptsTrack, LPSTR lpszErr ) {
	TRACE1("Track buffer offset %lu\n", (DWORD)(ptsTrack->
pTrackCurrent - ptsTrack->pTrackStart));
    TRACE1("Track total length %lu\n", ptsTrack->dwTrackLength);
	TRACE1("%hs\n", lpszErr);
}
////////////////////////////////////////////////////////////////////
void CMIDI :: OnMidiOutOpen() {
}
void CMIDI :: OnMidiOutDone(MIDIHDR & rHdr) {
	if( m_uCallbackStatus == STATUS_CALLBACKDEAD )
		return;
	++m_nEmptyBuffers;
	if( m_uCallbackStatus == STATUS_WAITINGFOREND ) {
		if( m_nEmptyBuffers < NUM_STREAM_BUFFERS )
			return;
		else {
			m_uCallbackStatus = STATUS_CALLBACKDEAD;
			Stop();
			SetEvent(m_hBufferReturnEvent);
			return;
		}
	}
	// This flag is set whenever the callback is waiting for all
buffers to
	// come back.
	if( m_uCallbackStatus == STATUS_KILLCALLBACK ) {
		// Count NUM_STREAM_BUFFERS-1 being returned for the
last time
		if( m_nEmptyBuffers < NUM_STREAM_BUFFERS )
			return;
		else {
			// Change the status to callback dead
			m_uCallbackStatus = STATUS_CALLBACKDEAD;
			SetEvent(m_hBufferReturnEvent);
			return;
		}
	}
	m_dwProgressBytes +=
m_StreamBuffers[m_nCurrentBuffer].mhBuffer.dwBytesRecorded;

	////////////////////////////////////////////////////////////////
// Fill an available buffer with audio data again...
	if( m_bPlaying && m_nEmptyBuffers ) {
		m_StreamBuffers[m_nCurrentBuffer].dwStartOffset = 0;
		m_StreamBuffers[m_nCurrentBuffer].dwMaxLength =
OUT_BUFFER_SIZE;
```

```
            m_StreamBuffers[m_nCurrentBuffer].tkStart = 0;
            m_StreamBuffers[m_nCurrentBuffer].dwBytesRecorded = 0;
            m_StreamBuffers[m_nCurrentBuffer].bTimesUp = FALSE;
            int nChkErr;
            if(( nChkErr = ConvertToBuffer( 0, &m_StreamBuffers
[m_nCurrentBuffer] )) != CONVERTERR_NOERROR ) {
                  if( nChkErr == CONVERTERR_DONE ) {
                        m_uCallbackStatus = STATUS_WAITINGFOREND;
                        return;
                  } else {
                        TRACE0("MidiProc() conversion pass
failed!\n");
                        return;
                  }
            }
      m_StreamBuffers[m_nCurrentBuffer].mhBuffer.dwBytesRecorded =
m_StreamBuffers[m_nCurrentBuffer].dwBytesRecorded;
            MMRESULT mmrRetVal;
            if( (mmrRetVal = midiStreamOut(m_hStream,
&m_StreamBuffers[m_nCurrentBuffer].mhBuffer, sizeof(MIDIHDR))) !=
MMSYSERR_NOERROR ) {
                  MidiError(mmrRetVal);
                  return;
            }
            m_nCurrentBuffer = ( m_nCurrentBuffer + 1 ) %
NUM_STREAM_BUFFERS;
            m_nEmptyBuffers--;
      }
}
void CMIDI :: OnMidiOutPositionCB(MIDIHDR & rHdr, MIDIEVENT &
rEvent) {
      if( MIDIEVENT_TYPE(rEvent.dwEvent) == MIDI_CTRLCHANGE )
      {
            if( MIDIEVENT_DATA1(rEvent.dwEvent) == MIDICTRL_VOLUME )
{
                  // Mask off the channel number and cache the
volume data byte
                  m_Volumes[MIDIEVENT_CHANNEL(rEvent.dwEvent)] =
DWORD(MIDIEVENT_VOLUME(rEvent.dwEvent)*100/VOLUME_MAX);
                  if( m_pWndParent && ::IsWindow(m_pWndParent->
GetSafeHwnd()) )
                        // Do not use SendMessage(), because a
change of the midi stream has no effect
                        // during callback handling, so if the owner
wants to adjust the volume, as a
                        // result of the windows message, (s)he will
not hear that change.
                        m_pWndParent->PostMessage(
                              WM_MIDI_VOLUMECHANGED,
                              WPARAM(this),
                              LPARAM(
                                    MAKELONG(
                                          WORD(MIDIEVENT_CHANNEL
(rEvent.dwEvent)),
                                          WORD(MIDIEVENT_VOLUME
(rEvent.dwEvent)*100/VOLUME_MAX)
                                    )
                              )
```

```
                        );
                }
        }
}


void CMIDI :: OnMidiOutClose() {
}
////////////////////////////////////////////////////////////////////////
void CMIDI :: MidiProc(HMIDIOUT hMidi, UINT uMsg, DWORD
dwInstanceData, DWORD dwParam1, DWORD dwParam2) {
      CMIDI * pMidi = (CMIDI *) dwInstanceData;
      ASSERT(pMidi != 0);
      MIDIHDR * pHdr = (MIDIHDR*) dwParam1;
      switch(uMsg) {
            case MOM_OPEN:
                  pMidi->OnMidiOutOpen();
                  break;
            case MOM_CLOSE:
                  pMidi->OnMidiOutClose();
                  break;
            case MOM_DONE:
                  ASSERT(pHdr != 0);
                  pMidi->OnMidiOutDone(*pHdr);
                  break;
            case MOM_POSITIONCB:
                  ASSERT(pHdr != 0);
                  pMidi->OnMidiOutPositionCB(*pHdr, *((MIDIEVENT*)
(pHdr->lpData + pHdr->dwOffset)));
                  break;
            default:
                  break;
      }
}
```

DirectSound.h

```
////////////////////////////////////////////////////////////////////////
// Copyright (C) 1998 by J rg K nig
// All rights reserved
//
// This file is part of the completely free tetris clone "CGTetris".
//
// This is free software.

// You may redistribute it by any means providing it is not sold for
profit
// without the authors written consent.
//
// No warrantee of any kind, expressed or implied, is included with
this
// software; use at your own risk, responsibility for damages (if
any) to
// anyone resulting from the use of this software rests entirely
with the
// user.
// Send bug reports, bug fixes, enhancements, requests, flames,
etc., and
```

```
// I'll try to keep a version up to date.  I can be reached as
follows:
//    J.Koenig@adg.de                 (company site)
//    Joerg.Koenig@rhein-neckar.de    (private site)
////////////////////////////////////////////////////////////////////
// DirectSound.h: interface for the CDirectSound class.
////////////////////////////////////////////////////////////////////
#if
!defined(AFX_DIRECTSOUND_H__A20FE86F_118F_11D2_9AB3_0060B0CDC13E__IN
CLUDED_)
#define
AFX_DIRECTSOUND_H__A20FE86F_118F_11D2_9AB3_0060B0CDC13E__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#include <mmsystem.h>
#include <dsound.h>
#pragma message("linking with Microsoft's DirectSound library ...")
#pragma comment(lib, "dsound.lib")
class CDirectSound
{
public:          // construction/destruction
      CDirectSound();
      virtual ~CDirectSound();
      // If the "pWnd" paramter is NULL, then AfxGetApp()->
GetMainWnd() will be used.
      BOOL  Create(LPCTSTR pszResource, CWnd * pWnd = 0);
      BOOL  Create(UINT uResourceID, CWnd * pWnd = 0) {
            return Create(MAKEINTRESOURCE(uResourceID), pWnd);
      }
      // Alternativly you can specify the sound by yourself
      // Note that the class does not copy the entire data ! Instead
      // a pointer to the given data will be stored !
      // You can load an entire WAV file into memory and then call
this
      // Create() method.
      BOOL  Create(LPVOID pSoundData, CWnd * pWnd = 0);
public:          // operations
      BOOL              IsValid() const;
      void              Play(DWORD dwStartPosition = 0, BOOL bLoop =
FALSE);
      void              Stop();
      void              Pause();
      void              Continue();
      CDirectSound &    EnableSound(BOOL bEnable = TRUE) {
                              m_bEnabled = bEnable;
                              if( ! bEnable )
                                    Stop();
                              return * this;
                        }
      BOOL              IsEnabled() const { return m_bEnabled; }
protected:  // implementation
      BOOL SetSoundData(LPVOID pSoundData, DWORD dwSoundSize);
      BOOL CreateSoundBuffer(WAVEFORMATEX * pcmwf);
      BOOL GetWaveData(void * pRes, WAVEFORMATEX * & pWaveHeader,
void * & pbWaveData, DWORD & cbWaveSize);

private:    // data member
```

```
      LPVOID m_pTheSound;
      DWORD m_dwTheSound;
      LPDIRECTSOUNDBUFFER m_pDsb;
      BOOL m_bEnabled;
      static LPDIRECTSOUND m_lpDirectSound;
      static DWORD m_dwInstances;
};
#endif //
!defined(AFX_DIRECTSOUND_H__A20FE86F_118F_11D2_9AB3_0060B0CDC13E__IN
CLUDED_)
```

Directsound.cpp

```
////////////////////////////////////////////////////////////////////
// Copyright (C) 1998 by J rg K nig
// All rights reserved
//
// This file is part of the completely free tetris clone "CGTetris".
//
// This is free software.
// You may redistribute it by any means providing it is not sold for
profit
// without the authors written consent.
//
// No warrantee of any kind, expressed or implied, is included with
this
// software; use at your own risk, responsibility for damages (if
any) to
// anyone resulting from the use of this software rests entirely
with the
// user.
//
// Send bug reports, bug fixes, enhancements, requests, flames,
etc., and
// I'll try to keep a version up to date.  I can be reached as
follows:
//    J.Koenig@adg.de                 (company site)
//    Joerg.Koenig@rhein-neckar.de    (private site)
////////////////////////////////////////////////////////////////////
// DirectSound.cpp: implementation of the CDirectSound class.
//
////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "DirectSound.h"
// The following macro is defined since DirectX 5, but will work
with
// older versions too.
#ifndef DSBLOCK_ENTIREBUFFER
      #define DSBLOCK_ENTIREBUFFER        0x00000002
#endif
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
static void DSError( HRESULT hRes ) {
      switch(hRes) {
            case DS_OK: TRACE0("NO ERROR\n"); break;
```

```
            case DSERR_ALLOCATED: TRACE0("ALLOCATED\n"); break;
            case DSERR_INVALIDPARAM: TRACE0("INVALIDPARAM\n");
break;
            case DSERR_OUTOFMEMORY: TRACE0("OUTOFMEMORY\n"); break;
            case DSERR_UNSUPPORTED: TRACE0("UNSUPPORTED\n"); break;
            case DSERR_NOAGGREGATION: TRACE0("NOAGGREGATION\n");
break;
            case DSERR_UNINITIALIZED: TRACE0("UNINITIALIZED\n");
break;
            case DSERR_BADFORMAT: TRACE0("BADFORMAT\n"); break;
            case DSERR_ALREADYINITIALIZED:
TRACE0("ALREADYINITIALIZED\n"); break;
            case DSERR_BUFFERLOST: TRACE0("BUFFERLOST\n"); break;
            case DSERR_CONTROLUNAVAIL: TRACE0("CONTROLUNAVAIL\n");
break;
            case DSERR_GENERIC: TRACE0("GENERIC\n"); break;
            case DSERR_INVALIDCALL: TRACE0("INVALIDCALL\n"); break;
            case DSERR_OTHERAPPHASPRIO: TRACE0("OTHERAPPHASPRIO\n");
break;
            case DSERR_PRIOLEVELNEEDED: TRACE0("PRIOLEVELNEEDED\n");
break;
            default: TRACE1("%lu\n",hRes);break;
      }
}
////////////////////////////////////////////////////////////////////
LPDIRECTSOUND CDirectSound::m_lpDirectSound;
DWORD CDirectSound::m_dwInstances;
CDirectSound::CDirectSound()
{
      m_lpDirectSound = 0;
      m_pDsb = 0;
      m_pTheSound = 0;
      m_dwTheSound = 0;
      m_bEnabled = TRUE;

      ++m_dwInstances;
}

CDirectSound::~CDirectSound()
{
      if( m_pDsb )
            m_pDsb->Release();

      if( !--m_dwInstances && m_lpDirectSound ) {
            m_lpDirectSound->Release();
            m_lpDirectSound = 0;
      }
}

BOOL CDirectSound::Create(LPCTSTR pszResource, CWnd * pWnd)
{
      ////////////////////////////////////////////////////////////////
      // load resource
      HINSTANCE hApp = ::GetModuleHandle(0);
      ASSERT(hApp);
      HRSRC hResInfo = ::FindResource(hApp, pszResource, TEXT
("WAVE"));
      if(hResInfo == 0)
```

```
            return FALSE;
      HGLOBAL hRes = ::LoadResource(hApp, hResInfo);
      if(hRes == 0)
            return FALSE;
      LPVOID pTheSound = ::LockResource(hRes);
      if(pTheSound == 0)
            return FALSE;
      return Create(pTheSound, pWnd);
}
BOOL CDirectSound :: Create(LPVOID pSoundData, CWnd * pWnd) {
      if(pWnd == 0)
            pWnd = AfxGetApp()->GetMainWnd();
      ASSERT(pWnd != 0);
      ASSERT(::IsWindow(pWnd->GetSafeHwnd()));
      ASSERT(pSoundData != 0);
      ////////////////////////////////////////////////////////////////
      // create direct sound object
      if( m_lpDirectSound == 0 ) {
            // Someone might use sounds for starting apps. This may
cause
            // DirectSoundCreate() to fail because the driver is
used by
            // anyone else. So wait a little before starting with
the work ...
            HRESULT hRes = DS_OK;
            short nRes = 0;
            do {
                  if( nRes )
                        ::Sleep(500);
                  hRes = ::DirectSoundCreate(0, &m_lpDirectSound,
0);
                  ++nRes;
            } while( nRes < 10 && (hRes == DSERR_ALLOCATED || hRes
== DSERR_NODRIVER) );

            if( hRes != DS_OK )
                  return FALSE;
            m_lpDirectSound->SetCooperativeLevel(pWnd->GetSafeHwnd
(), DSSCL_NORMAL);
      }
      ASSERT(m_lpDirectSound != 0);
      WAVEFORMATEX * pcmwf;
      if( ! GetWaveData(pSoundData, pcmwf, m_pTheSound,
m_dwTheSound) ||
            ! CreateSoundBuffer(pcmwf) ||
            ! SetSoundData(m_pTheSound, m_dwTheSound) )
            return FALSE;
      return TRUE;
}
BOOL CDirectSound :: GetWaveData(void * pRes, WAVEFORMATEX * &
pWaveHeader, void * & pbWaveData, DWORD & cbWaveSize) {
      pWaveHeader = 0;
      pbWaveData = 0;
      cbWaveSize = 0;
      DWORD * pdw = (DWORD *)pRes;
      DWORD dwRiff = *pdw++;
      DWORD dwLength = *pdw++;
      DWORD dwType = *pdw++;
```

