การพิมพ์ภาษาไทยสำหรับปาล์ม

Thai Palm Print

นายปิยะ ผลิเจริญผล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การพิมพ์ภาษาไทยสำหรับปาล์ม

Thai Palm Print

โดย

นายปิยะ ผลิเจริญผล

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพิมพ์ภาษาไทยบนปาล์ม

Thai Palm Print

ผู้จัดทำ

1. นาย ปิยะ ผลิเจริญผล รหัสประจำตัว 40010464

_____ อาจารย์ที่ปรึกษา

(อาจารย์ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

การพิมพ์ภาษาไทยบนปาล์ม

นายปิยะ ผลิเจริญผล 40010464
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ในปัจจุบันปาล์ม (Palm) เป็นออร์แกไนเซอร์ (Organizer) ที่ได้รับความนิยมเป็นอย่างมาก เนื่องจากขนาดเล็ก น้ำหนักเบา และราคาที่ไม่แพงจนเกินไป และมีความสามารถไม่เพียงแค่ออร์แกไนเซอร์ คือ สามารถเขียนโปรแกรมเพิ่มเติม ทำให้มีโปรแกรมสำหรับปาล์มออกมาสนับสนุนความต้องการของผู้ใช้เป็นจำนวนมาก

แต่เนื่องด้วยระบบปฏิบัติการบนปาล์ม (Palm OS) นั้น ไม่สนับสนุนในการพิมพ์เอกสาร ทำให้มีผู้พัฒนาโปรแกรมช่วยในงานพิมพ์บนปาล์มออกมา เช่น PalmPrint, IR Print suite แต่โปรแกรมดังกล่าวก็ยังไม่สนับสนุนภาษาไทย ไม่สามารถพิมพ์เอกสารภาษาไทยได้ เนื่องจากโปรแกรมไม่มีฟอนต์ภาษาไทย และไม่มีฟังก์ชั่นในการจัดพิมพ์ภาษาไทย นอกจากนั้นยังไม่สนับสนุนการพิมพ์โดยผ่านพอร์ตอนุกรมอย่างเต็มที่ กล่าวคือต้องอาศัยโปรแกรมเช่นไฮเปอร์เทอร์มินอล (HyperTerminal) ในการรับข้อมูลจากปาล์มแล้วส่งต่อไปยังเครื่องพิมพ์

ระบบภาษาไทยสำหรับปาล์มนั้น ปัจจุบันมีผู้ผลิตซอฟต์แวร์รายใหญ่อยู่สองค่าย คือ ThaiPos และ ThaiHack ซึ่งทำให้สามารถแสดงภาษาไทย อินพุตข้อมูลเป็นภาษาไทย โดยมีการกำหนดกราฟฟิตี้ (Graffiti) ภาษาไทย รวมทั้งแสดงเมนู และยูสเซอร์อินเทอร์เฟซเป็นภาษาไทย แต่ก็ยังไม่สนับสนุนในเรื่องการพิมพ์เอกสารภาษาไทย

จากแนวคิดตามข้างต้น ทำให้เกิดปริญญานิพนธ์นี้ขึ้น เพื่อพัฒนาโปรแกรมที่สามารถพิมพ์เอกสารเป็นภาษาไทย จากเครื่องปาล์มได้

Thai Palm Print

Mr. Piya	Plicharoenpon	
Mr. Thana	Hongsuwan	Advisor
Mr. Akkradach	Watcharapupong	Advisor

Abstract

At present Palm is a very popular organizer because of its small size, light weight and affordable price. Its capability is not only an organizer but also further more program can be written on it. Therefore a lot of programs for Palm are launched to serve the users' requirement.

As Palm Operating System (Palm OS) does not support printing, programs for Palm printing such as Palm Print, IR Print Suite are developed. Still they do not support Thai printing because of the lack of Thai font and multi-levels in printing Thai and printing via serial port must be used toward Teminal Emulator such as HyperTerminal to redirect data from Palm to printer.

There are two software, ThaiPos and ThaiHack, that help to display Thai, input Thai and assign Thai Graffiti and to display Thai user interface but they do not help to print Thai.

This project comes after this concept that is to develop program which enable to print Thai from Palm.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วย เนื่องจากได้รับคำแนะนำ สนับสนุน และคำปรึกษาจาก อาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง อีกทั้งคณาจารย์ทุกท่านที่ได้อบรมสั่งสอน ทำให้ผู้จัดทำมีความรู้ความสามารถที่เพียงพอในการจัดทำปริญญานิพนธ์นี้

ขอขอบพระคุณภาควิชาวิศวกรรมศาสตร์ และห้องวิจัยระบบรักษาความปลอดภัย (ISAG) ที่ได้เอื้อเฟื้อสถานที่ และอำนวยความสะดวกต่างๆในการทำปริญญานิพนธ์นี้ รวมทั้งเป็นแหล่งความรู้สำคัญของผู้จัดทำปริญญานิพนธ์นี้เสมอมา

สุดท้ายนี้ขอขอบพระคุณ บิดามารดา ที่ได้ให้กำเนิด คอยสั่งสอน ดูแล ให้การศึกษา พร้อมทั้งสนับสนุนกิจกรรมต่างๆ และขอบคุณเพื่อนๆ ที่ให้ข้อคิดเห็น และเป็นกำลังใจ ผู้จัดทำกราบขอขอบพระคุณมา ณ ที่นี้ด้วย

ปิยะ ผลิเจริญผล

สารบัญ

สารบัญภาพประกอบ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันปาล์ม (Palm) เป็นออร์แกไนเซอร์ (Organizer) ที่ได้รับความนิยมเป็นอย่างมาก เนื่องจากขนาดเล็ก น้ำหนักเบา และราคาที่ไม่แพงจนเกินไป และมีความสามารถไม่เพียงแค่ออร์แกไนเซอร์ คือ สามารถเขียนโปรแกรมเพิ่มเติม ทำให้มีโปรแกรมสำหรับปาล์มออกมาสนับสนุนความต้องการของผู้ใช้เป็นจำนวนมาก

แต่เนื่องด้วยระบบปฏิบัติการบนปาล์ม (Palm OS) นั้น ไม่สนับสนุนในการพิมพ์เอกสาร ทำให้มีผู้พัฒนาโปรแกรมช่วยในงานพิมพ์บนปาล์มออกมา เช่น PalmPrint, IR Print suite แต่โปรแกรมดังกล่าวก็ยังไม่สนับสนุนภาษาไทย ไม่สามารถพิมพ์เอกสารภาษาไทยได้ เนื่องจากโปรแกรมไม่มีฟอนต์ภาษาไทย และไม่มีฟังก์ชันในการจัดพิมพ์ภาษาไทย นอกจากนั้นยังไม่สนับสนุนการพิมพ์โดยผ่านพอร์ตอนุกรมอย่างเต็มที่ กล่าวคือต้องอาศัยโปรแกรมเช่นไฮเปอร์เทอร์มินอล (HyperTerminal) ในการรับข้อมูลจากปาล์มแล้วส่งต่อไปยังเครื่องพิมพ์

ระบบภาษาไทยสำหรับปาล์มนั้น ปัจจุบันมีผู้ผลิตซอฟต์แวร์รายใหญ่อยู่สองค่าย คือ ThaiPos และ ThaiHack ซึ่งทำให้สามารถแสดงภาษาไทย อินพุตข้อมูลเป็นภาษาไทย โดยมีการกำหนดกราฟฟิตี้ (Graffiti) ภาษาไทย รวมทั้งแสดงเมนู และยูสเซอร์อินเทอร์เฟซเป็นภาษาไทย แต่ก็ยังไม่สนับสนุนในเรื่องการพิมพ์เอกสารภาษาไทย

จากแนวคิดตามข้างต้น ทำให้เกิดปริญญานิพนธ์นี้ขึ้น เพื่อพัฒนาโปรแกรมที่สามารถพิมพ์เอกสารเป็นภาษาไทย จากเครื่องปาล์มได้

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์คือ

(1) หาหนทางในการพิมพ์เอกสารภาษาไทยจากเครื่องปาล์ม
(2) เพื่อเข้าใจการทำงานของปาล์ม และการเขียนโปรแกรมบนปาล์ม
(3) เพื่อเข้าใจการสั่งงาน และควบคุมการทำงานของเครื่องพิมพ์
(4) เพื่อสร้างโปรแกรมที่มีความสามารถในการพิมพ์เอกสารจากแอพพลิเคชันมาตรฐานของปาล์มที่เป็นภาษาไทยได้

1.3 ขอบเขตของปริญญานิพนธ์

ขอบเขตการทำงานของปริญญานิพนธ์นี้ ได้แก่

(1) ศึกษาการทำงานของปาล์มโอเอส (Palm OS) และแอพพลิเคชันบนปาล์ม เพื่อเลือกวิธีการที่ดีที่สุดนำมาใช้ออกแบบและพัฒนาโปรแกรม

(2) ศึกษาการพัฒนาโปรแกรมบนเครื่องปาล์ม และสภาวะแวดล้อมในการพัฒนาโปรแกรมบนปาล์ม พร้อมทั้งเลือกวิธีการที่ใช้ในการพัฒนาโปรแกรม

(3) ศึกษาการสั่งงานเครื่องพิมพ์ และการจัดรูปแบบเอกสาร

(4) ออกแบบ และพัฒนาโปรแกรมที่สามารถพิมพ์เอกสารจากแอพพลิเคชันมาตรฐานของปาล์มที่เป็นภาษาไทยได้

1.4 ขั้นตอนการดำเนินงาน

(1) ศึกษาการทำงานของปาล์ม สถาปัตยกรรมของปาล์ม เพื่อหาหนทางที่ดีที่สุดในการพิมพ์เอกสารภาษาไทยจากปาล์ม

(2) ศึกษาการพัฒนาโปรแกรมบนเครื่องปาล์ม เพื่อหาวิธีการที่เหมาะสมที่สุดในการพัฒนาโปรแกรมโดยไม่ยาก หรือไม่ซับซ้อนเกินไป แต่สนับสนุนการเขียนโปรแกรมตามที่ต้องการใช้งานได้

(3) ขั้นออกแบบโปรแกรม ออกแบบโปรแกรม การทำงานของโปรแกรม หน้าจอของโปรแกรม ให้ผู้ใช้สามารถใช้งานง่าย ไม่ยุ่งยากซับซ้อน เรียนรู้ง่าย

(4) พัฒนาโปรแกรม ตามที่ได้ออกแบบไว้

(5) ทดสอบโปรแกรม ว่าสามารถใช้งานได้ตรงถูกต้องตามความต้องการ และมีคุณภาพในการพิมพ์เป็นที่น่าพอใจ

บทที่ 2
ปาล์ม

2.1 ประวัติความเป็นมาของปาล์ม

ในปี 1994 บริษัท Palm Computing เป็นบริษัทเล็กๆ ก่อตั้งโดย Jeff Hawkins และ Donna Dubinsky เดิมทีขายโปรแกรมจดจำลายมือ (Handwriting recognition: HWR) ให้กับ Apple Newton MessagePad ซึ่งเรียกว่า 'Graffiti' (หรือ Unistroke โดยซีรอกซ์) กราฟฟิตีนี้ขายดีมากและเป็นที่นิยมสำหรับผู้ใช้เครื่องนิวตัน ปาล์มจึงเสี่ยงผลิตพีดีเอ (PDA) ของตัวเองออกวางตลาด เป็นออร์แกไนเซอร์ขนาดเล็ก มีชื่อเรียกว่าไพลอต (Pilot) และมีกราฟฟิตี้เป็นโปรแกรมจดจำลายมือหลัก ในขณะนั้นไพล็อตนี้ถูกกล่าวว่าเป็นเครื่องมือในการโฆษณาผลิตภัณฑ์หลักของบริษัทคือกราฟฟิตี้ ปรากฏว่าผิดคาด เพราะออร์แกไนเซอร์ขนาดเล็กสร้างความประทับแก่ผู้ใช้ซึ่งไม่เคยใช้ผลิตภัณฑ์ประเภทนี้มาก่อน และสามารถแลกเปลี่ยนข้อมูลกับเครื่องแมคอินทอช และพีซีได้เป็นอย่างดี ราคาถูก ใช้งานง่าย เพราะอิงอยู่บนซอฟต์แวร์จดจำลายมือที่มีประสิทธิภาพ ทั้งหมดนี้คือสาเหตุที่ทำให้ปาล์มประสบความสำเร็จ จากนั้นปาล์มคอมพิวติง ถูกซื้อโดยบริษัท ยูเอส โรโบติกส์ (US Robotics Inc.) ต่อมาถูกซื้อโดย 3Com อีกทอดหนึ่ง Jeff และ Donna ก็เริ่มต้นใหม่อีกครั้ง เหมือนวันวาน โดยก่อตั้งบริษัทแฮนด์สปริง (HandSpring)

ต่อมา 3Com แยกตัวมาเป็นบริษัท Palm Inc. เพื่อหันมาผลิตผลิตภัณฑ์ปาล์มโดยเฉพาะ มีการเปลี่ยนชื่อเครื่องปาล์ม 'ไพล็อต' เนื่องจากยูเอส โรโบติกส์ ซึ่งต่อมากลายเป็น 3Com ถูกฟ้องโดย Pilot Pen Corperation เนื่องจากใช้ชื่อเลียนแบบ สุดท้ายก็ถูกเปลี่ยนชื่อเป็นปาล์มไพลอต (PalmPilot) สั้นๆ คือปาล์ม(Palm) หลังจากเสร็จสิ้นคดีความเรื่องชื่อของไพลอต ในปี 1998 3Com ก็ได้สู้ความกับไมโครซอฟต์อีกครั้ง บิล เกต ต้องการวางตลาดผลิตภัณฑ์ในชื่อปาล์มพีซี PalmPC ในที่สุดไมโครซอฟต์โดนบังคับให้เปลี่ยนชื่อจากปาล์มพีซี เป็น ปาล์มไซส์พีซี (Palm-Sized PC) และในที่สุดก็กลายเป็นพ็อกเก็ตพีซี (Pocket PC)

ปาล์มโอเอส (PalmOS) ถูกก่อตั้งด้วย Hawkins และ Dubinsky ยังยึดส่วนแบ่งตลาดผลิตภัณฑ์ซึ่งช่วยให้ผู้ใช้สามารถจัดการข้อมูลส่วนตัว ได้โดยไม่ยุ่งยาก

2.2 การใช้งานปาล์ม

ปาล์มเป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่หลักในการเป็นออร์แกไนเซอร์ ก็คือการจัดระบบระเบียบส่วนตัว แต่ด้วยความ สามารถที่มากกว่านั้นปาล์มก็มีคุณสมบัติในการเป็นคอมพิวเตอร์ขนาดเล็ก (ขนาดปาล์ม จึงได้ชื่อว่า Palm-sized Computers) หรือ ขนาดพกพาไปไหนมาไหนได้ (จึงได้ชื่อว่า Handheld Computers) คุณลักษณะของอุปกรณ์อิเล็กทรอนิกส์จำพวกที่ทำหน้าที่ ในการ เป็นออร์แกไนเซอร์ ก็สามารถเรียกได้ว่าเป็นอุปกรณ์จำพวก PDA (Personal Digital Assistants) ก็คือเลขานุการส่วนตัวในรูปแบบของ เครื่องมือทางดิจิตอลนั่นเอง

ปาล์มมีโปรแกรมสำเร็จรูปมาตรฐาน มาพร้อมกับระบบปฏิบัติการ 5 โปรแกรม คือ

1. **Address Book** : ใช้บันทึกที่อยู่ เบอร์โทรศัพท์ การติดต่อ รายละเอียดของบุคคล
2. **Date Book** : ใช้บันทึกการนัดหมาย หรือกิจกรรมที่จะต้องทำ ตารางเวลา
3. **Expense** : ใช้บันทึกรายการใช้จ่าย
4. **Memo Pad** : ใช้บันทึกข้อความสั้น บันทึกช่วยจำ
5. **Todo List** : ใช้บันทึกงานที่จะต้องทำ

ปาล์มสามารถแสดงผลเป็นกราฟฟิก โดยผ่านแอลซีดี (LCD) สามารถอินพุตข้อมูลโดยสัมผัสที่หน้าจอ และสามารถป้อนข้อมูลเป็นตัวเลขและตัวอักษร โดยปาล์มสามารถรู้จำลายเขียนที่เรียกว่ากราฟฟิตี้ (Graffiti)

มีแท่นใช้สำหรับชาร์จไฟ และพอร์ตอนุกรมสำหรับต่อกับคอมพิวเตอร์เรียกว่าเครเดิล (Cradle)

