ระบบอ่านลายมือเขียนภาษาไทย

THAI HANDWRITING RECOGNITION SYSTEM

สุรีรัตน์ เรืองอมรรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2549

ระบบอ่านลายมือเขียนภาษาไทย

THAI HANDWRITING RECOGNITION SYSTEM

โดย

สุรีรัตน์ เรืองอมรรัตน์

อาจารย์ที่ปรึกษา

รศ.ดร. บุญธีร์ เครือตราชู

ผศ. กฤตวัน ศิริบูรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบอ่าลายมือเขียนภาษาไทย

THAI HANDWRITTEN RECOGNITION SYSTEM

ผู้จัดทำ

1. นางสาวสุรีรัตน์ เรืองอมรรัตน์ รหัสนักศึกษา 46010887

_____ อาจารย์ที่ปรึกษา

(รศ.ดร. บุญธีร์ เครือตราชู)

_____ อาจารย์ที่ปรึกษา

(ผศ. กฤตวัน ศิริบูรณ์)

ระบบอ่านลายมือเขียนภาษาไทย

สุรีรัตน์ เรืองอมรรัตน์ 46010887
รศ.ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา
ผศ. กฤตวัน ศิริบูรณ์ อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2549

บทคัดย่อ

ในปัจจุบันคอมพิวเตอร์ได้เข้ามามีบทบาทต่อชีวิตประจำวันมากขึ้นและยังมีแนวโน้มที่จะช่วยอำนวยความสะดวกต่อผู้ใช้งานมากยิ่งขึ้นอีกด้วย จึงทำให้มีการพัฒนาระบบเพื่อช่วยให้การติดต่อระหว่างผู้ใช้งานและเครื่องคอมพิวเตอร์มีความง่ายในการใช้มากยิ่งขึ้น ถึงแม้ปัจจุบันนี้จะมีเทคโนโลยีการนำเข้าข้อมูลตัวอักษรอยู่หลายรูปแบบ และเทคโนโลยีที่ใช้งานอย่างแพร่หลายในปัจจุบันนี้คือการใช้คีย์บอร์ด ซึ่งการใช้งานนั้นผู้ใช้ต้องจดจำปุ่มบนแป้นพิมพ์ทั้งยังต้องฝึกฝนเพื่อให้ใช้งานได้อย่างคล่องแคล่ว ทำให้เกิดความไม่สะดวกต่อผู้ที่เริ่มต้นใช้งาน ดังนั้นวิธีการนำเข้าข้อมูลตัวอักษรที่มีความใกล้เคียงธรรมชาติ และเป็นสิ่งที่ผู้ใช้คุ้นเคยมาก่อน เช่น การใช้ลายมือเขียนของผู้ใช้เอง จึงเป็นสิ่งที่น่าจะช่วยให้ผู้ใช้รู้สึกสะดวกสบายและเคยชินมากขึ้นดั้งนั้นโครงงานนี้จึงนำเสนอระบบอ่านลายมือเขียนภาษาไทย เพื่ออ่านลายมือเขียนภาษาไทยของผู้ใช้งานแล้วแปลงเป็นอักษรตัวพิมพ์ภาษาไทย(Printed Character) บนคอมพิวเตอร์แบบออนไลน์ซึ่งประมวลผลลายมือเขียนภาษาไทยเป็นอักษรตัวพิมพ์ในขณะเขียน

THAI HANDWRITING RECOGNITION SYSTEM

Sureerat Reaung-a-mornrat	46010887
Assoc.Prof.Dr. Boontee Kruatrathue	Advisor
Asst.Prof. Kritawan Siriboon	Co-Advisor
Academic Year 2006	

ABSTRACT

The fact that people try to make life easy and comfortable is generally known. For that reason many new inventions have made such as intelligent automobile, mobile phone, PDA, etc. And, nowadays, a computer tends to take too many roles in every people's life: communicating to foreigners via the internet, making any reports or documents, calculating and doing any clerical tasks. It makes our life easier but it is still difficult to use computers. There are many way to input data into the computer: typing, using mouse, etc. Thai Language is the one of the most difficult languages in the world. It not only consists about eighty-six different characters but also has many levels of characters- some character write above the line, some character write on the line , and some character write below the line, which make it difficult to type. If someone wants to type Thai fluently, they have to remember the positions of each character on the keyboard and practice quite hard. This program was developed to help people - who have just started using a computer or cannot type fluently - input data to the computer via writing which is a natural skill for everybody. The program receives written characters and then immediately transforms them to printed characters.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก รศ.ดร. บุญธีร์ เครือตราชู ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์, ผ.ศ. กฤตวัน ศิริบูรณ์ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์ทั้งสองท่าน และขอขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

สุรีรัตน์ เรืองอมรรัตน์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันเทคโนโลยีต่างๆที่มีการพัฒนาขึ้นนั้นล้วนมีจุดประสงค์หรือเป้าหมายอย่างเดียวกัน คือ พยายามอำนวยความสะดวกให้แก่ผู้ใช้งานให้มากที่สุด ในขณะที่ใช้เวลาในการเรียนรู้วิธีใช้งานให้น้อยที่สุด เนื่องจากคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น เดิมคอมพิวเตอร์แสดงผลแบบตัวอักษร(text-base) ซึ่งยากต่อการใช้งานและใช้เวลาในการเรียนรู้นานจึงมีการใช้คอมพิวเตอร์ในวงจำกัดเท่านั้น ภายหลังมีการพัฒนาให้คอมพิวเตอร์แสดงผลแบบรูปภาพ (graphic)และปรับปรุงส่วนแสดงผลให้ง่ายต่อการใช้งานทำให้คอมพิวเตอร์ได้รับความนิยมมากขึ้นและมีการนำไปใช้งานในวงการต่างๆอย่างแพร่หลาย แต่อย่างไรก็ตามการใช้งานคอมพิวเตอร์ยังคงสร้างความยุ่งยากและเป็นเรื่องลำบากของผู้ใช้งานบางคนอยู่ โดยเฉพาะอย่างยิ่งผู้ที่เริ่มใช้งานคอมพิวเตอร์ใหม่ๆ การจะใช้งานคอมพิวเตอร์ให้มีประสิทธิผลและก่อประโยชน์สูงสุดนั้น การพิมพ์เร็วถือเป็นปัจจัยหนึ่งที่สำคัญ

ทักษะการพิมพ์เร็วต้องอาศัยการฝึกฝนซึ่งมักต้องใช้ระยะเวลานาน นอกจากนี้การพิมพ์อักษรภาษาไทยซึ่งมีถึง 84 ตัว โดยประมาณไม่ใช่เรื่องง่ายเลย แตกต่างจากการเขียนตัวอักษรซึ่งเป็นทักษะที่ทุกคนโดยส่วนใหญ่ได้ฝึกฝนและพัฒนามาตั้งแต่เด็ก การเขียนอักษรจึงถือเป็นเรื่องง่ายเมื่อเปรียบเทียบกับหาแป้นของตัวอักษรที่ต้องการบนคีย์บอร์ด จึงมีแนวคิดที่ว่าหากสามารถป้อนข้อมูลเข้าสู่เครื่องคอมพิวเตอร์ผ่านการเขียนได้น่าจะทำให้การใช้งานคอมพิวเตอร์สะดวกและง่ายดายกว่าที่เป็นอยู่ในปัจจุบันมากยิ่งขึ้น

ดังนั้นโครงงานนี้จึงนำเสนอระบบอ่านลายมือเขียนภาษาไทยซึ่งเป็นการนำเข้าข้อมูลตัวอักษรลายมือเขียน(Hand-writing Character) แล้วประมวลผลให้ตัวอักษรตัวพิมพ์(Printing Character)ที่มีลักษณะใกล้เคียงกับตัวอักษรลายมือเขียนที่อ่านเข้าไปมากที่สุดออกมา ผู้ใช้งานคอมพิวเตอร์จะรู้สึกเป็นธรรมชาติเพราะ การเขียนเป็นทักษะพื้นฐานที่ได้รับการฝึกฝนมาตั้งแต่เด็กและเป็นทักษะที่มีความคุ้นเคยเป็นอย่างดี จึงน่าที่จะทำให้การใช้งานคอมพิวเตอร์เป็นเรื่องที่ง่ายขึ้นกว่าเดิม

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาและพัฒนาโปรแกรมโดยใช้ CLI (Common Language Infrastructure) ใน .NET Framework

1.2.2 เพื่อพัฒนาระบบสำหรับเก็บโปรโตไทป์ (Prototype) อักษรภาษาไทยเพิ่มเติม

1.2.3 เพื่อศึกษาและพัฒนาระบบที่สามารถรับข้อมูลลายมือเขียนตัวอักษรภาษาไทยได้คราวละหลายๆตัวอักษร

1.2.4 สร้างระบบรับตัวอักษรลายมือเขียนภาษาไทย (Handwritten Character) โดยระบบสามารถรับและประมวลผลตัวอักษรได้ทุกชนิดทั้งอักษรไทยที่สามารถเขียนให้เสร็จในครั้งเดียว (1 stroke) , ตัวอักษรภาษาไทยที่ต้องเขียนหลายครั้ง และ ตัวอักษรภาษาไทยบนและใต้บรรทัด

1.2.5 สร้างระบบรับตัวอักษรลายมือเขียนภาษาไทย (Handwriting Character) ผ่านทางการลากเมาส์ แล้วนำไปประมวลผลให้ตัวอักษรตัวพิมพ์ (Printing Character) แบบออนไลน์ โดยรับตัวอักษรลายมือเขียนภาษาไทยหลายตัวอักษรได้

1.2.6 สร้างระบบฐานข้อมูลซึ่งเก็บลายมือเขียนภาษาไทยเพื่อนำไปใช้ในการพัฒนาอัลกอริทึมอื่นๆสำหรับการรู้จำลายมือเขียนภาษาไทยเพิ่มเติมได้

1.2.7 สร้างระบบฐานข้อมูลซึ่งเก็บลายมือเขียนภาษาไทยเพื่อนำไปใช้ในการพัฒนาอัลกอริทึมอื่นๆสำหรับการรู้จำลายมือเขียนภาษาไทยเพิ่มเติมได้

1.3 ขอบเขตของโครงงาน

1.3.1พัฒนาระบบอ่านลายมือเขียนภาษาไทยซึ่งสามารถอ่านลายมือเขียนตัวอักษรที่ตำแหน่งต่าง ๆ ของบรรทัดได้ คือ ในเส้นบรรทัด, บนบรรทัด และใต้บรรทัด

1.3.2 ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านลายมือเขียนที่ไม่ใช่ตัวอักษรโดด (Multiple Stroke Character) คือ เขียนไม่เสร็จในครั้งเดียวได้ ได้แก่ ญ, ฐ, ษ, ศ และ ส แต่ในการเขียนตัวอักษรที่เป็นตัวโดดจะต้องไม่มีการยกมือหรือปากกาในเวลาเขียน

1.3.3ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านตัวอักษรที่ไม่มีการเปลี่ยนแปลงมุมในระหว่างการเขียนได้ คือ ไม้เอก และ ไม้จัตวา

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถพัฒนาแอพพลิเคชั่นโดยใช้ .NET Framework ซึ่งช่วยให้การพัฒนาโปรแกรมเป็นไปได้อย่างรวดเร็ว

1.4.2 สามารถนำแนวคิดการโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ไปประยุกต์ใช้ในการพัฒนาโปรแกรม ได้

1.4.3 เข้าใจวิธีการและสามารถทำการดีบักโปรแกรมได้ดีขึ้น

1.4.4 เข้าใจขั้นตอนการพัฒนาระบบ โดยแบ่งระบบออกเป็นระบบย่อยพัฒนาระบบย่อยทดสอบระบบย่อยแล้วนำระบบย่อยที่พัฒนาเสร็จแล้วมารวมกัน แล้วทดสอบระบบหลังจากการรวมอีกครั้ง

1.4.5 เข้าใจและสามารถพัฒนาแอพพลิเคชั่นที่ติดต่อกับระบบฐานข้อมูลได้

1.5 System Requirement

1.5.1 ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านและประมวลผลออกมาเป็นตัวอักษรตัวพิมพ์ให้ครบทุกตัวอักษรในภาษาไทย

1.5.2 ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านข้อมูลตัวอักษรที่ตำแหน่งต่าง ๆ ของบรรทัด คือ ในเส้นบรรทัด, บนบรรทัด และใต้บรรทัด และต้องอ่านข้อมูลตัวอักษรได้ 5 บรรทัด

1.5.3 ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านลายมือเขียนอักษรภาษาไทยที่ไม่เป็นตัวอักษรโดด (Multiple Stroke Character) ได้ คือ ญ, ฐ, ย, ศ และ ส

1.5.4 ระบบอ่านลายมือเขียนภาษาไทยต้องสามารถอ่านลายมือเขียนอักษรภาษาไทยที่ไม่มีการเปลี่ยนแปลงมุมระหว่างการเขียนได้ คือ ไม้เอก และไม้จัตวา

1.5.5 ระบบอ่านลายมือเขียนภาษาไทยสามารถจัดเก็บข้อมูลที่ได้จากการใช้งานระบบ ซึ่งได้แก่ ลำดับของพิกัด(x, y) ตามการลากเมาส์ และตัวอักษรตัวพิมพ์ (Printed Character) ลงในฐานข้อมูลได้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ประกอบด้วยเนื้อหา 5 บท ได้แก่