```
	if( dwRiff != mmioFOURCC('R', 'I', 'F', 'F') )
		return FALSE;		// not even RIFF
	if( dwType != mmioFOURCC('W', 'A', 'V', 'E') )
		return FALSE;		// not a WAV
	DWORD * pdwEnd = (DWORD *)((BYTE *)pdw + dwLength-4);
	while( pdw < pdwEnd ) {
		dwType = *pdw++;
		dwLength = *pdw++;
		switch( dwType ) {
			case mmioFOURCC('f', 'm', 't', ' '):
				if( !pWaveHeader ) {
					if( dwLength < sizeof(WAVEFORMAT) )
						return FALSE;		// not a WAV
					pWaveHeader = (WAVEFORMATEX *)pdw;

					if( pbWaveData && cbWaveSize )
						return TRUE;
				}
				break;
			case mmioFOURCC('d', 'a', 't', 'a'):
				pbWaveData = LPVOID(pdw);
				cbWaveSize = dwLength;
				if( pWaveHeader )
					return TRUE;
				break;
		}
		pdw = (DWORD *)((BYTE *)pdw + ((dwLength+1)&~1));
	}
	return FALSE;
}
BOOL CDirectSound::CreateSoundBuffer(WAVEFORMATEX * pcmwf)
{
	DSBUFFERDESC dsbdesc;
	// Set up DSBUFFERDESC structure.
	memset(&dsbdesc, 0, sizeof(DSBUFFERDESC)); // Zero it out.
	dsbdesc.dwSize = sizeof(DSBUFFERDESC);
	// Need no controls (pan, volume, frequency).
	dsbdesc.dwFlags = DSBCAPS_STATIC;		// assumes that the
sound is played often
	dsbdesc.dwBufferBytes = m_dwTheSound;
	dsbdesc.lpwfxFormat = pcmwf;	// Create buffer.
	HRESULT hRes;
	if( DS_OK != (hRes = m_lpDirectSound->CreateSoundBuffer
(&dsbdesc, &m_pDsb, 0)) ) {
		// Failed.
		DSError(hRes);
		m_pDsb = 0;
		return FALSE;
	}
	return TRUE;
}
BOOL CDirectSound::SetSoundData(void * pSoundData, DWORD
dwSoundSize) {
	LPVOID lpvPtr1;
	DWORD dwBytes1;
	// Obtain write pointer.
	HRESULT hr = m_pDsb->Lock(0, 0, &lpvPtr1, &dwBytes1, 0, 0,
DSBLOCK_ENTIREBUFFER);
```

```
    // If DSERR_BUFFERLOST is returned, restore and retry lock.
      if(DSERR_BUFFERLOST == hr) {
            m_pDsb->Restore();
            hr = m_pDsb->Lock(0, 0, &lpvPtr1, &dwBytes1, 0, 0,
DSBLOCK_ENTIREBUFFER);
      }
      if(DS_OK == hr) {
            // Write to pointers.
            ::CopyMemory(lpvPtr1, pSoundData, dwBytes1);
            // Release the data back to DirectSound.
            hr = m_pDsb->Unlock(lpvPtr1, dwBytes1, 0, 0);
            if(DS_OK == hr)
            return TRUE;
      }
      // Lock, Unlock, or Restore failed.
      return FALSE;
}
void CDirectSound::Play(DWORD dwStartPosition, BOOL bLoop)
{
      if( ! IsValid() || ! IsEnabled() )
            return;           // no chance to play the sound ...
      if( dwStartPosition > m_dwTheSound )
            dwStartPosition = m_dwTheSound;
      m_pDsb->SetCurrentPosition(dwStartPosition);
      if( DSERR_BUFFERLOST == m_pDsb->Play(0, 0, bLoop ?
DSBPLAY_LOOPING : 0) ) {
            // another application had stolen our buffer
            // Note that a "Restore()" is not enough, because
            // the sound data is invalid after Restore().
            SetSoundData(m_pTheSound, m_dwTheSound);

            // Try playing again
            m_pDsb->Play(0, 0, bLoop ? DSBPLAY_LOOPING : 0);
      }
}
void CDirectSound::Stop()
{
      if( IsValid() )
            m_pDsb->Stop();
}
void CDirectSound::Pause()
{
      Stop();
}
void CDirectSound::Continue()
{
      if( IsValid() ) {
            DWORD dwPlayCursor, dwWriteCursor;
            m_pDsb->GetCurrentPosition(&dwPlayCursor,
&dwWriteCursor);
            Play(dwPlayCursor);
      }
}
BOOL CDirectSound::IsValid() const
{
      return (m_lpDirectSound && m_pDsb && m_pTheSound &&
m_dwTheSound) ? TRUE : FALSE;
}
```

DIBSectionLite.h

```
#if
!defined(AFX_CDIBSECTIONLITE_H__35D9F3D4_B960_11D2_A981_2C4476000000
__INCLUDED_)
#define
AFX_CDIBSECTIONLITE_H__35D9F3D4_B960_11D2_A981_2C4476000000__INCLUDE
D_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// DIBSectionLite.h : header file
//
// Copyright   Dundas Software Ltd. 1999, All Rights Reserved
// Properties:
//    NO    Abstract class (does not have any objects)
//    NO    Derived from CWnd
//    NO    Is a CWnd.
//    NO    Two stage creation (constructor & Create())
//    NO    Has a message map
//    NO    Needs a resource (template)
//    YES   Persistent objects (saveable on disk)
//    YES   Uses exceptions
////////////////////////////////////////////////////////////////////
// Desciption :
// DIBSection is DIBSection wrapper class for Win32 platforms.
// This class provides a simple interface to DIBSections including
loading,
// saving and displaying DIBsections.
// Using DIBSection :
// This class is very simple to use. The bitmap can be set using
either SetBitmap()
// (which accepts either a Device dependant or device independant
bitmap, or a
// resource ID) or by using Load(), which allows an image to be
loaded from disk.
// To display the bitmap simply use Draw or Stretch.
//
// eg.
//
//      CDIBSectionLite dibsection;
//      dibsection.Load(_T("image.bmp"));
//      dibsection.Draw(pDC, CPoint(0,0));  // pDC is of type CDC*
//
//      CDIBSectionLite dibsection;
//      dibsection.SetBitmap(IDB_BITMAP);
//      dibsection.Draw(pDC, CPoint(0,0));  // pDC is of type CDC*
//
// The CDIBSectionLite API includes many methods to extract
information about the
// image, as well as palette options for getting and setting the
current palette.
//
// Author: Chris Maunder (cmaunder@dundas.com)
// Date  : 12 April 1999
#include <vfw.h>
#pragma comment(lib,
"vfw32")////////////////////////////////////////////////////////////
```

```
// defines
#define DS_BITMAP_FILEMARKER  ((WORD) ('M' << 8) | 'B')    // is
always "BM" = 0x4D42
/////////////////////////////////////////////////////////////////////
// BITMAPINFO wrapper
struct DIBINFO : public BITMAPINFO
{
      RGBQUAD      arColors[255];    // Color table info - adds an
extra 255 entries to palette

      operator LPBITMAPINFO()          { return (LPBITMAPINFO) this;
}
      operator LPBITMAPINFOHEADER()    { return &bmiHeader;
}
      RGBQUAD* ColorTable()            { return bmiColors;
}
};
/////////////////////////////////////////////////////////////////////
// LOGPALETTE wrapper
struct PALETTEINFO : public LOGPALETTE
{
    PALETTEENTRY arPalEntries[255];                 // Palette entries

    PALETTEINFO() { palVersion = (WORD) 0x300; }

    operator LPLOGPALETTE()   { return (LPLOGPALETTE) this;
}
    operator LPPALETTEENTRY() { return (LPPALETTEENTRY)
(palPalEntry); }
};
/////////////////////////////////////////////////////////////////////
// CDIBSectionLite object
class CDIBSectionLite : public CObject
{
// Construction
public:
      CDIBSectionLite();
      virtual ~CDIBSectionLite();
    void DeleteObject();
// static helpers
public:
    static int BytesPerLine(int nWidth, int nBitsPerPixel);
    static int NumColorEntries(int nBitsPerPixel);
    static PALETTEENTRY ms_StdColours[];
    static BOOL UsesPalette(CDC* pDC) { return (pDC->GetDeviceCaps
(RASTERCAPS) & RC_PALETTE); }
    static BOOL CreateHalftonePalette(CPalette& palette, int
nNumColours);
// Attributes
public:
    HBITMAP      GetSafeHandle() const       { return (this)?
m_hBitmap : NULL;        }
    CSize        GetSize() const             { return CSize(GetWidth
(), GetHeight());  }
    int          GetHeight() const           { return
m_DIBinfo.bmiHeader.biHeight;    }
    int          GetWidth() const            { return
m_DIBinfo.bmiHeader.biWidth;     }
```

```
    int           GetPlanes() const           { return
m_DIBinfo.bmiHeader.biPlanes;    }
    int           GetBitCount() const         { return
m_DIBinfo.bmiHeader.biBitCount;  }
    LPVOID        GetDIBits()                 { return m_ppvBits;
}
    LPBITMAPINFO GetBitmapInfo()              { return  (BITMAPINFO*)
m_DIBinfo;        }
    DWORD         GetImageSize() const        { return
m_DIBinfo.bmiHeader.biSizeImage; }
    LPBITMAPINFOHEADER GetBitmapInfoHeader() { return
(BITMAPINFOHEADER*) m_DIBinfo;   }
    BOOL SetBitmap(UINT nIDResource);
    BOOL SetBitmap(LPCTSTR lpszResourceName);
    BOOL SetBitmap(HBITMAP hBitmap, CPalette* pPalette = NULL);
    BOOL SetBitmap(LPBITMAPINFO lpBitmapInfo, LPVOID lpBits);
    CPalette *GetPalette()  { return &m_Palette; }
    BOOL SetPalette(CPalette* pPalette);
    BOOL SetLogPalette(LOGPALETTE* pLogPalette);
      BOOL SetDither(BOOL bDither);
    BOOL GetDither();
// Operations
public:
    BOOL Load(LPCTSTR lpszFileName);
    BOOL Save(LPCTSTR lpszFileName);
    BOOL Draw(CDC* pDC, CPoint ptDest, BOOL bForceBackground =
FALSE);
    BOOL Stretch(CDC* pDC, CPoint ptDest, CSize size, BOOL
bForceBackground = FALSE);
// Overrideables
// Implementation
public:
#ifdef _DEBUG
      virtual void AssertValid() const;
      virtual void Dump(CDumpContext& dc) const;
#endif
// Implementation
protected:
    void _ShowLastError();
    BOOL CreatePalette();
    BOOL FillDIBColorTable(UINT nNumColours, RGBQUAD *pRGB);
protected:
    HBITMAP  m_hBitmap;          // Handle to DIBSECTION
    DIBINFO  m_DIBinfo;          // Bitmap header & color table info
    VOID    *m_ppvBits;          // Pointer to bitmap bits
    UINT     m_iColorDataType;   // color data type (palette or RGB
values)
    UINT     m_iColorTableSize;  // Size of color table
    CPalette m_Palette;          // Color palette
    BOOL     m_bDither;          // Use DrawDib routines for
dithering?
    HDRAWDIB m_hDrawDib;         // handle to a DrawDib DC

private:
    HBITMAP  m_hOldBitmap;      // Storage for previous bitmap in
Memory DC
};
//{{AFX_INSERT_LOCATION}}
```

```
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_CDIBSECTIONLITE_H__35D9F3D4_B960_11D2_A981_2C4476000000
__INCLUDED_)
```

DIBSectionLite.cpp

```
// CDIBSectionLite.cpp : implementation file
//
// General purpose DIBsection wrapper class for Win9x, NT 4.0
//
// Author       : Chris Maunder (cmaunder@dundas.com)
// Date         : 17 May 1999
//
// Copyright © Dundas Software Ltd. 1999, All Rights Reserved
//
// This code may be used in compiled form in any way you desire.
This
// file may be redistributed unmodified by any means PROVIDING it is
// not sold for profit without the authors written consent, and
// providing that this notice and the authors name is included. If
// the source code in this file is used in any commercial
application
// then a simple email would be nice.
//
// This file is provided "as is" with no expressed or implied
warranty.
// The author accepts no liability for any damage, in any form,
caused
// by this code. Use it at your own risk and as with all code expect
bugs!
// It's been tested but I'm not perfect.
//
// Please use and enjoy. Please let me know of any
bugs/mods/improvements
// that you have found/implemented and I will fix/incorporate them
into this
// file.
//
// History :  25 May 1999 - First release
//             4 Jun 1999 - Fixed SetBitmap bug
#include "stdafx.h"
#include "DIBSectionLite.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// Standard colours
PALETTEENTRY CDIBSectionLite::ms_StdColours[] = {
    { 0x00, 0x00, 0x00, 0 },     // First 2 will be palette for
monochrome bitmaps
    { 0xFF, 0xFF, 0xFF, 0 },

    { 0x00, 0x00, 0xFF, 0 },     // First 16 will be palette for 16
colour bitmaps
    { 0x00, 0xFF, 0x00, 0 },
```

```
    { 0x00, 0xFF, 0xFF, 0 },
    { 0xFF, 0x00, 0x00, 0 },
    { 0xFF, 0x00, 0xFF, 0 },
    { 0xFF, 0xFF, 0x00, 0 },
    { 0x00, 0x00, 0x80, 0 },
    { 0x00, 0x80, 0x00, 0 },
    { 0x00, 0x80, 0x80, 0 },
    { 0x80, 0x00, 0x00, 0 },
    { 0x80, 0x00, 0x80, 0 },
    { 0x80, 0x80, 0x00, 0 },
    { 0x80, 0x80, 0x80, 0 },
    { 0xC0, 0xC0, 0xC0, 0 },
};
////////////////////////////////////////////////////////////////////////
// CDIBSectionLite static functions
// --- In  : nBitsPerPixel - bits per pixel
// --- Out :
// --- Returns :The number of colours for this colour depth
// --- Effect : Returns the number of color table entries given the
number
//              of bits per pixel of a bitmap
/*static*/ int CDIBSectionLite::NumColorEntries(int nBitsPerPixel)
{
    int nColors = 0;
    switch (nBitsPerPixel)
    {
        case 1:  nColors = 2;   break;
        case 4:  nColors = 16;  break;
        case 8:  nColors = 256; break;
        case 16:
        case 24:
        case 32: nColors = 0;   break; // 16,24 or 32 bpp have no
color table
        default:
           ASSERT(FALSE);
    }
    return nColors;
}
// --- In  : nWidth - image width in pixels
//           nBitsPerPixel - bits per pixel
// --- Out :
// --- Returns : Returns the number of storage bytes needed for each
scanline
//               in the bitmap
// --- Effect :
/*static*/ int CDIBSectionLite::BytesPerLine(int nWidth, int
nBitsPerPixel)
{
    int nBytesPerLine = nWidth * nBitsPerPixel;
    nBytesPerLine = ( (nBytesPerLine + 31) & (~31) ) / 8;

    return nBytesPerLine;
}
// --- In  : palette - reference to a palette object which will be
filled
//           nNumColours - number of colour entries to fill
// --- Out :
// --- Returns : TRUE on success, false otherwise
```

```
// --- Effect : Creates a halftone color palette independant of
screen color depth.
//              palette will be filled with the colors, and
nNumColours is the No.
//              of colors to file. If nNumColoursis 0 or > 256, then
256 colors are used.
/*static*/ BOOL CDIBSectionLite::CreateHalftonePalette(CPalette&
palette, int nNumColours)
{
    palette.DeleteObject();
    int nNumStandardColours = sizeof(ms_StdColours) / sizeof
(ms_StdColours[0]);
    int nIndex = 0;
    int nNumEntries = nNumColours;
    if (nNumEntries <= 0 || nNumEntries > 256)
        nNumEntries = 256;
    PALETTEINFO PalInfo;
    PalInfo.palNumEntries = (WORD) nNumEntries;
    // The standard colours (16)
    for (int i = 0; i < nNumStandardColours; i++)
    {
        if (nIndex >= nNumEntries) break;
        memcpy( &(PalInfo.palPalEntry[nIndex]), &(ms_StdColours[i]),
sizeof(PALETTEENTRY) );
        nIndex++;
    }
    // A colour cube (6 divisions = 216)
    for (int blue = 0; blue <= 5; blue++)
        for (int green = 0; green <= 5; green++)
            for (int red = 0; red <= 5; red++)
            {
                if (nIndex >= nNumEntries)
                    break;
                PalInfo.palPalEntry[nIndex].peRed   = (BYTE)
((red*255)/5);
                PalInfo.palPalEntry[nIndex].peGreen = (BYTE)
((green*255)/5);
                PalInfo.palPalEntry[nIndex].peBlue  = (BYTE)
((blue*255)/5);
                PalInfo.palPalEntry[nIndex].peFlags = 0;
                nIndex++;
            }