สามารถถ่ายข้อมูลจากปาล์มเข้าเครื่องคอมพิวเตอร์หรือถ่ายข้อมูลจากเครื่องคอมพิวเตอร์สู่ปาล์ม เพื่อปรับปรุงข้อมูลให้ทันสมัยหรือทำสำเนาข้อมูลเรียกว่าการฮอตซิงก์ (Hot sync)

ปาล์มมีความสามารถเชื่อมต่อกับโมเด็ม หรือโทรศัพท์มือถือ ทำให้สามารถท่องเว็บเพจผ่านเครื่องปาล์ม ส่งถ่ายข้อมูล รวมทั้งฮอตซิงก์ในระยะไกลได้ ปาล์มบางรุ่นมีเครือข่ายไร้สายในตัวเอง เพียงแค่ชักเสาอากาศก็สามารถเชื่อมต่อเข้าสู่เครือข่ายได้

ลักษณะพิเศษของปาล์ม

1. ปาล์มเป็นเครื่องที่ถูกออกแบบมาให้มีลักษณะการใช้งานที่ง่าย ไม่ซับซ้อน สิ่งที่ทำให้ปาล์มมีลักษณะเด่นก็คือปาล์มโอเอสที่มีความเสถียรสูงโดยมีรายงานจาก การทดลองหลายแห่งในต่างประเทศแล้วว่าปาล์มโอเอสเป็นระบบปฏิบัติการที่ใช้งานง่ายโดยผู้ใช้ไม่จำเป็นต้องมีความรู้ทางด้านคอมพิวเตอร์มากนัก และระบบปฏิบัติการที่ไม่ซับซ้อนนี้เองจึงเป็นส่วนผลักดันการในด้านลดปัญหาการใช้งานในตัวผู้ใช้เอง จึงเป็นการทำให้เพิ่มประสิทธิภาพในการทำงานของผู้ใช้ให้มากขึ้น

2. ปาล์มมีการทำงานที่รวดเร็วไม่ต้องเสียเวลาในการประมวลผลนาน ถึงแม้ว่าซีพียูของปาล์มจะมีความเร็วต่ำ

3. ปาล์มมีอัตราการสิ้นเปลืองพลังงานน้อย สามารถใช้งานได้ถึง 2 อาทิตย์ ต่อการชาร์จไฟ 1 ครั้ง

4. ปาล์มใช้ระบบปฏิบัติการที่รวดเร็ว ไม่ต้องรอการประมวลผลที่นาน โดยคุณสามารถเรียกใช้งานได้สะดวก เพียงแต่เปิดสวิทช์ใช้งานของเครื่อง คุณก็สามารถใช้งานได้ทันที

5. ปาล์มมีพันธมิตรมากมาย ทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์เช่น ซอฟต์แวร์ของปาล์มจะมีออกใหม่ทุกวันโดยซอฟต์แวร์ส่วนมากจะเป็นทั้ง ฟรีแวร์และแชร์แวร์ ที่คุณสามารถลองดาวน์โหลดไปใช้ก่อนได้ และราคาซอฟต์แวร์ของปาล์มนั้นมีราคาไม่สูงจนเกินไป โดยจะมีราคาเฉลี่ยอยู่ประมาณ 3-25 $ ซึ่งคุณสามารถซื้อ ซอฟต์แวร์ที่ต้องการผ่านทางเว็บไซต์ได้ ซึ่งเว็บไซต์เกี่ยวกับปาล์ม ในปัจจุบันนั้นมีมากกว่า 2000 แห่ง ทั่วโลก

6. โปรแกรมต่างๆในปาล์มใช้พื้นที่ในการเก็บไม่มาก โดยหน่วยความจำขนาด 2 เมกะไบต์ นั้น สามารถเก็บโปรแกรมใช้งานได้มากถึง 30 โปรแกรม หรือหากเป็นรุ่นที่มีหน่วยความจำ 8 เมกะไบต์ ก็จะสามารถเก็บโปรแกรมได้นับร้อยโปรแกรมเลยทีเดียว

7. การซิงโครไนซ์ข้อมูลที่ง่ายและสะดวก การปรับปรุงข้อมูลให้ตรงกันระหว่างปาล์มกับเครื่อง คอมพิวเตอร์นั้นเป็นสิ่งที่ต้องคำนึงสำหรับการใช้งานของผู้ใช้ เพื่อข้อมูลที่ถูกต้องและแน่นอน สำหรับปาล์มเองแล้ว ปาล์มมีความสามารถในการซิงโครไนซ์ข้อมูลกับซอฟต์แวร์ได้หลายประเภท เช่น MS Outlook , LotusNotes GroupWise หรือแม้แต่ซอฟต์แวร์ฟรี ที่มาพร้อมกับปาล์มอย่างปาล์มเดสก์ท็อป (Palm Desktop) พร้อมกับระบบการสำรองข้อมูลที่มีประสิทธิภาพเพื่อลดการสูญหายของข้อมูลที่สำคัญ

8. กราฟฟิตี้ที่ยอดเยี่ยม สำหรับการป้อนข้อมูลลงใน เครื่องพีดีเอทั่วไปนั้นจะสามารถทำได้หลายวิธี ซึ่งการป้อนข้อมูลด้วยการเขียนนั้นจะเรียกว่ากราฟฟิตี้ ซึ่งเป็นการเขียนด้วยภาษาที่แตกต่างจากลายมือทั่วไปสักเล็กน้อย ซึ่งผู้ใช้สามารถเรียนรู้ด้วยเวลาอันสั้น และสำหรับปาล์มเองแล้วการป้อนข้อมูลด้วยการเขียน หรือกราฟฟิตี้นั้นจะสามารถอ่านลายมือของผู้ใช้ได้ถูกต้องและแม่นยำกว่าพีดีเอรุ่นอื่นๆ

2.3 ส่วนประกอบด้านฮาร์ดแวร์ของปาล์ม

ปาล์มใช้หน่วยประมวลผลกลางของ Motorola Dragonball EZ (MC68EZ328) ทำงานที่ความถี่ 16 เมกะเฮิรตซ์ มี มีการเก็บค่าเป็นแ บบ Big-Endian ซึ่งตรงกันข้ามกับหน่วยประมวลผลตระกูล x86 มีรอมสำหรับเก็บระบบปฏิบัติการ และ แรมเพื่อเก็บโปรแกรมและข้อมูล โดยขนาดของแรมนี่มีตั้งแต่ 2, 4, 8 เมกะไบต์ ขนาดแรมที่ใหญ่ขึ้นทำให้รองรับโปรแกรมที่มีขนาดใหญ่ขึ้นได้จำนวนมากขึ้น สามารถต่ออุปกรณ์อินพุต/เอาท์พุต โดยผ่านพอร์ตอนุกรม หรือ อินฟาเรด (สำหรับเครื่องปาล์มคอมแพตทิเบิลบางรุ่นยังสามารถใช้งานสมาร์ทสล๊อต หรือแฟลชเมโมรี่ได้ด้วย)

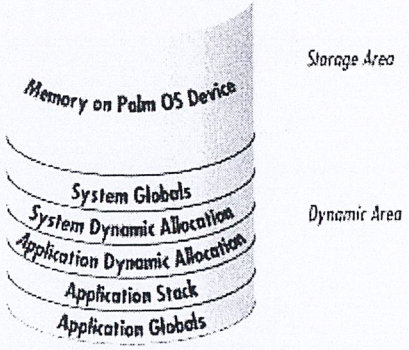
2.4 หน่วยความจำของปาล์ม

สามารถแบ่งหน่วยความจำของปาล์มออกได้เป็นสองประเภทคือ

- Dynamic Memory Allocation ใช้เก็บค่าตัวแปรเช่นสแต็ก เมื่อโปรแกรมทำงานอาจจะมีการจองหน่วยความจำส่วนนี้ขึ้นมาเพื่อเก็บข้อมูล ซึ่งจะถูกลบไปเมื่อจบโปรแกรมหรือรีเซ็ต คล้ายกับแรมของระบบปฏิบัติการทั่วไป
- Permanent storage ใช้เก็บโปรแกรมที่ดาวน์โหลดลงปาล์ม และข้อมูลที่ผู้ใช้สร้างขึ้น เช่น Todo, Address book หน่วยความจำส่วนนี้จะไม่ถูกลบเมื่อรีเซ็ต คล้ายฮาร์ดดิสก์

แอดเดรสของหน่วยความจำที่หน่วยประมวลผลกลางของปาล์มอ้างได้มีขนาด 32 บิต ทำให้อ้างอิงหน่วยความจำที่มีขนาดใหญ่ที่สุดได้ถึง 4 กิกะไบต์ แม้ว่าบัสภายนอกของปาล์มมีขนาด 16 บิต เพื่อลดต้นทุนในการผลิต ถึงกระนั้นก็ยังไม่ทำให้การทำงานด้อยลงแต่อย่างใด เมื่อมีการอ้างอิงหน่วยความจำแบบ 32 บิต หน่วยควบคุมการทำงานของบัสจะแตกการอ้างอิงของบัสออกมาเป็น 16 บิต เมื่อทำการเขียน/อ่านโดยอัตโนมัติ

หน่วยความจำทั้งสองนี้จะถูกจองเรียกชังก์ (Chunk) คือส่วนย่อยๆของหน่วยความจำ ในดาต้าเบสอาจจะประกอบด้วยชังก์นี้หลายๆอัน



รูปที่ 1-1 การแบ่งหน่วยความจำของปาล์ม

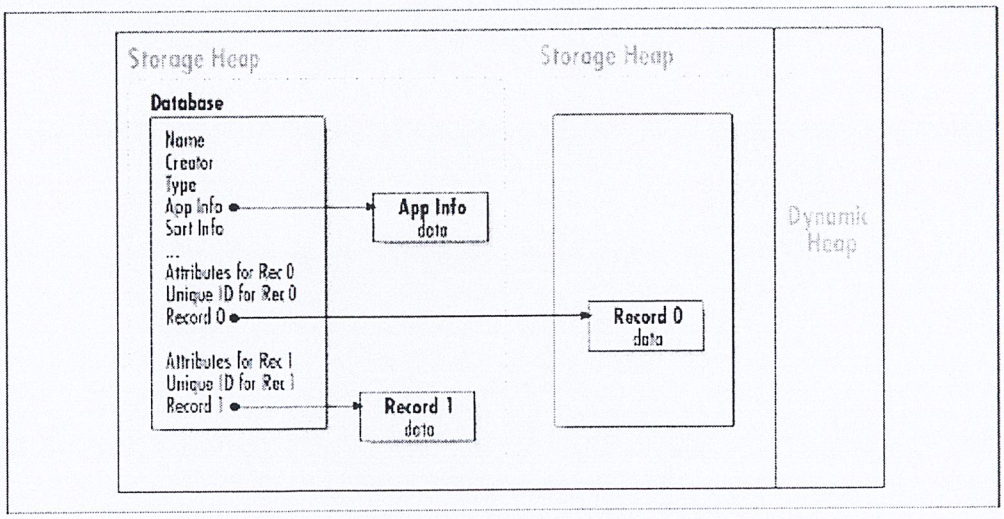
2.5 ฮีป

ฮีป (Heap) คือพื้นที่ในหน่วยความจำที่ติดกันกระกอบด้วยชังก์หลายๆชังก์ เมื่อแอพพลิเคชันทำงานกับหน่วยความจำ (จอง,ขยาย,ล็อก) จะทำงานกับชังก์แอพพลิเคชันสามารถระบุได้ว่าจองภายในสตอเรจฮีป (Storage heap) หรือ ไดนามิกฮีป (Dynamic heap) โดยเมโมรี่แมเนเจอร์ (Memory Manager) จะจัดการแต่ละส่วนแยกกัน และมีการปรับย้ายเมื่อเกิดแฟรกเมนต์ (Fragment) เป็นการเพิ่มเนื้อที่ว่างให้ใหญ่ขึ้น

ระบบจะใช้ไดนามิกฮีปนี้เพื่อเก็บข้อมูลไดนามิก เช่น ตัวแปรโกลบอล เมื่อระบบต้องการใช้หน่วยความจำไดนามิกเช่น (ทีซีพี/ไอพี , ไออาร์ดีเอ) สแต็กของแอพพลิเคชัน หน่วยความจำชั่วคราว หน่วยความจำเมื่อใช้ฟังก์ชัน memHandleNew

อีกส่วนหนึ่งของหน่วยความจำที่ไม่ถูกใช้เป็นไดนามิกฮีป จะถูกกำหนดให้เป็นสตอเรจฮีป โดยจะใช้หน่วยความจำส่วนนี้ในการเก็บข้อมูลที่ไม่สูญหาย เช่น กรนัดหมาย เมมโม สมุดปันทึก สามารถเข้าถึงข้อมูลส่วนนี้ได้โดยดาต้าแมเนเจอร์หรือรีซอร์สแมเนเจอร์ ขึ้นอยู่กับว่าข้อมูลในส่วนนี้เป็นด้าต้าเบส หรือ รีซอร์ส

2.6 ไฟล์และดาต้าเบส



รูปที่ 1-2 การจัดเก็บดาต้าเบส และเรคอร์ด

ข้อแตกต่างของปาล์มโอเอสกับระบบปฏิบัติการทั่วไปอีกอย่างหนึ่งคือ ปาล์มโอเอสไม่จำเป็นต้องสำเนาหน่วยความจำจากหน่วยความจำถาวรไปยังหน่วยความจำแบบไดนามิก (Dynamic Memory) ก่อนเพื่อรันโปรแกรม อนึ่งปาล์มสามารถรันโปรแกรมได้จากหน่วยความจำถาวร โดยตรง ทำให้ประหยัดหน่วยความจำไม่ต้องสินเปลืองหน่วยความจำอีกส่วนเพื่อใช้เป็นบัฟเฟอร์

ข้อมูลของปาล์มถูกแตกเป็นส่วนย่อยๆหลายส่วน ที่มีขนาดคงที่ เรียกเรคอร์ด ซึ่งสามารถกระจายไปเนื้อที่หน่วยความจำ ทำให้การเพิ่ม ลบ เปลี่ยนขนาดไม่กระทบกับเรคอร์ดอื่น เรคอร์ดก็คือชังก์ ซึ่งถูกจองโดยเมโมรีแมเนเจอร์นั่นเอง

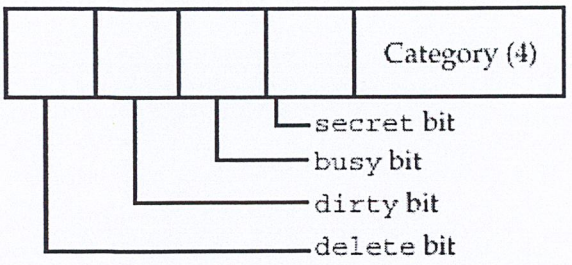
ดาต้าเบสคือเรคอร์ดหลายๆเรคอร์ดที่มีความสัมพันธ์กัน โดยเรคอร์ดหนึ่งๆจะเป็นส่วนประกอบของดาต้าเบสได้ตัวเดียว แต่ดาต้าเบสหนึ่งสามารถมีหลายเรคอร์ด ดาต้าเบสนี้ก็คล้ายกับไฟล์ในระบบปฏิบัติการอื่นๆนั่นเอง

เฮดเดอร์ของดาต้าเบสจะประกอบด้วย

- Name เป็นฟิลด์ซึ่งเก็บชื่อของดาต้าเบส
- attributes เป็นฟิลด์ซึ่งใช้เก็บแฟลกของดาต้าเบส
- version เก็บเวอร์ชันของดาต้าเบสนั้น
- modificationNumber จะเพิ่มขึ้นทุกครั้งที่มีการแก้ไข เปลี่ยนแปลง เพิ่มเติมเรคอร์ดในดาต้าเบสนั้นๆ ทำให้ทราบได้ว่ามีการแก้ไขเปลี่ยนแปลงสำหรับดาต้าเบสที่มีการใช้งานร่วมกัน
- appInfoID โดยทั่วไปประกอบด้วยชื่อประเภทของดาต้าเบสนั้น
- sortInfoID ใช้เก็บลำดับของเรคอร์ดที่เรียงกันในลักษณะต่างกัน เช่น Address Book อาจจะเรียงกันด้วยชื่อบริษัท หรือชื่อคน โดยส่วนใหญ่แล้วโปรแกรมจะไม่ใช้
- numRecord เก็บจำนวนเรคอร์ดที่เป็นสมาชิกของดาต้าเบสนั้น
- type มีขนาด 4 ไบต์ ใช้แยกแยะ ดาต้าเบสที่ใช้เก็บโปรแกรม และดาต้าเบสที่ใช้เก็บข้อมูล
- creator มีขนาด 4 ไบต์ ใช้บอกว่าดาต้าเบสนั้นเป็นดาต้าเบสของแอพพลิเคชันใด เมื่อลบแอพพลิเคชันใดๆ ปาล์มโอเอสจะลบดาต้าเบสที่มี creator ตรงกับ แอพพลิเคชันนั้นๆด้วย แต่ละโปรแกรมจึงจำเป็นต้องมี creator ไม่ซ้ำกัน ซึ่งสามารถลงทะเบียนได้ที่ http://www.palm.com/dev/crid