บทที่ 1 บทนำ เป็นการอธิบายโครงงาน วัตถุประสงค์ของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ และขอบเขตของโครงงาน

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง ประกอบด้วย ทฤษฎีพื้นฐานที่เกี่ยวข้องกับโครงงานนี้

บทที่ 3 เทคโนโลยีและเครื่องมือในการพัฒนา

บทที่ 4 การออกแบบและพัฒนา

บทที่ 5 บทสรุป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

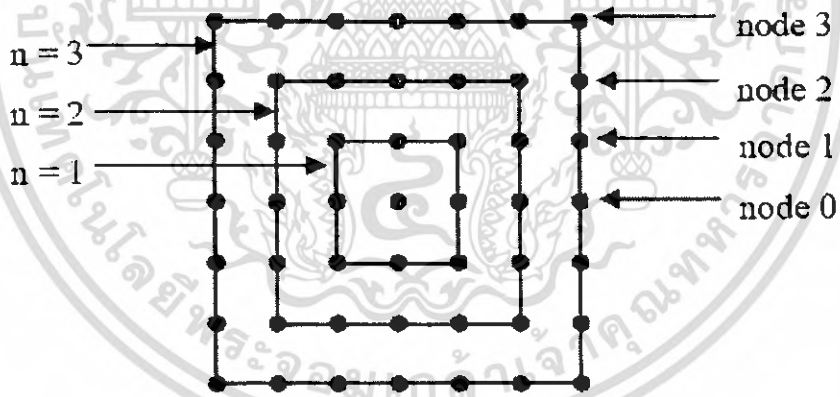
2.1 ทฤษฎีที่ใช้ในการพัฒนาโปรแกรมในส่วนรู้จำตัวอักษร(โครงงานเดิมพัฒนาโดยอ. คำเพ็ด บุญนะดี)

ข้อมูลที่ได้จากการเขียนตัวอักษรภาษาไทยเข้าสู่คอมพิวเตอร์ในรูปเซตของคู่อันดับหรือพิกัด(x, y) ซึ่งได้จากการ Sampling ตามเวลาของอุปกรณ์อิเล็คโทรนิค ข้อมูลนี้จะถูกนำมาเข้ารหัสที่เรียกว่า การเข้ารหัสลูกโซ่แบบGCC (Generalize Chain Code) รหัสลูกโซ่แบบGCC ประกอบด้วยตัวแปรสองตัว คือ

n : ตัวแปรกำหนดวงรหัสที่ มีลำดับเป็น n

node : แต่ละวงรหัสประกอบไปด้วย node ทั้งหมด M node เมื่อ M = (8 x n) และ n = 1,2,3,...

บนพื้นฐานที่กำหนดให้วงรหัสมีจุดใจกลางเพียงจุดเดียวซึ่งก็คือ จุดพิกัด (x, y) ตำแหน่งที่พิจารณาไปก่อนหน้านี้ทำให้ทุกๆลายมือเขียนสามารถที่จะเข้ารหัสได้



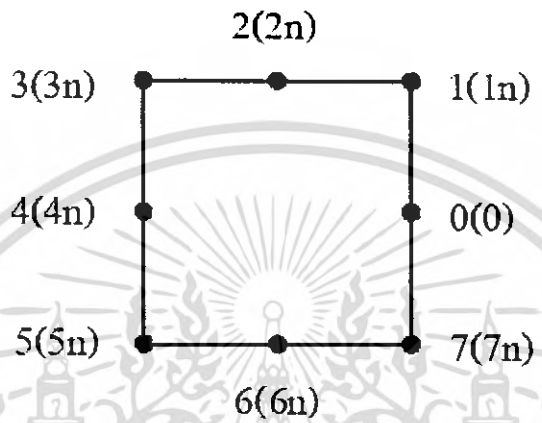
รูปที่ 2.1 แสดงส่วนประกอบของรหัสลูกโซ่แบบ GCC วงรหัสของและจำนวน node ที่มีในแต่ละวงรหัส

ในการคำนวณหารหัสลูกโซ่แบบ GCC สิ่งหนึ่งที่จะต้องหาก็คือ เวกเตอร์จากจุดกึ่งกลางวงรหัส (คือจุดที่ทำการเข้ารหัสไปก่อนหน้า) ไปยัง node ที่อยู่ใกล้ที่สุดที่ตัดกับวงรหัส โดยการจะกำหนดว่าnodeนั้นอยู่วงรหัสที่เท่าใดสามารถคำนวณได้จากผลต่างตามแกนx และ แกน y ของจุด2 จุดที่มีลำดับติดกันตามสมการที่ (1) โดยผลต่างตามแนวแกนใดมีค่ามากที่สุดก็จะนำมากำหนดลำดับวงรหัส

$$n = |\max(x2 - x1)| \quad \text{หรือ} \quad n = |\max(y2 - y1)| \qquad (1)$$

จากนั้นทำ Normalized GCC (NGCC) เพื่อลดจำนวนวงรหัสและเพื่อให้มี node เหลือเพียง 8 node หลัก โดย NGCC เป็นรหัส Ci ซึ่งคำนวณได้จาก

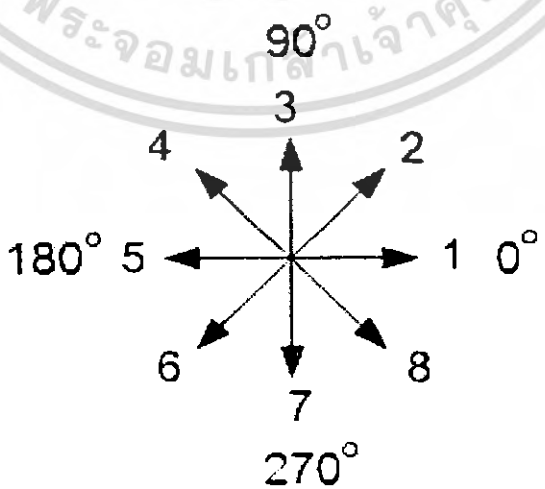
$$\mathbf{Ci = node/n} \qquad (2)$$



รูปที่ 2.2 GCC เมื่อทำการ Normalized แล้วจะเหลือเพียง 8 node ต่อวงรหัส

รหัส NGCC Ci จะอธิบายถึงค่าAbsoluteของทุกเส้นเชื่ อมต่อ (Link) ของ NGCC โดยไม่ขึ้นอยู่กับการกำหนดวงรหัส

ตัวอย่าง ถ้ารหัส NGCC มี เส้นเชื่ อมต่อที่มีค่า 0.78 เราจะได้ รหัสลูกโซ่ในระบบเป็น 0.78 + 1 = 1.78 เพราะรหัสลูกโซ่ที่นำมาใช้ในที่นี้จะเริ่มต้นด้วย 1 จนถึง 8.99 ซึ่งรหัสลูกโซ่ที่มีค่า 1.87 ก็จะได้เส้นเชื่อมต่ออยู่ระหว่างสองnodeที่ติดกันคือ node 0 และ node 1 และมุมของเส้นเชื่อมต่อนี้จะมีค่าอยู่ในช่วงของ $0 < \theta < 45$ องศาซึ่งกาเข้ารหัส NGCC จะเทียบเท่ากับการทำ quantization มุมดังแสดงในรูปที่ 3



รูปที่ 2.3 รหัสลูกโซ่แบบ GCC ที่ทำการ quantization แล้ว

ตัวอย่างการหารหัสลูกโซ่แบบ GCC ของตัวอักษรลายมือเขียนโดยสมมุติว่าจุดที่ Sampling ได้จากเขียนนั้นมีจุดดังนี้P1(344,183) ; P2 (343,186);P3 (342,187); P4 (340,188) ในการคำนวณหารหัสลูกโซ่ระหว่างจุด P1 และ P2 มีดังนี้
ก่อนอื่นจะคำนวณหาวงรหัสก่อนว่าอยู่ในวงรหัสที่เท่าไร ซึ่งคำนวณดังนี้

$$n = |MAX(P2(X) - P1(X))| = |MAX(343 - 344)| = 1 \quad \text{หรือ}$$
$$n = |MAX(P2(Y) - P1(Y))| = |MAX(186 - 183)| = 3$$

ซึ่งจะเลือกค่าที่มากสุดเป็นวงรหัสในที่นี้คือ 3 หลังจากนั้นหาระหว่างสองจุดซึ่งคำนวณหาได้ดังนี้

$$\text{Length} = \sqrt{1^2 + 3^2} = 3.1622$$

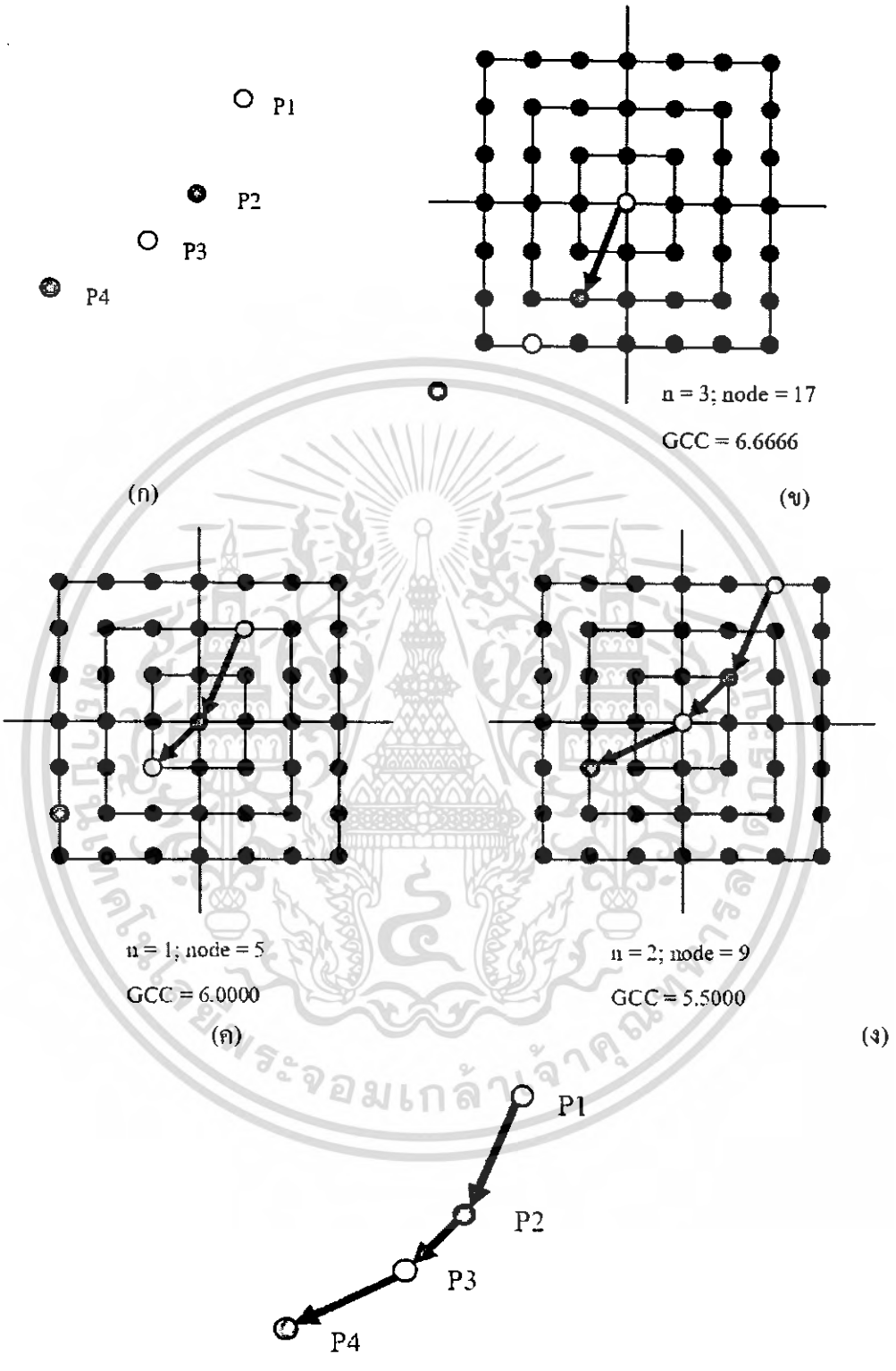
จากนั้นหาว่าจุดที่สนใจนั้นตกอยู่ใน node ที่เท่าใดในวงรหัสที่ 3 ซึ่งในนี้เป็น node ที่ 17 ดังรูปที่ 4 (ข) แล้วหาว่าเป็นรหัสลูกโซ่ที่เท่าใดซึ่งคำนวณจะได้จาก

$$\mathbf{Ci = node/n = 17 / 3 = 5.6666}$$

แต่เนื่องจากรหัส GCC ที่ใช้นั้นเริ่มต้นจาก 1 เป็นต้นไปดังรหัส

$$\mathbf{GCC = Ci + 1 = 5.6666 + 1 = 6.6666}$$

และจุดต่อไปก็จะทำเช่นเดียวกันนี้ไปเรื่อยไปจนครบทุกจุด



รูปที่ 2.4 แสดงตัวอย่างการเข้ารหัสลูกโซ่แบบ GCC

เนื่องจากการรู้จำตัวอักษรเป็นขบวนการสำคัญที่ทำให้สามารถประมวลผลตัวอักษรลายมือเป็นตัวอักษรตัวพิมพ์ได้ถูกต้อง ดังนั้นการดึงเอาคุณลักษณะของตัวอักษร (Feature Extraction) เป็นขบวนการที่สำคัญ เพราะระบบการรู้จำขึ้นอยู่กับคุณสมบัติหลักๆสองประการ คือ