    // A grey scale (24 divisions)
    for (int grey = 0; grey <= 23; grey++)
    {
        if (nIndex >= nNumEntries)
            break;
        PalInfo.palPalEntry[nIndex].peRed   = (BYTE) (grey*255/23);
        PalInfo.palPalEntry[nIndex].peGreen = (BYTE) (grey*255/23);
        PalInfo.palPalEntry[nIndex].peBlue  = (BYTE) (grey*255/23);
        PalInfo.palPalEntry[nIndex].peFlags = 0;
        nIndex++;
    }

    return palette.CreatePalette((LPLOGPALETTE) PalInfo);
}
```

```
// CDIBSectionLite
CDIBSectionLite::CDIBSectionLite()
{
    m_hBitmap = NULL;
    m_bDither = FALSE;
    m_hDrawDib = NULL;
    DeleteObject(); // This will initialise to a known state - ie.
empty
}
CDIBSectionLite::~CDIBSectionLite()
{
    DeleteObject();
}
// --- In  :
// --- Out :
// --- Returns :
// --- Effect : Resets the object to an empty state, and frees all
memory used.
void CDIBSectionLite::DeleteObject()
{
    if (m_hBitmap)
        ::DeleteObject(m_hBitmap);
    m_hBitmap = NULL;
    m_ppvBits = NULL;
    memset(&m_DIBinfo, 0, sizeof(m_DIBinfo));
    if (m_hDrawDib)
        DrawDibClose(m_hDrawDib);
    m_hDrawDib = NULL;

    m_iColorDataType = DIB_RGB_COLORS;
    m_iColorTableSize = 0;
}
// CDIBSectionLite diagnostics
#ifdef _DEBUG
void CDIBSectionLite::AssertValid() const
{
    ASSERT(m_hBitmap);

    DIBSECTION ds;
    DWORD dwSize = GetObject( m_hBitmap, sizeof(DIBSECTION), &ds );
    ASSERT(dwSize == sizeof(DIBSECTION));

    ASSERT(0 <= m_iColorTableSize && m_iColorTableSize <= 256);
      CObject::AssertValid();
}
void CDIBSectionLite::Dump(CDumpContext& dc) const
{
      CObject::Dump(dc);
}
#endif //_DEBUG
// --- In  : bDither - whether or not dithering should be enabled
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Turns dithering on by using the DrawDib functions
instead of
//              the GDI functions
BOOL CDIBSectionLite::SetDither(BOOL bDither)
```

```
{
    BOOL bResult = TRUE;
    // Check for a no-change op.
    if ( (m_bDither == bDither) &&
         ((m_hDrawDib && bDither) || (!m_hDrawDib && !bDither)) )
        return bResult;
    if (m_hDrawDib != NULL)
    {
        DrawDibClose(m_hDrawDib);
        m_hDrawDib = NULL;
    }
    if (bDither)
    {
        if ( (m_hDrawDib = DrawDibOpen()) == NULL )
            bResult = FALSE;
    }
    m_bDither = (m_hDrawDib != NULL);
    return bResult;
}
// --- In  :
// --- Out :
// --- Returns : TRUE if dithering is used
// --- Effect : Returns whether or not the DrawDib functions (and
hence dithering)
//              is being used.
BOOL CDIBSectionLite::GetDither()
{
    return (m_bDither && (m_hDrawDib != NULL));
}
// CDIBSectionLite operations
// --- In  : pDC - Pointer to a device context
//           ptDest - point at which the topleft corner of the image
is drawn
//           bForceBackground - Specifies whether the palette is
forced to be
//                              a background palette
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Draws the image 1:1 on the device context
BOOL CDIBSectionLite::Draw(CDC* pDC, CPoint ptDest, BOOL
bForceBackground /*=FALSE*/ )
{
    if (!m_hBitmap)
        return FALSE;
    CSize size = GetSize();
    CPoint SrcOrigin = CPoint(0,0);
    BOOL bResult = FALSE;
    if (GetDither() && GetBitCount() >= 8)
    {
        DrawDibSetPalette( m_hDrawDib, (HPALETTE)m_Palette);
        DrawDibRealize( m_hDrawDib,  pDC->GetSafeHdc(),
bForceBackground);
        bResult = DrawDibDraw(m_hDrawDib, pDC->GetSafeHdc(),
                              ptDest.x, ptDest.y, size.cx, size.cy,
                              GetBitmapInfoHeader(), GetDIBits(),
                              SrcOrigin.x, SrcOrigin.y, size.cx,
size.cy,
                              0/*DDF_HALFTONE*/);
```

```
    }
    else
    {
        CPalette* pOldPalette = NULL;
        if (m_Palette.m_hObject && UsesPalette(pDC))
        {
            pOldPalette = pDC->SelectPalette(&m_Palette,
bForceBackground);
            pDC->RealizePalette();
        }
        bResult = SetDIBitsToDevice(pDC->GetSafeHdc(),
                                    ptDest.x, ptDest.y,
                                    size.cx, size.cy,
                                    SrcOrigin.x, SrcOrigin.y,
                                    SrcOrigin.y, size.cy -
                                    SrcOrigin.y,
                                    GetDIBits(), GetBitmapInfo(),
                                    m_iColorDataType);

        if (pOldPalette)
            pDC->SelectPalette(pOldPalette, FALSE);
    }
        return bResult;
}
// --- In  : pDC - Pointer to a device context
//           ptDest - point at which the topleft corner of the image
is drawn
//           size - size to stretch the image
//           bForceBackground - Specifies whether the palette is
forced to be
//                              a background palette
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Stretch draws the image to the desired size on the
device context
BOOL CDIBSectionLite::Stretch(CDC* pDC, CPoint ptDest, CSize size,
                              BOOL bForceBackground /*=FALSE*/)
{
    if (!m_hBitmap)
        return FALSE;
    CSize imagesize = GetSize();
    CPoint SrcOrigin = CPoint(0,0);
    BOOL bResult = FALSE;
    if (GetDither() && GetBitCount() >= 8)
    {
        DrawDibSetPalette( m_hDrawDib, (HPALETTE)m_Palette);
        DrawDibRealize( m_hDrawDib,  pDC->GetSafeHdc(),
bForceBackground);
        bResult = DrawDibDraw(m_hDrawDib, pDC->GetSafeHdc(),
                              ptDest.x, ptDest.y, size.cx, size.cy,
                              GetBitmapInfoHeader(), GetDIBits(),
                              SrcOrigin.x, SrcOrigin.y,
                              imagesize.cx, imagesize.cy,
                              0/*DDF_HALFTONE*/);
    }
    else
    {
        CPalette* pOldPalette = NULL;
```

```
        if (m_Palette.m_hObject && UsesPalette(pDC))
        {
            pOldPalette = pDC->SelectPalette(&m_Palette,
bForceBackground);
            pDC->RealizePalette();
        }

        pDC->SetStretchBltMode(COLORONCOLOR);
        bResult = StretchDIBits(pDC->GetSafeHdc(),
                                ptDest.x, ptDest.y,
                                size.cx, size.cy,
                                SrcOrigin.x, SrcOrigin.y,
                                imagesize.cx, imagesize.cy,
                                GetDIBits(), GetBitmapInfo(),
                                m_iColorDataType, SRCCOPY);
        if (pOldPalette)
            pDC->SelectPalette(pOldPalette, FALSE);
    }
    return bResult;
}
// Setting the bitmap...
// --- In  : nIDResource - resource ID
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Initialises the bitmap from a resource. If failure,
then object is
//                initialised back to an empty bitmap.
BOOL CDIBSectionLite::SetBitmap(UINT nIDResource)
{
    return SetBitmap(MAKEINTRESOURCE(nIDResource));
}
// --- In  : lpszResourceName - resource name
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Initialises the bitmap from a resource. If failure,
then object is
//                initialised back to an empty bitmap.
BOOL CDIBSectionLite::SetBitmap(LPCTSTR lpszResourceName)
{
    HBITMAP hBmp = (HBITMAP)::LoadImage(AfxGetResourceHandle(),
                                        lpszResourceName,
                                        IMAGE_BITMAP,
                                        0,0,
                                        LR_CREATEDIBSECTION);
    if (!hBmp) return FALSE;
    BOOL bResult = SetBitmap(hBmp);
    ::DeleteObject(hBmp);
    return bResult;
}

// --- In  : lpBitmapInfo - pointer to a BITMAPINFO structure
//           lpBits - pointer to image bits
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Initialises the bitmap using the information in
lpBitmapInfo to determine
//              the dimensions and colours, and the then sets the
bits from the bits in
```

```
//lpBits. If failure, then object is initialised back to an empty
bitmap.
BOOL CDIBSectionLite::SetBitmap(LPBITMAPINFO lpBitmapInfo, LPVOID
lpBits)
{
    DeleteObject();
    if (!lpBitmapInfo || !lpBits)
        return FALSE;
    HDC hDC = NULL;
    try {
        BITMAPINFOHEADER& bmih = lpBitmapInfo->bmiHeader;
        // Compute the number of colours in the colour table
        m_iColorTableSize = NumColorEntries(bmih.biBitCount);
        DWORD dwBitmapInfoSize = sizeof(BITMAPINFO) +
m_iColorTableSize*sizeof(RGBQUAD);
        // Copy over BITMAPINFO contents
        memcpy(&m_DIBinfo, lpBitmapInfo, dwBitmapInfoSize);
        // Should now have all the info we need to create the
sucker.
        //TRACE(_T("Width %d, Height %d, Bits/pixel %d, Image Size
%d\n"),
        //      bmih.biWidth, bmih.biHeight, bmih.biBitCount,
bmih.biSizeImage);
        // Create a DC which will be used to get DIB, then create
DIBsection
        hDC = ::GetDC(NULL);
        if (!hDC)
        {
            TRACE0("Unable to get DC\n");
            AfxThrowResourceException();
        }
        m_hBitmap = CreateDIBSection(hDC, (const BITMAPINFO*)
m_DIBinfo,
                                     m_iColorDataType, &m_ppvBits,
NULL, 0);
        ::ReleaseDC(NULL, hDC);
        if (!m_hBitmap)
        {
            TRACE0("CreateDIBSection failed\n");
            AfxThrowResourceException();
        }
        DWORD dwImageSize = m_DIBinfo.bmiHeader.biSizeImage;
        if (dwImageSize == 0)
        {
            int nBytesPerLine = BytesPerLine(lpBitmapInfo->
bmiHeader.biWidth,
                                             lpBitmapInfo->
bmiHeader.biBitCount);
            dwImageSize = nBytesPerLine * lpBitmapInfo->
bmiHeader.biHeight;
        }
        // Flush the GDI batch queue
        GdiFlush();
        memcpy(m_ppvBits, lpBits, dwImageSize);
        if (!CreatePalette())
        {
            TRACE0("Unable to create palette\n");
            AfxThrowResourceException();
```

```
        }
    }
    catch (CException *e)
    {
        e->Delete();
        _ShowLastError();
        if (hDC)
            ::ReleaseDC(NULL, hDC);
        DeleteObject();
        return FALSE;
    }
    return TRUE;
}
// --- In  : hBitmap - handle to image
//           pPalette - optional palette to use when setting image
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Initialises the bitmap from the HBITMAP supplied. If
failure, then
//              object is initialised back to an empty bitmap.
BOOL CDIBSectionLite::SetBitmap(HBITMAP hBitmap, CPalette* pPalette
/*= NULL*/)
{
    DeleteObject();
    if (!hBitmap)
        return FALSE;
    // Get dimensions of bitmap
    BITMAP bm;
    if (!::GetObject(hBitmap, sizeof(bm),(LPVOID)&bm))
        return FALSE;
    bm.bmHeight = abs(bm.bmHeight);
       CWindowDC dc(NULL);
    CPalette* pOldPalette = NULL;
    try {
        m_iColorTableSize = NumColorEntries(bm.bmBitsPixel);
        // Initialize the BITMAPINFOHEADER in m_DIBinfo
        BITMAPINFOHEADER& bih = m_DIBinfo.bmiHeader;
        bih.biSize          = sizeof(BITMAPINFOHEADER);
        bih.biWidth         = bm.bmWidth;
        bih.biHeight        = bm.bmHeight;
        bih.biPlanes        = 1;                    // Must always be 1
according to docs
        bih.biBitCount      = bm.bmBitsPixel;
        bih.biCompression   = BI_RGB;
        bih.biSizeImage     = BytesPerLine(bm.bmWidth,
bm.bmBitsPixel) * bm.bmHeight;
        bih.biXPelsPerMeter = 0;
        bih.biYPelsPerMeter = 0;
        bih.biClrUsed       = 0;
        bih.biClrImportant  = 0;
        // calls GetDIBits with NULL bits pointer to fill in the
BITMAPINFOHEADER data
        if (!::GetDIBits(dc.m_hDC, hBitmap, 0, bm.bmHeight, NULL,
m_DIBinfo, m_iColorDataType))
        {
            TRACE0("Unable to GetDIBits\n");
            AfxThrowResourceException();
        }
```

```
        // If the driver did not fill in the biSizeImage field, then
compute it
        // Each scan line of the image is aligned on a DWORD (32bit)
boundary
        if (bih.biSizeImage == 0)
            bih.biSizeImage = BytesPerLine(bih.biWidth,
bih.biBitCount) * bih.biHeight;
        if (pPalette)
            SetPalette(pPalette);
        else
            CreateHalftonePalette(m_Palette, m_iColorTableSize);

        if (m_Palette.GetSafeHandle())
        {
            pOldPalette = dc.SelectPalette(&m_Palette, FALSE);
            dc.RealizePalette();
        }
        // Create it!
        m_hBitmap = CreateDIBSection(dc.m_hDC,
                                     (const BITMAPINFO*) m_DIBinfo,
                                     m_iColorDataType,
                                     &m_ppvBits,
                                     NULL, 0);
        if (pOldPalette)
            dc.SelectPalette(pOldPalette, FALSE);
        pOldPalette = NULL;
        if (!m_hBitmap)
        {
            TRACE0("Unable to CreateDIBSection\n");
            AfxThrowResourceException();
        }
        // Need to copy the supplied bitmap onto the newly created
DIBsection
        CDC memDC, CopyDC;
        if (!CopyDC.CreateCompatibleDC(&dc) ||
!memDC.CreateCompatibleDC(&dc))
        {
            TRACE0("Unable to create compatible DC's\n");
            AfxThrowResourceException();
        }
        if (m_Palette.GetSafeHandle())
        {
            memDC.SelectPalette(&m_Palette, FALSE);
memDC.RealizePalette();
            CopyDC.SelectPalette(&m_Palette, FALSE);
CopyDC.RealizePalette();
        }
        GdiFlush();
        HBITMAP hOldMemBitmap  = (HBITMAP) SelectObject(memDC.m_hDC,
hBitmap);
        HBITMAP hOldCopyBitmap = (HBITMAP) SelectObject
(CopyDC.m_hDC, m_hBitmap);

        CopyDC.BitBlt(0,0, bm.bmWidth, bm.bmHeight, &memDC, 0,0,
SRCCOPY);
        SelectObject(memDC.m_hDC, hOldMemBitmap);
        SelectObject(CopyDC.m_hDC, hOldCopyBitmap);
        if (m_Palette.GetSafeHandle())
```