แต่ละเรคอร์ดจะถูกเก็บเป็นเรคอร์ดลิสต์ มีข้อมูลดังต่อไปนี้

- local ID ตำแหน่งที่เรคอร์ดนั้นถูกเก็บ
- Unique ID มีขนาด 3 ไบต์ ใช้อ้างอิงเรคอร์ดนั้นๆ ในดาต้าเบสใดๆจะไม่มี ID ของเรคอร์ดที่ซ้ำกัน ซึ่งค่าตัวเลขนี้ถูกกำหนดโดย Data Manager ตั้งแต่เมื่อเรคอร์ดนั้นถูกสร้างขึ้น
- Attribute มีขนาด 1 ไบต์ ประกอบด้วย 4 บิต คือ Delete bit, Archive bit, Busy Bit, Secret (หรือ Private) Bit



รูปที่ 1-3 แอตทริบิวต์ของเรคอร์ดในดาต้าเบสของปาล์ม

เมื่อผู้ใช้ลบ หรือสำเนาเรคอร์ดบนปาล์มโอเอส จะดำเนินการคือ

- delete bit จะถูกเซ็ต เมื่อแต่ยังปรากฏอยู่ในดาต้าเบสเฮดเดอร์ จนกว่าจะมีการฮอตซิงก์ครั้งต่อไป
- dirty bit จะถูกเซ็ต เมื่อมีการแก้ไขเรคอร์ด
- busy bit จะถูกเซ็ต เมื่อมีการล็อกเพื่อเขียนหรืออ่านเรคอร์ด
- secret bit ถ้าถูกเซ็ต เรคอร์ดนั้นจะไม่ถูกแสดงจนกว่าจะมีการป้อนพาสเวิร์ดถูกต้อง

เมื่อเรคอร์ดถูกลบ เนื้อที่หน่วยความจำที่ใช้เก็บข้อมูลจะถูกฟรี local ID ของเรคอร์ดนั้นจะถูกตั้งค่าเป็น 0 และ delete bit จะถูกเซ็ต เมื่อมีการซิงโครไนซ์ระหว่างปาล์มกับพีซี จะเป็นการง่ายที่จะทราบว่าเรคอร์ดถูกลบ

บทที่ 3
การพัฒนาโปรแกรมบนปาล์ม

3.1 ข้อแตกต่างของการพัฒนาโปรแกรมบนปาล์ม

- **ขนาดหน้าจอ** ปาล์มมีขนาดหน้าจอเล็กกว่าการเขียนโปรแกรมบนเดสก์ท็อป เพราะฉะนั้นจำเป็นต้องมีการออกแบบยูสเซอร์อินเทอร์เฟซให้รัดกุม กะทัดรัด

- **ความเร็วในการทำงาน** สำหรับโปรแกรมบนเดสก์ผู้ใช้อาจจะสามารถรอได้ แต่โปรแกรมบนเครื่องคอมพิวเตอร์มือถือ ซึ่งผู้ใช้ต้องใช้ช่วยในการทำงาน ช่วยติดต่อสื่อสารกับบุคคล ผู้ใช้จึงไม่สามารถรอได้ ควรออกแบบให้ผู้ใช้ ใช้งานได้ง่าย สะดวกที่สุด

- **การเชื่อมต่อกับคอมพิวเตอร์** แอพพลิเคชันบนปาล์มต้องสามารถเชื่อมต่อกับคอมพิวเตอร์เพื่อ ส่งถ่ายข้อมูลมายังคอมพิวเตอร์

- **การป้อนข้อมูล** ปาล์มไม่มีคีย์บอร์ดหรือเมาส์ การป้อนข้อมูลให้ปาล์มทำโดยการสัมผัสจอหรือใช้กราฟฟิตี้ ฉะนั้นไม่ควรให้ผู้ใช้ป้อนตัวอักษรยาวเกินไป ยูสเซอร์อินเทอร์เฟซใช้งานได้ง่ายที่สุด

- **พลังงาน** เครื่องปาล์มทำงานโดยใช้พลังงานจากแบตเตอรี่ ต่างจากเดสก์ท็อป ดังนั้นงานที่ใช้การคำนวณมากๆ หรือใช้เวลานานๆควรปล่อยให้เป็นหน้าที่ของเครื่องพีซี

- **หน่วยความจำ** ปาล์มโอเอสมีข้อจำกัดในด้านหน่วยความจำ แตกต่างกับเดสก์ท็อปอีกทั้งไม่มีดิสก์ไดรว์เหมือนเครื่องเดสก์ท็อป ดังนั้นจำเป็นต้องมีการบริหารหน่วยความจำที่ดี เนื่องจากมีข้อจำกัดในด้านพลังงาน และหน่วยความจำจึงต้องมีการอ็อพติไมซ์แอพพลิเคชันให้เร็วและมีประสิทธิภาพมากที่สุดเท่าที่จะเป็นไปได้ โดยคำนึงถึงขนาดของฮีปเป็นอันดับแรก รองลงมาคือความเร็วและขนาดของโปรแกรมตามลำดับ

- **ระบบไฟล์** เนื่องจากปาล์มมีหน่วยความจำ และสามารถซิงโครไนซ์กับเดสก์ท็อปได้ จึงไม่มีการเก็บไฟล์เป็นแบบไฟล์ซิสเต็ม แต่ถูกเก็บภายในหน่วยความจำเป็นเรคอร์ดและรวมเรคอร์ดที่เกี่ยวข้องกันเป็นดาต้าเบส ดาต้าเบสนี้ก็คือไฟล์ของปาล์มนั่นเองโดยจะแตกเป็นส่วนๆไม่รวมกันเป็นส่วนเดียว เพื่อเป็นการประหยัดเนื้อที่

- **แบ็กเวิร์ดคอมแพตทิเบิล (Backward Compatible)** แอพพลิเคชันบนปาล์มต้องสามารถใช้ได้กับปาล์มโอเอสหลายๆเวอร์ชัน และฮาร์ดแวร์หลายๆชนิด การอัพเกรดทำได้ยากสำหรับปาล์มดังนั้นจึงต้องออกแบบโปรแกรมให้คอมแพตทิเบิลกับปาล์มโอเอสและ/หรือฮาร์ดแวร์ เวอร์ชันก่อนหน้าด้วย

3.2 การออกแบบโปรแกรมที่ดี

3.2.1 ทำงานเข้ากันได้กับปาล์มโอเอส

แอพพลิเคชันจะทำงานร่วมกับปาล์มโอเอสได้ดี มีข้อพึงปฏิบัติคือ

- มีการจัดการ lunch code sysAppLaunchCmdNormalLaunch
- ไม่ทำงานกับ lunch code อื่น ที่ไม่จำเป็นต้องใช้งาน
- จัดการกับค่าตัวแปรของระบบที่ผู้ใช้ตั้งไว้ เช่น รูปแบบวันที่ รูปแบบเวลา
- รับเมสเสจของระบบเช่น สัญญาณเตือน ตั้งปลุก เตือนแบตเตอรี่อ่อน
- ไม่ทำงานก้าวก่าย เปลี่ยนแปลงพื้นที่เขียน ปุ่มเปิด/ปิด
- ไม่ทำงานก้าวก่าย การชิฟตัวอักษรกราฟฟิตี้
- บันทึกค่าพรีเฟอร์เรนซ์ (Preference) ของโปรแกรมก่อนเลิกการทำงาน
- ถ้ามีการใช้พอร์ตอนุกรม ต้องมีการฟรีพอร์ตหลังจากเลิกใช้งาน เพื่อให้โปรแกรมอื่นเช่น ฮอตซิงก์ ใช้งานพอร์ตได้
- ไพรเวทเรคอร์ด (Private Record) ไม่ควรถูกค้นหา ควรจะถูกซ่อนไว้เมื่อไม่มีการป้อนยูสเซอร์และพาสเวิร์ดที่ถูกต้อง
- ชื่อของโปรแกรม ที่ตั้งในพาเนล PlamRez จะถูกใช้เมื่อฮอตซิงก์ ใช้ใน About Box ใช้เมื่อดูเนื้อที่หน่วยความจำ และปรากฏในดาต้าเบสเฮดเดอร์
- ตั้งค่าชื่อโปรแกรมให้ตรงกันทุกๆที่ที่มีการแสดงค่า รวมทั้งในดาต้าเบส
- ดาต้าเบสที่ถูกลบจะต้องไม่ถูกแสดงผล
- เคตากอรี (Categories) ต้องไม่เกิน 15 เมื่อ "Unfiled" ก็จะมีเคตากอรี่เป็น 16 พอดี
- คำนึงถึงสถานะมาตรฐานของโปรแกรม เช่น เมื่อเปิดโปรแกรม Datebook จะมีการเลือกวันที่ปัจจุบันเป็นมาตรฐาน

3.2.2 กฎการตั้งชื่อ

- ฟังก์ชันต้องขึ้นต้นด้วยตัวใหญ่
- ฟังก์ชันที่มีเรียกใช้งานแมเนเจอร์ต่างๆ จะขึ้นต้นด้วยตัวอักษรตัวใหญ่สองถึงสามตัวเช่น "Ctl" สำหรับ Control Function ,"Ftr" สำหรับฟังก์ชันที่เป็นฟังก์ชันในกลุ่ม Feature Manager
- เหตุการณ์และค่าคงตัวจะขึ้นด้วยตัวอักษรตัวเล็ก
- ตัวแปรสตักเจอร์ขึ้นต้นด้วยตัวอักษรตัวเล็ก
- ตัวแปร โกลบอล (Global variable) จะขึ้นต้นด้วยตัวอักษรตัวใหญ่
- การประการประเภทตัวแปร (Typedef) จะขึ้นต้นด้วยตัวอักษรตัวใหญ่ และจบด้วยคำว่า "Type" เช่น "DateFormatType"

- สมาชิกของตัวแปรประเภทเอนิวเมอเรต (Enumerated) จะนำหน้าด้วยตัวอักษรตัวเล็ก แล้วตามด้วยชื่อที่ขึ้นต้นด้วยตัวอักษรตัวใหญ่ ดังตัวอย่าง

```
enum formObjects {
   frmFieldObj,
   frmControlObj,
   frmListObj,
   frmTableObj,
   frmBitmapObj,
   frmLineObj,
   frmFrameObj,
   frmRectangleObj,
   frmLabelObj,
   frmTitleObj,
   frmPopupObj,
   frmGraffitiStateObj,
```

รูปที่ 3-1 ตัวอย่างการประกาศสมาชิกประเภทเอนิวเมอเรต

3.2.3 การเพิ่มประสิทธิภาพโปรแกรม

เนื่องจากข้อจำกัดด้านทรัพยากรของปาล์ม ทั้งหน่วยความจำและฮีป การบริหารทรัพยากรจึงเป็นสิ่งสำคัญอย่างยิ่งยวด ในการเขียนโปรแกรมบนปาล์ม

- การจองหน่วยความจำ พยายามให้เกิดแฟรกเมนต์น้อยที่สุด
- เรียงข้อมูลเมื่อต้องการใช้ เรียงข้อมูลเมื่อต้องการใช้เท่านั้นเป็นการประหยัดหน่วยความจำและทำให้โปรแกรมง่ายขึ้น
- หน่วยความจำไดนามิก อาจทำให้เกิดปัญหาคอขวด จึงควรหลีกเลี่ยงการพุทข้อมูลประเภทสตรักเจอร์ขนาดใหญ่ลงใน สแต็ก
- ปรับขนาดรูทีนย่อย ไม่ควรให้มีการกระโดดเกิน 32 K
- ประหยัดฮีปที่สุดท่าที่จะเป็นไปได้โดย
  - จองหน่วยความจำเป็นชังก์ แทนที่จะประกาศเป็นหน่วยความจำโกลบอล
  - สลับหน้าจอยูสเซอร์อินเทอร์เฟซ แทนที่จะเปิดหลายๆหน้าจอซ้อนกัน
  - แก้ไขข้อมูลในดาต้าเบสโดยตรง มิใช่ทำสำเนามาอีกชุดหนึ่งเพื่อแก้ไข
- ไม่ใช้เนื้อที่สแต็กเก็บข้อมูลขนาดใหญ่ การดีบักโปรแกรมที่ผิดพลาดจากฮีปทำได้ยากมาก

3.2.4 กำหนดค่า Creator ID

Creator ID มีความสำคัญ ใช้เป็นตัวเลขบ่งชี้ว่าดาต้าเบสกับแอพพลิเคชันที่สัมพันธ์กันดังที่กล่าวไว้แล้วดังข้างต้น จึงควรมีการกำหนดค่า Creator ID โดยสามารถจดทะเบียนค่า Creator ID นี้ได้จากเว็บไซต์ http://www.palm.com/devzone/crid/cridsub.html

3.2.5 การทำงานกับดาต้าเบส

การทำงานกับดาต้าเบสทำให้โปรแกรมทำงานได้เร็วขึ้น และไม่เกิดปัญหาจากการซิงโครไนซ์

- เมื่อผู้ใช้ลบเรคอร์ด ใช้ฟังก์ชัน **DmDeleteRecord** เพื่อลบข้อมูล มิใช่ **DmRemoveRecord** ซึ่งจะลบเรคอร์ดนั้นออกจากหน่วยความจำซึ่งอาจจะเกิดปัญหาเมื่อทำการฮอตซิงก์
- เก็บข้อมูลในดาต้าเบสให้กระทัดรัดที่สุด ปาล์มจะไม่มีการบีบอัดข้อมูลที่เก็บในหน่วยความจำ และการเก็บข้อมูลที่ใหญ่ทำให้เสียเวลาในการซิงโครไนซ์มาก
- ดาต้าเบสควรจะมีชนิดเดียวกันตลอดทั้งเรคอร์ด มิใช่ข้อบังคับตายตัว แต่จะช่วยลดโอเวอร์เฮดในการจัดการเรคอร์ดในดาต้าเบส

3.2.6 เขียนโปรแกรมให้ทนทาน

- ป้องกันโปรแกรมทำงานผิดพลาด โดยเพิ่มเติมส่วนตรวจสอบ อาจจะทำได้โดยใช้ฟังก์ชัน **ErrNonFatalDisplayIf** หรือ **ErrFatalDisplayIf**
- หลีกเลี่ยงการโพลลิ่ง (Polling) ถี่ๆ
- ระวังการอ่านหรือเขียนหน่วยความจำที่มีค่าเป็นนัล (Null) ควรมีการตรวจสอบว่าสามารถจองหน่วยความจำได้จริง
- ไม่เข้าถึงฮาร์ดแวร์หรือหน่วยความจำโดยตรง เรียกผ่านฟังก์ชันที่ปาล์มโอเอสจัดเตรียมไว้ให้จะเป็นการปลอดภัยมากกว่า
- ค่าพรีเฟอร์เรนซ์ของแอพพลิเคชันอาจจะมีการเปลี่ยนแปลงได้ เมื่อทำงานบนปาล์มโอเอสต่างเวอร์ชันกัน เพราะฉะนั้นควรมีการเขียนโปรแกรมเพื่อป้องกันความผิดพลาดนี้ โดยอาจจะไม่อนุญาตให้รันโปรแกรมบนเวอร์ชันที่ไม่ได้รับการทดสอบ

3.3 แนวทางการพัฒนาโปรแกรมบนปาล์ม

ภาษาที่ใช้ แอพพลิเคชันปาล์มสามารถพัฒนาได้โดยใช้ภาษาได้หลายภาษาแล้วแต่ความถนัดของผู้เขียนโปรแกรม และขนาดของงาน แบ่งการพัฒนาออกเป็นระดับคือ

1. ภาษาระดับต่ำ สามารถพัฒนาแอพพลิเคชันบนปาล์มได้โดยใช้ภาษาแอสเซมบลี้ (Assembly) หรือภาษาฟอร์ท (Forth)
2. ภาษาระดับกลาง สามารถพัฒนาได้โดยใช้ภาษาซี ซึ่งเป็นที่นิยมมากกว่า โดยมีคอมไพล์เลอร์ที่ได้รับความนิยมคือ Codewarrior (โดยบริษัท Metrowork) และ GCC (Gnu License) ซึ่งสามารถดาวน์โหลดได้ฟรี (Codewarrior สามารถดาวน์โหลดได้เฉพาะ Codewarrior Lite version)
3. การพัฒนาระดับฟอร์ม สามารถพัฒนาในลักษณะวิชวล คล้ายวิชวลเบสิก โดยสามารถสร้างฟอร์ม และเพิ่มเติมคอมโพเนนต์ต่างๆลงในฟอร์ม จากนั้นก็ใส่เหตุการที่สัมพันธ์กับฟอร์มนั้นๆ โปรแกรมที่สนับสนุนการเขียนโปรแกรมในลักษณะนี้ได้แก่ Pendragon Form และ Sattelite Form