1.การเลือกคุณลักษณะเด่นของตัวอักษร (Feature)

2. การเลือก Recognizers ที่จะทำการรู้จำคุณลักษณะเด่นของตัวอักษรแต่ละตัว

ดังนั้นการหาและการเลือกคุณลักษณะเด่นของตัวอักษรจึงเป็นส่วนที่สำคัญเพราะคุณลักษณะเด่นจะบ่งบอกถึงตัวอักษรตัวนั้นๆและตัวอักษรแต่ละตัวก็จะมีคุณลักษณะเด่นเฉพาะตัวที่แตกต่างกัน

การหาคุณลักษณะเด่นของตัวอักษรนั้นจะหาโดยแบ่งตัวอักษรออกเป็นเซ็กเมนต์จุดแบ่งเซ็กเมนต์ของตัวอักษรนั้นจะเป็นจุดเปลี่ยนทิศทางในการเขียนตัวอักษรตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา โดยพิจารณาทิศทางการเขียนจากค่า Diffcode ซึ่งค่า Diffcode คือค่าความแตกต่างของรหัสลูกโซ่ GCC สองตัวที่อยู่ติดกันที่เชื่อมต่อกัน

$$\mathbf{Diffcodei = GCCi+1 - GCCi} \quad (3)$$

จุดแบ่งเซ็กเมนต์จะเป็นจุดที่มีการเปลี่ยนแปลงของค่า Diffcode โดยจะมีการเปลี่ยนแปลงค่าจากค่าบวกที่อยู่ติดกันหลายๆค่าเป็นลบที่ติดอยู่กันหลายๆค่า หรือเปลี่ยนจากลบที่อยู่ติดกันหลายๆค่าเป็นบวกที่อยู่ติดกันหลายๆค่า การวาด
เส้นโค้งในทิศทางทวนเข็มนาฬิกาจะทำให้รหั สลูกโซ่ GCC มีค่าเพิ่มขึ้นทำให้ค่าความแตกต่าง Diffcode มีค่าเป็นบวก ในขณะที่การวาดเส้นโค้งไปในทิศตามเข็มนาฬิกาจะทำให้รหัสลูกโซ่ GCC มีค่าลดลงทำให้ค่าความแตกต่าง Diffcodeนั้นมีค่าเป็นลบทั้งหมด

เมื่อได้จุดแบ่งเซ็กเมนต์ของตัวอักษรแล้วทำการหาค่าสะสมของ Diffcode ในแต่ละเซ็กเมนต์ โดยจะได้ค่าสะสมเป็นบวกหากเซ็กเมนต์นั้นเขียนไปในทิศทวนเข็มนาฬิกา และค่าสะสมของ Diffcode เป็นค่าลบหาก Segmnet นั้นเขียนไปในทิศตามเข็มนาฬิกา แต่เนื่องจากการหาในแบบนี้ยังมีบางตัวอักษรที่ต่างกันแต่เมื่อทำการหาเซ็กเมนต์แล้วได้จำนวนของเซ็กเมนต์เท่ากันและมีค่าสะสมความแตกต่าง Diffcode ที่ใกล้เคียงกัน ทำให้เราไม่สามารถแยกความแตกต่างของตัวอักษรดังกล่าวออกจากกันได้

Cumulativ Diffcode					
ธ	1	2	3	4	5
seg 1	-0.584182	-0.612194	-0.637535	-0.596017	-0.612194
seg 2	0.47638	0.427905	0.427905	0.394501	0.427905
จ					
seg 1	-0.584182	-0.622869	-0.601073	-0.584182	-0.601073
seg 2	0.50133	0.584182	0.457577	0.414182	0.405177
ม					
seg 1	-0.552867	-0.584182	-0.616955	-0.584182	-0.622869
seg 2	0.47638	0.368578	0.33985	0.427905	0.448843
ย					
seg 1	-0.566144	-0.507643	-0.660669	-0.612194	-0.584182
seg 2	0.448843	0.465003	0.47638	0.427905	0.427905
ล					
seg 1	-0.552866	-0.627444	-0.517364	-0.50133	-0.552867
seg 2	0.457577	0.497357	0.47638	0.489854	0.517364

ตารางที่ 1 แสดงค่าสะสมความแตกต่าง Diffcode ที่ใกล้เคียงกันของตัวอักษรที่ต่างกัน

ในการที่จะแยกตัวอักษรที่ต่างกันแต่มีจำนวนเซ็กเมนต์และค่าสะสมความแตกต่าง Diffcode ใกล้เคียงกันหรือเท่ากันออกจากกัน จะใช้ค่าความยาวในการลากตามแกน X และแกน Y จากซ้ายไปขวา ขวามาซ้าย จากบนลงล่าง และล่างขึ้นบน คิดเป็นเปอร์เซ็นต์ของความยาวในแต่ละเซ็กเมนต์ มาช่วยในการแยกตัวอักษรเหล่านี้ออกจากกัน

DiffLengthX เป็นความยาวระหว่างสองจุดที่อยู่ติดกันในแกน X

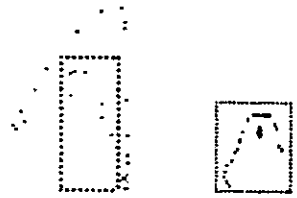
DiffLengthY เป็นความยาวระหว่างสองจุดที่อยู่ติดกันในแกน Y

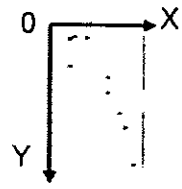
$$\mathbf{DiffLengthX\ [i] = X\ [i+1] - X[i]}$$

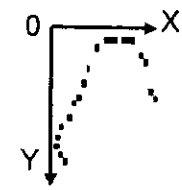
$$\mathbf{DiffLengthY\ [i] = Y\ [i+1] - Y[i]}$$

(4)

ในแต่ละเซ็กเมนต์หา ผลรวมของ DiffLengthX ที่เป็นบวก, DiffLengthXที่เป็นลบ, DiffLengthY และ DiffLengthY ที่เป็นลบแล้วคิดเป็นเปอร์เซ็นต์เทียบกับความยาวทั้งหมดของเซ็กเมนต์นั้น







Segment	1			
จ	X+	X-	Y+	Y-
	39	-23	76	-33
	X+(%)	X-(%)	Y+(%)	Y-(%)
	62.9	37.1	69.72	30.28

Segment	1			
ล	X+	X-	Y+	Y-
	22	-3	7	-17
	X+(%)	X-(%)	Y+(%)	Y-(%)
	88	12	29.17	70.83

รูปที่ 2.5 แสดงการเปรียบเทียบความแตกต่างของค่า X Y ของตัวอักษร จ และ ล ในเซ็กเมนต์แรก

จ										
	Segment 1					Segment 2				
ตัวที่	sumdiff	X + (%)	X - (%)	Y + (%)	Y - (%)	sumdiff	X + (%)	X - (%)	Y + (%)	Y - (%)
1	-0.584182	70.4545	29.54545	83.63636	16.36364	0.50133	20.83333	79.16667	13.72549	86.27451
2	-0.622869	84.6154	15.38462	85.71429	14.28571	0.584182	13.46154	86.53846	18.86792	81.13208
3	-0.601073	81.25	18.75	85	15	0.457577	7.8125	92.1875	15.04425	84.95575
4	-0.584182	81.6667	18.33333	83.82353	16.17647	0.414182	22	78	32.06107	67.93893
5	-0.601073	63.4921	36.50794	79.54545	20.45455	0.405177	26.19048	73.80952	30.90909	69.09091

ล										
	Segment 1					Segment 2				
ตัวที่	sumdiff	X + (%)	X - (%)	Y + (%)	Y - (%)	sumdiff	X + (%)	X - (%)	Y + (%)	Y - (%)
1	-0.552866	71.0526	28.94737	40.90909	59.09091	0.457577	18.33333	81.66667	33.33333	66.66667
2	-0.627444	70.3704	29.62963	38.46154	61.53846	0.497357	28.57143	71.42857	27.27273	72.72727
3	-0.517364	67.6471	32.35294	37.5	62.5	0.47638	32.39437	67.60563	25	75
4	-0.50133	64.5161	35.48387	46.15385	53.84615	0.489854	32.43243	67.56757	26.47059	73.52941
5	-0.552867	71.4286	28.57143	35.29412	64.70588	0.517364	26.15385	73.84615	24.32432	75.67568

ตารางที่ 2 แสดงความแตกต่างของตัวอักษร จ และ ล ที่ใช้ความยาวในแกน X Y ในการแยก

การหาคุณลักษณะเด่นของตัวอักษรโดยใช้จุดเปลี่ยนทิศทางในการเขียนตัวอักษรเป็นจุดแบ่งเซ็กเมนต์ซึ่งพิจารณาจากค่า Diffcode ที่มีเครื่องหมายที่ต่างกัน และการที่ผลรวมของค่า Diffcodeไม่ขึ้นกับการหมุน ทำให้เส้นโค้งที่มีรูปร่างที่เหมือนกันแต่มีขนาดที่แตกต่าง แบ่งเซ็กเมนตแล้วให้ผลที่เหมือนกันนั้นคือขนาดไม่มีผลต่อการแบ่งเซ็กเมนต์ถึงแม้ว่าการหาคุณลักษณะเด่นของตัวอักษรแบบนี้ทำให้ตัวอักษรที่ไม่เหมือนกันมีจำนวนเซ็กเมนต์ที่เท่ากันและผลรวมค่าความแตกต่าง Diffcode ในเซ็กเมนต์ใกล้เคียงกันก็ตาม แต่ก็สามารถแยกอักษรดังกล่าวออกจากกันได้โดยการหาความยาวตามแกน X แกน Y และการหาความยาวของแต่

ละเซ็กเมนต์ก ซึ่งทำให้การหาคุณลักษณะเด่นของตัวอักษรในลักษณะนี้สามารถแยกความแตกต่างของตัวอักษรได้ถูกต้องแม่นยำมากขึ้น และมีความยืดหยุ่นเพียงพอสำหรับตัวอักษรที่มีการหมุน หรือ มีการ
เปลี่ยนแปลงรูปร่างที่ไม่คงที่

2.2 ทฤษฎีที่ใช้ในการพัฒนาโครงงานนี้

2.2.1 ตัวอักษร แบ่งเป็นประเภทต่างๆดังนี้

1. อักษรโดด คือ ตัวอักษรที่เขียนเสร็จได้ในครั้งเดียว
เช่น น.หนู ก.ไก่ สระอา สระอิ เป็นต้น

2. อักษรที่อาจเกิดจากอักษรโดดมากกว่า 1 ตัว ได้แก่
สระ อะ สระอี สระอือ สระแอ สระอำ ไม้จัตวา ย.ฤๅษี ศ.ศาลา
ส.เสือ ญ.หญิง และ ฐ.ฐาน

3. อักษรที่เกิดจากการเขียนอักษรโดดมากกว่า 2 ตัว
ได้แก่ สระอือ

2.2.2 การวิเคราะห์อักษรที่ไม่เป็นอักษรโดด

สามารถทำได้ 2 วิธีคือ

2.2.2.1 หาเอาท์พุทอักษรโดดก่อนตรวจการซ้อนทับ (Overlapping) ของลายมือเขียน เป็นวิธีที่ใช้ในระบบปัจจุบัน

ข้อดี

- สามารถเขียนตัวอักษร Single Stroke 2 ตัวทับกันได้

ข้อเสีย

- หากเอาท์พุทอักษรโดดที่ได้จากการรู้จำผิดพลาดก็จะทำให้เอาท์พุทที่แท้จริงผิดพลาดด้วย

2.2.2.2 ตรวจการซ้อนทับ (Overlapping) ก่อนประมวลผลหาอักษรเอาท์พุท

ข้อดี

- เอาท์พุทขึ้นกับผลการ Recognition น้อยกว่าวิธีการแรก
- จำนวน State ลดลงทำให้ความซับซ้อนของโปรแกรมลดลง