```
        {
            memDC.SelectStockObject(DEFAULT_PALETTE);
            CopyDC.SelectStockObject(DEFAULT_PALETTE);
        }
    }
    catch (CException *e)
    {
        e->Delete();
        _ShowLastError();
        if (pOldPalette)
            dc.SelectPalette(pOldPalette, FALSE);
        DeleteObject();
        return FALSE;
    }
    return TRUE;
}
// Persistance...
// --- In  : lpszFileName - image filename
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Loads the bitmap from a bitmap file with the name
lpszFileName.
// If failure, then object is initialised back to an empty bitmap.
BOOL CDIBSectionLite::Load(LPCTSTR lpszFileName)
{
    CFile file;
    if (!file.Open(lpszFileName, CFile::modeRead))
        return FALSE;
    // Get the current file position.
    DWORD dwFileStart = file.GetPosition();
    // The first part of the file contains the file header.
      // This will tell us if it is a bitmap, how big the header is,
and how big
    // the file is. The header size in the file header includes the
color table.
      BITMAPFILEHEADER BmpFileHdr;
    int nBytes;
    nBytes = file.Read(&BmpFileHdr, sizeof(BmpFileHdr));
    if (nBytes != sizeof(BmpFileHdr))
    {
        TRACE0("Failed to read file header\n");
        return FALSE;
    }

    // Check that we have the magic 'BM' at the start.
    if (BmpFileHdr.bfType != DS_BITMAP_FILEMARKER)
    {
        TRACE0("Not a bitmap file\n");
        return FALSE;
    }

    // Read the header (assuming it's a DIB).
      DIBINFO     BmpInfo;
    nBytes = file.Read(&BmpInfo, sizeof(BITMAPINFOHEADER));
    if (nBytes != sizeof(BITMAPINFOHEADER))
    {
        TRACE0("Failed to read BITMAPINFOHEADER\n");
        return FALSE;
```

```
    }
    // Check that we have a real Windows DIB file.
    if (BmpInfo.bmiHeader.biSize != sizeof(BITMAPINFOHEADER))
    {
        TRACE0(" File is not a Windows DIB\n");
        return FALSE;
    }
    // See how big the color table is in the file (if there is one).
    int nColors = NumColorEntries(BmpInfo.bmiHeader.biBitCount);
      if (nColors > 0)
    {
        // Read the color table from the file.
        int nColorTableSize = nColors * sizeof(RGBQUAD);
            nBytes = file.Read(BmpInfo.ColorTable(),
nColorTableSize);
            if (nBytes != nColorTableSize)
        {
                TRACE0("Failed to read color table\n");
                return FALSE;
        }
      }
      // So how big the bitmap surface is.
    int nBitsSize = BmpFileHdr.bfSize - BmpFileHdr.bfOffBits;

    // Allocate the memory for the bits and read the bits from the
file.
    BYTE* pBits = (BYTE*) malloc(nBitsSize);
    if (!pBits)
    {
        TRACE0("Out of memory for DIB bits\n");
        return FALSE;
    }
    // Seek to the bits in the file.
    file.Seek(dwFileStart + BmpFileHdr.bfOffBits, CFile::begin);
    // read the bits
    nBytes = file.Read(pBits, nBitsSize);
    if (nBytes != nBitsSize)
    {
        TRACE0("Failed to read bits\n");
        free(pBits);
            return FALSE;
    }
      // Everything went OK.
      BmpInfo.bmiHeader.biSizeImage = nBitsSize;

    if (!SetBitmap((LPBITMAPINFO) BmpInfo, pBits))
    {
        TRACE0("Failed to set bitmap info\n");
        free(pBits);
            return FALSE;
    }
    free(pBits);
    return TRUE;
}
// --- In  : lpszFileName - image filename
// --- Out :
// --- Returns : Returns TRUE on success, FALSE otherwise
// --- Effect : Saves the image to file.
```

```
BOOL CDIBSectionLite::Save(LPCTSTR lpszFileName)
{
    BITMAPFILEHEADER   hdr;
    LPBITMAPINFOHEADER lpbi = GetBitmapInfoHeader();
    if (!lpbi || !lpszFileName)
        return FALSE;
    CFile file;
    if (!file.Open(lpszFileName,
CFile::modeWrite|CFile::modeCreate))
        return FALSE;
    DWORD dwBitmapInfoSize = sizeof(BITMAPINFO) +
m_iColorTableSize*sizeof(RGBQUAD);
    DWORD dwFileHeaderSize = dwBitmapInfoSize + sizeof(hdr);
    // Fill in the fields of the file header
    hdr.bfType        = DS_BITMAP_FILEMARKER;
    hdr.bfSize        = dwFileHeaderSize + lpbi->biSizeImage;
    hdr.bfReserved1   = 0;
    hdr.bfReserved2   = 0;
    hdr.bfOffBits     = dwFileHeaderSize;
    // Write the file header
    file.Write(&hdr, sizeof(hdr));
    // Write the DIB header
    file.Write(lpbi, dwBitmapInfoSize);
    // Write DIB bits
    file.Write(GetDIBits(), lpbi->biSizeImage);
    return TRUE;
}
// CDIBSectionLite palette stuff
// --- In  :
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Creates the palette from the DIBSection's color
table. Assumes
//              m_iColorTableSize has been set and the DIBsection
m_hBitmap created
BOOL CDIBSectionLite::CreatePalette()
{
    m_Palette.DeleteObject();
    if (!m_hBitmap)
        return FALSE;
    // Create a 256 color halftone palette if there is no color
table in the DIBSection
    if (m_iColorTableSize == 0)
        return CreateHalftonePalette(m_Palette, 256);
    HDC hDC = ::GetDC(NULL);
    if (!hDC)
        return FALSE;
    // Get space for the colour entries
    RGBQUAD *pRGB = new RGBQUAD[m_iColorTableSize];
    if (!pRGB)
    {
        ReleaseDC(NULL, hDC);
        return CreateHalftonePalette(m_Palette, m_iColorTableSize);
    }
    // Create a memory DC compatible with the current DC
    CDC MemDC;
    MemDC.CreateCompatibleDC(CDC::FromHandle(hDC));
    if (!MemDC.GetSafeHdc())
```

```
    {
        delete [] pRGB;
        ::ReleaseDC(NULL, hDC);
        return CreateHalftonePalette(m_Palette, m_iColorTableSize);
    }
    ::ReleaseDC(NULL, hDC);
   HBITMAP hOldBitmap = (HBITMAP) ::SelectObject(MemDC.GetSafeHdc(),
m_hBitmap);
    if (!hOldBitmap)
    {
        delete [] pRGB;
        return CreateHalftonePalette(m_Palette, m_iColorTableSize);
    }
    // Get the colours used.
    int nColours = ::GetDIBColorTable(MemDC.GetSafeHdc(), 0,
m_iColorTableSize, pRGB);
    // Clean up
    ::SelectObject(MemDC.GetSafeHdc(), hOldBitmap);
    if (!nColours)   // No colours retrieved => the bitmap in the DC
is not a DIB section
    {
        delete [] pRGB;
        return CreateHalftonePalette(m_Palette, m_iColorTableSize);
    }

    // Create and fill a LOGPALETTE structure with the colours used.
    PALETTEINFO PaletteInfo;
    PaletteInfo.palNumEntries = (WORD) m_iColorTableSize;
    for (int i = 0; i < nColours; i++)
    {
        PaletteInfo.palPalEntry[i].peRed   = pRGB[i].rgbRed;
        PaletteInfo.palPalEntry[i].peGreen = pRGB[i].rgbGreen;
        PaletteInfo.palPalEntry[i].peBlue  = pRGB[i].rgbBlue;
        PaletteInfo.palPalEntry[i].peFlags = 0;
    }
    for (; i < (int) m_iColorTableSize; i++)
    {
        PaletteInfo.palPalEntry[i].peRed   = 0;
        PaletteInfo.palPalEntry[i].peGreen = 0;
        PaletteInfo.palPalEntry[i].peBlue  = 0;
        PaletteInfo.palPalEntry[i].peFlags = 0;
    }
    delete [] pRGB;
    // Create Palette!
    return m_Palette.CreatePalette(&PaletteInfo);
}
// --- In  : pPalette - new palette to use
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Sets the current palette used by the image from the
supplied CPalette,
//              and sets the color table in the DIBSection
BOOL CDIBSectionLite::SetPalette(CPalette* pPalette)
{
    m_Palette.DeleteObject();
    if (!pPalette)
        return FALSE;
```

```
    UINT nColours = pPalette->GetEntryCount();
    if (nColours <= 0 || nColours > 256)
        return FALSE;
    // Get palette entries
    PALETTEINFO pi;
    pi.palNumEntries = (WORD) pPalette->GetPaletteEntries(0,
nColours, (LPPALETTEENTRY) pi);
    // TODO: If pi.palNumEntries < m_iColorTableSize, then fill in
blanks with 0's
    return SetLogPalette(&pi);
}
// --- In  : pLogPalette - new palette to use
// --- Out :
// --- Returns : TRUE on success
// --- Effect : Sets the current palette used by the image from the
supplied LOGPALETTE
BOOL CDIBSectionLite::SetLogPalette(LOGPALETTE* pLogPalette)
{
    ASSERT(pLogPalette->palVersion == (WORD) 0x300);
    UINT nColours = pLogPalette->palNumEntries;
    if (nColours <= 0 || nColours > 256)
    {
        CreatePalette();
        return FALSE;
    }
    // Create new palette
    m_Palette.DeleteObject();
    if (!m_Palette.CreatePalette(pLogPalette))
    {
        CreatePalette();
        return FALSE;
    }
    if (m_iColorTableSize == 0)
        return TRUE;
    // Set the DIB colours
    RGBQUAD RGBquads[256];
    for (UINT i = 0; i < nColours; i++)
    {
        RGBquads[i].rgbRed   = pLogPalette->palPalEntry[i].peRed;
        RGBquads[i].rgbGreen = pLogPalette->palPalEntry[i].peGreen;
        RGBquads[i].rgbBlue  = pLogPalette->palPalEntry[i].peBlue;
        RGBquads[i].rgbReserved = 0;
    }
    return FillDIBColorTable(nColours, RGBquads);
}
// --- In  : nNumColours - number of colours to set
//           pRGB - colours to fill
// --- Out :
// --- Returns : Returns TRUE on success
// --- Effect : Sets the colours used by the image. Only works if #
colours <= 256
BOOL CDIBSectionLite::FillDIBColorTable(UINT nNumColours, RGBQUAD
*pRGB)
{
    if (!m_hBitmap || !pRGB || !nNumColours || !m_iColorTableSize
        || nNumColours > 256)
        return FALSE;
```

```
    // Create a memory DC compatible with the screen
    HDC hDC = ::GetDC(NULL);
    if (!hDC) return FALSE;
    CDC MemDC;
    MemDC.CreateCompatibleDC(CDC::FromHandle(hDC));
    ::ReleaseDC(NULL, hDC);
    if (!MemDC.GetSafeHdc())
        return FALSE;
    HBITMAP hOldBitmap = (HBITMAP) ::SelectObject(MemDC.GetSafeHdc
(), m_hBitmap);
    // Set the bitmap colours.
    int nColours = ::SetDIBColorTable(MemDC.GetSafeHdc(), 0,
nNumColours, pRGB);
    // Clean up
    if (hOldBitmap)
        ::SelectObject(MemDC.GetSafeHdc(), hOldBitmap);
    MemDC.DeleteDC();
    return (nColours > 0);
}
#ifdef _DEBUG
// Makes trace windows a little bit more informative...
void CDIBSectionLite::_ShowLastError()
{
    LPVOID lpMsgBuf;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,
                  NULL, GetLastError(), MAKELANGID(LANG_NEUTRAL,
SUBLANG_DEFAULT),
                  (LPTSTR) &lpMsgBuf, 0, NULL);
    TRACE1("Last error: %s\n", lpMsgBuf);
    LocalFree(lpMsgBuf);
}
#else
void CDIBSectionLite::_ShowLastError() {}
#endif
```

Paper.h

```
#if
!defined(AFX_PAPER_H__C74A42E1_D291_11D3_8AFC_102749C10001__INCLUDED
_)
#define AFX_PAPER_H__C74A42E1_D291_11D3_8AFC_102749C10001__INCLUDED_
#include "Frame.h"
#include <afxtempl.h>
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Paper.h : header file
// CPaper window
class CPaper : public CStatic
{
// Construction
public:
      CPaper();
      CPaper(CString FileName);
// Attributes
public:
      CList <CFrame*, CFrame*> m_FrameList;
```

```
      CList <HBITMAP, HBITMAP> m_ActBitmap;
      POSITION m_Pos;
      int m_FrameCount;
      int m_ActCount;
      CSize m_Size;
      CBitmap m_Paper;
      CString m_ScriptName;
      CreateFrame();
      DestroyFrame();
      DrawPaper(HDC hDC,UINT x,UINT y,UINT w,UINT h);
      SetFrameAction();
      HBITMAP GetImageAt(int i);
// Operations
public:
// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CPaper)
      public:
      virtual BOOL Create(LPCTSTR lpszClassName, LPCTSTR
lpszWindowName, DWORD dwStyle, const RECT& rect, CWnd* pParentWnd,
UINT nID, CCreateContext* pContext = NULL);
      //}}AFX_VIRTUAL
// Implementation
public:
      virtual ~CPaper();
      // Generated message map functions
protected:
      //{{AFX_MSG(CPaper)
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_PAPER_H__C74A42E1_D291_11D3_8AFC_102749C10001__INCLUDED
_)
```

Paper.cpp

```
// Paper.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "Paper.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// CPaper
CPaper::CPaper()
{
      m_ScriptName.Empty();
}
CPaper::CPaper(CString FileName)
{
```

```
	m_ScriptName = FileName;
	CreateFrame();
	//DestroyFrame();
}
CPaper::~CPaper()
{
}
BEGIN_MESSAGE_MAP(CPaper, CStatic)
	//{{AFX_MSG_MAP(CPaper)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CPaper message handlers
CPaper::CreateFrame()
{
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	pApp->m_pszProfileName=_tcsdup(_T(m_ScriptName));
	m_Size.cx = pApp->GetProfileInt("Paper","Width",0);
	m_Size.cy = pApp->GetProfileInt("Paper","Height",0);
	HDC dc = CreateDC("DISPLAY",NULL,NULL,NULL);
	CDC DC;
	DC.Attach(dc);
	m_Paper.CreateCompatibleBitmap(&DC, m_Size.cx, m_Size.cy);
	DeleteDC(dc);
	DC.DeleteDC();
	m_FrameCount = pApp->GetProfileInt("Paper","Frame",0);
	m_ActCount = pApp->GetProfileInt("Paper","Action",0);
	CFrame *Frame;
	CRect rect;
	CString framefilename,section,framepath;
	UINT colorref,action;
	HBITMAP hBitmap;
	for (int i=0;i<m_FrameCount;i++)
	{
		section.Format("Frame%d",i+1);
		rect.left = pApp->GetProfileInt(LPCTSTR(section),"X",0);
		rect.top = pApp->GetProfileInt(LPCTSTR(section),"Y",0);
		rect.right = pApp-
>GetProfileInt(LPCTSTR(section),"Width",0);
		rect.bottom = pApp->GetProfileInt(LPCTSTR
(section),"Height",0);
		framefilename = pApp->GetProfileString(LPCTSTR
(section),"File");
		colorref = pApp-
>GetProfileInt(LPCTSTR(section),"ColorRef",0);
		action = pApp-
>GetProfileInt(LPCTSTR(section),"Action",0);
		framepath.Format("%s\\%s\\%s",pApp->GetFrameStylePath
(),pApp->GetFrameFileName(pApp->GetFrameSelect()),framefilename);
		hBitmap = ::LoadBitmapFile(framepath.GetBuffer(0));
		Frame = new CFrame();
		Frame->Create(NULL,NULL,SS_BLACKFRAME,rect, this,NULL);
		Frame->SetFrameRect(rect);
		Frame->SetColorRef(colorref);
		Frame->SetFrame(hBitmap);
		Frame->SetAction(action);
		m_FrameList.AddTail(Frame);
```