สำหรับโครงงานนี้เลือกใช้โค้ดวอริเออร์เนื่องจากใช้งานง่าย และเหมาะสมกับขนาดของงานสามารถเขียนโปรแกรมในระดับของฟังก์ชันระบบของปาล์มโอเอสได้ อีกทั้งโปรแกรมโค้ดวอริเออร์ยังมีความสามารถไม่เป็นเพียงคอมไพล์เลอร์ โค้ดวอริเออร์ยังสามารถทำงานเป็นซอร์สโค้ดอีดิทเตอร์ โปรเจ็กแมเนเจอร์ และดีบักเกอร์ได้ภายในตัวเองอีกด้วย จึงเป็นคอมไพล์เลอร์ที่ทำงานได้สะดวก และคล่องตัวสูง

3.4 ซอร์สโค้ดของโปรแกรมบนปาล์ม

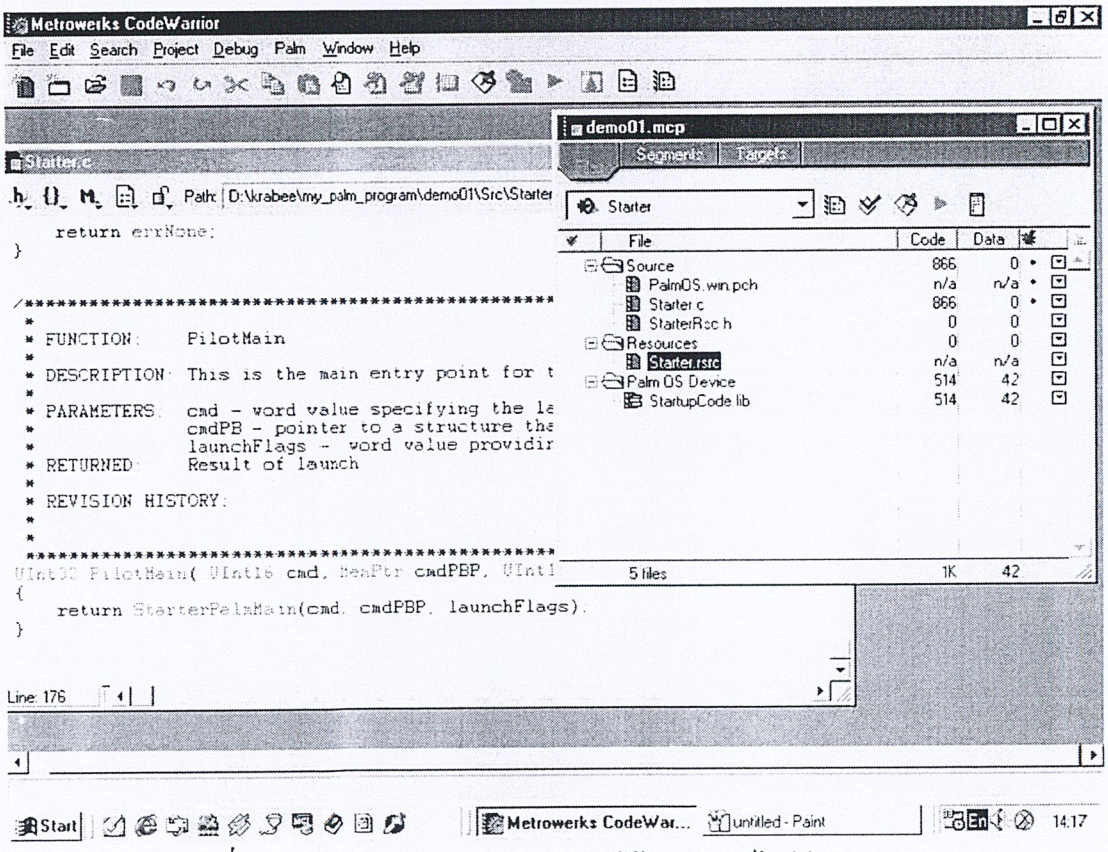
ซอร์สโค้ดของโปรแกรมบนปาล์มโอเอสมี 2 ส่วนคือ

- Resource file เป็นส่วนที่เก็บลักษณะของรีซอร์สต่างๆ ของโปรแกรม ได้แก่ ฟอร์ม และปุ่ม เมนูต่างๆ รวมทั้งคุณสมบัติของรีซอร์สนั้นๆด้วย รีซอร์สในปาล์มจะมีเลขประจำของตัวเองซึ่งไม่ซ้ำกัน เป็นตัวเลขที่ใช้อ้างอิง
- ส่วนภาษาซี รวมทั้ง Include file เป็นส่วนที่ควบคุมการทำงานของแอพพลิเคชันนั้นๆ เหมือนการพัฒนาโปรแกรมบนแพลตฟอร์มอื่นทั่วไป

3.5 โปรแกรมโค้ดวอริเออร์ (Codewarrior)

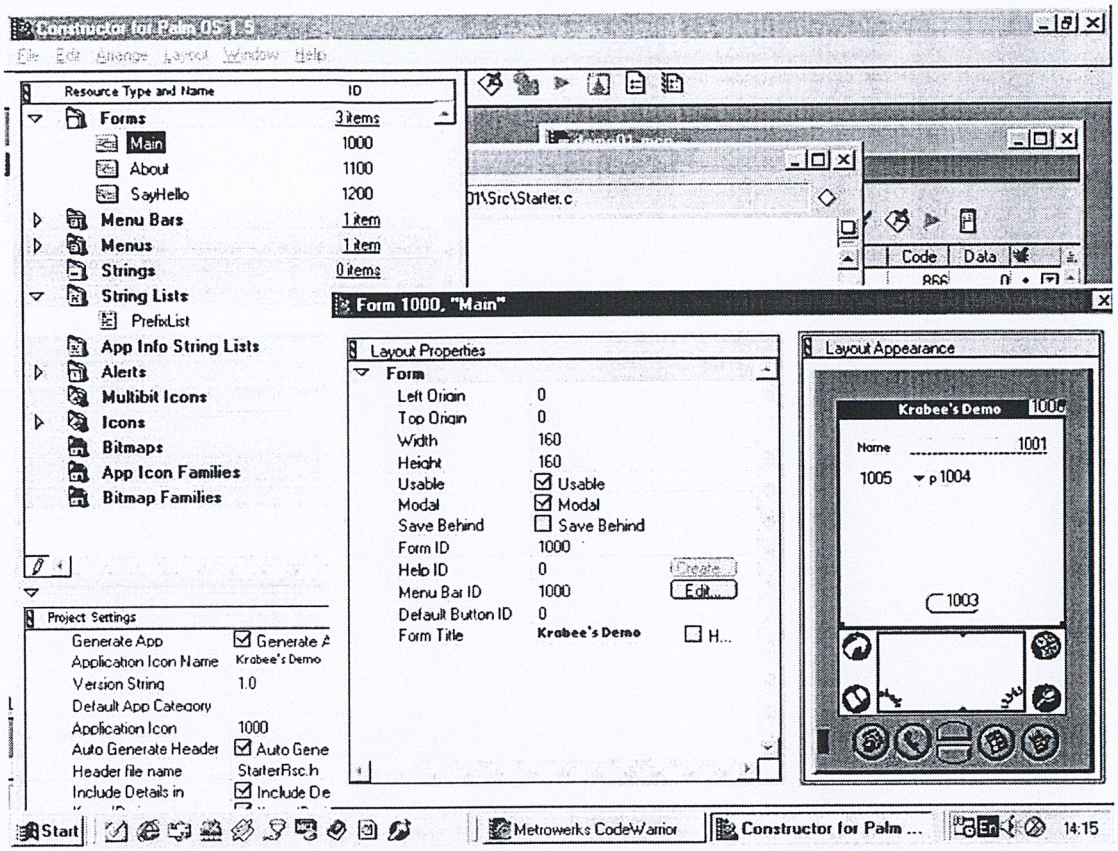
โปรแกรมโค้ดวอริเออร์ ประกอบด้วยส่วนสำคัญสองส่วนคือ

1. โค้ดวอริเออร์ไอดีอี (Codewarrior IDE) ใช้สำหรับแก้ไขโปรแกรม แก้ไขโปรเจ็ก ดีบักโปรแกรม หาข้อผิดพลาด โดยมีหน้าจอที่สำคัญคือหน้าจอโปรเจ็ก จะแสดงไฟล์ที่เกี่ยวข้องกับโปรเจ็ก สามารถดับเบิลคลิ๊กเพื่อแก้ไขไฟล์ หรือแก้ไขฟอร์มได้ อีกหน้าจอหนึ่งคืออีดิทเตอร์ใช้สำหรับแก้ไขซอร์สโค้ด



รูปที่ 3-2 หน้าจอการทำงานของโปรแกรมโค้ดวอริเออร์ไอดีอี

2. คอนสตรักเตอร์ (Constructure) ใช้สำหรับแก้ไขรีซอร์ส ได้แก่ ยูสเซอร์อินเทอร์เฟซ ปุ่มเมนูต่างๆ



รูปที่ 3-3 หน้าจอการทำงานของโปรแกรมโค้ดวอริเออร์คอนสตรักเตอร์

3.6 แนวทางการดีบักโปรแกรม

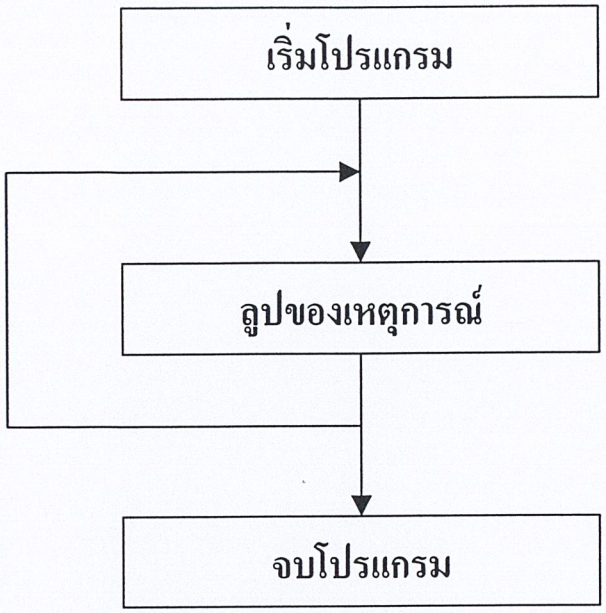
หลังจากเขียนโปรแกรม คอมไพล์โปรแกรมเสร็จเรียบ จะได้ไฟล์ที่มีนามสกุล .prc สามารถดาวน์โหลดไปยังปาล์มได้โดยใช้โปรแกรมอินสตอลแมเนเจอร์ (Install Manager) หรืออาจจะทดลองรันโปรแกรมโดยใช้ปาล์มอีมูเลเตอร์ (Palm OS Emulator) โดยสามารถดาวน์โหลดได้จาก http://www.palmos.com/dev โดยสามารถผสานการทำงานของปาล์มโอเอสอีมูเลเตอร์เข้ากับโปรแกรมโค้ดวอริเออร์ได้ สามารถ Step, Trace โปรแกรมได้ พร้อมทั้งตรวจสอบการทำงานได้จากปาล์มโอเอสอีมูเลเตอร์

3.7 หลักการทำงานของโปรแกรมบนปาล์ม

แอพพลิเคชันบนปาล์มทำงานแบบซิงเกิลเทรด และเป็นแบบอีเวนต์ไดรเวน คือทำงานตามอีเวนต์ที่เข้ามา สามารถรันโปรแกรมได้ทีละโปรแกรมเท่านั้น การพัฒนาโปรแกรมบนปาล์มจำเป็นต้องทำความเข้าใจการทำงานของปาล์มโอเอสก่อน

- แอพพลิเคชันบนปาล์มแต่ละตัวจะเริ่มการทำงานที่ฟังก์ชัน PilotMain คล้ายกับฟังก์ชัน main ในโปรแกรมภาษาซีทั่วไป จึงต้องปรากฏฟังก์ชัน PilotMain นี้ในทุกๆแอพพลิเคชัน เมื่อสั่งให้แอพพลิเคชันทำงาน ปาล์มโอเอสจะส่ง lunch code เพื่อให้โปรแกรมเริ่มต้นการทำงาน หรืออาจจะใช้เป็นประโยชน์ในการดีบักโปรแกรมอีกด้วย
- ปาล์มโอเอสมีลักษณะการทำงานแบบอีเวนต์ไดร์เวน แอพพลิเคชันบนปาล์มจึงต้องประกอบด้วยอีเวนต์ลูป (Event loop) วนลูปเพื่อตรวจสอบอีเวนต์ที่เข้ามา แล้วตอบสนองฟังก์ชันนั้นๆ จนกว่าจะพบอีเวนต์ appStopEvent จึงออกจากโปรแกรม โดยการทำงานอาจจะไม่เพียงตอบสนองอีเวนต์ อาจจะสร้างอีเวนต์ใหม่ส่งไปยังอีเวนต์คิว ซึ่งอีเวนต์ลูปนี้จะทำงานก็ต่อเมื่อเริ่มการทำงานของแอพพลิเคชันโดยมี lunch code เป็น normal lunch เท่านั้น
- แอพพลิเคชันบนปาล์มประกอบด้วยฟอร์มหลายๆฟอร์ม ซึ่งฟอร์มนี้เทียบได้กับวินโดวส์ในเดสก์ทอปแอพพลิเคชัน โดยแต่ละฟอร์มจะประกอบด้วยรีซอร์สต่างๆ เช่นปุ่ม เลเบิลต่างๆ ซึ่งรีซอร์สนี้จะมี ID ประจำตัวซึ่งไม่ซ้ำกัน เพื่อใช้สำหรับอ้างอิง
- การเข้าถึงหน่วยความจำของปาล์ม สามารถทำได้โดยผ่านดาต้าแมนเนจเมนท์ (Data Management)

3.8 ลูปของเหตุการณ์ (Event loop)



รูปที่ 3-4 ลูปของเหตุการณ์

โปรแกรมบนปาล์มมีสามส่วนคือ

1. **เริ่มต้นโปรแกรม** ส่วนอินนิเชียลไลซ์ (Initialize) จะจองหน่วยความจำ อ่านค่าพรีเฟอร์เรนซ์ของโปรแกรม จัดเตรียมค่าตัวแปรให้พร้อมสำหรับการทำงาน โปรแกรมบนปาล์มจะเริ่มต้นการทำงานที่ฟังก์ชัน PilotMain() ซึ่งคล้ายกับ main() ของโปรแกรมภาษาซีทั่วไป จากนั้น หลังจากตรวจสอบเวอร์ชันของปาล์มโอเอส เพื่อป้องการการทำงานที่ผิดพลาดเนื่องจากรันโปรแกรมในปาล์มโอเอสที่ไม่สนับสนุนฟังก์ชัน