ข้อเสีย

- อาจจะเกิดความผิดพลาดในกรณีที่เขียนตัวอักษร Single Stroke 2 ตัวซ้อนทับกัน

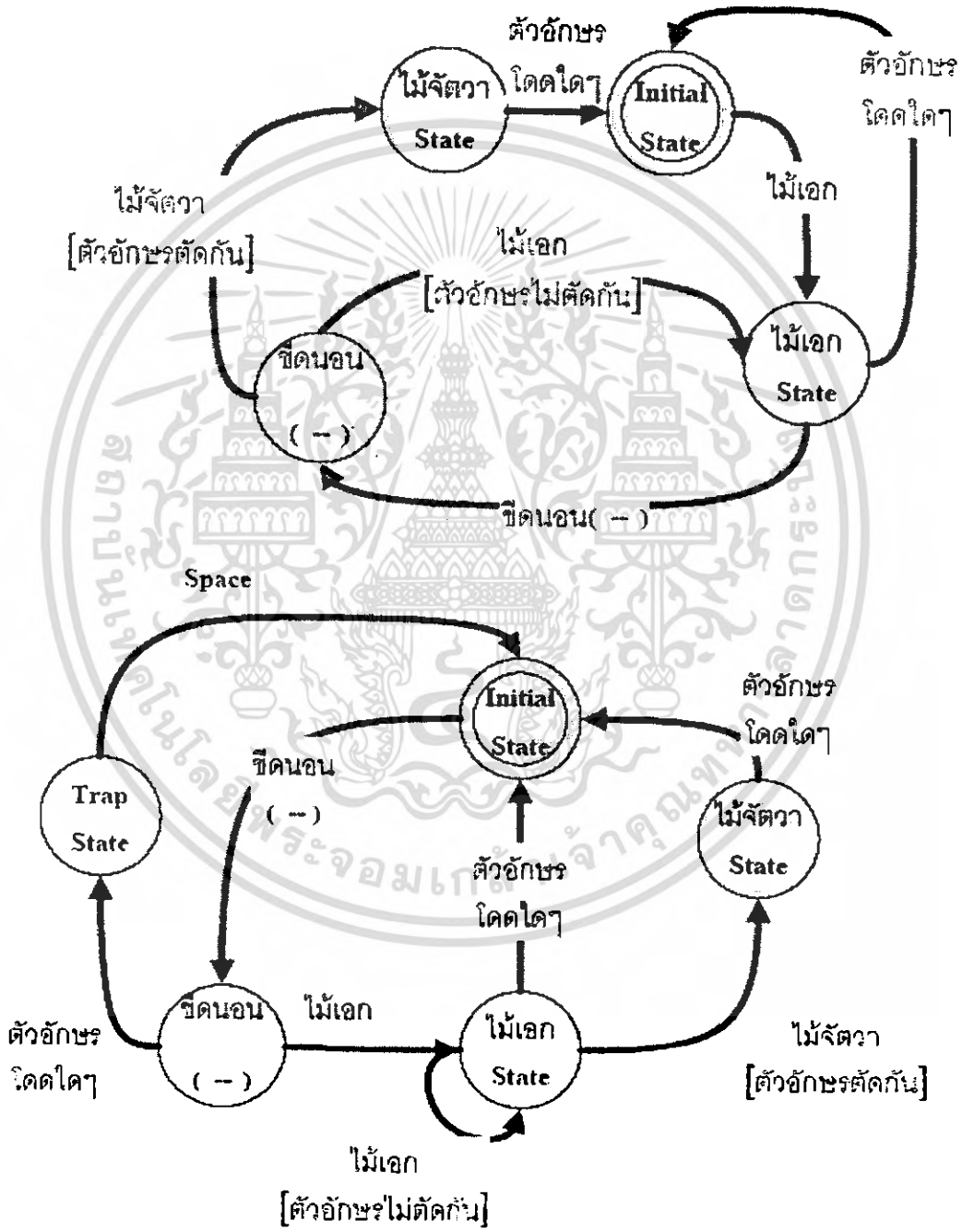
2.2.2.1 หาเอาท์พุทอักษรโดดก่อนตรวจการซ้อนทับ (Overlapping) ของลายมือเขียน เป็นวิธีที่ใช้ในระบบปัจจุบัน

ระบบวิเคราะห์อักษร Multiple Strokes ทำโดยเทียบอักษรโดดก่อนหน้า กับ อักษรโดดปัจจุบัน

การจัดการกับกรณีต่างๆข้างต้นโดยการใช้ if else จะทำให้โปรแกรมเข้าใจยากและซับซ้อนเพราะต้องคำนึงถึงหลายกรณี

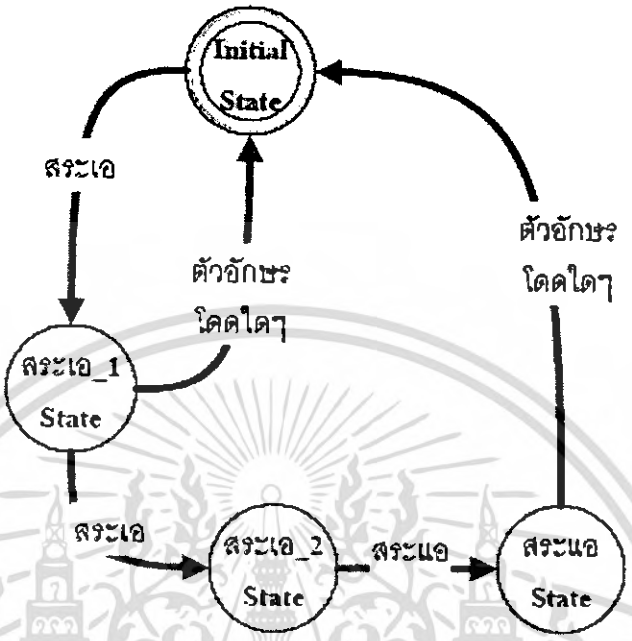
ด้วยเหตุนี้จึงนำ State Machine มาใช้ในการจัดการดังประกอบด้วย state และ transition ต่างๆ ดังนี้