```
	}
	m_Pos = m_FrameList.GetHeadPosition();
}
CPaper::DestroyFrame()
{
	m_Pos = m_FrameList.GetHeadPosition();
	CFrame *Frame;
	while (!m_FrameList.IsEmpty())
	{
		Frame = m_FrameList.GetHead();
		DeleteObject(Frame);
		m_FrameList.RemoveHead();
		delete Frame;
	}
}
CPaper::DrawPaper(HDC hDC, UINT x, UINT y, UINT w, UINT h)
{
	CFrame *Frame;
	m_Pos = m_FrameList.GetHeadPosition();
	HDC hdc;
	hdc = CreateCompatibleDC(hDC);
	SelectObject(hdc, HBITMAP(m_Paper));
	SetMapMode(hdc, GetMapMode(hDC));
	for (int i=0;i<m_FrameCount;i++)
	{
		Frame = m_FrameList.GetAt(m_Pos);
		//Frame->DrawFrame(::GetWindowDC(m_hWnd));
		Frame->DrawFrame(hdc);
		m_FrameList.GetNext(m_Pos);
	}
	DeleteDC(hdc);
	m_Pos = m_FrameList.GetHeadPosition();
	//BitBlt(hDC,0,0,100,100,::GetWindowDC(m_hWnd),0,0,SRCCOPY);
	HBITMAP bitmap = HBITMAP(m_Paper);
	//SetClipboardData(CF_BITMAP, bitmap);

	DrawStretchBitmap(hDC, bitmap, x, y, w, h,SRCCOPY);
	//DrawTransparentBitmap(hDC, bitmap, 0, 0, , ,
}
CPaper::SetFrameAction()
{
	CFrame *Frame;
	m_Pos = m_FrameList.GetHeadPosition();
	for (int i=1;i<=m_FrameCount;i++)
	{
		CString Section;
		Section.Format("Frame%d",i);
		Frame = m_FrameList.GetAt(m_Pos);
		Frame->SetImage(GetImageAt(Frame->GetAction()));
		m_FrameList.GetNext(m_Pos);
	}
	m_Pos = m_FrameList.GetHeadPosition();
}
HBITMAP CPaper::GetImageAt(int i)
{
	POSITION pos;
	pos = m_ActBitmap.GetHeadPosition();
```

```
      for (int j=1;j<i;j++)
      {
            m_ActBitmap.GetNext(pos);
      }
      return m_ActBitmap.GetAt(pos);
}
BOOL CPaper::Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName,
DWORD dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID,
CCreateContext* pContext)
{
      // TODO: Add your specialized code here and/or call the base
class
      return CWnd::Create(lpszClassName, lpszWindowName, dwStyle,
rect, pParentWnd, nID, pContext);
}
```

Frame.h

```
#if
!defined(AFX_FRAME_H__D1635245_D245_11D3_8AFC_80614AC10801__INCLUDED
_)
#define AFX_FRAME_H__D1635245_D245_11D3_8AFC_80614AC10801__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Frame.h : header file
//////////////////////////////////////////////////////////////////////
// CFrame window
class CFrame : public CStatic
{
// Construction
public:
      CFrame();
// Attributes
public:
      SetImage(HBITMAP hBitmap);
      SetFrame(HBITMAP hBitmap);
      SetFrameRect(CRect FrameRect);
      SetColorRef(COLORREF ColorRef);
      SetAction(int Action);
      HBITMAP           GetImage();
      HBITMAP           GetFrame();
      CRect       GetFrameRect();
      COLORREF    GetColorRef();
      int               GetAction();
      DrawFrame(HDC hDC);
// Operations
public:
// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CFrame)
      public:
      virtual BOOL Create(LPCTSTR lpszClassName, LPCTSTR
lpszWindowName, DWORD dwStyle, const RECT& rect, CWnd* pParentWnd,
UINT nID, CCreateContext* pContext = NULL);
      //}}AFX_VIRTUAL
// Implementation
public:
```

```
      virtual ~CFrame();
      // Generated message map functions
protected:
      HBITMAP           m_hImage;
      HBITMAP           m_hFrame;
      CRect       m_FrameRect;
      COLORREF    m_ColorRef;
      int               m_Action;
      //{{AFX_MSG(CFrame)
      afx_msg void OnPaint();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_FRAME_H__D1635245_D245_11D3_8AFC_80614AC10801__INCLUDED
_)
```

Frame.cpp

```
// Frame.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "Frame.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CFrame
CFrame::CFrame()
{
      m_hImage = NULL;
      m_hFrame = NULL;
      m_FrameRect.SetRectEmpty();
      m_ColorRef = NULL;
}
CFrame::~CFrame()
{
}
BEGIN_MESSAGE_MAP(CFrame, CStatic)
      //{{AFX_MSG_MAP(CFrame)
      ON_WM_PAINT()
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CFrame message handlers
HBITMAP CFrame::GetFrame()
{
      return m_hFrame;
}
HBITMAP CFrame::GetImage()
{
      return m_hImage;
```

```
}
CRect CFrame::GetFrameRect()
{
	return m_FrameRect;
}
int CFrame::GetAction()
{
	return m_Action;
}
COLORREF CFrame::GetColorRef()
{
	return m_ColorRef;
}
CFrame::SetFrame(HBITMAP hBitmap)
{
	m_hFrame = hBitmap;
}
CFrame::SetImage(HBITMAP hBitmap)
{
	m_hImage = hBitmap;
}
CFrame::SetFrameRect(CRect FrameRect)
{
	m_FrameRect.CopyRect(LPCRECT(FrameRect));
	/*RECT rect;
	rect.bottom = m_FrameRect.bottom;
	rect.left = m_FrameRect.left;
	rect.right = m_FrameRect.right;
	rect.top = m_FrameRect.top;
	Create(NULL, SS_BLACKFRAME, rect,NULL);*/
}
CFrame::SetColorRef(COLORREF ColorRef)
{
	m_ColorRef = ColorRef;
}
CFrame::DrawFrame(HDC hDC)
{
	::DrawStretchBitmap(hDC, m_hImage, m_FrameRect.left,
		m_FrameRect.top, m_FrameRect.right, m_FrameRect.bottom);
/*	::DrawTransparentBitmap(hDC, m_hImage, m_FrameRect.left,
		m_FrameRect.top, m_FrameRect.right, m_FrameRect.bottom,
m_ColorRef);*/
	::DrawTransparentBitmap(hDC, m_hFrame, m_FrameRect.left,
		m_FrameRect.top, m_FrameRect.right, m_FrameRect.bottom,
m_ColorRef);
	/*::DrawBitmap(hDC, m_hFrame, m_FrameRect.left,
		m_FrameRect.top, m_FrameRect.Width(), m_FrameRect.Height
());*/
}
CFrame::SetAction(int Action)
{
	m_Action = Action;
}
void CFrame::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	// TODO: Add your message handler code here
	DrawFrame(dc.m_hDC);
```

```
      // Do not call CStatic::OnPaint() for painting messages
}
BOOL CFrame::Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName,
DWORD dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID,
CCreateContext* pContext)
{
      // TODO: Add your specialized code here and/or call the base
class
      return CWnd::Create(lpszClassName, lpszWindowName, dwStyle,
rect, pParentWnd, nID, pContext);
}
```

NewOpenDlg.h

```
#if
!defined(AFX_NEWOPENDLG_H__42C41C81_D16F_11D3_8AFC_00634AC10801__INC
LUDED_)
#define
AFX_NEWOPENDLG_H__42C41C81_D16F_11D3_8AFC_00634AC10801__INCLUDED_
#include "MyDialog.h"
#include "MyButton.h"
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// NewOpenDlg.h : header file
//////////////////////////////////////////////////////////////////////
// CNewOpenDlg dialog
class CNewOpenDlg : public CMyDialog
{
// Construction
public:
      CNewOpenDlg(CWnd* pParent = NULL);   // standard constructor
// Dialog Data
      //{{AFX_DATA(CNewOpenDlg)
      enum { IDD = IDD_NEW_OPEN_DIALOG };
      CMyButton   m_OpenButton;
      CMyButton   m_NewButton;
      //}}AFX_DATA


// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CNewOpenDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
      //}}AFX_VIRTUAL

// Implementation
protected:
      // Generated message map functions
      //{{AFX_MSG(CNewOpenDlg)
      virtual BOOL OnInitDialog();
      afx_msg void OnNewButton();
      afx_msg void OnOpenButton();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_NEWOPENDLG_H__42C41C81_D16F_11D3_8AFC_00634AC10801__INC
LUDED_)
```

NewOpenDlg.cpp

```
// NewOpenDlg.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "NewOpenDlg.h"
#include "StickerDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// CNewOpenDlg dialog
CNewOpenDlg::CNewOpenDlg(CWnd* pParent /*=NULL*/)
      : CMyDialog(CNewOpenDlg::IDD, pParent)
{
      //{{AFX_DATA_INIT(CNewOpenDlg)
            // NOTE: the ClassWizard will add member initialization
here
      //}}AFX_DATA_INIT
}
void CNewOpenDlg::DoDataExchange(CDataExchange* pDX)
{
      CMyDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CNewOpenDlg)
      DDX_Control(pDX, IDC_OPEN_BUTTON, m_OpenButton);
      DDX_Control(pDX, IDC_NEW_BUTTON, m_NewButton);
      //}}AFX_DATA_MAP
BEGIN_MESSAGE_MAP(CNewOpenDlg, CMyDialog)
      //{{AFX_MSG_MAP(CNewOpenDlg)
      ON_BN_CLICKED(IDC_NEW_BUTTON, OnNewButton)
      ON_BN_CLICKED(IDC_OPEN_BUTTON, OnOpenButton)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////
// CNewOpenDlg message handlers
BOOL CNewOpenDlg::OnInitDialog()
{
      CDialog::OnInitDialog();
      // TODO: Add extra initialization here
      CStickerApp *pApp = (CStickerApp*) AfxGetApp();
      //for bitmap
}
      HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;
hButtonAll = ::LoadBitmapFile(pApp-
>New_Open_Dialog_New_Button.GetBuffer(0));
      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->New_Open_Dialog_New_Width, pApp->
      New_Open_Dialog_New_Height);//-1);
```

```
hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->New_Open_Dialog_New_Height, pApp->
New_Open_Dialog_New_Width, pApp->New_Open_Dialog_New_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->New_Open_Dialog_New_Height*2, pApp->
New_Open_Dialog_New_Width, pApp->New_Open_Dialog_New_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->New_Open_Dialog_New_Height*3, pApp->
New_Open_Dialog_New_Width, pApp->New_Open_Dialog_New_Height*4-1);
      m_NewButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_NewButton.SizeToContent();
      hButtonAll = ::LoadBitmapFile(pApp-
>New_Open_Dialog_Open_Button.GetBuffer(0));
      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->New_Open_Dialog_Open_Width, pApp->
New_Open_Dialog_Open_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->New_Open_Dialog_Open_Height, pApp->
New_Open_Dialog_Open_Width, pApp->New_Open_Dialog_Open_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->New_Open_Dialog_Open_Height*2, pApp->
New_Open_Dialog_Open_Width, pApp->New_Open_Dialog_Open_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->New_Open_Dialog_Open_Height*3, pApp->
New_Open_Dialog_Open_Width, pApp->New_Open_Dialog_Open_Height*4-1);
      m_OpenButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_OpenButton.SizeToContent();
      //for cursor
      m_NewButton.SetCursor(::LoadCursorFromFile(pApp->
New_Open_Dialog_New_Cur.GetBuffer(0)));
      m_OpenButton.SetCursor(::LoadCursorFromFile(pApp->
New_Open_Dialog_Open_Cur.GetBuffer(0)));
      SetCursor(::LoadCursorFromFile(pApp-
>New_Open_Dialog_Cur.GetBuffer(0)));
      //for sound
      m_NewButton.SetSound(IDR_DEFAULT_SOUND);
      m_OpenButton.SetSound(IDR_DEFAULT_SOUND);

      m_NewButton.MoveWindow(pApp->New_Open_Dialog_New_X,
                              pApp->New_Open_Dialog_New_Y,
                              pApp->New_Open_Dialog_New_Width,
                              pApp-
>New_Open_Dialog_New_Height);
      m_OpenButton.MoveWindow(pApp->New_Open_Dialog_Open_X,
                              pApp->New_Open_Dialog_Open_Y,
                              pApp-
>New_Open_Dialog_Open_Width,
                              pApp-
>New_Open_Dialog_Open_Height);
      HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>New_Open_Dialog_Background.GetBuffer(0));
      SetBitmap(hBitmap);
      SetSound();//IDR_NEW_OPEN_DLG);
      SetMidi();//IDR_DEFAULT_MIDI);
```

```
	ShowWindow(FALSE);
	MoveWindow(0,0,pApp->New_Open_Dialog_Background_Width,pApp->
New_Open_Dialog_Background_Height);
	return TRUE;  // return TRUE unless you set the focus to a
control
		// EXCEPTION: OCX Property Pages should return
FALSE
}
void CNewOpenDlg::OnNewButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	pApp->m_New = TRUE;
	ShowWindow(FALSE);
}
void CNewOpenDlg::OnOpenButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	pApp->m_New = FALSE;
	pApp->FindSticker();
	ShowWindow(FALSE);
	CStickerDlg *dlg = (CStickerDlg*) GetParent();
	dlg->m_OpenDlg.ShowWindow(TRUE);
	//CDC *dc = dlg->m_OpenDlg.m_Preview.GetDC();
	POSITION pos = pApp->m_StickerList.GetHeadPosition();
	dlg->m_OpenDlg.m_Pos = pos;
	//HBITMAP hbitmap = pApp->m_StickerList.GetAt(pos);
	//dlg->m_OpenDlg.m_Preview.SetBitmap(hbitmap);
	//DrawStretchBitmap(dc->m_hDC, hbitmap,0,0,0,0,SRCCOPY);
}
```

OpenDlg.h

```
#if
!defined(AFX_OPENDLG_H__18E2D683_1EFF_11D4_881E_A0234AC1010C__INCLUD
ED_)
#define
AFX_OPENDLG_H__18E2D683_1EFF_11D4_881E_A0234AC1010C__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// OpenDlg.h : header file
#include "MyDialog.h"
#include "MyButton.h"
/////////////////////////////////////////////////////////////////////////
// COpenDlg dialog
class COpenDlg : public CMyDialog
{
// Construction
public:
	COpenDlg(CWnd* pParent = NULL);   // standard constructor
	UINT m_Count;
	POSITION m_Pos;
// Dialog Data
	//{{AFX_DATA(COpenDlg)
	enum { IDD = IDD_OPEN_DIALOG };
	CStatic	m_Preview;
```

```
	CMyButton	m_NextButton;
	CMyButton	m_DoneButton;
	CMyButton	m_BackButton;
	//}}AFX_DATA
// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(COpenDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV
support
	//}}AFX_VIRTUAL
// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(COpenDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
	afx_msg void OnOpenNextButton();
	afx_msg void OnOpenBackButton();
	afx_msg void OnOpenDoneButton();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_OPENDLG_H__18E2D683_1EFF_11D4_881E_A0234AC1010C__INCLUD
ED_)
```

OpenDlg.cpp

```
// OpenDlg.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "OpenDlg.h"
#include "StickerDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// COpenDlg dialog
COpenDlg::COpenDlg(CWnd* pParent /*=NULL*/)
	: CMyDialog(COpenDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(COpenDlg)
	// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}
void COpenDlg::DoDataExchange(CDataExchange* pDX)
{
	CMyDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(COpenDlg)
	DDX_Control(pDX, IDC_OPEN_PREVIEW, m_Preview);
	DDX_Control(pDX, IDC_OPEN_NEXT_BUTTON, m_NextButton);
	DDX_Control(pDX, IDC_OPEN_DONE_BUTTON, m_DoneButton);
```