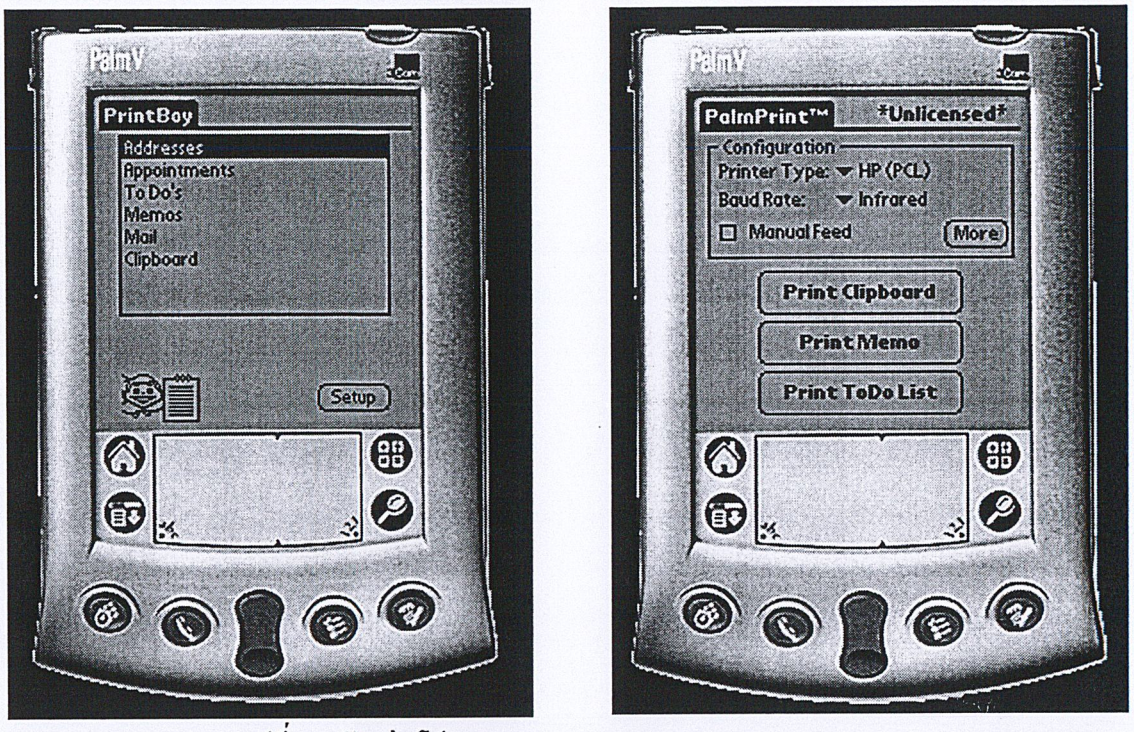
2. **ลูปของเหตุการณ์ (Event loop)** อนึ่งการทำงานของปาล์มเป็นแบบอีเวนต์ไดรเวน (Event-driven) คือ โปรแกรมจะอ่านเหตุการณ์จากคิวของเหตุการณ์ (Event queue) แล้วตอบสนองเหตุการณ์นั้นๆ วนอ่านเหตุการณ์และตอบสนองต่อเหตุการณ์นั้นไปเรื่อยๆ จนกว่าจะได้รับเหตุการณ์ appStopEvent จากคิวของเหตุการณ์จึงออกจากการทำงานในขั้นตอนนี้

3. **จบโปรแกรม** ก่อนออกจากโปรแกรม โปรแกรมจะทำการบันทึกค่าพรีเฟอร์เรนซ์ของโปรแกรม จากนั้นก็คืนตัวแปรที่ได้จองไว้

สามารถดูตัวอย่างซอร์สโค้ดอย่างง่ายของโปรแกรมบนปาล์มได้ในภาคผนวก ก เพื่อสร้างความเข้าใจเกี่ยวกับลูปของเหตุการณ์ได้ดียิ่งขึ้น

บทที่ 4
แนวคิดของโปรแกรม

4.1 การพิมพ์เอกสารบนปาล์ม



รูปที่ 4-1 ตัวอย่างโปรแกรม PrintBoy และ PalmPrint

เนื่องจากปาล์มโอเอสไม่สนับสนุนการพิมพ์เอกสาร จึงมีผู้พัฒนาแอพพลิเคชัน เพื่อช่วยพิมพ์เอกสารจากปาล์มเช่น Palm Print, IR Print suit โปรแกรมข้างต้นมีความสามารถคล้ายๆกันคือ สามารถพิมพ์ข้อมูลจาก Clipboard, Notepad, Address book สามารถเลือกชนิดของเครื่องพิมพ์ได้ว่าเป็นเครื่องพิมพ์ชนิดใด เช่น ESC/P2, PCL หรือ Generic (Raw text) จากการทดลองใช้โปรแกรมถ้าหากเลือกชนิดเครื่องพิมพ์เป็น Generic และใช้กับเครื่องพิมพ์ที่มีฟอนต์ภาษาไทยเท่านั้นจึงสามารถพิมพ์ภาษาไทยได้

การส่งข้อมูลของโปรแกรมเหล่านี้คล้ายกันคือส่งโดย IrDA ไปยังเครื่องพิมพ์โดยตรง และส่งผ่าน RS-232 ซึ่งวิธีนี้ต้องพึ่งฮาร์ดแวร์แปลงจาก RS-232 เป็น Parallel port แล้วส่งไปยังเครื่องพิมพ์หรือไม่ในขณะที่เสียบอยู่กับเครดิล จากนั้นก็ใช้โปรแกรมรับข้อมูลผ่านพอร์ตอนุกรม เช่น ไฮเปอร์เทอร์มินอลแล้วค่อยส่งข้อมูลดังกล่าวไปยังเครื่องพิมพ์อีกครั้ง

4.2 ปัญหาการพิมพ์เอกสารบนปาล์ม

สามารถสรุปปัญหาการพิมพ์บนเครื่องปาล์มออกมาเป็นข้อได้ดังนี้

1. การพิมพ์ยังไม่สนับสนุนภาษาไทย คือไม่สามารถพิมพ์ภาษาไทย รวมทั้งจัดการการจัดระดับการพิมพ์ และการตัดคำ ไม่มีฟอนต์ภาษาไทย
2. การพิมพ์สำหรับเครื่องพิมพ์ที่ไม่มี IrDA นั้นยังยุ่งยากอยู่ กล่าวคือต้องอาศัยโปรแกรมอย่าง เช่น ไฮเปอร์เทอร์มินอล มาช่วย ถ้าหากใช้เป็นโปรแกรมขนาดเล็ก ที่สามารถทำงานได้เฉพาะก็ยิ่งจำอำนวยความสะดวกแก่ผู้ใช้ยิ่งขึ้น

4.3 หลักการทำงานของโปรแกรมไทยปาล์มพรินต์

การพิมพ์เอกสารของโปรแกรมไทยปาล์มพรินต์มีข้อแตกต่างกับโปรแกรมพิมพ์เอกสารบนปาล์มตัวอื่นคือ จะใช้จัดรูปแบบเอกสารเป็น RTF ก่อนแล้วจึงส่งให้เครื่องคอมพิวเตอร์พิมพ์ แตกต่างกับโปรแกรมอื่นคือจะแปลงเป็นภาษาของเครื่องพิมพ์แล้วจึงส่งไปยังเครื่องคอมพิวเตอร์หรือเครื่องพิมพ์โดยตรง

การส่งข้อมูลจากปาล์มมายังคอมพิวเตอร์มีการส่งข้อมูลเป็น RS-232 ผ่านเครเดิ้ล มิได้ใช้คอนดิท(Conduit) ไม่ต้องเสียเวลารอการฮอตซิงค์ทุกครั้งที่ต้องการพิมพ์เอกสาร ขณะพิมพ์งานจึงต้องปิดโปรแกรมที่ใช้พอร์ตอนุกรม เช่น ฮอตซิงค์แมเนเจอร์

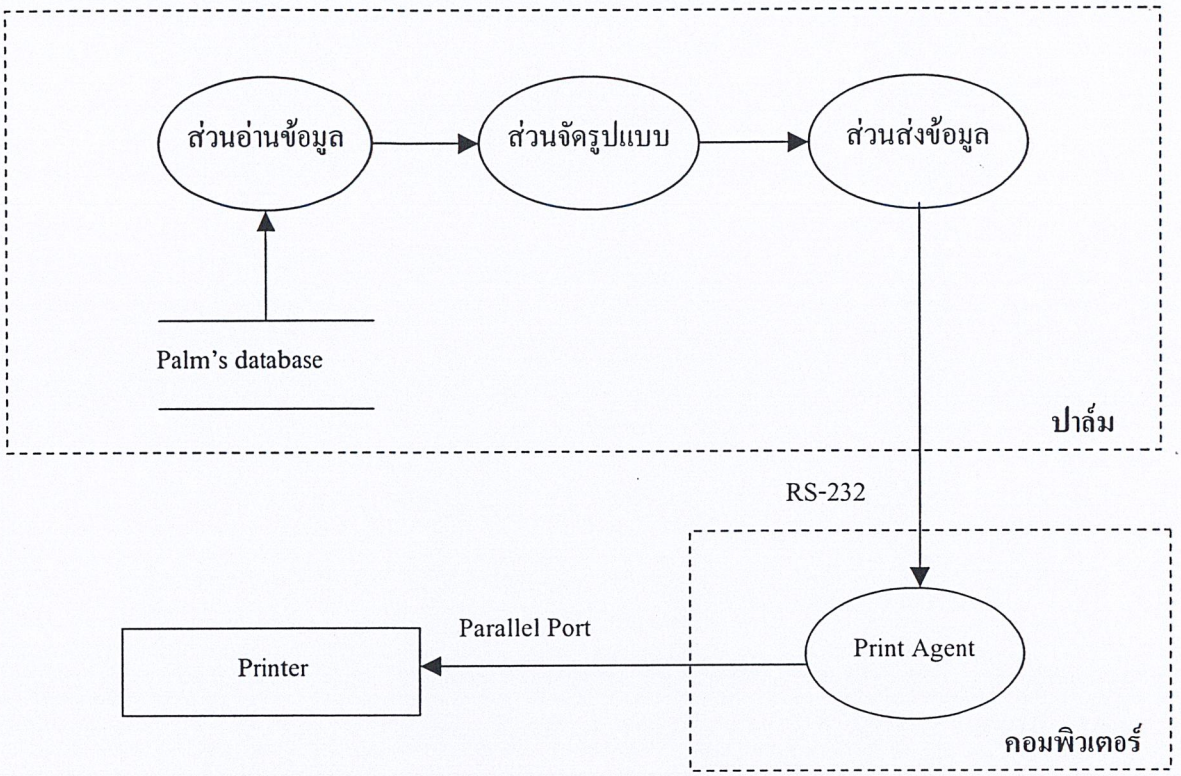
ข้อมูลของแอพพลิเคชันในปาล์มโอเอส จะถูกเก็บในรูปของดาต้าเบส ซึ่งประกอบด้วยเรคอร์ดย่อยๆ โดยโปรแกรมแต่ละตัวที่เป็นผู้สร้างดาต้าเบส นั้นจะมี Creator ID ที่แตกต่างกัน อาศัย Creator ID นี่เองเป็นตัวคัดกรองในการเปิดดาต้าเบสของแอพพลิเคชันที่ต้องการ

เมื่ออ่านข้อมูลออกมาจากดาต้าเบสแล้ว โปรแกรมก็จะนำข้อมูลที่ได้จากดาต้าเบสนั้นไปจัดรูปแบบ เป็น RTF (Rich Text Format)

เมื่อจัดรูปแบบเสร็จก็จะส่งข้อมูลมาให้กับเครื่องคอมพิวเตอร์ โดยผ่านพอร์ตอนุกรม โดยฝั่งวินโดวส์จะมีโปรแกรมขนาดเล็กทำงานอยู่ในเทรย์บาร์ (Tray Bar) ของวินโดวส์ และให้คอมพิวเตอร์ส่งต่อไปยังเครื่องพิมพ์อีกที

4.4 ส่วนประกอบของโปรแกรม

โปรแกรมประกอบด้วยส่วนประกอบใหญ่ๆ 4 ส่วนคือ



รูปที่ 4-2 แผนภาพแสดงการทำงานของโปรแกรมไทยปาล์มพรินต์

1. ส่วนอ่านข้อมูล จะเรียกใช้ฟังก์ชันของ Palm OS เพื่ออ่านค่าจากดาต้าเบส ของแอพพลิเคชันมาตรฐานของปาล์มเนื่องจากดาต้าเบสของแอพพลิเคชันแต่ละตัวมีรูปแบบในการจัดเก็บที่แตกต่างกัน เพราะฉะนั้นจึงต้องมีตัวอ่านดาต้าเบสที่แตกต่างกันสำหรับแอพพลิเคชันแต่ละตัวเช่น Memo pad จะมี Creator ID เป็น "memo" และมี Type เป็น "DATA" การอ่านข้อมูลส่วนนี้สามารถทำได้ผ่าน Database Manager โดยใช้ฟังก์ชัน **DmOpenDatabaseByTypeCreator** ก็จะหาดาต้าเบสที่ถูกสร้างด้วย Memo Pad จากนั้นจึงใช้ฟังก์ชัน **DmGetRecord** เพื่อเข้าถึงเรคอร์ดใน Database นั้นๆ บางดาต้าเบสของปาล์มเช่น Todo ยังจำเป็นที่จะต้อง แปลงรูปแบบในแต่ละเรคอร์ดให้อยู่ในรูปแบบที่สามารถอ่านเข้าใจได้ ข้อมูลที่อ่านได้จะอยู่ในรูปชังก์ของหน่วยความจำ ที่ถูกชี้โดยพอยน์เตอร์

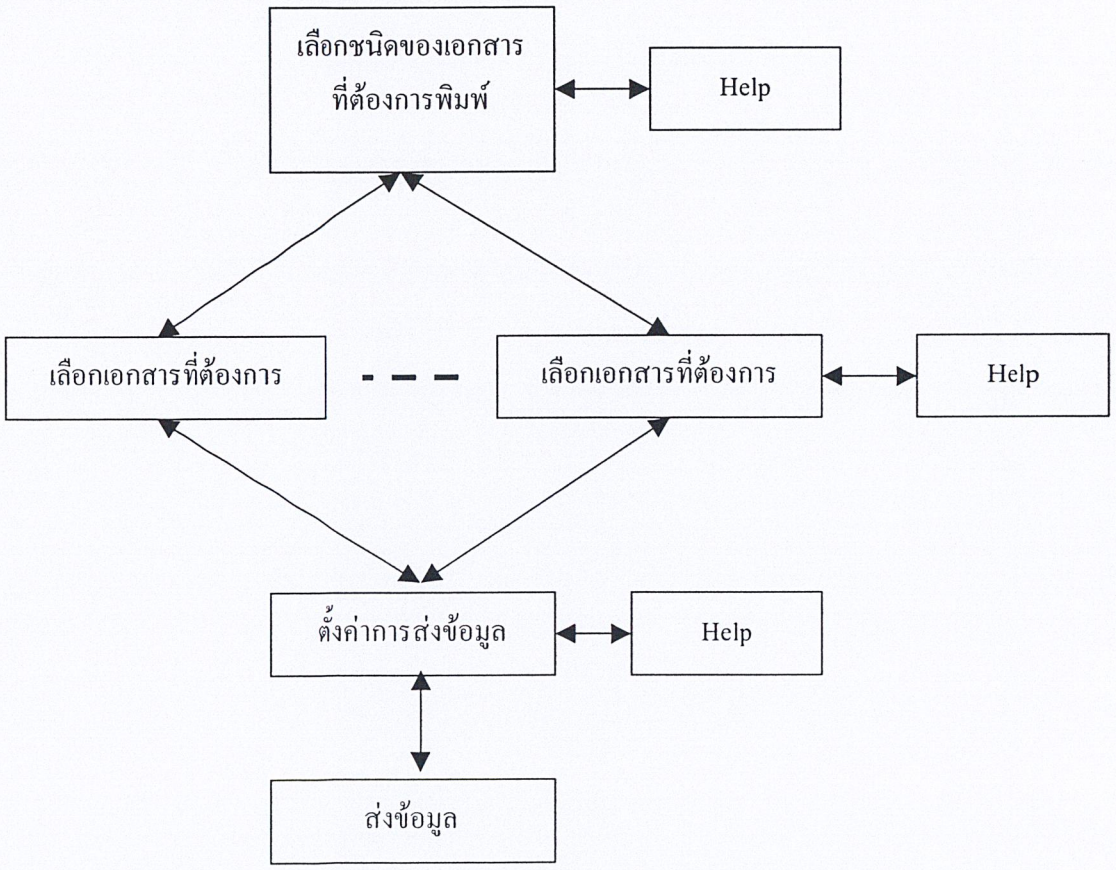
2. ส่วนจัดรูปแบบ จะนำข้อมูลที่อ่านได้มาจัดรูปแบบ เป็น RTF เตรียมเพื่อส่งข้อมูลไปยังคอมพิวเตอร์ มีการแทรกเฮดเดอร์ของ RTF รหัสควบคุมการพิมพ์

3. ส่วนส่งข้อมูล จะนำข้อมูลที่ผ่านการจัดรูปแบบ ส่งไปยังคอมพิวเตอร์ โดยผ่านทางพอร์ตอนุกรม โดยสามารถตั้งค่าความเร็วในการส่งข้อมูลได้ส่วนรับข้อมูลจากปาล์มส่งต่อไปยังเครื่องคอมพิวเตอร์

4. พรินท์เอเจนท์ (Print Agent) เป็นโปรแกรมรันบนวินโดวส์ สามารถสั่งพิมพ์จากปาล์ม โดยผ่านคอมพิวเตอร์ไปยังเครื่องพิมพ์ที่ต่ออยู่กับคอมพิวเตอร์เครื่องนั้นได้ (รวมทั้งเครื่องพิมพ์ที่ติดตั้งอยู่บนเครือข่ายด้วย) โปรแกรมนี้พัฒนาด้วยเดลไฟ มีคอมโพเนนต์ที่เกี่ยวข้องคือ Serial I/O เพื่อใช้สำหรับติดต่อกับพอร์ตอนุกรม