1. State Machine ไม้จัตวา เมื่อรับอินพุทตัวอักษรเป็น อักษรโคดไม้เอก และ ขีดนอน ที่หากตัดกันจะได้ตัวอักษรไม้จัตวา



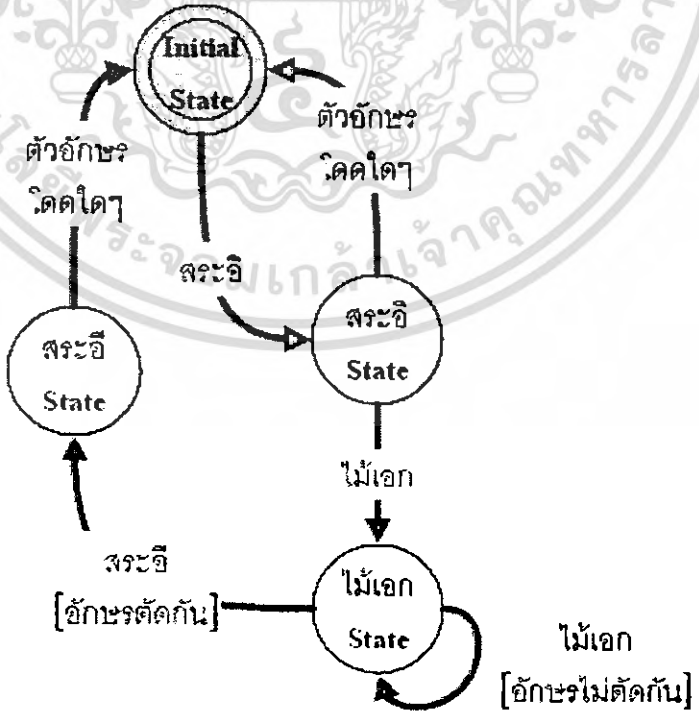
รูปที่ 2.6 แสดง State Machine ไม้จัตวา

2. State Machine สระแอ เมื่อรับอินพุทตัวอักษรเป็น อักษรโดดสระเอ 2 ตัว ทำให้ได้เอาพุทเป็นสระแอ



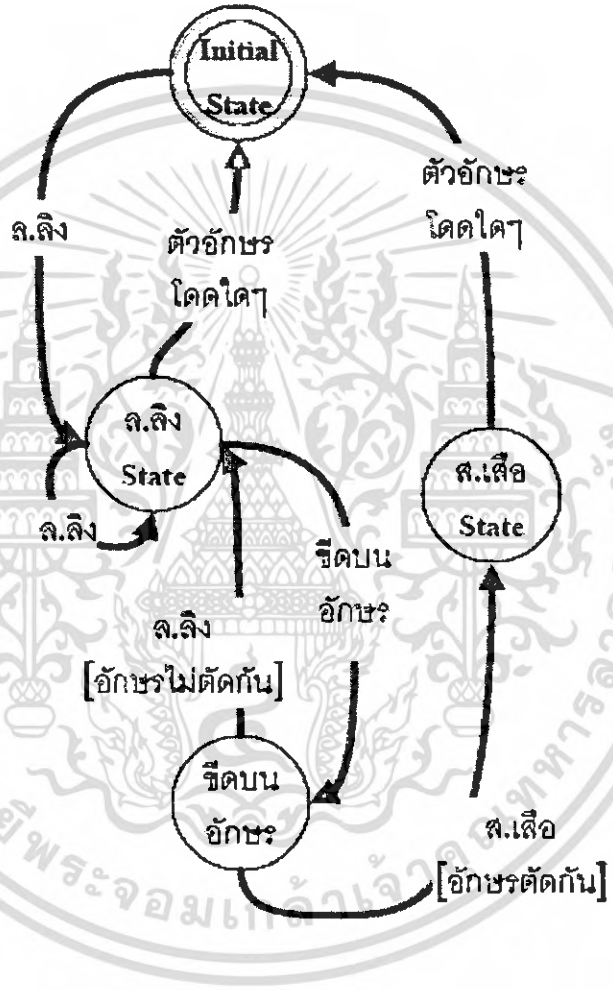
รูปที่ 2.7 แสดง State Machine สระแอ

3. State Machine สระอี เมื่อรับอินพุทตัวอักษรเป็น อักษรโดดไม้เอก และ สระอิ ซึ่งหากตัดกันจะได้สระอี



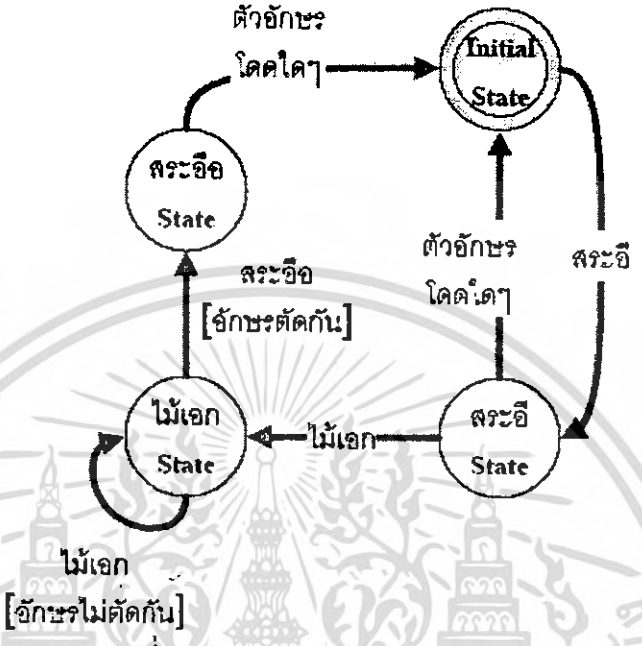
รูปที่ 2.8 แสดง State Machine สระอี

4. State Machine ส.เสือ เมื่อรับอินพุทตัวอักษรเป็น อักษรโตคล.ลิงและ ขีดบนตัวอักษร ที่หากตัดกันจะได้ตัวอักษรส.เสือ



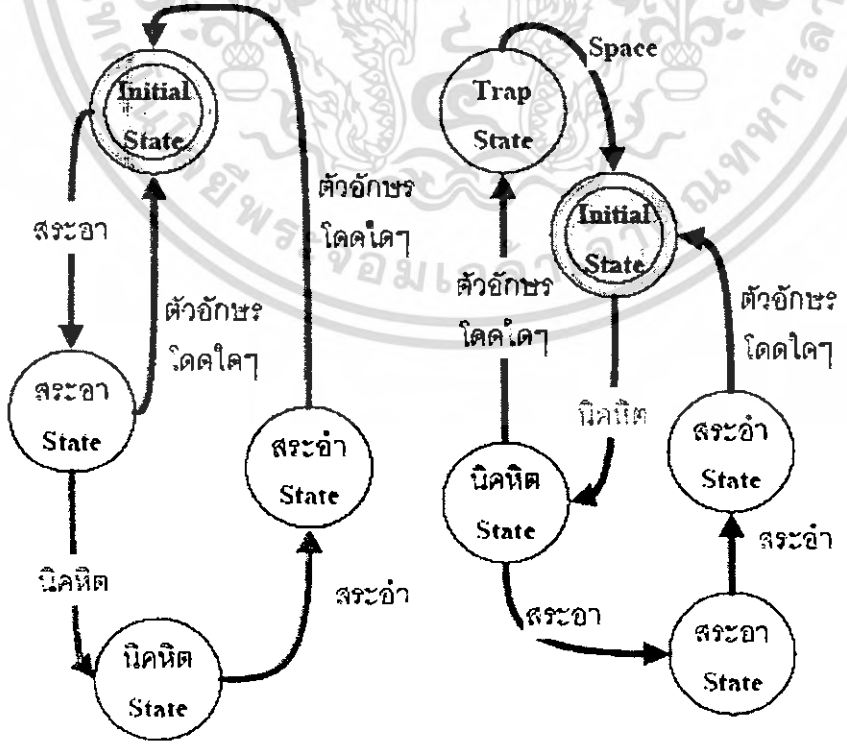
รูปที่ 2.9 แสดง State Machine ส.เสือ

5. State Machine สระออ เมื่อรับอินพุทตัวอักษรเป็น อักษรโดดไม้เอก และ สระอิ หรือ สระอีซึ่งหากตัดกันจะได้สระออ



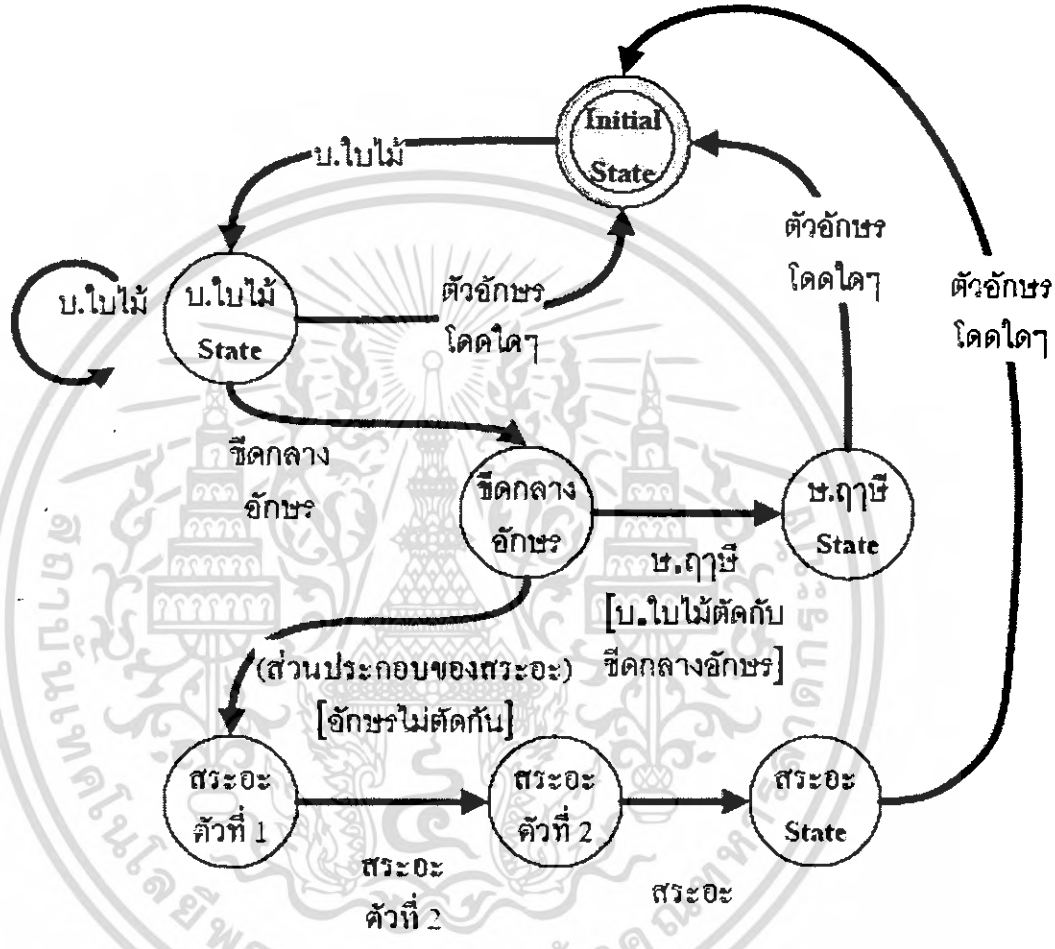
รูปที่ 2.10 แสดง State Machine สระออ

6. State Machine สระอำ เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด นิคหิต และ สระอา หรือ จะได้อักษรเอาท์พุทเป็นสระอำ



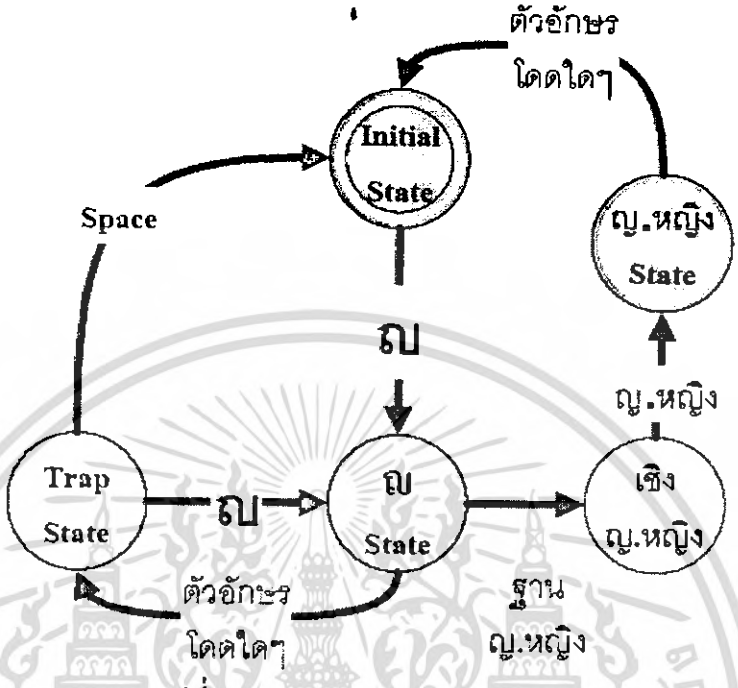
รูปที่ 2.11 แสดง State Machine สระอำ

7. State Machine ย.ฤๅษี เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด บ.ใบไม้ และ ขีดกลางตัวอักษร จะได้อักษรเอาท์พุทเป็น ย.ฤๅษี



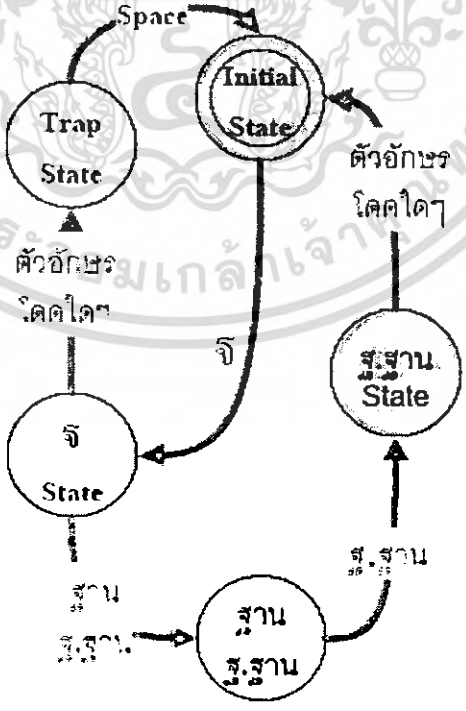
รูปที่ 2.12 แสดง State Machine ย.ฤๅษี

8. State Machine ญ.หญิง เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด ญ.หญิง และ เชิง ญ.หญิง จะได้อักษรเอาพุทเป็น ญ.หญิง



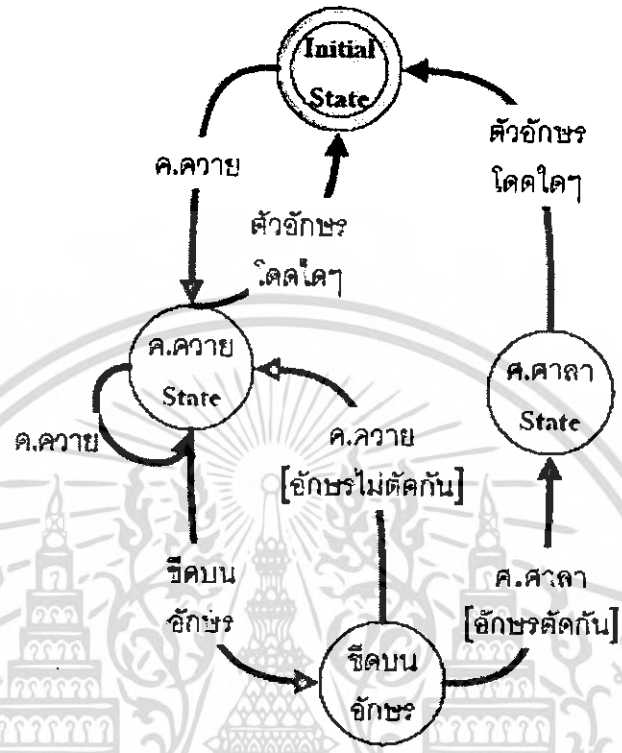
รูปที่ 2.13 แสดง State Machine ญ.หญิง

9. State Machine ฐ.ฐาน เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด ฐ.ฐาน และ ฐาน ฐ.ฐาน จะได้อักษรเอาพุทเป็น ฐ.ฐาน



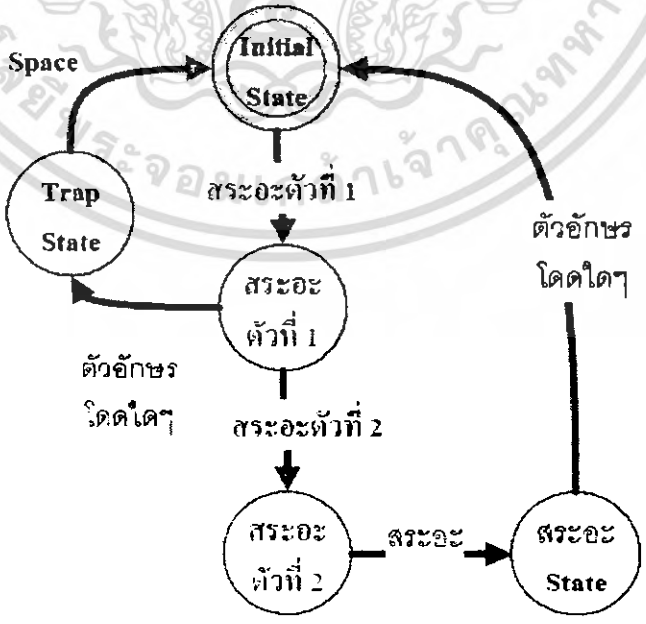
รูปที่ 2.14 แสดง State Machine ฐ.ฐาน

10. State Machine ศ.ศาลา เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด ค.ควาย และ ขีดบนตัวอักษร จะได้อักษรเอาพุทเป็น ศ.ศาลา



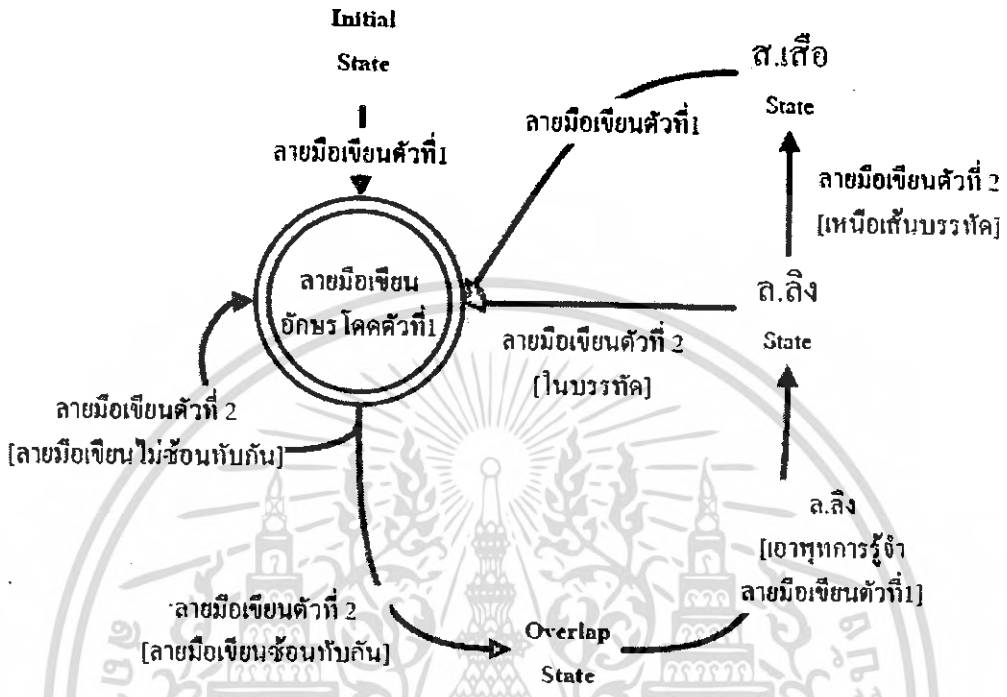
รูปที่ 2.15 แสดง State Machine ศ.ศาลา

11 State Machine สระอะ เมื่อรับอินพุทตัวอักษรเป็น อักษรโดด สระอะตัวที่1 และ สระอะตัวที่2 จะได้อักษรเอาท์พุทเป็น สระอะ



รูปที่ 2.16 แสดง State Machine สระอะ

2.2.2.2 ตรวจการซ้อนทับ (Overlapping) ก่อนประมวลผลหาอักษรเอาท์พุท

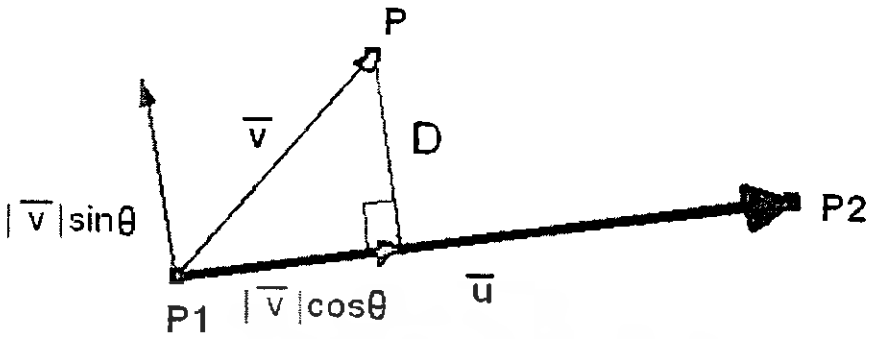


รูปที่ 2.17 แสดง State Machine ของอักษร ส.เสือ เมื่อใช้วิธีการที่ 2 ในการจัดการ

2.2.3 การจัดการตัวอักษรที่ไม่มีการเปลี่ยนแปลงมุมระหว่างการเขียน

อักษรที่ไม่มีการเปลี่ยนแปลงมุมระหว่างการเขียน คือ อักษรที่ไม่มีส่วนโค้งหรือไม่มีจุดหักเห ได้แก่ ไม้เอก และไม้จัตวา การวิเคราะห์อักษรที่ไม่มีส่วนโค้งเว้าทำได้โดยทดสอบความเป็นเส้นตรงของเซตของพิกัด(x, y) จากระยะห่างระหว่างจุดใดในเซตไปยังเส้นตรงที่ลากจากจุดเริ่มต้นไปยังจุดสุดท้าย