```
	DDX_Control(pDX, IDC_OPEN_BACK_BUTTON, m_BackButton);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(COpenDlg, CMyDialog)
	//{{AFX_MSG_MAP(COpenDlg)
	ON_WM_SHOWWINDOW()
	ON_BN_CLICKED(IDC_OPEN_NEXT_BUTTON, OnOpenNextButton)
	ON_BN_CLICKED(IDC_OPEN_BACK_BUTTON, OnOpenBackButton)
	ON_BN_CLICKED(IDC_OPEN_DONE_BUTTON, OnOpenDoneButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// COpenDlg message handlers
BOOL COpenDlg::OnInitDialog()
{
	CMyDialog::OnInitDialog();
	// TODO: Add extra initialization here
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	//for bitmap
	HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;
	hButtonAll = ::LoadBitmapFile(pApp-
>Open_Dialog_Next_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Open_Dialog_Next_Width, pApp->
Open_Dialog_Next_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Open_Dialog_Next_Height, pApp-
>Open_Dialog_Next_Width, pApp->Open_Dialog_Next_Height*2-1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Open_Dialog_Next_Height*2, pApp->
Open_Dialog_Next_Width, pApp->Open_Dialog_Next_Height*3-1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
		0, pApp->Open_Dialog_Next_Height*3, pApp->
Open_Dialog_Next_Width, pApp->Open_Dialog_Next_Height*4-1);
	m_NextButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
	m_NextButton.SizeToContent();
	hButtonAll = ::LoadBitmapFile(pApp-
>Open_Dialog_Back_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Open_Dialog_Back_Width, pApp->
Open_Dialog_Back_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Open_Dialog_Back_Height, pApp-
>Open_Dialog_Back_Width, pApp->Open_Dialog_Back_Height*2-1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Open_Dialog_Back_Height*2, pApp->
Open_Dialog_Back_Width, pApp->Open_Dialog_Back_Height*3-1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
		0, pApp->Open_Dialog_Back_Height*3, pApp->
Open_Dialog_Back_Width, pApp->Open_Dialog_Back_Height*4-1);
	m_BackButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
	m_BackButton.SizeToContent();
```

```
      hButtonAll = ::LoadBitmapFile(pApp-
>Open_Dialog_Done_Button.GetBuffer(0));
      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Open_Dialog_Done_Width, pApp->
Open_Dialog_Done_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Open_Dialog_Done_Height, pApp-
>Open_Dialog_Done_Width, pApp->Open_Dialog_Done_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Open_Dialog_Done_Height*2, pApp->
Open_Dialog_Done_Width, pApp->Open_Dialog_Done_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Open_Dialog_Done_Height*3, pApp->
Open_Dialog_Done_Width, pApp->Open_Dialog_Done_Height*4-1);
      m_DoneButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_DoneButton.SizeToContent();
            //for cursor
      m_NextButton.SetCursor(::LoadCursorFromFile(pApp->
Open_Dialog_Next_Cur.GetBuffer(0)));
      m_BackButton.SetCursor(::LoadCursorFromFile(pApp->
Open_Dialog_Back_Cur.GetBuffer(0)));
      m_DoneButton.SetCursor(::LoadCursorFromFile(pApp->
Open_Dialog_Done_Cur.GetBuffer(0)));
      SetCursor(::LoadCursorFromFile(pApp->Open_Dialog_Cur.GetBuffer
(0)));
      //for sound
      m_NextButton.SetSound(IDR_DEFAULT_SOUND);
      m_BackButton.SetSound(IDR_DEFAULT_SOUND);
      m_DoneButton.SetSound(IDR_DEFAULT_SOUND);
      m_NextButton.MoveWindow(pApp->Open_Dialog_Next_X,
                              pApp->Open_Dialog_Next_Y,
                              pApp->Open_Dialog_Next_Width,
                              pApp->Open_Dialog_Next_Height);
      m_BackButton.MoveWindow(pApp->Open_Dialog_Back_X,
                              pApp->Open_Dialog_Back_Y,
                              pApp->Open_Dialog_Back_Width,
                              pApp->Open_Dialog_Back_Height);
      m_DoneButton.MoveWindow(pApp->Open_Dialog_Done_X,
                              pApp->Open_Dialog_Done_Y,
                              pApp->Open_Dialog_Done_Width,
                              pApp->Open_Dialog_Done_Height);
      HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>Open_Dialog_Background.GetBuffer(0));
      SetBitmap(hBitmap);
      SetSound();//IDR_NEW_OPEN_DLG);
      SetMidi();//IDR_DEFAULT_MIDI);
      ShowWindow(FALSE);
      int width = pApp->Open_Dialog_Background_Width;
      int height = pApp->Open_Dialog_Background_Height;
      MoveWindow(0,0, width, height);
      m_BackButton.EnableWindow(FALSE);
      m_Preview.MoveWindow(pApp->Select_Frame_Dialog_Preview_X,
            pApp->Select_Frame_Dialog_Preview_Y,
            pApp->Select_Frame_Dialog_Preview_Width,
            pApp->Select_Frame_Dialog_Preview_Height);
```

```
	return TRUE;  // return TRUE unless you set the focus to a
control
                  // EXCEPTION: OCX Property Pages should return
FALSE
}
void COpenDlg::OnShowWindow(BOOL bShow, UINT nStatus)
{
	CMyDialog::OnShowWindow(bShow, nStatus);
	// TODO: Add your message handler code here
	if (bShow)
	{
		CStickerApp *pApp = (CStickerApp*) AfxGetApp();
		m_Pos = pApp->m_StickerList.GetHeadPosition();
		HBITMAP bitmap = pApp->m_StickerList.GetAt(m_Pos);
		m_Count = 1;
		//m_Preview.SetBitmap(bitmap);
	//::DrawStretchBitmap(::GetDC(m_Preview.m_hWnd),bitmap,0,0,100
,100,SRCCOPY);
	}
}
void COpenDlg::OnOpenNextButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	if (m_Count<pApp->GetOpenCount())
	{
		pApp->m_StickerList.GetNext(m_Pos);
		HBITMAP bitmap = pApp->m_StickerList.GetAt(m_Pos);
		RECT rect;
		m_Preview.GetWindowRect(&rect);
	::DrawStretchBitmap(::GetDC(m_Preview.m_hWnd),bitmap,0,0,
		rect.right-rect.left,rect.bottom-rect.top,SRCCOPY);
		m_BackButton.EnableWindow(TRUE);
		m_Count++;
		if (m_Count>=pApp->GetOpenCount())
			m_NextButton.EnableWindow(FALSE);
	}
	else
		m_NextButton.EnableWindow(FALSE);
}
void COpenDlg::OnOpenBackButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	if (m_Count>1)
	{
		pApp->m_StickerList.GetPrev(m_Pos);
		HBITMAP bitmap = pApp->m_StickerList.GetAt(m_Pos);
		RECT rect;
		m_Preview.GetWindowRect(&rect);
	::DrawStretchBitmap(::GetDC(m_Preview.m_hWnd),bitmap,0,0,
		rect.right-rect.left,rect.bottom-rect.top,SRCCOPY);
		m_NextButton.EnableWindow(TRUE);
		m_Count--;
		if (m_Count<=1)
			m_BackButton.EnableWindow(FALSE);
	}
	else
```

```
            m_BackButton.EnableWindow(FALSE);
}
void COpenDlg::OnOpenDoneButton()
{
      // TODO: Add your control notification handler code here
      CStickerApp *pApp = (CStickerApp*) AfxGetApp();
      m_Pos = pApp->m_StickerList.GetHeadPosition();
      for (int i=1;i<m_Count;i++)
            pApp->m_StickerList.GetNext(m_Pos);
      CStickerDlg *dlg = (CStickerDlg*)GetParent();
      dlg->m_Step2Button.EnableWindow(FALSE);
      dlg->m_Step4Button.EnableWindow(TRUE);
      ShowWindow(FALSE);
}
```

SelectFrame.h

```
#if
!defined(AFX_SELECTFRAME_H__09DF1B88_D1A2_11D3_8AFC_E0614AC10801__IN
CLUDED_)
#define
AFX_SELECTFRAME_H__09DF1B88_D1A2_11D3_8AFC_E0614AC10801__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SelectFrame.h : header file
#include "MyDialog.h"
#include "MyButton.h"
#include <afxtempl.h>
/////////////////////////////////////////////////////////////////////////
// CSelectFrame dialog
class CSelectFrame : public CMyDialog
{
// Construction
public:
      CSelectFrame(CWnd* pParent = NULL);   // standard constructor
      int m_StyleCount;
      int m_StyleSelect;
      CList<HBITMAP,HBITMAP> m_List;
      POSITION m_Pos;
// Dialog Data
      //{{AFX_DATA(CSelectFrame)
      enum { IDD = IDD_SELECT_FRAME_DIALOG };
      CStatic     m_Preview;
      CMyButton   m_NextButton;
      CMyButton   m_DoneButton;
      CMyButton   m_BackButton;
      //}}AFX_DATA
// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CSelectFrame)
      public:
      virtual BOOL DestroyWindow();
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
      //}}AFX_VIRTUAL
```

```
// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(CSelectFrame)
	afx_msg void OnDoneButton();
	virtual BOOL OnInitDialog();
	afx_msg void OnNextButton();
	afx_msg void OnBackButton();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_SELECTFRAME_H__09DF1B88_D1A2_11D3_8AFC_E0614AC10801__IN
CLUDED_)
```

SelectFrame.cpp

```
// SelectFrame.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "SelectFrame.h"
#include "StickerDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// CSelectFrame dialog
CSelectFrame::CSelectFrame(CWnd* pParent /*=NULL*/)
	: CMyDialog(CSelectFrame::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSelectFrame)
		// NOTE: the ClassWizard will add member initialization
here
	//}}AFX_DATA_INIT
	m_StyleCount = 0;
	m_StyleSelect = 0;
}
void CSelectFrame::DoDataExchange(CDataExchange* pDX)
{
	CMyDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSelectFrame)
	DDX_Control(pDX, IDC_PREVIEW, m_Preview);
	DDX_Control(pDX, IDC_NEXT_BUTTON, m_NextButton);
	DDX_Control(pDX, IDC_DONE_BUTTON, m_DoneButton);
	DDX_Control(pDX, IDC_BACK_BUTTON, m_BackButton);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CSelectFrame, CMyDialog)
	//{{AFX_MSG_MAP(CSelectFrame)
	ON_BN_CLICKED(IDC_DONE_BUTTON, OnDoneButton)
	ON_BN_CLICKED(IDC_NEXT_BUTTON, OnNextButton)
	ON_BN_CLICKED(IDC_BACK_BUTTON, OnBackButton)
	//}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////
// CSelectFrame message handlers
void CSelectFrame::OnDoneButton()
{
	// TODO: Add your control notification handler code here
	ShowWindow(FALSE);
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	pApp->SetFrameSelect(m_StyleSelect);
	CString FileName;
	FileName.Format("%s\\%s\\Script.STK",pApp->GetFrameStylePath
(),pApp->GetFrameFileName(m_StyleSelect));
	//pApp->m_pszProfileName=_tcsdup(_T(LPCTSTR(FileName)));
	pApp->SetScriptFileName(FileName);
	CStickerDlg *dlg = (CStickerDlg*)GetParent();
	dlg->m_Step2Button.EnableWindow(FALSE);
	pApp->CreatePaper();
}
BOOL CSelectFrame::OnInitDialog()
{
	CMyDialog::OnInitDialog();
		// TODO: Add extra initialization here
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	//for bitmap
	HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;
	hButtonAll = ::LoadBitmapFile(pApp-
>Select_Frame_Dialog_Next_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Select_Frame_Dialog_Next_Width, pApp->
Select_Frame_Dialog_Next_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Select_Frame_Dialog_Next_Height, pApp->
Select_Frame_Dialog_Next_Width, pApp-
>Select_Frame_Dialog_Next_Height*2-1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Select_Frame_Dialog_Next_Height*2, pApp->
Select_Frame_Dialog_Next_Width, pApp-
>Select_Frame_Dialog_Next_Height*3-1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
		0, pApp->Select_Frame_Dialog_Next_Height*3, pApp->
Select_Frame_Dialog_Next_Width, pApp-
>Select_Frame_Dialog_Next_Height*4-1);
	m_NextButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
	m_NextButton.SizeToContent();
	hButtonAll = ::LoadBitmapFile(pApp-
>Select_Frame_Dialog_Back_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Select_Frame_Dialog_Back_Width, pApp->
Select_Frame_Dialog_Back_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Select_Frame_Dialog_Back_Height, pApp->
Select_Frame_Dialog_Back_Width, pApp-
>Select_Frame_Dialog_Back_Height*2-1);
```

```
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Select_Frame_Dialog_Back_Height*2, pApp->
Select_Frame_Dialog_Back_Width, pApp-
>Select_Frame_Dialog_Back_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Select_Frame_Dialog_Back_Height*3, pApp->
Select_Frame_Dialog_Back_Width, pApp-
>Select_Frame_Dialog_Back_Height*4-1);
      m_BackButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_BackButton.SizeToContent();
      hButtonAll = ::LoadBitmapFile(pApp-
>Select_Frame_Dialog_Done_Button.GetBuffer(0));
      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->Select_Frame_Dialog_Done_Width, pApp->
Select_Frame_Dialog_Done_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Select_Frame_Dialog_Done_Height, pApp->
Select_Frame_Dialog_Done_Width, pApp-
>Select_Frame_Dialog_Done_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->Select_Frame_Dialog_Done_Height*2, pApp->
Select_Frame_Dialog_Done_Width, pApp-
>Select_Frame_Dialog_Done_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->Select_Frame_Dialog_Done_Height*3, pApp->
Select_Frame_Dialog_Done_Width, pApp-
>Select_Frame_Dialog_Done_Height*4-1);
      m_DoneButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_DoneButton.SizeToContent();
            //for cursor
      m_NextButton.SetCursor(::LoadCursorFromFile(pApp->
Select_Frame_Dialog_Next_Cur.GetBuffer(0)));
      m_BackButton.SetCursor(::LoadCursorFromFile(pApp->
Select_Frame_Dialog_Back_Cur.GetBuffer(0)));
      m_DoneButton.SetCursor(::LoadCursorFromFile(pApp->
Select_Frame_Dialog_Done_Cur.GetBuffer(0)));
      SetCursor(::LoadCursorFromFile(pApp-
>Select_Frame_Dialog_Cur.GetBuffer(0)));
      //for sound
      m_NextButton.SetSound(IDR_DEFAULT_SOUND);
      m_BackButton.SetSound(IDR_DEFAULT_SOUND);
      m_DoneButton.SetSound(IDR_DEFAULT_SOUND);
      m_NextButton.MoveWindow(pApp->Select_Frame_Dialog_Next_X,
                              pApp-
>Select_Frame_Dialog_Next_Y,
                              pApp-
>Select_Frame_Dialog_Next_Width,
                              pApp-
>Select_Frame_Dialog_Next_Height);
      m_BackButton.MoveWindow(pApp->Select_Frame_Dialog_Back_X,
                              pApp-
>Select_Frame_Dialog_Back_Y,
                              pApp-
>Select_Frame_Dialog_Back_Width,
```