4.5 การออกแบบยูสเซอร์อินเทอร์เฟซ

โปรแกรมไทยปาล์มพรินต์ ออกแบบยูสเซอร์อินเทอร์เฟซแบบวิซาร์ต คือจะแนะนำผู้ใช้ตั้งแต่เลือกชนิดเอกสารที่ต้องการพิมพ์ เลือกเอกสาร ตั้งค่าพารามิเตอร์ในการส่งข้อมูล สามารถย้อนไปทำขั้นตอนก่อนหน้าได้ ทำให้ผู้ใช้ใช้งานได้ง่าย ไม่ต้องมีความรู้ในการใช้งานมาก่อน พร้อมทั้งมีรายการช่วยเหลือเมื่อผู้ใช้ต้องการ



รูปที่4-3 แผนภาพแสดงยูสเซอร์อินเทอร์เฟซของโปรแกรมไทยปาล์มพรินต์

1. เลือกชนิดเอกสารที่ต้องการพิมพ์ เลือกชนิดของเอกสารของปาล์มทีต้องการพิมพ์ เช่น TodoList, MemoPad ยูสเซอร์อินเทอร์เฟซที่ใช้เลือกเป็นลิสต์ สามารถเพิ่มเติมได้ ในกรณีปรับปรุงให้พิมพ์เอกสารหลายชนิดมากขึ้น เมื่อกด Next ก็จะนำไปสู่ขั้นต่อไป
2. เลือกเอกสารที่ต้องการพิมพ์ เอกสารบางประเภทจะต้องเลือกเอกสารย่อย เช่น เมมโมแพดจะต้องเลือกพิมพ์เมมโมไหน สำหรับเอกสารบางชนิดอาจจะไม่มีหน้าจอนี้ ชนิดของเอกสารที่เลือกในหน้าจอที่แล้วจะนำมาสู่หน้าจอการเลือกเอกสารที่แตกต่างกัน
3. ตั้งค่าการส่งข้อมูล ตั้งค่าความเร็วในการส่งของพอร์ตอนุกรม และคอนฟิกูเรชันของพอร์ต เป็นขั้นตอนที่อาจจะสร้างความสับสนให้กับผู้ใช้ กล่าวคือถ้าหากความเร็วไม่ตรงกันกับคอมพิวเตอร์อาจจะทำให้การส่งไม่ได้ หรือส่งข้อมูลผิดพลาด เพราะฉะนั้นจึงมีส่วนช่วยเหลืออธิบายให้ผู้ใช้เข้าใจได้ง่าย
4. ส่งข้อมูล เป็นหน้าจอแสดงสถานะของการส่งข้อมูล และแจ้งเตือนผู้ใช้เมื่อส่งข้อมูลเสร็จเรียบร้อยแล้ว
5. หน้าจอช่วยเหลือ เป็นส่วนที่ช่วยเหลือให้ผู้ใช้สามารถใช้งานโปรแกรมได้
6. หน้าจออะเบ้าต์ (About) แสดงข้อมูลของโปรแกรม รายละเอียดของโปรแกรม การติดต่อกับผู้พัฒนาโปรแกรม เมื่อโปรแกรมเกิดปัญหา

บทที่ 5
การทดสอบโปรแกรมไทยปาล์มพรินต์

6.1 ความต้องการของโปรแกรมไทยปาล์มพรินต์

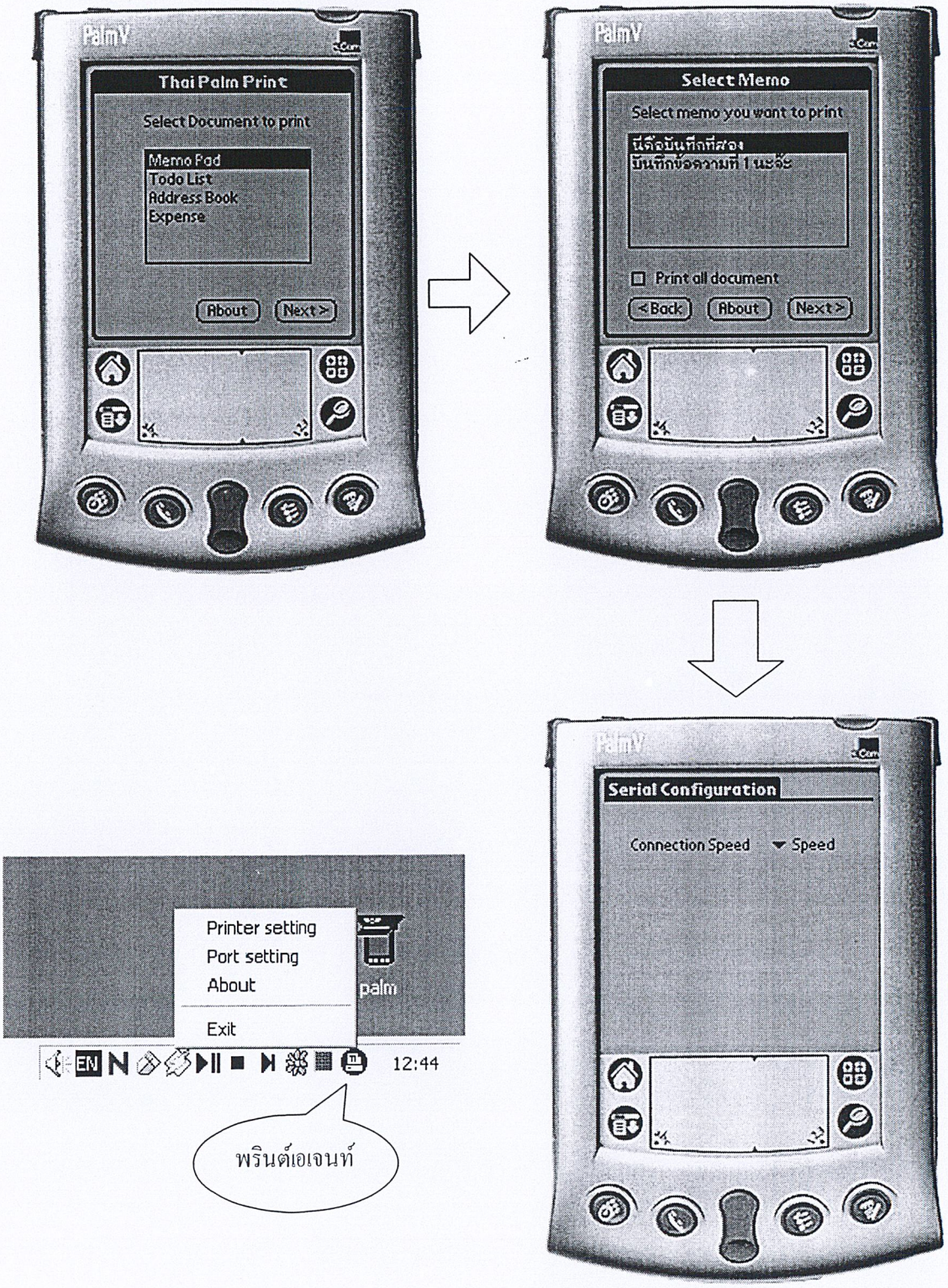
1. ระบบปฏิบัติการวินโดส์ 95,98,NT
2. เนื้อที่ในฮาร์ดดิสก์ 5 เมกะไบต์
3. หน่วยความจำ 32 เมกะไบต์
4. พอร์ตอนุกรม
5. เครื่องพิมพ์ชนิดใดก็ได้ (พร้อมไดรเวอร์สำหรับวินโดวส์)
6. ปาล์มโอเอสเวอร์ชัน 3.0 ขึ้นไป

6.2 คุณสมบัติ

1. สามารถพิมพ์เอกสารจากแอพพลิเคชันมาตรฐานของปาล์ม ได้แก่ Address Book, Date Book, Expense, Memo Pad, Todo List
2. พิมพ์ผ่านเครื่องพิมพ์ที่ต่ออยู่กับเครื่องพีซี ที่ใช้ระบบปฏิบัติการวินโดวส์

6.1 หน้าจอโปรแกรมไทยปาล์มพรินต์

ตัวอย่างหน้าจอบางส่วนของโปรแกรมไทยปาล์มพรินต์สำหรับการพิมพ์เอกสาร เมโมแพดของปาล์มโอเอส และโปรแกรมพรินต์เอเจนท์



รูปที่ 6-1 ภาพแสดงหน้าจอการทำงานของโปรแกรมไทยปาล์มพรินต์

บทที่ 6
สรุปและวิจารณ์

5.1 ปัญหาและอุปสรรค

ปัญหาและอุปสรรคในการพัฒนาระบบพิมพ์ภาษาไทยบนปาล์มนี้คือ

1. เป็นแพลตฟอร์มใหม่ ทำให้มีผู้ที่สามารถให้คำปรึกษาได้น้อย และมีหนังสืออ้างอิงน้อย แต่นับว่าเป็นโชคดีที่สามารถหาข้อมูล เอกสารอ้างอิงได้มากในอินเตอร์เน็ต ทำให้พอช่วยในการศึกษาได้
2. ผู้พัฒนาไม่มีงบประมาณพอที่จะซื้อเครื่องปาล์มเพื่อเขียนโปรแกรม การพัฒนาโปรแกรมในระยะแรกจึงพัฒนาและทดสอบบนปาล์มโอเอสอีมูเลเตอร์เป็นหลัก

5.2 แนวทางการวิจัยและพัฒนาต่อ

1. ระบบพิมพ์ภาษาไทยนี้พัฒนาขึ้นให้ใช้กับแอพพลิเคชันมาตรฐานของปาล์มเท่านั้น ในการพัฒนาต่ออาจจะขยายความสามารถโดยให้สามารถให้ใช้งานได้กับแอพพลิเคชันตัวอื่นๆ ของปาล์ม
2. พัฒนาออกแบบโปรแกรมให้มีลักษณะเป็นโมดูล สามารถเพิ่มเติมโมดูลได้เพื่อรองรับแอพพลิเคชันใหม่ๆของปาล์ม
3. พัฒนาให้ใช้ได้กับระบบปฏิบัติการอื่นนอกจากวินโดวส์

5.3 เปรียบเทียบกับโปรแกรมอื่น

1. การพิมพ์เอกสารสำหรับโปรแกรมปาล์มพรินท์ และ IR Print suit ไม่สามารถพิมพ์ภาษาไทยได้ แต่โปรแกรมไทยปาล์มพรินต์สามารถพิมพ์ภาษาไทยได้
2. โปรแกรมปาล์มพรินท์อื่นๆ สามารถพิมพ์ ได้เฉพาะเครื่องพิมพ์ที่ใช้ภาษาหลักๆ เท่านั้น เช่น ESC/P2 ,PCL แต่โปรแกรมไทยปาล์มพรินต์สามารถใช้พิมพ์ได้กับเครื่องพิมพ์ทุกชนิดที่ติดตั้งไดรเวอร์บนวินโดวส์ รวมทั้งเครื่องพิมพ์ที่ติดตั้งอยู่บนเครือข่ายด้วย
3. โปรแกรมไทยปาล์มพรินต์สามารถใช้งานได้ง่ายกว่าไม่ต้องอาศัยไฮเปอร์เทอร์มินอล เพื่อรับข้อมูลจากพอร์ตอนุกรม เพียงเปิดโปรแกรมพรินต์เอเจนท์ ก็สามารถพิมพ์งานได้ทันที
4. โปรแกรมไทยปาล์มพรินต์สามารถพิมพ์งานได้มีคุณภาพสูง เพราะจัดการพิมพ์เป็น RTF
5. โปรแกรมปาล์มพรินต์ไม่สามารถใช้งานกับพรินเตอร์ผ่านพอร์ตอินฟราเรดได้เหมือนโปรแกรมพิมพ์งานอื่นๆ
6. โปรแกรมไทยปาล์มพรินต์ยังใช้งานได้เฉพาะกับระบบปฏิบัติการวินโดวส์อยู่

ภาคผนวก ก

ตัวอย่างซอร์สโค้ดโปรแกรมบนปาล์มเพื่อแสดงลูปของเหตุการณ์

```
DWord PilotMain(Word launchCode, Ptr cmdPBP, Word launchFlags)
{
  Err err;

   if (launchCode == sysAppLaunchCmdNormalLaunch) {
    if ((err = StartApplication()) == 0) {
      EventLoop();
      StopApplication();
    }
  }
  return err;
}


static void EventLoop(void)
{
  EventType  event;
  Word     error;

  do {
   EvtGetEvent(&event, evtWaitForever);          //   system routine
   if (! SysHandleEvent(&event))                 //system routine
     if (! MenuHandleEvent(0, &event, &error))   // system routine
       if (! ApplicationHandleEvent(&event))     // routine we write
         FrmDispatchEvent(&event);               // system routine
  } while (event.eType != appStopEvent);
}


static Err StartApplication(void)
```

```
{
  FrmGotoForm(HelloWorldForm);
  return 0;
}


static void StopApplication(void)
{
}


static Boolean ApplicationHandleEvent(EventPtr event)
{
  FormPtr  frm;
  Int    formId;
  Boolean  handled = false;


  if (event->eType == frmLoadEvent) {
  // Load the form resource specified in the event and activate the form.
    formId = event->data.frmLoad.formID;
    frm = FrmInitForm(formId);
    FrmSetActiveForm(frm);


  // Set the event handler for the form.  The handler of the currently
  // active form is called by FrmDispatchEvent each time it gets an event.
    switch (formId) {
    case HelloWorldForm:
      FrmSetEventHandler(frm, MyFormHandleEvent);
      break;
    }
    handled = true;
  }
  return handled;
}
```

ภาคผนวก ข

อ้างอิงฟังก์ชันและโครงสร้างข้อมูลที่เรียกใช้ในโปรแกรม

ฟังก์ชันที่เรียกใช้ในโปรแกรม

DmOpenDatabaseByTypeCreator	
การใช้งาน	เปิดดาต้าเบสโดยที่มีชนิด และผู้สร้างตามที่ระบุไว้ ถ้าดาต้าเบสในเป็นรีซอร์ส จะเปิดโดยยึดตามโลแคลปัจจุบัน
โพรโตไทป์	DmOpenRef **DmOpenDatabaseByTypeCreator**(UInt32 type, UInt32 creator, UInt16 mode)
พารามิเตอร์	-> ประเภทของดาต้าเบส -> ผู้สร้างดาต้าเบส -> โหมดของการเปิดดาต้าเบส
ผลลัพธ์	ถ้าหากไม่พบดาต้าเบสที่ต้องการเปิดฟังก์ชันนี้จะรีเทิร์นค่า 0 และสามารถตรวจสอบเหตุผลของความผิดพลาดได้จากฟังก์ชัน **DmGetLastErr**
หมายเหตุ	ถ้าใช้รูทีนนี้เพื่อเปิดดาต้าเบสแบบอ่านอย่างเดียว การเปิดจะยึดดาต้าเบสตามโลแคลปัจจุบัน ดูฟังก์ชัน DmOpenDatabase เพิ่มเติมเกี่ยวกับโอเวอร์เลย์และดาต้าเบสประเภทรีซอร์ส

DmGetResource	
การใช้งาน	หาดาต้าเบสของรีซอร์สทุกตัวที่เปิด คืนค่าเป็นแฮนเดิ้ลของรีซอร์สที่มีชนิดและID ตามที่กำหนด
โพรโตไทป์	MemHandle **DmGetResource** (DmResType type,DmResID resID)
พารามิเตอร์	-> ชนิดของรีซอร์ส -> ID ของรีซอร์ส
ผลลัพธ์	แฮนเดิ้ลของรีซอร์ส ถ้าหาไม่พบจะรีเทิร์นเป็นนัล และสามารถตรวจสอบเหตุผลของความผิดพลาดได้จากฟังก์ชัน **DmGetLastErr**
หมายเหตุ	การหาดาต้าเบสของรีซอร์สจะเริ่มหาจากดาต้าเบสที่เพิ่งจะถูกเปิดล่าสุด โดยยึดตามชนิดและID ถ้าหาพบจะคืนค่าเป็นแฮนเดิ้ล แอพพลิเคชันควรเรียกใช้ฟังก์ชัน **DmReleaseResource** เมื่อไม่ต้องการเข้าถึงรีซอร์สนั้น จะทำให้รีซอร์สแฮนเดิ้ลไม่ถูกล็อก ฟังก์ชันนี้จะคืนค่ารีซอร์สที่อยู่ในโอเวอร์เลย ถ้าโอเวอร์เลย์มีรีซอร์สที่มีชนิดและ ID ตรงตามที่กำหนด ถ้าโอเวอร์เลย์ที่เปิดไม่มีรีซยร์สที่ตรงตามชนิดและID ฟังก์ชันนี้จะคืนค่าจากดาต้าเบสหลัก