2.2.3.1. การระยะห่างจากจุดใดๆไปยังเส้นตรง



รูปที่ 2.18 แสดงการหาระยะห่างจากจุด P ไปยังเส้นตรง P_1P_2

การหาระยะห่างจากจุด P ไปยังเส้นตรง P_1P_2

a) สร้างเวคเตอร์ u ซึ่งเป็นเวคเตอร์จากจุด P_1 ไปยัง P_2

b) สร้างเวคเตอร์ v ซึ่งเป็นเวคเตอร์จากจุด P_1 ไปยัง P

c) ให้ระยะห่างจากจุด **P** ไปยังเส้นตรง P_1P_2 เป็น D

$$D = |\overline{v}|\sin\theta$$

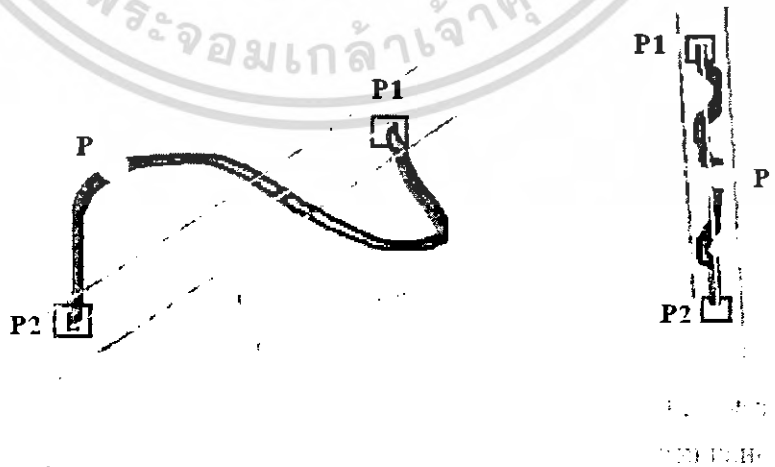
และเนื่องจาก $|\overline{u} \times \overline{v}| = |\overline{u}||\overline{v}|\sin\theta$

ดังนั้น ระยะห่างจากจุด P ไปยังเส้นตรง P_1P_2 คำนวณได้จาก

$$D = \frac{|\overline{u} \times \overline{v}|}{|\overline{u}|}$$

2.2.3.2. การวิเคราะห์ความเป็นเส้นตรงของเซตของพิกัด(x, y)

a) ถ้ากำหนดให้จุด P_1 เป็นจุดเริ่มต้นของเส้น และ P_2 เป็นจุดสุดท้ายดังรูป



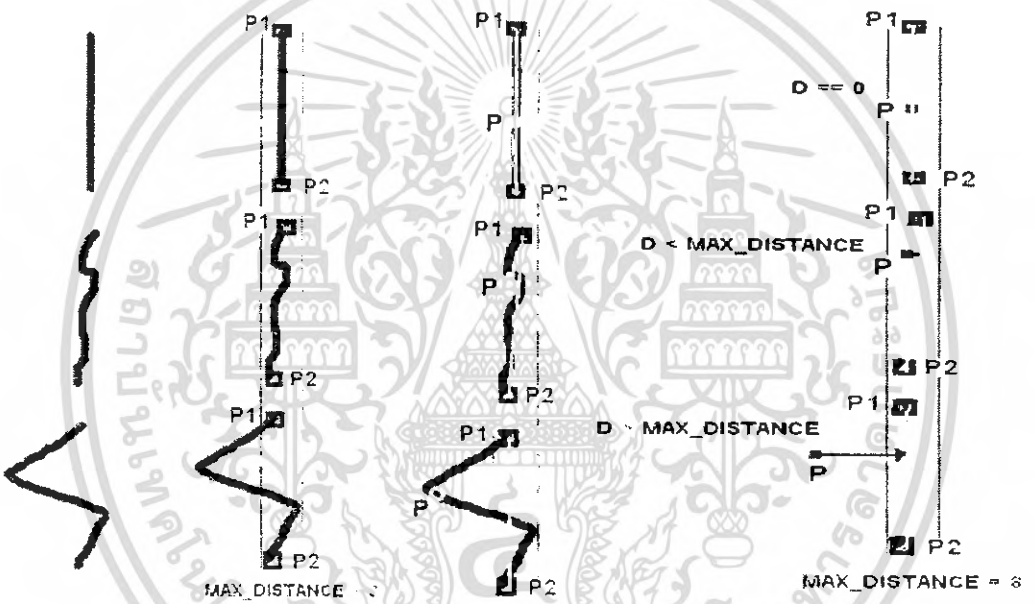
รูปที่ 2.19 แสดงการวิเคราะห์ความเป็นเส้นตรงของเส้นลายมือเขียน

b) สร้างเส้นตรงจากP_1 ไปยัง P_2

c) กำหนดให้จุดใดๆในเส้น เป็นจุด P

d) หาระยะห่างจากจุด P ไปยังเส้นตรง P_1P_2

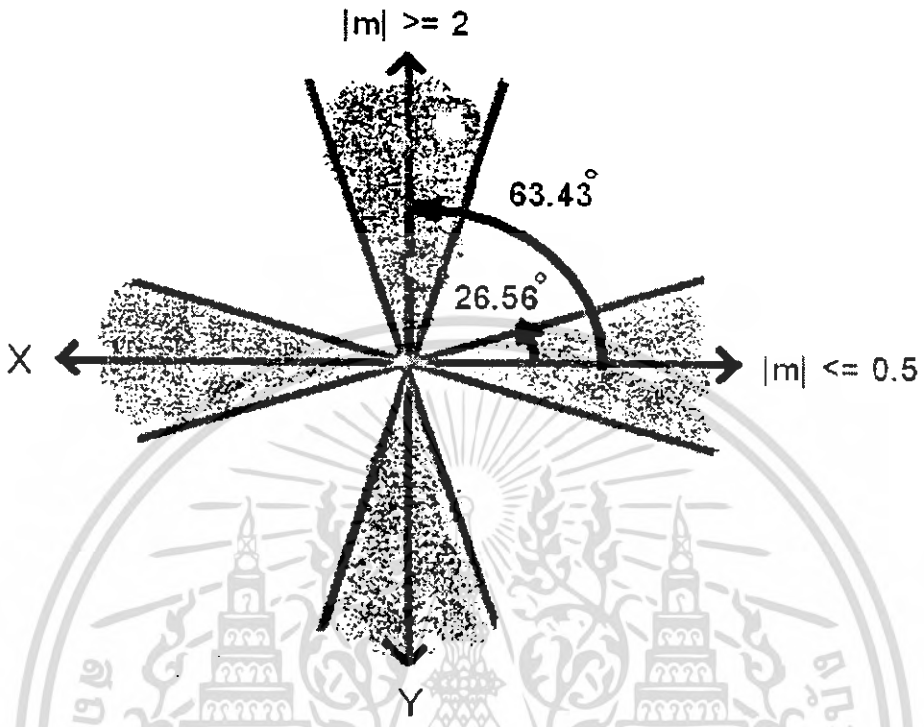
e) หากมีจุด P ใดๆ ที่มีระยะห่างไปยังเส้นตรง มากกว่า ค่า MAX_DISTANCE ซึ่งเป็นระยะห่างสูงสุดที่เป็นไปได้ที่กำหนดไว้ล่วงหน้าแล้ว (ค่า threshold) แสดงว่าเส้นดังกล่าวไม่เป็นเส้นตรง



รูปที่ 2.20 แสดงการวิเคราะห์ความเป็นเส้นตรงของเส้นลายมือเขียน

เมื่อผ่านการทดสอบนี้เซตของพิกัด(x, y)ที่ไม่เป็นเส้นตรงจะผ่านฟังก์ชันนี้ออกไปและเข้าสู่การวิเคราะห์หาอักษรด้วยพิมพ์ด้วยวิธีปกติ (เรียกใช้โมดูลเดิมในการวิเคราะห์หาเอาท์พุท) แต่ถ้าหากเซตของพิกัด(x, y)เป็นเส้นตรงจะเข้าสู่การวิเคราะห์หาอักษรตัวพิมพ์ด้วยฟังก์ชันที่ใช้

2.2.3.3. ทฤษฎีตรีโกณมิติและความชันของเส้นตรง มาช่วยในการวิเคราะห์ดังนี้



รูปที่ 2.21 แสดงการวิเคราะห์รหัส ASCII ของเส้นลายมือเขียนที่เป็นเส้นตรง

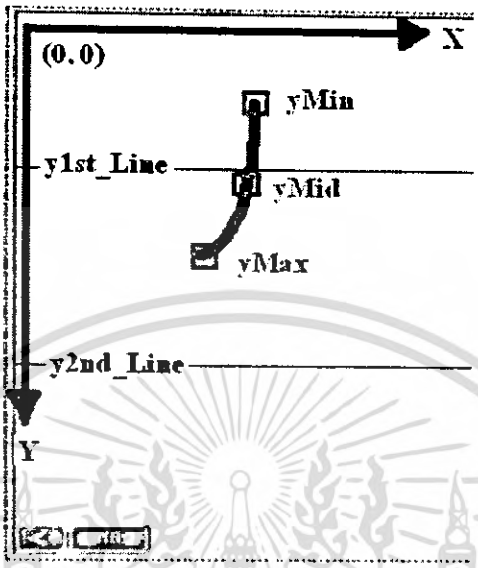
a) หากเส้นตรงที่ได้มีความชันมากกว่าหรือเท่ากับสอง

| m | >= 2 แสดงว่า เส้นเอียงเป็นมุมประมาณ 63.43 องศา กับแนวแกนX ประมาณให้เป็นเส้นตรงตามแนวแกนY และให้เอาท์พุทเป็นไม้เอก

b) หากเส้นตรงที่ได้มีความชันน้อยกว่าหรือเท่ากับสอง

| m | <= 0.5 แสดงว่า เส้นเอียงเป็นมุมประมาณ 26.56 องศา กับแนวแกนX ประมาณให้เป็นเส้นตรงตามแนวแกนX และให้เอาท์พุทเป็นขีดกลางของไม้จัตวา

2.2.4 การแยกบรรทัดตัวอักษร



รูปที่ 2.22 แสดงการแยกบรรทัดของลายมือเขียน

กำหนดให้จุดเริ่มต้นของเส้นบรรทัดแรก อยู่ที่ตำแหน่ง (0, y1st_Line) และจุดเริ่มต้นของเส้นบรรทัดที่สอง อยู่ที่ตำแหน่ง (0, y2nd_Line)

A.ค่า yMax คือ ค่า y สูงสุดจากเส้นลายมือเขียน

B. ค่า yMin คือ ค่า y ต่ำสุดจากเส้นลายมือเขียน

C. จะได้ค่า yMid ซึ่งเป็น y ที่จุดกึ่งกลางเส้นลายมือเขียนมีค่า

yMid = (yMax + yMin) / 2

D. นำค่า yMid ที่ได้มาเปรียบเทียบเพื่อหาตำแหน่งของตัวอักษรดังนี้

a) หาก yMid < y1st_Line อักษรอยู่เหนือบรรทัด

b) หาก yMid > y2nd_Line อักษรอยู่ใต้บรรทัด

c) y1st_Line < = yMid <= y2nd_Line อักษรอยู่ในบรรทัด

บทที่ 3
เทคโนโลยีและเครื่องมือในการพัฒนา

3.1 เทคโนโลยีดอทเน็ต

ดอทเน็ดเฟรมเวิร์ก คือโครงร่างการพัฒนาโปรแกรมคอมพิวเตอร์ ที่ถูกออกแบบมาเพื่อให้อำนวยความสะดวกในการพัฒนาโปรแกรมสมัยใหม่ที่ใช้งานในระบบเครือข่ายโดยมีเป้าหมายการทำงานหลัก 3 ข้อ ได้แก่