```
                                    pApp-
>Select_Frame_Dialog_Back_Height);
      m_DoneButton.MoveWindow(pApp->Select_Frame_Dialog_Done_X,
                                    pApp-
>Select_Frame_Dialog_Done_Y,
                                    pApp-
>Select_Frame_Dialog_Done_Width,
                                    pApp-
>Select_Frame_Dialog_Done_Height);
      HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>Select_Frame_Dialog_Background.GetBuffer(0));
      SetBitmap(hBitmap);
      SetSound();//IDR_NEW_OPEN_DLG);
      SetMidi();//IDR_DEFAULT_MIDI);
      ShowWindow(FALSE);
      int width = pApp->Select_Frame_Dialog_Background_Width;
      int height = pApp->Select_Frame_Dialog_Background_Height;
      MoveWindow(0,0, width, height);
      CString Path;
      m_StyleCount = pApp->GetFrameCount();
      for (int i=0;i<m_StyleCount;i++)
      {
            Path.Format("%s%s%s%s",pApp->GetFrameStylePath(),"\\",
                  pApp->GetFrameFileName(i),"\\View.BMP");
            m_List.AddTail(::LoadBitmapFile((char*)LPCSTR(Path)));
            Path.Empty();
      }
      m_BackButton.EnableWindow(FALSE);
      m_Pos = m_List.GetHeadPosition();
      m_Preview.SetBitmap(m_List.GetAt(m_Pos));
      m_Preview.MoveWindow(pApp->Select_Frame_Dialog_Preview_X,
            pApp->Select_Frame_Dialog_Preview_Y,
            pApp->Select_Frame_Dialog_Preview_Width,
            pApp->Select_Frame_Dialog_Preview_Height);
      return TRUE;  // return TRUE unless you set the focus to a
control
                  // EXCEPTION: OCX Property Pages should return
FALSE
}
void CSelectFrame::OnNextButton()
{
      // TODO: Add your control notification handler code here
      m_BackButton.EnableWindow(TRUE);

      HBITMAP hBitmap;

      if (m_StyleSelect<m_StyleCount-1)
      {
            m_StyleSelect++;
            m_List.GetNext(m_Pos);
            hBitmap = m_List.GetAt(m_Pos);
            m_Preview.SetBitmap(hBitmap);
            if (m_StyleSelect==(m_StyleCount-1))
                  m_NextButton.EnableWindow(FALSE);
      }
}
void CSelectFrame::OnBackButton()
{
```

```
	// TODO: Add your control notification handler code here
	m_NextButton.EnableWindow(TRUE);
	HBITMAP hBitmap;
	if (m_StyleSelect>0)
	{
		m_StyleSelect--;
		m_List.GetPrev(m_Pos);
		hBitmap = m_List.GetAt(m_Pos);
		m_Preview.SetBitmap(hBitmap);
		if (m_StyleSelect==0)
			m_BackButton.EnableWindow(FALSE);
	}
}
BOOL CSelectFrame::DestroyWindow()
{
	// TODO: Add your specialized code here and/or call the base
class
	m_Pos = m_List.GetHeadPosition();
	while (!m_List.IsEmpty())
	{
		DeleteObject(m_List.GetHead());
		m_List.RemoveHead();
	}
	return CMyDialog::DestroyWindow();
}
```

CaptureDlg.h

```
#if
!defined(AFX_CAPTUREDLG_H__6DC14161_D381_11D3_8AFC_303A49C10001__INC
LUDED_)
#define
AFX_CAPTUREDLG_H__6DC14161_D381_11D3_8AFC_303A49C10001__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// CaptureDlg.h : header file
//
#include "MyDialog.h"
#include "Frame.h"
#include "MyButton.h"
/////////////////////////////////////////////////////////////////////////
// CCaptureDlg dialog
class CCaptureDlg : public CMyDialog
{
// Construction
public:
	CCaptureDlg(CWnd* pParent = NULL);   // standard constructor
	HWND  Capture_Window;
	int		m_Capture_Count;
	//CFrame  Frame;
// Dialog Data
	//{{AFX_DATA(CCaptureDlg)
	enum { IDD = IDD_CAPTURE_DIALOG };
	CMyButton   m_CaptureButton;
	//}}AFX_DATA
// Overrides
	// ClassWizard generated virtual function overrides
```

```
	//{{AFX_VIRTUAL(CCaptureDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
	//}}AFX_VIRTUAL
// Implementation
protected:
	// Generated message map functions
	//{{AFX_MSG(CCaptureDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnCaptureButton();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_CAPTUREDLG_H__6DC14161_D381_11D3_8AFC_303A49C10001__INC
LUDED_)
```

CaptureDlg.cpp

```
// CaptureDlg.cpp : implementation file
#include "stdafx.h"
#include "Sticker.h"
#include "CaptureDlg.h"
#include <vfw.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////
// CCaptureDlg dialog
CCaptureDlg::CCaptureDlg(CWnd* pParent /*=NULL*/)
	: CMyDialog(CCaptureDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCaptureDlg)
	//}}AFX_DATA_INIT
	m_Capture_Count = 0;
}
void CCaptureDlg::DoDataExchange(CDataExchange* pDX)
{
	CMyDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCaptureDlg)
	DDX_Control(pDX, IDC_CAPTURE_BUTTON, m_CaptureButton);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CCaptureDlg, CMyDialog)
	//{{AFX_MSG_MAP(CCaptureDlg)
	ON_BN_CLICKED(IDC_CAPTURE_BUTTON, OnCaptureButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////
// CCaptureDlg message handlers
BOOL CCaptureDlg::OnInitDialog()
{
```

```
	CMyDialog::OnInitDialog();
	// TODO: Add extra initialization here
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	//for bitmap
	HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;
	hButtonAll = ::LoadBitmapFile(pApp-
>Capture_Dialog_Capture_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->Capture_Dialog_Capture_Width, pApp->
Capture_Dialog_Capture_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Capture_Dialog_Capture_Height, pApp->
Capture_Dialog_Capture_Width, pApp->Capture_Dialog_Capture_Height*2-
1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->Capture_Dialog_Capture_Height*2, pApp->
Capture_Dialog_Capture_Width, pApp->Capture_Dialog_Capture_Height*3-
1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
		0, pApp->Capture_Dialog_Capture_Height*3, pApp->
Capture_Dialog_Capture_Width, pApp->Capture_Dialog_Capture_Height*4-
1);
	m_CaptureButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
	m_CaptureButton.SizeToContent();
		//for cursor
	m_CaptureButton.SetCursor(::LoadCursorFromFile(pApp->
Capture_Dialog_Capture_Cur.GetBuffer(0)));
	SetCursor(::LoadCursorFromFile(pApp-
>Capture_Dialog_Cur.GetBuffer(0)));

	//for sound
	m_CaptureButton.SetSound(IDR_DEFAULT_SOUND);

	m_CaptureButton.MoveWindow(pApp->Capture_Dialog_Capture_X,
				pApp->Capture_Dialog_Capture_Y,
				pApp-
>Capture_Dialog_Capture_Width,
				pApp-
>Capture_Dialog_Capture_Height);
	HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>Capture_Dialog_Background.GetBuffer(0));
	SetBitmap(hBitmap);
	SetSound();//IDR_NEW_OPEN_DLG);
	SetMidi();//IDR_DEFAULT_MIDI);
	ShowWindow(FALSE);
	int width = pApp->Capture_Dialog_Background_Width;
	int height = pApp->Capture_Dialog_Background_Height;
	MoveWindow(0,0, width, height);
	Capture_Window = capCreateCaptureWindow((LPTSTR)TEXT("Capture
Window"),
					WS_CHILD | WS_VISIBLE,
					0, 0, 800, 600,
	this->m_hWnd, (int) 0);
```

```
	capDriverConnect(Capture_Window, 0); //0 = default driver
	capPreviewRate(Capture_Window, 66);
	capOverlay(Capture_Window, TRUE);
	capPreview(Capture_Window, TRUE) ;
	BITMAPINFO video_format;
	capGetVideoFormat(Capture_Window,&video_format,sizeof
(video_format));
	video_format.bmiHeader.biHeight = pApp-
>Capture_Dialog_Preview_Height;
	video_format.bmiHeader.biWidth = pApp-
>Capture_Dialog_Preview_Width;
	capSetVideoFormat(Capture_Window,&video_format,sizeof
(video_format));
	::MoveWindow(Capture_Window,pApp->Capture_Dialog_Preview_X,
		pApp->Capture_Dialog_Preview_Y,
		pApp->Capture_Dialog_Preview_Width,
		pApp->Capture_Dialog_Preview_Height,TRUE);
	CRect rect;
	rect.left = pApp->Capture_Dialog_Preview_X;
	rect.top = pApp->Capture_Dialog_Preview_Y;
	rect.right = pApp->Capture_Dialog_Preview_Width;
	rect.bottom = pApp->Capture_Dialog_Preview_Height;
	return TRUE;  // return TRUE unless you set the focus to a
control
		// EXCEPTION: OCX Property Pages should return
FALSE
}
void CCaptureDlg::OnCaptureButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp;
	pApp = (CStickerApp*)AfxGetApp();
	if (m_Capture_Count<pApp->pPaper->m_ActCount) //pApp->
GetFrameCount())
	{
		int result;
		result = capGrabFrameNoStop(Capture_Window);
		//result = capEditCopy(Capture_Window);
		capFileSaveDIB(Capture_Window,"C:/test.bmp");
		//::UpdateWindow(Capture_Window);
		//HBITMAP hBitmap = (HBITMAP)GetClipboardData
(CF_BITMAP);
		HBITMAP hBitmap = ::LoadBitmapFile("C:/test.bmp");
		//::DrawBitmap(::GetWindowDC(m_hWnd),hBitmap,
		//    0,0,0,0);
		pApp->pPaper->m_ActBitmap.AddTail(hBitmap);
		//SetRedraw();
		m_Capture_Count++;
		if (m_Capture_Count==pApp->pPaper->m_ActCount)
		{
			pApp->pPaper->SetFrameAction();
			m_Capture_Count = 0;
			ShowWindow(FALSE);
			//pApp->pPaper->DrawPaper(::GetDC(m_hWnd));
		}
	}
	else
		ShowWindow(FALSE);
```

```
}
```

PrintPreviewDlg.h

```
#if
!defined(AFX_PRINTPREVIEWDLG_H__F4BEDA61_1620_11D4_881C_F0384AC1010C
__INCLUDED_)
#define
AFX_PRINTPREVIEWDLG_H_F4BEDA61_1620_11D4_881C_F0384AC1010C__INCLUDED
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// PrintPreviewDlg.h : header file
#include "MyDialog.h"
#include "MyButton.h"
/////////////////////////////////////////////////////////////////////
// CPrintPreviewDlg dialog
class CPrintPreviewDlg : public CMyDialog
{
// Construction
public:
      CPrintPreviewDlg(CWnd* pParent = NULL);   // standard
constructor
// Dialog Data
      //{{AFX_DATA(CPrintPreviewDlg)
      enum { IDD = IDD_PRINT_PREVIEW_DIALOG };
      CStatic     m_PrintPreview;
      CMyButton   m_PreviewButton;
      CMyButton   m_PrintButton;
      //}}AFX_DATA
// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CPrintPreviewDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV
support
      //}}AFX_VIRTUAL
// Implementation
protected:
      // Generated message map functions
      //{{AFX_MSG(CPrintPreviewDlg)
      virtual BOOL OnInitDialog();
      afx_msg void OnPrintButton();
      afx_msg void OnPreviewButton();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.
#endif //
!defined(AFX_PRINTPREVIEWDLG_H__F4BEDA61_1620_11D4_881C_F0384AC1010C
__INCLUDED_)
```

PrintPreviewDlg.cpp

```
// PrintPreviewDlg.cpp : implementation file
#include "stdafx.h"
```

```
#include "Sticker.h"
#include "PrintPreviewDlg.h"
#include "DIBSectionLite.h"
#include "StickerDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CPrintPreviewDlg dialog
CPrintPreviewDlg::CPrintPreviewDlg(CWnd* pParent /*=NULL*/)
	: CMyDialog(CPrintPreviewDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CPrintPreviewDlg)
	//}}AFX_DATA_INIT
}
void CPrintPreviewDlg::DoDataExchange(CDataExchange* pDX)
{
	CMyDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CPrintPreviewDlg)
	DDX_Control(pDX, IDC_PRINT_PREVIEW, m_PrintPreview);
	DDX_Control(pDX, IDC_PREVIEW_BUTTON, m_PreviewButton);
	DDX_Control(pDX, IDC_PRINT_BUTTON, m_PrintButton);
	//}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CPrintPreviewDlg, CMyDialog)
	//{{AFX_MSG_MAP(CPrintPreviewDlg)
	ON_BN_CLICKED(IDC_PRINT_BUTTON, OnPrintButton)
	ON_BN_CLICKED(IDC_PREVIEW_BUTTON, OnPreviewButton)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CPrintPreviewDlg message handlers
BOOL CPrintPreviewDlg::OnInitDialog()
{
	CMyDialog::OnInitDialog();
	// TODO: Add extra initialization here
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	//for bitmap
	HBITMAP hButton, hButtonSel, hButtonFocus, hButtonDisabled,
hButtonAll;
	hButtonAll = ::LoadBitmapFile(pApp-
>PrintPreview_Dialog_Print_Button.GetBuffer(0));
	hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, 0, pApp->PrintPreview_Dialog_Print_Width, pApp->
PrintPreview_Dialog_Print_Height);//-1);
	hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->PrintPreview_Dialog_Print_Height, pApp->
PrintPreview_Dialog_Print_Width, pApp-
>PrintPreview_Dialog_Print_Height*2-1);
	hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
		0, pApp->PrintPreview_Dialog_Print_Height*2, pApp->
PrintPreview_Dialog_Print_Width, pApp-
>PrintPreview_Dialog_Print_Height*3-1);
	hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
```

```
            0, pApp->PrintPreview_Dialog_Print_Height*3, pApp->
PrintPreview_Dialog_Print_Width, pApp-
>PrintPreview_Dialog_Print_Height*4-1);
      m_PrintButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_PrintButton.SizeToContent();
      hButtonAll = ::LoadBitmapFile(pApp-
>PrintPreview_Dialog_PrintPreview_Button.GetBuffer(0));
      hButton = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, 0, pApp->PrintPreview_Dialog_PrintPreview_Width,
pApp->PrintPreview_Dialog_PrintPreview_Height);//-1);
      hButtonSel = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->PrintPreview_Dialog_PrintPreview_Height, pApp->
PrintPreview_Dialog_PrintPreview_Width, pApp-
>PrintPreview_Dialog_PrintPreview_Height*2-1);
      hButtonFocus = ::CreateSubBitmap(::GetDC(m_hWnd), hButtonAll,
            0, pApp->PrintPreview_Dialog_PrintPreview_Height*2,
pApp->PrintPreview_Dialog_PrintPreview_Width, pApp-
>PrintPreview_Dialog_PrintPreview_Height*3-1);
      hButtonDisabled = ::CreateSubBitmap(::GetDC(m_hWnd),
hButtonAll,
            0, pApp->PrintPreview_Dialog_PrintPreview_Height*3,
pApp->PrintPreview_Dialog_PrintPreview_Width, pApp-
>PrintPreview_Dialog_PrintPreview_Height*4-1);
      m_PreviewButton.LoadBitmaps(hButton, hButtonSel, hButtonFocus,
hButtonDisabled);
      m_PreviewButton.SizeToContent();
      //for cursor
      m_PrintButton.SetCursor(::LoadCursorFromFile(pApp->
PrintPreview_Dialog_Print_Cur.GetBuffer(0)));
      m_PreviewButton.SetCursor(::LoadCursorFromFile(pApp->
PrintPreview_Dialog_PrintPreview_Cur.GetBuffer(0)));
      SetCursor(::LoadCursorFromFile(pApp-
>PrintPreview_Dialog_Cur.GetBuffer(0)));
      //for sound
      m_PrintButton.SetSound(IDR_DEFAULT_SOUND);
      m_PreviewButton.SetSound(IDR_DEFAULT_SOUND);
      m_PrintButton.MoveWindow(pApp->PrintPreview_Dialog_Print_X,
                               pApp-
>PrintPreview_Dialog_Print_Y,
                               pApp-
>PrintPreview_Dialog_Print_Width,
                               pApp-
>PrintPreview_Dialog_Print_Height);
      m_PreviewButton.MoveWindow(pApp-
>PrintPreview_Dialog_PrintPreview_X,
                               pApp-
>PrintPreview_Dialog_PrintPreview_Y,
                               pApp-
>PrintPreview_Dialog_PrintPreview_Width,
                               pApp-
>PrintPreview_Dialog_PrintPreview_Height);
      HBITMAP hBitmap = ::LoadBitmapFile(pApp-
>PrintPreview_Dialog_Background.GetBuffer(0));
      SetBitmap(hBitmap);
      SetSound();//IDR_NEW_OPEN_DLG);
      SetMidi();//IDR_DEFAULT_MIDI);
      ShowWindow(FALSE);
```