EvtGetEvent	
การใช้งาน	คืนค่าเหตุการณ์ถัดไปที่เกิดขึ้น
โพรโตไทป์	void EvtGetEvent (EventType *event, Int32 timeout)
พารามิเตอร์	event พอยน์เตอร์ชี้ไปที่สตรักเจอร์ที่เก็บเหตุการณ์ที่คืนค่ากลับมา timeout เวลาที่รอให้เกิดใดเหตุการณ์ขึ้น (-1 คือไม่มีกำหนด)
ผลลัพธ์	ไม่คืนค่า
หมายเหตุ	ตั้งค่า timeout= -1 เมื่อรันโปรแกรมเพื่อรอรับอินพุตจากผู้ใช้ สำหรับแอพพลิเคชันที่มีการเคลื่อนไหว ให้ค่า timeout >= 0

MemHandleResize	
การใช้งาน	เปลี่ยนขนาดของชังก์
โพรโตไทป์	Err MemHandleResize (MemHandle h, UInt32 newSize)
พารามิเตอร์	-> แฮนเดิ้ลของชังก์ -> newSize ขนาดใหม่ที่ต้องการ
ผลลัพธ์	0 เมื่อไม่เกิดข้อผิดพลาด **memErrInvalidParam** พารามิเตอร์ที่ส่งให้ไม่ถูกต้อง **memErrNotEnoughSpace** ฮีปไม่เพียงพอสำหรับขยายขนาดของชังก์ **memErrChunkLocked** ไม่สามารถขยายชังก์ได้เนื่องจากชังก์ถูกล็อก

DmGetRecord	
การใช้งาน	คืนค่าแฮนเดิ้ลของเรคอร์ดตามค่าอินเด็กซ์ที่ต้องการ และเซ็ตค่า busy
โพรโตไทป์	MemHandle DmGetRecord (DmOpenRef dbP,UInt16 index)
พารามิเตอร์	-> dbP เป็นตัวแปรประเภท DmOpenRef ระบุดาต้าเบสที่ต้องการเปิด -> index ลำดับเรคอร์ดที่ต้องการเปิด
ผลลัพธ์	คืนค่าแฮนเดิ้ลของข้อมูลของเรคอร์ด ถ้ามีการใช้ฟังก์ชัน DmGetRecord กับเรคอร์ดเดียวกันก่อนที่จะปล่อยเรคอร์ดจะคืนค่าเป็นนัล และสามารถตรวจสอบเหตุผลของความผิดพลาดได้จากฟังก์ชัน **DmGetLastErr**
หมายเหตุ	ถ้าเรคอร์ดเป็นเรคอร์ดที่ถูกเก็บในรอม รูทีนนี้จะสร้างแฮนเดิ้ลปลอมชี้ แล้วเก็บแฮนเดิ้ลจริงในตัวแปรสตรักเจอร์ DmAccessType ฟังก์ชัน **DmReleaseRecord** จะต้องถูกเรียกใช้หลังจากเสร็จสิ้นการแสดง หรือการแก้ไขเรคอร์ดนั้น

DmGetLastErr	
การใช้งาน	คืนค่าความผิดพลาดจากฟังก์ชันดาต้าแมเนเจอร์ที่ถูกเรียกล่าสุด
โพรโตไทป์	Err DmGetLastErr (void)
พารามิเตอร์	ไม่มี
ผลลัพธ์	เลขแสดงความผิดพลาดจากฟังก์ชันดาต้าแมเนเจอร์ที่ถูกเรียกล่าสุด

DmReleaseRecord	
การใช้งาน	เคลียร์ busy bit ในเรคอร์ดที่กำหนด และเซ็ตค่า dirty bit ถ้าหาก dirty เป็นจริง
โพรโตไทป์	Err DmReleaseRecord (DmOpenRef dbP, UInt16 index,Boolean dirty)
พารามิเตอร์	-> dbP เป็นตัวแปรประเภท DmOpenRef ระบุดาต้าเบสที่ต้องการ -> index เรคอร์ดที่ต้องการปลดล็อก -> dirty ถ้าเป็นจริง dirty bit จะถูกเซ็ตในเรคอร์ดแอททริบิวต์ของเรคอร์ดนั้น
ผลลัพธ์	คืนค่า errNone ถ้าไม่มีข้อผิดพลาด หรือ dmErrIndexOutOfRange ถ้าอินเด็กซ์นั้นเลบขอบเขตจริง การปลดล็อกบางครั้งจะแสดงข้อความผิดพลาดแทนที่จะคืนค่าความผิดพลาด
หมายเหตุ	เรียกใช้รูทีนนี้หลังจากใช้ฟังก์ชัน **DmGetRecord** หรือ สร้างเรคอร์ดโดยใช้คำสั่ง **DmNewRecord**

DmReleaseResource	
การใช้งาน	คืนรีซอร์สที่จองโดยคำสั่ง DmGetResource
โพรโตไทป์	Err **DmReleaseResource** (MemHandle resourceH)
พารามิเตอร์	-> resourceH แฮนเดิ้ลของรีซอร์ส
ผลลัพธ์	คืนค่า errNONE ถ้าไม่มีข้อผิดพลาด

ErrFatalDisplayIf	
การใช้งาน	มาโครเพื่อแสดงไดอะล็อกแสดงข้อผิดพลาด เมื่อ condition เป็นจริง และการตรวจสอบข้อผิดพลาดถูกเซ็ต บางส่วนหรือทั้งหมด
โพรโตไทป์	ErrFatalDisplayIf (condition, msg)
พารามิเตอร์	-> condition เป็นตัวแปรบูลีน ถ้าเป็นจริงจะแสดงข้อผิดพลาด -> msg ข้อความแสดงความผิดพลาด เป็นสตริง
ผลลัพธ์	ไม่คืนค่าใด
หมายเหตุ	มาโครนี้จะถูกคอมไพล์รวมกับโปรแกรมเมื่อมีการประกาศ ERROR_CHECK_LEVEL เป็น 1 หรือ 2 (ERROR_CHECK_PARTIAL หรือ ERROR_CHECK_FULL)

FrmDeleteForm	
การใช้งาน	คืนหน่วยความจำที่ใช้โดยฟอร์ม และหน่วยความจำที่ออปเจ็กบนฟอร์มใช้
โพรโตไทป์	void FrmDeleteForm (FormType *formP)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType)
ผลลัพธ์	ไม่คืนค่าใด

FrmDoDialog	
การใช้งาน	แสดงโมดาลไดอะล็อก(Modal dialog) จนกว่าผู้ใช้จะกดปุ่มในไดอะล็อกนั้น
โพรโตไทป์	UInt16 FrmDoDialog (FormType *formP)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType)
ผลลัพธ์	คืนค่า ID ของปุ่มที่ถูกกด

FrmDrawForm	
การใช้งาน	วาดออบเจ็กบนฟอร์มและกรอบของฟอร์ม
โพรโตไทป์	void FrmDrawForm (FormType *formP)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType)
ผลลัพธ์	ไม่คืนค่าใด
หมายเหตุ	ถ้าแอททริบิวต์ saveBehind ของฟอร์มถูกเซ็ต และฟอร์มแสดงอยู่ ฟังก์ชันนี้จะเก็บค่าบิตที่อยู่หลังฟอร์มลงในฟิลด์ bitsBehindForm ในสตรักเจอร์ FormType คำสั่งนี้ใช้กับอีเวนต์ frmOpenEvent ถ้ามีการวาดใดๆเพิ่มเติมควรทำหลังจากเรียกใช้ฟังก์ชันนี้ และเชื่อมโยงสิ่งที่วาดเพิ่มเติมนั้นกับอีเวนต์ frmUpdateEvent เหมือนๆกับที่ทำกับ frmOpenEvent และต้องคืนค่าจริง เพื่อแจ้งว่า frmUpdateEvent ทำงานเรียบร้อย โดยปกติแล้วอีเวนต์ frmUpdateEvent จะเรียก FrmDrawForm ดังนั้นถ้าอนุญาตให้มีการคืนค่าเท็จ สิ่งที่วาดเพิ่มเติมลงไปนั้นก็จะไม่ถูกวาด

FrmGetActiveForm	
การใช้งาน	คืนค่าฟอร์มที่กำลังแอกทีฟ
โพรโตไทป์	FormType *FrmGetActiveForm (void)
พารามิเตอร์	ไม่มี
ผลลัพธ์	ไม่คืนค่าใด
หมายเหตุ	คืนค่าพอยน์เตอร์ที่ชี้ออบเจ็กของฟอร์มที่กำลังแอกทีฟ

FrmGetObjectBounds	
การใช้งาน	หาค่าขอบเขตของออบเจ็ก จากค่าฟอร์มและอินเด็กซ์
โพรโตไทป์	void FrmGetObjectBounds (const FormType *formP,UInt16 ObjIndex, RectangleType *rP)
พารามิเตอร์	-> formP พอยน์เตอร์ชี้ไปที่ออปเจ็กของฟอร์ม (มีชนิดเป็น FormType สตรักเจอร์) -> ObjIndex อินเด็กซ์ของออปเจ็กในฟอร์ม โดยสามารถหาได้จาก FrmGetObjetIndex <- rp พอยเตอร์ชี้ไปยังสตรักเจอร์ RecTangleType ที่วัตถุนั้นวางอยู่
ผลลัพธ์	ไม่คืนค่าใดๆ ขอบเขตของออบเจ็กจะถูกรีเทิร์นในค่าตัวแปร r

FrmGetObjectIndex	
การใช้งาน	คืนค่าลำดับของออบเจ็กในฟอร์ม
โพรโตไทป์	UInt16 FrmGetObjectIndex (const FormType *formP,UInt16 objID)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType) -> objID ID ของออบเจ็กในฟอร์ม
ผลลัพธ์	คืนค่าลำดับของออบเจ็ก (ออบเจ็กแรกเริ่มต้นด้วยเลข 0)

FrmGetObjectPtr	
การใช้งาน	คืนค่าพอย์นเตอร์ที่ชี้ไปที่ดาต้าสตรักเจอร์ของออบเจ็กในฟอร์ม
โพรโตไทป์	void *FrmGetObjectPtr (const FormType *formP, UInt16 objIndex)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType) -> objIndex ลำดับของออบเจ็กในฟอร์ม สามารถหาได้โดยใช้ฟังก์ชัน **FrmGetObjectIndex**
ผลลัพธ์	คืนค่าพอยน์เตอร์ที่ชี้ไปที่ออบเจ็กในฟอร์ม

FrmGotoForm	
การใช้งาน	ส่งเหตุการณ์ frmCloseEvent ไปที่ฟอร์มปัจจุบัน แล้วส่งเหตุการณ์ frmLoadEvent และ frmOpenEvent ไปยังฟอร์มที่กำหนด
โพรโตไทป์	void FrmGotoForm (UInt16 formId)
พารามิเตอร์	-> formId ID ของฟอร์มที่ต้องการแสดง
ผลลัพธ์	ไม่มีการคืนค่า
หมายเหตุ	โดยทั่วไปแล้วตัวจัดการเหตุการณ์ของฟอร์ม (FrmHandleEvent) จะลบและคืนหน่วยความจำของฟอร์มเมื่อได้รับเหตุการณ์ frmCloseEvent

FrmInitForm	
การใช้งาน	โหลดและอินนิเชียลไลซ์ฟอร์มรีซอร์ส
โพรโตไทป์	FormType *FrmInitForm (UInt16 rscID)
พารามิเตอร์	-> rscID ID ของฟอร์ม
ผลลัพธ์	คืนค่าพอยน์เตอร์ชี้ดาต้าสตรักเจอร์ของฟอร์ม ทั้งแสดงข้อความผิดพลาด ถ้าหากฟอร์มนั้นถูกอินนิเชียลมาก่อนแล้ว

FrmSetActiveForm	
การใช้งาน	เซ็ตค่าแอกทีฟฟอร์ม การป้อนข้อมูล (ปุ่มและปากกา) จะทำกับฟอร์มที่กำลังแอกทีฟ
โพรโตไทป์	void FrmSetActiveForm (FormType *formP)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType)
ผลลัพธ์	ไม่มีการคืนค่า
หมายเหตุ	เหตุการณ์ penDownEvent ภายนอกฟอร์มแต่ภายในพื้นที่แสดงผลจะถูกเพิกเฉย

FrmSetEventHandler	
การใช้งาน	ลงทะเบียนตัวจัดการเหตุการณ์สำหรับกฟอร์มที่กำหนด
โพรโตไทป์	void FrmSetEventHandler (FormType *formP,FormEventHandlerType *handler)
พารามิเตอร์	-> formP พอยน์เตอร์ลี้ไปที่ออปเจ็กของฟอร์ม (สตรักเจอร์ประเภท FormType) -> handler ตำแหน่งของฟังก์ชันของตัวจัดการเหตุการของฟอร์มนั้นๆ
ผลลัพธ์	ไม่มีการคืนค่า

MemHandleFree	
การใช้งาน	คืนหน่วยความจำจากชังก์
โพรโตไทป์	Err **MemHandleFree** (MemHandle h)
พารามิเตอร์	-> h คือ ชังก์ที่แฮนเดิ้ลไว้
ผลลัพธ์	ไม่มีการคืนค่า
หมายเหตุ	คืนค่า 0 เมื่อไม่มีข้อผิดพลาด หรือ memErrInvalidParam เมื่อเกิดข้อผิดพลาดขึ้น

MemHandleLock	
การใช้งาน	ล็อกชังก์ พร้อมทั้งพอยน์เตอร์ที่ชี้ไปยังข้อมูลชังก์นั้น
โพรโตไทป์	MemPtr **MemHandleLock** (MemHandle h)
พารามิเตอร์	-> h แฮนเดิ้ลของชังก์
ผลลัพธ์	คืนค่าพอยน์เตอร์ที่ชี้ไปที่ชังก์
หมายเหตุ	เมื่อเรียกใช้ฟังก์ชัน **MemHandleLock** ต้องมีการเรียกฟังก์ชัน **MemHandleUnlock** คู่กันเพื่อปลดล็อก

MemHandleNew	
การใช้งาน	จองชังก์หน่วยความจำในไดนามิกฮีป และคืนค่าแฮนเดิ้ล
โพรโตไทป์	MemHandle MemHandleNew (UInt32 size)
พารามิเตอร์	-> size ขนาดของชังก์ที่ต้องการ
ผลลัพธ์	คืนค่าแฮนเดิ้ลชังก์ หรือ 0 ถ้าไม่สามารถจองได้
หมายเหตุ	ใช้คำสั่งนี้เพื่อจองหน่วยความจำไดนามิก ก่อนที่จะเขียนดาต้าลงในชังก์ที่จอง ต้องใช้คำสั่ง MemHandleLock เพื่อล็อกชังก์และหาค่าพอยน์เตอร์ที่ชี้มัน

MenuEraseStatus	
การใช้งาน	ลบสถานะเมนูสถานะ
โพรโตไทป์	void MenuEraseStatus (MenuBarType *menuP)
พารามิเตอร์	-> menuP พอยต์เตอร์ชี้ไปที่ตัวแปรชนิด MenuBarType หรือ NULL หมายถึงเมนูปัจจุบัน
ผลลัพธ์	ไม่คืนค่าใดๆ

SysFormPointerArrayToStings	
การใช้งาน	เปลี่ยนอาเรย์ของพอยน์เตอร์เป็นสตริง ใช้เพื่อตั้งค่าสมาชิกในลิสต์
โพรโตไทป์	MemHandle SysFormPointerArrayToStrings (Char* c,Int16 stringCount)
พารามิเตอร์	c พอยน์เตอร์ชี้ไปที่บล็อกของสตริงโดยสตริงปิดท้ายด้วยนัล stringCount จำนวนของสตริงในบล็อก
ผลลัพธ์	แฮนเดิ้ลที่ไม่ถูกล็อกชี้ไปที่อาเรย์ของพอยเตอร์ที่ชี้ไปที่สตริง อาเรย์ที่คืนมาชี้ไปที่สตริง

LstGetSelection	
การใช้งาน	คืนค่าตัวเลือกที่ถูกเลือกในลิสต์
โพรโตไทป์	Int16 LstGetSelection (const ListType *listP)
พารามิเตอร์	-> size ขนาดของชังก์ที่ต้องการ
ผลลัพธ์	listP พอยน์เตอร์ชี้ลิสต์ออบเจ็ก
หมายเหตุ	คืนค่าตัวเลือกที่ถูกเลือก เรียงตามลำดับเริ่มจาก 0 และคืนค่า noListSelection เมื่อไม่มีตัวเลือกใดที่ถูกเลือก