1. การพัฒนาโปรแกรมในรูปแบบของเวบเซอร์วิส ซึ่งจะเป็นหลักของโปรแกรมต่างๆ ที่ใช้งานบนอินเตอร์เน็ตเวบเซอร์วิสจะช่วยให้การติดต่อสื่อสารระหว่างแอพลิเคชั่นบนอินเตอร์เน็ตนั้นง่ายขึ้น และเป็นระบบมากยิ่งขึ้น
2. เวบเซอร์วิสขั้นพื้นฐานเช่น การตรวจสอบชื่อผู้ใช้ที่ล็อกอินเข้าสู่ระบบ มีการพัฒนาให้เป็นมาตรฐาน และสามารถนำไปใช้ได้ทั่วไปบนอินเตอร์เน็ต
3. เครื่องคอมพิวเตอร์และอุปกรณ์พกพาต่างๆ ที่ต่อเชื่อมกับอินเตอร์เน็ตได้ เช่น พีดีเอและโทรศัพท์มือถือ จะมีบทบาทและประโยชน์มากขึ้นเมื่อสามารถติดต่อใช้งานโปรแกรมต่างๆ บนอินเตอร์เน็ตได้

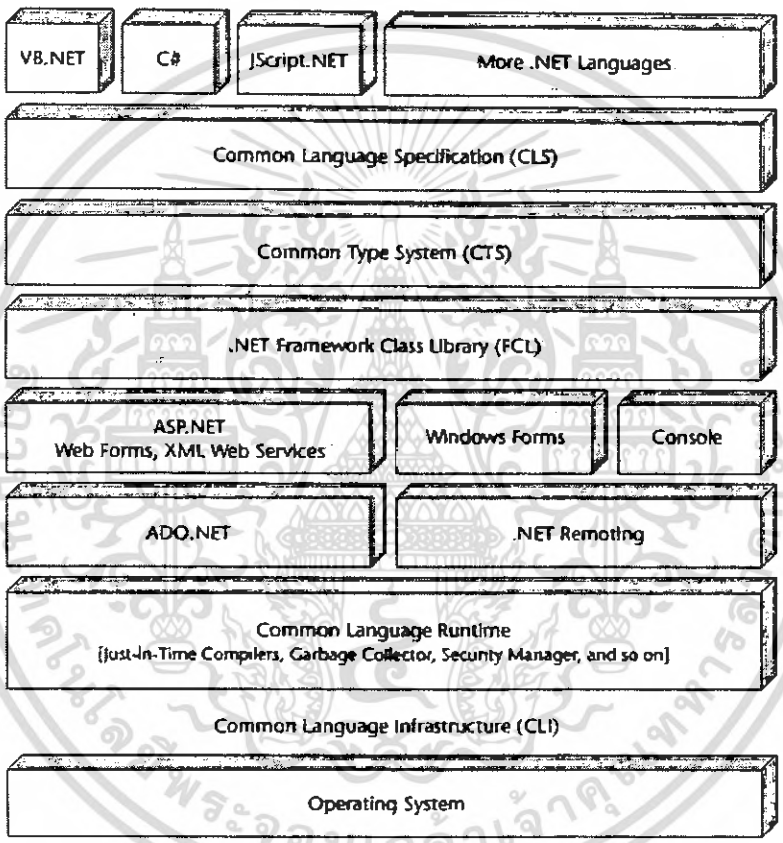
การที่จะทำให้ระบบหลายๆ ระบบทำงานต่อเชื่อมกันได้อย่างราบรื่นนั้น ไมโครซอร์ฟได้พัฒนารูปแบบการพัฒนาระบบคอมพิวเตอร์ขึ้นมา เรียกว่า ดอทเน็ดเฟรมเวิร์ก ซึ่งมีผู้คิดค้นพัฒนาจากหลายหน่วยงาน ยกตัวอย่างเช่น ซันไมโครซิสเต็มส์ บริษัทไอบีเอ็ม เป็นต้น แต่ว่าไมโครซอร์ฟได้นำแนวคิดเหล่านั้นมาออกแบบให้อยู่ในรูปแบบที่สามารถต่อเชื่อมกันได้ง่ายขึ้น และเป็นระบบมากขึ้น โดยดอทเน็ดประกอบไปด้วยส่วนประกอบต่างๆ หลายส่วนด้วยกัน และส่วนประกอบเหล่านี้ถูกออกแบบมาเพื่อให้ทำงานได้เข้ากันได้ดีขึ้น

3.1.1 ส่วนประกอบของดอทเน็ดเฟรมเวิร์ก

ส่วนประกอบหลักของ ดอทเน็ดเฟรมเวิร์กแบ่งเป็นชั้นๆ ได้แก่

1. Common Language Runtime (CLR) ถือเป็นรากฐานของแพลตฟอร์มดอทเน็ต โดยเป็นส่วนพื้นฐานที่ติดต่อกับระบบปฏิบัติการวินโดวส์ ทำหน้าที่เป็น run-time environment ให้กับโปรแกรมที่เขียนขึ้นสำหรับใช้บนดอทเน็ต โดยเป็น execution engine ในการประมวลผลและจัดการโปรแกรมที่คอมไพล์แล้ว ให้ทำงานได้บนวินโดวส์ CLR มีส่วนของคอมไพเลอร์ทั้งที่เป็นแบบปกติ คือคอมไพล์ก่อนที่จะนำโปรแกรมไปใช้ และแบบ Just-In-Time คือคอมไพล์เมื่อจะใช้โปรแกรมนั้นๆ หลังจากนั้นถ้าต้องการนำส่วนอื่นมาใช้งานอีกก็จะแปลงเพิ่มเฉพาะในส่วนนั้น ซึ่งช่วยให้โปรแกรมทำงานได้เร็วขึ้น เนื่องจากไม่ต้องรอให้แปลงเสร็จ

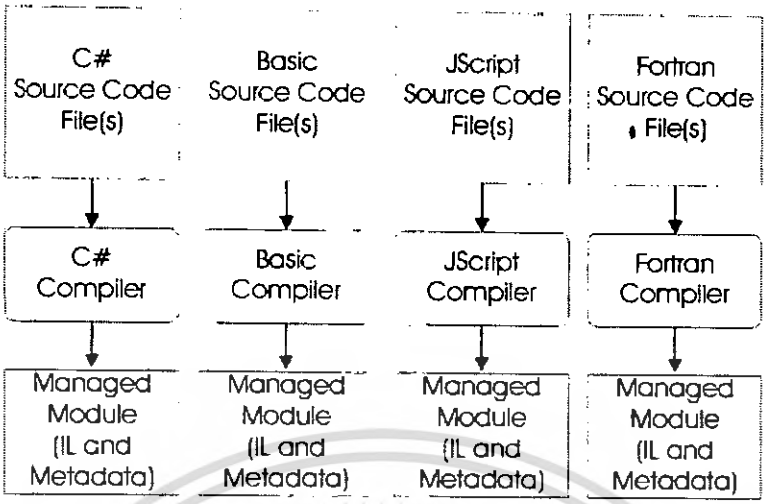
ทั้งหมดก่อนจึงจะทำงานได้ มีส่วนของการจัดการหน่วยความจำ ที่เอาไว้สำหรับจัดสรรหน่วยความจำของเครื่องให้กับโปรแกรม รวมไปถึงการจัดสรรหน่วยความจำให้กับโปรแกรมต่างๆ และคืนหน่วยความจำที่ไม่ถูกใช้งานแล้วให้กับระบบ (Garbage Collection) การจัดการกับข้อผิดพลาดต่างๆ (Exception Handling) รวมถึงดูแลเรื่องความปลอดภัย (security management) ด้วย ส่วนของ Common Type Systems (CTS) ทำให้ภาษาต่าง ๆ ที่เขียนขึ้นบน .Net สามารถทำงานร่วมกันได้ เพราะขนาด และรูปแบบของข้อมูลที่เก็บไว้นั้นเป็นรูปแบบเดียวกัน



รูปที่ 3.1 แสดงสถาปัตยกรรมของดอทเน็ตเฟรมเวิร์ก

2. Base Classes เป็นคลาสไลบรารี่พื้นฐาน ที่โปรแกรมต่างๆ ไม่ว่าจะเขียนด้วยภาษาใดบนดอทเน็ตก็สามารถใช้ร่วมกันได้ เช่น การติดต่อระบบฐานข้อมูล การติดต่อกับ ระบบไฟล์ของเซิร์ฟเวอร์ เป็นต้น

3. Programming Languages เป็นเซ็ตของภาษาคอมพิวเตอร์ ที่ถูกออกแบบมาเพื่อการเขียนโปรแกรมบนดอทเน็ตเฟรมเวิร์ก โดยมีภาษาหลักๆ ได้แก่ VB.Net ซึ่งเป็นตัวที่พัฒนาต่อมาจาก VB C# ซึ่งเป็นภาษาใหม่ที่มีรูปแบบการเขียนใกล้เคียงกับ Java และ C++ Visual C++ และ JScript.Net ส่วนภาษาอื่นๆ นั้น มีบริษัท หรือหน่วยงานอื่นๆ เป็นผู้พัฒนาขึ้น ซึ่งมีอีกจำนวนหลายภาษา



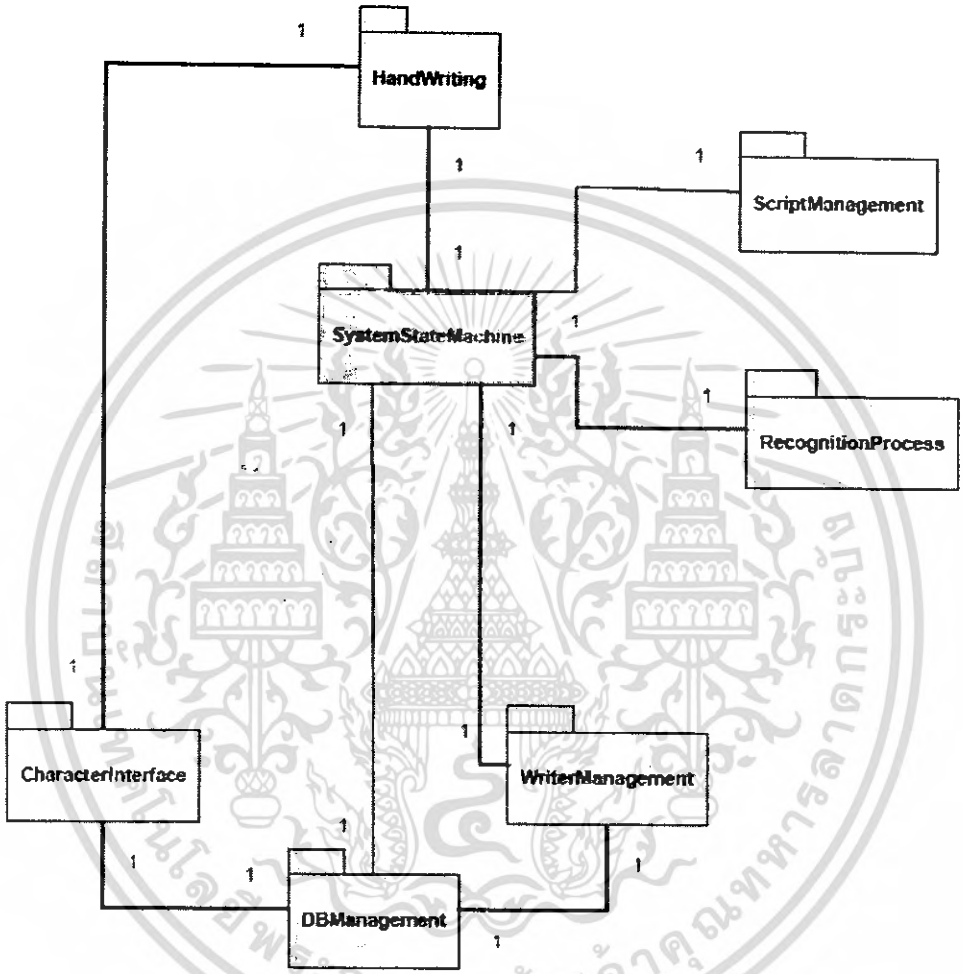
รูปที่ 3.2 แสดงการแปลจากโค้ดให้อยู่ในรูปของ IL

สำหรับบนดอทเน็ตเฟรมเวิร์กนั้นไม่ว่าจะเขียนโปรแกรมด้วยภาษาใดก็ตามคอมไพล์เลอร์ใน CLR ก็จะคอมไพล์โปรแกรมนั้นให้อยู่ในรูปของ Intermediate Language (IL) ซึ่งจะถูกนำไปแปลเป็นภาษาเครื่อง (Native Code) อีกทีเมื่อตอนที่นำไปใช้