```
	int width = pApp->PrintPreview_Dialog_Background_Width;
	int height = pApp->PrintPreview_Dialog_Background_Height;
	MoveWindow(0,0, width, height);
	CRect rect;
	rect.left = pApp->Capture_Dialog_Preview_X;
	rect.top = pApp->Capture_Dialog_Preview_Y;
	rect.right = pApp->Capture_Dialog_Preview_Width;
	rect.bottom = pApp->Capture_Dialog_Preview_Height;
	m_PrintPreview.MoveWindow(&rect);
	return TRUE;  // return TRUE unless you set the focus to a
control
                  // EXCEPTION: OCX Property Pages should return
FALSE
}
void CPrintPreviewDlg::OnPrintButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	CDC dc;
	CPrintDialog dlg (FALSE);
	dlg.GetDefaults ();
	DEVMODE *DeviceMode;
	DeviceMode = dlg.GetDevMode();
	dc.Attach (dlg.GetPrinterDC ());
	dlg.m_pd.nMinPage = dlg.m_pd.nMaxPage = 1;
	dlg.m_pd.nFromPage = dlg.m_pd.nToPage = 1;
	dc.SetMapMode(MM_ISOTROPIC);
	switch (DeviceMode->dmYResolution)
	{
		case 360 :
			dc.SetViewportExt(500,500);
		break;
		case 720 :
			dc.SetViewportExt(1000,1000);
		break;
	}
	DOCINFO di;
	::ZeroMemory (&di, sizeof (DOCINFO));
	di.cbSize = sizeof (DOCINFO);
	di.lpszDocName = "Print Sticker";
	dc.StartDoc (&di);
	dc.StartPage();
	HBITMAP bitmap;
	if (pApp->m_New)
		bitmap = GetInvertedBitmap( HBITMAP(pApp->pPaper->
m_Paper), TRUE);
	else
	{
		CStickerDlg *StkDlg = (CStickerDlg*) GetParent();
		HBITMAP bmp;
		bmp = pApp->m_StickerList.GetAt(StkDlg-
>m_OpenDlg.m_Pos);
		bitmap = GetInvertedBitmap( bmp, TRUE);
	}
	SetBitmapPixels(&dc, bitmap,
			RGB(255,255,255),
				pApp->pPaper->m_Size.cx/2,
				pApp->pPaper->m_Size.cy/2);
```

```
	dc.EndPage();
	dc.EndDoc();
	//for scan frame
	CFileFind file;
	CString Path;
	int count = 0;
	Path.Format("%s%s%d%s",LPCTSTR(pApp->GetOpenStickerPath
()),"\\Sticker",count,".BMP");
	while (file.FindFile(LPCSTR(Path)))
	{
		count++;
		Path.Format("%s%s%d%s",LPCTSTR(pApp->GetOpenStickerPath
()),"\\Sticker",count,".BMP");
	};

	if (pApp->m_New)
	{
		CDIBSectionLite dib;
		dib.SetBitmap(HBITMAP(pApp->pPaper->m_Paper));
		dib.Save(LPCSTR(Path));
	}
	else
		pApp->ClearStickerList();

	pApp->pPaper->DestroyFrame();
	ShowWindow(FALSE);
}
void CPrintPreviewDlg::OnPreviewButton()
{
	// TODO: Add your control notification handler code here
	CStickerApp *pApp = (CStickerApp*) AfxGetApp();
	if (pApp->m_New)
	{
		pApp->pPaper->DrawPaper(::GetDC(m_hWnd),
			pApp->Capture_Dialog_Preview_X, pApp->
Capture_Dialog_Preview_Y,
			pApp->Capture_Dialog_Preview_Width, pApp->
Capture_Dialog_Preview_Height);
	}
	else
	{
		CStickerDlg *dlg = (CStickerDlg*)GetParent();
		HBITMAP bitmap;
		bitmap = pApp->m_StickerList.GetAt(dlg-
>m_OpenDlg.m_Pos);
		DrawStretchBitmap(::GetDC(m_hWnd), bitmap,
			pApp->Capture_Dialog_Preview_X, pApp->
Capture_Dialog_Preview_Y,
			pApp->Capture_Dialog_Preview_Width, pApp->
Capture_Dialog_Preview_Height,SRCCOPY);
	}
}
```

การติดต่อ Registry

```
void CToolDlg::OnToolDoneButton()
{
	// TODO: Add your control notification handler code here
	CButton *StartWindow,*StartSticker;
	CStickerApp *pApp;
	pApp = (CStickerApp*) AfxGetApp();
	CString SubKey("Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce");
	CString Data;
	Data.Format(pApp->GetAppPath()+"\\Sticker.EXE");

	HKEY hKey;
	RegOpenKey( HKEY_LOCAL_MACHINE, LPCTSTR(SubKey), &hKey);

	pApp->m_pszProfileName=_tcsdup(_T("MySticker.INI"));
	StartWindow = (CButton*) GetDlgItem(IDC_START_WINDOW);
	StartSticker = (CButton*) GetDlgItem(IDC_START_STICKER);
	if (StartWindow->GetCheck() == 1)
	{
		pApp->WriteProfileInt("Environtment","StartupType",0);
		RegDeleteValue( hKey, "TheSticker");
	}
	if (StartSticker->GetCheck() == 1)
	{
		pApp->WriteProfileInt("Environtment","StartupType",1);
		RegSetValueEx( hKey, "TheSticker",0, REG_SZ,
		(const unsigned char *) LPCSTR(Data), (DWORD) (Data.GetLength()+1) *
		sizeof(TCHAR));
	}
```

```
        RegCloseKey(hKey);
        ShowWindow(FALSE);
}
```

ยกตัวอย่างดังโปรแกรมข้างบนซึ่งเป็นการตั้งค่า Registry และการลบ Registry ซึ่งตัวอย่างเป็นการตั้งค่า Registry เพื่อทำการรันโปรแกรมสติกเกอร์ก่อน Explorer โดยทำการตั้งค่า Registry HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce ซึ่งมีรายละเอียดของฟังก์ชันที่ใช้ทั้งหมดในตัวอย่างดังนี้

```
LONG RegOpenKey(
  HKEY hKey,      // HANDLE ของคีย์ที่ต้องการเปิด
  LPCTSTR lpSubKey, // ชื่อของคีย์ย่อย
  PHKEY phkResult  // ค่า HANDLE ของคีย์ที่ทำการเปิด
);

LONG RegDeleteValue(
  HKEY hKey,        // ค่า HANDLE ของคีย์ที่ต้องการลบ
  LPCTSTR lpValueName  //ชื่อของคีย์ที่ต้องการลบ
);
LONG RegSetValueEx(
  HKEY hKey,        // ค่า HANDLE ของคีย์ที่ต้องการตั้งค่า
  LPCTSTR lpValueName, // ชื่อของคีย์ที่ต้องการตั้งค่า
  DWORD Reserved,    // สงวนไว้
  DWORD dwType,      // แฟลกบอกชนิดของข้อมูล REG_SZ = ข้อความ
  CONST BYTE *lpData,  // ข้อมูลที่ต้องการตั้งค่า
  DWORD cbData       // ขนาดของข้อมูล
);
LONG RegCloseKey(
  HKEY hKey  // ค่า HANDLE ของคีย์ที่ต้องการปิด
);
```

การรับภาพจากการ์ด Capture

```
HWND Capture_Window;
Capture_Window = capCreateCaptureWindow((LPTSTR)TEXT("Capture Window"),
                                WS_CHILD | WS_VISIBLE,0, 0, 800, 600, this->m_hWnd, (int) 0);
capDriverConnect(Capture_Window, 0); //0 = default driver
capPreviewRate(Capture_Window, 66);
capPreview(Capture_Window, TRUE) ;
capGrabFrameNoStop(Capture_Window);
capEditCopy(Capture_Window);
capDriverDisconnect(Capture_Window);
```

จากส่วนของโปรแกรมข้างต้นจะเห็นว่าขั้นแรกจะต้องสร้างวินโดว์ที่ใช้ในการติดต่อกับการ์ด Capture จากนั้นทำการติดต่อกับไดร์เวอร์โดยระบุ Index ของไดร์เวอร์ที่ต้องการติดต่อ จากนั้นทำการตั้งค่าการแสดงผลแบบ Preview เมื่อต้องการจับภาพนิ่งสามารถใช้ Macro capGrabFrameNoStop และทำการเก็บภาพลงใน Clipboard โดยใช้ Macro capEditCopy เมื่อต้องการเลิกการติดต่อการ์ด Capture ต้องใช้ Macro capDriverDisconnect เพื่อทำการปล่อยไดร์เวอร์ให้โปรแกรมอื่นใช้งานได้ต่อไป

ภาคผนวก ค
รายละเอียดและคุณสมบัติของอุปกรณ์

FLYVIDEO '98

Lifeview's new Flyvideo 98 boasts the latest in TV viewing, video capture, and Internet roadcast reception. Just add this Plug and Play compatible board into your PC and you are ready to go! Flyvideo 98 has the latest tuner technology to allow crisp TV or video viewing in a sizable, scaleable window on your PC monitor up to full screen! Now you can capture high quality still video or full-motion video footage! Add these video images to your multimedia presentations, demos, web page, or just make them for fun!

FlyVideo 98 also includes a remote control unit so you can enjoy viewing without having to be near the keyboard and mouse. So sit back and enjoy the show!

FlyVideo 98 is also compliant with ITU H.324 standards. This means that you can user virtually any videoconferencing or videophone applications available today for live video sessions over a network or the Internet!

Flyvideo 98 comes with an intuitive interface making all capture and viewing functions a snap to do. You can make all fine tuning adjustments, volume control, and viewing styles right on screen!

SPECIFICATIONS

TV Broadcast

- 125 channel built-in and cable ready tuner
- Full NTSC broadcast support (PAL and SECAM available separately)

Video

- Supports up to 1280x1024 and 24-bit desktop viewing
- DirectX support

Full Motion Video Capture

- Sizable capture up to 640x480 resolution
- Supports 16, 24, or 32-bit color

- Capture up to 30 fps
- Captures to AVI file format

Still Frame Video Capture

- Sizable capture up to 640x480 resolution
- Supports 16, 24, or 32-bit color
- Save to BMP file or Clipboard

External Interface

- 75 ohm IEC coaxial input (cable TV)
- Composite (RCA) input
- S-Video (SVHS) input
- Audio input and line audio output

Dimensions and Weight

- 11 1/16 " x 6 5/8 " x 2 ?"
- 1 lb., 9.7 oz

SYSTEM REQUIREMENTS

- 133 MHz CPU or higher
- IBM PC or compatible computer system.
- 16 MB RAM
- 10 MB minimum available disk space for application and driver installation
- PCI or AGP SVGA graphics card with DirectX support
- Windows 95/98 support
- One available PCI slot
- Sound card (for audio capture)

PACKAGE CONTENTS

- FlyVideo '98 PCI compliant TV tuner and video capture card
- Audio loopback cable
- Remote Controller Unit (with 2 AA batteries included)
- User's Guide
- Installation CD-ROM

บรรณานุกรม

ชัยวุฒิ จันมา. **การเขียนโปรแกรมภาษา C++** . กรุงเทพฯ : หจก. ไทยเจริญการพิมพ์. 2538

3D Engine. **การเขียนโปรแกรมกราฟฟิกและเกมคอมพิวเตอร์ด้วยภาษา C.** กรุงเทพฯ : สงวนศักดิ์การพิมพ์. 2537

สมพันธุ์ ชาญศิลป์. **เอกสารประกอบการสอนวิชาภาษาซี.** ขอนแก่น : หน่วยสารบรรณงานบริการและธุรการ คณะวิศวกรรมศาสตร์ มหาลัยขอนแก่น. 2537

พันเอก เจนวิทย์ เหลืองอร่าม. **การใช้ Turbo C++ เขียนโปรแกรมภาษา C.** กรุงเทพฯ : ธรรมสารการพิมพ์. 2538

นิรุธ อำนวยศิลป์. **คู่มือการเขียนโปรแกรมด้วย Visual C++.** กรุงเทพฯ : สุทธิสารการพิมพ์. 2541

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายรัฐศักดิ์ ตั้งทักษ์ไกร
วันเดือนปีเกิด	11 กันยายน 2521
สถานที่เกิด	จังหวัดอุบลราชธานี
ภูมิลำเนาเดิม	242 หมู่ที่ 15 ต. ขามใหญ่ อ. เมือง จ. อุบลราชธานี
ที่อยู่ปัจจุบัน	261 ยุคลรัตน์คอนโดมิเนียม ซ. ดับเพลิง แขวงทับยาว
	เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	7380006-9 ต่อ 239
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านหนองฮาง
มัธยมศึกษาตอนต้น	โรงเรียนเบ็ญจะมะมหาราชอุบลราชธานี
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคอุบลราชธานี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคอุบลราชธานี
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ความสำเร็จร้อยละ 99 มาจากความยากลำบาก

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	ว่าที่ ร.ต. วัชรพล แซ่ตั้น
วันเดือนปีเกิด	16 พฤศจิกายน 2520
สถานที่เกิด	จังหวัดสตูล
ภูมิลำเนาเดิม	133/2 หมู่ที่ 2 ต. เจ๊ะบิลัง อ. เมือง จ. สตูล 91000
อยู่ปัจจุบัน	318/3 ซ. จินดานิเวศน์ แขวงลาดกระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7391029 , (01) 8197430
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านเจ๊ะบิลัง
มัธยมศึกษาตอนต้น	โรงเรียนพิมานพิทยาสรรค์
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคสตูล
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตภาคใต้สงขลา
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ค่าของคนอยู่ที่ผลของงาน

ประวัติผู้แต่ง

ชื่อ นำปริญญานิพนธ์	นายสมเพ็ชร เครือทองศรี
วันเดือนปีเกิด	1 เมษายน 2521
สถานที่เกิด	จังหวัดเลย
ภูมิลำเนาเดิม	32/30 ถนน นกแก้ว ตำบลกุดป่อง อำเภอเมือง จังหวัดเลย
ที่อยู่ปัจจุบัน	101/321 หมู่บ้านฟ้าคลองหลวง ถนน พหลโยธิน ตำบลคลองหนึ่ง อำเภอคลองหลวง จังหวัดปทุมธานี 12120
โทรศัพท์	02-9052274 , 01-8075874
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านหนองผักก้าม
มัธยมศึกษาตอนต้น	โรงเรียนเลยพิทยาคม
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคเลย
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคเลย
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	รองชนะเลิศอันดับที่ 1 ในการประกวดโครงงานวิทยาศาสตร์ระดับชาติ
ทุนการศึกษา	ทุนการศึกษาระดับประกาศนียบัตรวิชาชีพ
คติพจน์	ขยัน ตั้งใจ มุ่งมั่น สิ่งที่มุ่งหวังอยู่ไม่ไกล

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายอัคร เพชรมณี
วันเดือนปีเกิด	30 ตุลาคม 2520
สถานที่เกิด	จังหวัดสงขลา
ภูมิลำเนาเดิม	34/1 ซ. 14 ถนนทะเลหลวง ตำบลบ่อยาง อำเภอเมือง จังหวัดสงขลา
อยู่ปัจจุบัน	318/3 ซ. จินดานิเวศน์ แขวง ลาดกระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(01)8912693
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนอนุบาลนราธิวาส
มัธยมศึกษาตอนต้น	โรงเรียนมหาวชิราวุธ จังหวัดสงขลา
มัธยมศึกษาตอนปลาย	โรงเรียนมหาวชิราวุธ จังหวัดสงขลา
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตนนทบุรี
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ฝันให้ไกลไปให้ถึง