LstSetListChoices	
การใช้งาน	ตั้งค่าสมาชิกของลิสต์ ด้วยอาเรย์ของสตริงที่ผ่านให้กับฟังก์ชันนี้
โพรโตไทป์	void LstSetListChoices (ListType *listP, Char **itemsText, UInt16 numItems)
พารามิเตอร์	listP พอยน์เตอร์ชี้ที่ออบเจ็กของลิสต์ itemsText พอยน์เตอร์ชี้อาร์เรย์ของสตริง numItems จำนวนของทางเลือกในสิลต์
ผลลัพธ์	ไม่คืนค่าใดๆ

SysLibFind	
การใช้งาน	คืนค่าตำแหน่งที่ไลบรารี่ถูกโหลด โดยให้ชื่อของไลบรารี่
โพรโตไทป์	Err SysLibFind (const Char* nameP,UInt16* refNumP)
พารามิเตอร์	nameP พอยน์เตอร์ชี้ชื่อของไลบรารี่ refNumP พอยน์เตอร์ชี้ตำแหน่งที่ไลบรารี่ถูกโหลด (ถ้าไม่สามารถหาได้ตัวแปรนี้จะไม่ถูกประกาศ)
ผลลัพธ์	คืนค่าแฮนเดิ้ลชังก์ หรือ 0 ถ้าไม่สามารถจองได้
หมายเหตุ	ใช้คำสั่งนี้เพื่อจองหน่วยความจำไดนามิก ก่อนที่จะเขียนดาต้าลงในชังก์ที่จอง ต้องใช้คำสั่ง MemHandleLock เพื่อล็อกชังก์และหาค่าพอยน์เตอร์ที่ชี้มัน

MemHandleNew	
การใช้งาน	จองชังก์หน่วยความจำในไดนามิกฮีป และคืนค่าแฮนเดิ้ล
โพรโตไทป์	MemHandle MemHandleNew (UInt32 size)
พารามิเตอร์	-> size ขนาดของชังก์ที่ต้องการ
ผลลัพธ์	คืนค่า 0 ถ้าไม่มีข้อผิดพลาด หรือไม่ คืน sysErrorLibNotFound (เมื่อไลบรารี่ไม่ถูกโหลด) หรือข้อผิดพลาดอื่นๆ ที่คืนค่ามาจากจุดไลบรารี่ถูกโหลด

SerClose	
การใช้งาน	คืนพอร์ตอนุกรมซึ่งถูกจองด้วยฟังก์ชัน SerOpen
โพรโตไทป์	Err SerClose (UInt16 refNum)
พารามิเตอร์	-> refNum ตำแหน่งของไลบรารี่ของพอร์ตอนุกรม
ผลลัพธ์	0 ไม่เกิดข้อผิดพลาด serErrNotOpen พอร์ตไม่ได้ถูกเปิด serErrStillOpenPort พอร์ตถูกใช้งานด้วยโปรเซสอื่นอยู่
หมายเหตุ	การเปิดพอร์ตอนุกรมทำให้การใช้พลังงานของเครื่องสูงขึ้น เพราะฉะนั้นพึงเปิดพอร์ตเมื่อต้องการใช้เท่านั้น

SerOpen	
การใช้งาน	จองและเปิดพอร์ตอนุกรมโดยกำหนดอัตราบอด และใช้ค่าดีฟอล์ต
โพรโตไทป์	Err SerOpen (UInt16 refNum, UInt16 port,UInt32 baud)
พารามิเตอร์	-> refnum เลขอ้างอิงของไลบรารี่ของพอร์ตอนุกรม -> port หมายเลขพอร์ต -> baud อัตราบอด
ผลลัพธ์	คืนค่า 0 ถ้าไม่มีความผิดพลาด **serErrAlreadyOpen** พอร์ตถูกเปิดอยู่แล้ว สามารถใช้งานพอร์ตร่วมกันได้ (ไม่แนะนำให้ทำ) **serErrBadParam** พารามิเตอร์ไม่ถูกต้อง **memErrNotEnoughSpace** หน่วยความจำไม่เพียงพอ
หมายเหตุ	ปัจจุบันปาล์มมีพอร์ตอนุกรมเพียงพอร์ตเดียวคือ พอร์ตหมายเลข 0 อัตราบอด ก็เช่น (300, 1200, 2400, 9600, 19200, 38400, 57600,...) ปาล์มได้รับการทดสอบอัตราบอดมาตรฐาน ในช่วง 300 – 57600 ใช้ CTS แฮนด์เชค เมื่ออัตราบอดสูงกว่า 19200 ถ้าคืนฟังก์ชัน SerOpen นี้คืนค่า 0 หรือ serErrAlreadyOpen แสดงว่าพอร์ตถูกเปิดอยู่

SerSetSetting	
การใช้งาน	ตั้งค่าพอร์ตอนุกรม
โพรโตไทป์	Err SerSetSettings (UInt16 refNum,SerSettingsPtr settingsP)
พารามิเตอร์	-> refnum เลขอ้างอิงของไลบรารี่ของพอร์ตอนุกรม <-> SettingP พอยน์เตอร์ชี้ สตรักเจอร์ SerSettingType
ผลลัพธ์	0 ไม่เกิดความผิดพลาด **serErrNotOpen** พอร์ตไม่ได้เปิด **serErrBadParam** พารามิเตอร์ไม่ถูกต้อง
หมายเหตุ	แอททริบิวต์ที่ถูกตั้งค่าด้วยฟังก์ชันนี้รวมทั้งอัตราบอด CTS ไทม์เอาท์, แฮนด์เชค, รูปแบบข้อมูล หาข้อมูลเพิ่มเติมได้จากสตรักเจอร์ SerSettingType

SerSend	
การใช้งาน	ส่งข้อมูลหนึ่งไบต์หรือมากกว่าผ่านพอร์ตอนุกรม
โพรโตไทป์	UInt32 SerSend (UInt16 refNum, void *bufP,UInt32 count, Err *errP)
พารามิเตอร์	-> refnum เลขอ้างอิงของไลบรารี่ของพอร์ตอนุกรม -> bufP พอยน์เตอร์ชี้ไปที่ข้อมูลที่ต้องการส่ง -> count จำนวนไบต์ที่ต้องการส่ง <-> หมายเลขความผิดพลาด
ผลลัพธ์	จำนวนไบต์ที่ถูกส่ง ค่า errP 0 ไม่เกิดความผิดพลาด serErrTimeOut แฮนด์เชคไทม์เอาท์ (Handshake timeout)

โครงสร้างข้อมูลที่เกี่ยวข้อง

ControlType

```
typedef struct {
        UInt16 id;
        RectangleType bounds;
        Char * text;
        ControlAttrType attr;
        ControlStyleType style;
        FontID font;
        UInt8 group;
        UInt8 reserved;
} ControlType;
```

ControlPtr

```
typedef ControlType* ControlPtr;
```

FormType

```
typedef struct {
        WindowType window;
        UInt16 formId;
        FormAttrType attr;
        WinHandle bitsBehindForm;
        FormEventHandlerType * handler;
        UInt16 focus;
        UInt16 defaultButton;
        UInt16 helpRscId;
        UInt16 menuRscId;
        UInt16 numObjects;
        FormObjListType * objects;
} FormType;
```

FormPopupType

```
typedef struct {
        UInt16 controllD;
        UInt16 listID;
} FormPopupType;
```

FromLabelType

```
typedef struct {
        UInt16 id;
        PointType pos;
        FormObjAttrType attr;
        FontID fontID;
        UInt8 reserved;
        Char * text;
} FormLabelType;
```

ListType

```
typedef struct {
        UInt16 id;
        RectangleType bounds;
        ListAttrType attr;
        Char ** itemsText;
        Int16 numItems;
        Int16 currentItem;
        Int16 topItem;
        FontID font;
        UInt8 reserved;
        WinHandle popupWin;
        ListDrawDataFuncPtr drawItemCallback;
} ListType;
```

ListAttrTypes

```
typedef struct {
        UInt16 usable :1;
        UInt16 enabled :1;
        UInt16 visible :1;
        UInt16 poppedUp :1;
        UInt16 hasScrollBar:1.
        UInt16 search :1;
        UInt16 reserved :2;
} ListAttrType;
```

eventsEnum

```
enum events {
        nilEvent = 0,
        penDownEvent,
        penUpEvent,
        penMoveEvent,
        keyDownEvent,
        winEnterEvent,
        winExitEvent,
        ctlEnterEvent,
        ctlExitEvent,
        ctlSelectEvent,
        ctlRepeatEvent,
        lstEnterEvent,
        lstSelectEvent,
        lstExitEvent,
        popSelectEvent,
        fldEnterEvent,
        fldHeightChangedEvent,
        fldChangedEvent,
        tblEnterEvent,
        tblSelectEvent,
        daySelectEvent,
        menuEvent,
        appStopEvent = 22,
        frmLoadEvent,
        frmOpenEvent,
        frmGotoEvent,
        frmUpdateEvent,
        frmSaveEvent,
        frmCloseEvent,
        frmTitleEnterEvent,
        frmTitleSelectEvent,
        tblExitEvent,
```

```
        sclEnterEvent,
        sclExitEvent,
        sclRepeatEvent,
        tsmFepModeEvent,
        menuCmdBarOpenEvent = 0x0800,
        menuOpenEvent,
        menuCloseEvent,
        frmGadgetEnterEvent,
        frmGadgetMiscEvent,
        firstINetLibEvent = 0x1000,
        firstWebLibEvent = 0x1100,
        firstUserEvent = 0x6000
    } eventsEnum;
```

EventType

```
    typedef struct {
        eventsEnum eType;
        Boolean penDown;
        UInt8 tapCount;
        Int16 screenX;
        Int16 screenY;
        union{
            ...
        } data;
    } EventType;
```

คำอธิบาย

eType	ค่าคงตัว eventsEnum บอกชนิดของเหตุการณ์
penDown	เป็นจริงเมื่อปากกาถูกกดในขณะที่เกิดเหตุการณ์นั้นๆขึ้น
tapCount	จำนวนของการแตะหน้าจอในตำแหน่งใดๆ ใช้สำหรับฟิลด์คือ เมื่อผู้ใช้แตะครั้งแรกเป็นการเลือกฟิลด์นั้นๆ ครั้งที่สองเป็นการเลือกคำ ครั้งที่สามเป็นการเลือกทั้งบรรทัด
screenX	บอกตำแหน่งของปากกาในหน่วยพิกเซล นับจากด้านซ้ายไปขวา
screenX	บอกตำแหน่งของปากกาในหน่วยพิกเซล นับจากด้านบนลงล่าง
data	บอกข้อมูลเพิ่มเติมของเหตุการณ์นั้นๆ ซึ่งจะสัมพันธ์กับ eType

SerSettingsType

ใช้เก็บค่าแอตทริบิวต์ของพอร์ตอนุกรม ถูกใช้ในคำสั่ง SerGetSettings และ SerSetSettings. SerSettingsPtr เป็นพอยต์เตอร์ชี้ไปยัง SerSettingsType

```
typedef struct SerSettingsType {
        UInt32 baudRate;
        UInt32 flags;
        Int32 ctsTimeout;
} SerSettingsType;

typedef SerSettingsType* SerSettingsPtr;
```

คำอธิบาย

baudrate	อัตราบอด
flags	แฟลกเบ็ดเตล็ด
ctsTimeout	เวลาที่สามารถรอ CTS เมื่อต้องการส่งข้อมูล ใช้เมื่อมีการใช้งานแฟลก serSettingsFlagCTSAutoM

MenuBarType

```
typedef struct {
        WinHandle barWin;
        WinHandle bitsBehind;
        WinHandle savedActiveWin;
        WinHandle bitsBehindStatus;
        MenuBarAttrType attr;
        Int16 curMenu;
        Int16 curItem;
        Int32 commandTick;
        Int16 numMenus;
        MenuPullDownPtr menus;
} MenuBarType;
```

MenuBarPtr

```
typedef MenuBarType *MenuBarPtr;
```

SelectDayType

```
typedef enum {
        selectDayByDay,         // return d/m/y
        selectDayByWeek,        // return d/m/y with d as same day of the week
        selectDayByMonth        // return d/m/y with d as same day of the month
} SelectDayType;
```

DaySelectorType

```
typedef struct DaySelectorType {
        RectangleTypebounds;
        Booleanvisible;
        UInt8 reserved1;
        Int16visibleMonth;      // month actually displayed
        Int16visibleYear;       // year actually displayed
        DateTimeTypeselected;
        SelectDayTypeselectDayBy;
        UInt8 reserved2;
} DaySelectorType;
```

HMSTime

```
typedef struct {
        UInt8 hours;
        UInt8 minutes;
        UInt8 seconds;
        UInt8 reserved;
} HMSTime;
```

TimeFormatType

```
typedef enum{
        tfColon,
        tfColonAMPM,            // 1:00 pm
        tfColon24h,             // 13:00
        tfDot,
        tfDotAMPM,              // 1.00 pm
```

```
        tfDot24h,           // 13.00
        tfHoursAMPM,        // 1 pm
        tfHours24h,         // 13
        tfComma24h          // 13,00
    } TimeFormatType;
    typedef TimeFormatType* TimeFormatPtr;
```

DaylightSavingsTypes

```
    typedef enum {
        dsNone,             //Daylight Savings Time not
                            //observed
        dsUSA,              //United States Daylight
                            //Savings Time
        dsAustralia,        //Australian Daylight
                            //Savings Time
        dsWesternEuropean,  //Western European Daylight
                            //Savings Time
        dsMiddleEuropean,   //Middle European Daylight
                            //Savings Time
        dsEasternEuropean,  //Eastern European Daylight
                            //Savings Time
        dsGreatBritain,     //Great Britain and Eire
                            //Daylight Savings Time
        dsRumania,          //Rumanian Daylight Savings
                            //Time
        dsTurkey,           //Turkish Daylight Savings
                            //Time
        dsAustraliaShifted  //Australian Daylight
                            //Savings Time with shift
                            //in 1986
    } DaylightSavingsTypes;
```

DateFormatType

```
typedef enum {
        dfMDYWithSlashes,           // 12/31/95
        dfDMYWithSlashes,           // 31/12/95
        dfDMYWithDots,              // 31.12.95
        dfDMYWithDashes,            // 31-12-95
        dfYMDWithSlashes,           // 95/12/31
        dfYMDWithDots,              // 95.12.31
        dfYMDWithDashes,            // 95-12-31
        dfMDYLongWithComma,         // Dec 31, 1995
        dfDMYLong,                  // 31 Dec 1995
        dfDMYLongWithDot,           // 31. Dec 1995
        dfDMYLongNoDay,             // Dec 1995
        dfDMYLongWithComma,         // 31 Dec, 1995
        dfYMDLongWithDot,           // 1995.12.31
        dfYMDLongWithSpace,         // 1995 Dec 31
        dfMYMed,                    // Dec '95
        dfMYMedNoPost               // Dec 95
                                    //added for French 2.0 ROM)
} DateFormatType;
```

DateTimeType

```
typedef struct{
        Int16 second;
        Int16 minute;
        Int16 hour;
        Int16 day;
        Int16 month;
        Int16 year;
        Int16 weekDay;   //Days since Sunday (0 to 6)
}DateTimeType;
typedef DateTimeType* DateTimePtr;
```

TimeType

```
typedef struct {
        UInt8 hours;
        UInt8 minutes;
}TimeType;
typedef TimeType * TimePtr;
```

DateType

```
typedef struct{
        UInt16 year :7; //years since 1904 (Mac format)
        UInt16 month:4;
        UInt16 day :5;
}DateType;
typedef DateType * DatePtr;
```

AppInfoType

```
typedef struct {
        UInt16 renamedCategories;
        Char categoryLabels
        [dmRecNumCategories]
        [dmCategoryLength];
        UInt8 categoryUniqIDs
        [dmRecNumCategories];
        UInt8 lastUniqID;
        UInt8 padding;
} AppInfoType;
```

AppInfoPtr

```
typedef AppInfoType *AppInfoPtr;
```

บรรณานุกรม

หนังสืออ้างอิง

[1] Peter G. Randall. *"Laser Printer Reference,The"*, Prentice Hall Computer

[2] Andrew Binstock, David Babcock, Marvin Luse. *"HP LaserJet Programming"*, Addison wesley, 1992

เว็บไซต์อ้างอิง

[1] http://www.palmos.com

[2] http://www.palm.com

[3] http://www.palmgear.com

[4] http://www.palmdino.com

[5] http://www.mrpalm.com

[6] http://www.cs.washington.edu/homes/swanson/palmos_dev_documentation/CombinedTOC.html

[7] http://www.eecs.harvard.edu/~nr/pilot/pdb/

[8] http://www.nicholson.com/rhn/pilot.html

[9] http://www.msu.edu/~luckie/palms.html