บทที่ 4

การออกแบบและพัฒนา

4.1 สถาปัตยกรรมซอฟแวร์ของระบบอ่านลายมือเขียนภาษาไทย



รูปที่ 4.1 แสดงความสัมพันธ์ระหว่างnamespace ต่างๆในระบบ

4.1.1 ระบบอ่านลายมือเขียนภาษาไทย ประกอบด้วยโมดูลการทำงาน 4 ส่วนหลักดังนี้

1. ส่วนติดต่อกับผู้ใช้งานระบบ (user interface)
2. ส่วนโปรแกรมหลัก (business process)
3. ส่วนติดต่อกับฐานข้อมูล (storage)
4. ส่วนจัดเก็บข้อมูลลงไฟล์เอกสาร (storage)

4.1.2 Namespace ในระบบ

-Handwriting: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่จัดการติดต่อกับผู้ใช้ระบบ ผ่านการรับข้อมูลจากผู้ใช้ระบบ และแสดงผลข้อมูลให้ผู้ใช้ระบบ ออกทางหน้าจอ

-ScriptManagement: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่จัดการกับอักษรต่างๆที่ต้องการให้ผู้ใช้ระบบเขียนตาม (อักษรสคริปต์)

-RecognotionProcess: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ โดยจัดการติดต่อกับระบบของโครงงานเดิมเพื่อแปลงตัวอักษรลายมือเขียนเป็นอักษรโดด

-SystemStateMachine: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ซึ่งจัดการตัวอักษรที่ไม่เป็นอักษรโดด และเป็นส่วนหลักของระบบในการควบคุมการประมวลผลโปรแกรมคล้าย main function

-WriterManagement: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่จัดการข้อมูลของผู้ใช้ระบบ โดยหากผู้ใช้ระบบเป็นผู้ใช้ใหม่ระบบจะเก็บข้อมูลของผู้ใช้ใหม่นั้นเข้าฐานข้อมูล

-CharacterInterface: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่เป็นสื่อกลางในการแปลงรหัสอักษรระหว่างอักษรจากฐานข้อมูลและอักษรที่ใช้แสดงผลทางหน้าจอ (แปลงอักษรระหว่างรหัส ASCII และ รหัส UNICODE)

-DBManagement: รวบรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ติดต่อกับฐานข้อมูล

4.1.3 Source Code ของระบบ ประกอบด้วยไฟล์ต่างๆดังนี้

Header File	Source File	Namespace	Class
Form1.h	Form1.cpp	Handwriting	1. Form1
MyControl.h	MyControl.cpp	Handwriting	1. MyControl
MyControlLine.h	MyControlLine.cpp	Handwriting	1. LineAbstraction, 2. DistanceBetween-WrittenLine
MyUI.h	MyUI.cpp	Handwriting	1. MyUI
PrototypeControl.h	PrototypeControl.cpp	Handwriting	1. PrototypeControl
PrototypeDisplay.h	PrototypeDisplay.cpp	Handwriting	1. PrototypeDisplay
CharacterDialogBox.h	CharacterDialogBox.cpp	Handwriting	1. CharacterDialogBox
DbUI.h	DbUI.cpp	Handwriting	1. DbUI
DbView.h	DbView.cpp	Handwriting	1. DbView

Header File	Source File	Namespace	Class
NewWriterDialog.h	NewWriterDialog.cpp	Handwriting	1. NewWriterDialog
NewWriterUI.h	NewWriterUI.cpp	Handwriting	1. NewWriterUI
WriterUI.h	WriterUI.cpp	Handwriting	1. WriterUI
DisplayWriterIdDialog.h	DisplayWriterIdDialog.cpp	Handwriting	1. DisplayWriterId-Dialog
SystemStateMachine.h	SystemStateMachine.cpp	SystemStateMachine	1. ProcessingMachine
SystemStateContent.h	SystemStateContent.cpp	SystemStateMachine	1. RemovedStated-Message, 2. RecogItem, 3. RecogList, 4. StateContent, 5. StateStack, 6. SystemState-StackInterface, 7. SystemStateStack,
SystemStateTable.h	SystemStateTable.cpp	SystemStateMachine	ไม่มี
ScriptProcess.h	ScriptProcess.cpp	ScriptManagement	1. ScriptProcess
Rccognition.h	Recognition.cpp	RecognotionProcess	1. PrototypeCharacter, 2. PrototypeList, 3. Recognition-Interface, 4. RecognitionProcess
CharacterInterface.h	CharacterInterface.cpp	CharacterInterface	1. CharacterInterface
ConstantIdentifier.h	ConstantIdentifier.cpp	ConstantIdentifier	ไม่มี
DBConnection.h	DBConnection.cpp	DBManagement	1. DBConnection
DBContent.h	DBContent.cpp	DBManagement	1. Writer, 2. CharacterDB
StringTransformation.h	StringTransformation.cpp	StringTransformation	ไม่มี
WriterManagement.h	WriterManagement.cpp	WriterManagement	1. WriterInterface, 2. WriterProfile

Header File	Source File	Namespace	Class
CurveList.h	CurveList.cpp	ไม่มี Namespace	1. CurveInfo, 2. CurveList
CurveListBuilder.h	CurveListBuilder.cpp	ไม่มี Namespace	1. CurveListBuilder
MousePoint.h	MousePoint.cpp	ไม่มี Namespace	1. Point, 2. PointList, 3. Vector

4.2 Class และ Interface ของคลาส

4.2.1. Handwriting: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดการเกี่ยวกับการติดต่อกับผู้ใช้งานระบบ การรับข้อมูลจากผู้ใช้งานระบบ และแสดงผลข้อมูลออกทางหน้าจอ

4.2.1. 1. Class Form เป็นคลาสจัดการกับ user interface สร้าง instance ของคลาสต่างๆ และสร้างความสัมพันธ์เชื่อมโยงระหว่าง instance เหล่านั้น เมื่อปิดโปรแกรม instance ต่างๆ ที่ถูกสร้างขึ้นจะถูกกำจัดโดย default destructor ของคลาสนี้

```
public ref class Form1 : pub_ic System::Windows::Forms::Form
{
private:
        MyUI^ myUI ;
        MyControl^ myControl;

        PrototypeControl^ protoControl;

        ProcessManageMent* processMan;
        SystemStateMachine::SystemStateStackInterface*
stackMan;
        RecognitionProcess::RecognitionInterface* recogMan;
        WriterManagement::WriterInterface* writerIn;
        LineAbstraction* disLine;
        SystemStateMachine::ProcessingMachine* coreMan;
        ScriptManagement::ScriptProcess* scriptMan;
        CurveListBuilder* curveLst;
    private:
        void characterDBClick(System::Object^  sender,
        System::EventArgs^  e);
        void writerDBClick(System::Object^  sender,
        System::EventArgs^  e);
        void newWriterClick(System::Object^  sender,
        System::EventArgs^  e);
```

4.2.1.2. Class WritingPadManagement (class LineAbstraction) เป็น Abstract Class จัดการกับเส้นบรรทัดใน Writing Pad โดย class ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class LineAbstraction
{
public:
  virtual void increaseLineIndex() = 0;
  virtual std::pair<int, int> getLinePair()= 0;
  virtual std::pair<int, int> getLinePair(unsigned index)= 0;
  virtual void setMyControl(HandWriting::MyControl^ mycon) = 0;
  virtual void createNewLine() = 0;
  virtual unsigned getLineIndex() const = 0;
  virtual unsigned getLineCount() = 0;
  virtual int getSpaceLine() const = 0;
  virtual void setSpaceLine(unsigned mul)= 0;
  virtual void clearWritingPad() = 0;
};
```

และ คลาสที่สืบทอด (Inherit) มาจากAbstract Class ข้างต้นประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class DistanceBetweenWrittenLine :public LineAbstraction
{
typedef gcroot<HandWriting::MyControl^> MyControlItem;
typedef std::vector<MyControlItem> MyControlList;
typedef std::vector<std::pair<int, int>> LineControlSet;

private:

  unsigned lineConIndex;
  LineControlSet lineCon;
  MyControlItem myCon;

public:

  DistanceBetweenWrittenLine(HandWriting::MyControl^ mycon):
        lineConIndex(0), lineCon(), myCon(mycon)
  {
        int controlHeight = myCon->getMyControlHeight();
        int countRow = 0;
        int space = ConstantIdentifier::SPACE_CHAR;
        while (controlHeight - space > 0){
              controlHeight -= space;
              ++countRow;
        }
        int yUp = 0;
        int yUnder = 0;
        for (int i = 1; i < countRow; i *= 2){
              yUp += ConstantIdentifier::SPACE_CHAR;
              yUnder = yUp + ConstantIdentifier::SPACE_CHAR;
              lineCon.push_back(std::make_pair(yUp, yUnder));
              yUp = yUnder;
        }
  }
  std::pair<int, int> getLinePair();
  std::pair<int, int> getLinePair(unsigned index);
  void createNewLine();
  void setMyControl(HandWriting::MyControl^ mycon);
```

```
unsigned getLineIndex() const;
unsigned getLineCount();
void increaseLineIndex();
int getSpaceLine() const;
void resetSpaceChar(int mul);
void setSpaceLine(unsigned mul);
void clearWritingPad();

};
```

```
virtual void increaseLineIndex() = 0;
```
เพิ่มจำนวนบรรทัด

```
virtual std::pair<int, int> getLinePair()= 0;
```
ให้ค่าแกน y ของเส้นบรรทัดคู่ที่ 1 โดยให้ค่าเป็น y_1, y_2

```
virtual std::pair<int, int> getLinePair(unsigned
index)= 0;
```
ให้ค่าแกน y ของเส้นบรรทัดคู่ใดๆตามที่กำหนดใน index โดยให้ค่าเป็น y_1, y_2

```
virtual void setMyControl(HandWriting::MyControl^ )
= 0;
```
สร้าง Association ไปยัง MyControl Instance ซึ่งเป็น object จัดการ Writing Pad

```
virtual void createNewLine() = 0;
```
สร้างเส้นบรรทัดคู่ใหม่

```
virtual unsigned getLineCount() = 0;
```
ให้ค่าจำนวนเส้นบรรทัดทั้งหมดที่มี

```
virtual int getSpaceLine() const = 0;
```
ให้ค่าความกว้างระหว่างเส้นบรรทัด

```
virtual void setSpaceLine(unsigned mul)= 0;
```
กำหนดค่าความกว้างระหว่างเส้นบรรทัด

```
virtual void clearWritingPad() = 0;
```
ลบลายมือเขียนบน Writing Pad

4.2.1.3. Class PrototypeControl คลาสจัดการกับ user interface เพื่อแสดงอักษรตัวอย่าง (prototype character) ที่มีลักษณะใกล้เคียงกับลายมือเขียนที่รับเข้าไปเป็นอินพุทมากที่สุด ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
public ref class PrototypeControl : public
System::Windows::Forms::UserControl
{
private:
   DisplayPrototype* disPtt;
   const MyPoint::PointList* prototypeLst;
public:
   void setDisplayPointList(const MyPoint::PointList* pntLst);
   void deleteDisplayPrototype();
   System::ComponentModel::Container ^components;
};
```

void setDisplayPointList
(const MyPoint::PointList);

กำหนดค่าให้โปรโตไทป์ลายมือเขียนที่ต้องการแสดงที่หน้าจอ Prototype Window

void deleteDisplayPrototype();

ลบโปรโตไทป์ลายมือเขียนที่แสดงที่หน้าจอ Prototype Window

4.2.1.4. Class WritingPad คลาสจัดการกับ user interface เพื่อแสดงอินพุทลายมือเขียน ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
public ref class MyControl : public
                        System::Windows::Forms::UserControl
{
private:

System::ComponentModel::Container ^components;
  bool leftClicked;
  bool newLine;
  MyControlImpl* myImpl_;
  const MyPoint::PointList* writtenPntLst;
  CurveListBuilder* curveLB;
  LineAbstraction* disLine;
  SystemStateMachine::ProcessingMachine* processMan;
SystemStateMachine::SystemStateStackInterface* stateStack;

  System::Windows::Forms::TextBox^ recogTextBox;
private:
  void adjustWritingArea(int x);
  void adjustWritingAreaVertical(int y);
  void myScroll(System::Object^ sender,
              System::Windows::Forms::ScrollEventArgs^ e);
  void myScrollValueChanged(System::Object^ sender,
                            System::EventArgs^ e);
void vScrollValueChanged(System::Object^  sender,
System::EventArgs^  e);
public:
  void setProcessingMachine
(SystemStateMachine::ProcessingMachine* processman);
  void setDistanceBetweenWrittenLine(LineAbstraction* line);
  void setCurveList(CurveListBuilder* curveLst);
  void setSystemStateStackInterface
(SystemStateMachine::SystemStateStackInterface* stack);
  void clearWritingPad();
  int getMyControlHeight();
```

};

void setProcessingMachine
(SystemStateMachine::ProcessingMachine);*

สร้าง Association ไปยัง ProcessingMachine Instance ซึ่งเป็นobject ที่ควบคุมการประมวลผลโปรแกรมโดยรวม

void setDistanceBetweenWrittenLine
(LineAbstraction);*

สร้าง Association ไปยัง LineAbstraction Instance ซึ่งเป็น object จัดการการตีเส้นบรรทัดบน Writing Pad

void setCurveList(CurveListBuilder);*

สร้าง Association ไปยัง CurveListBuilder Instance ซึ่งเป็น object จัดการและเก็บข้อมูลลายมือเขียนบน Writing Pad

void setSystemStateStackInterface
(SystemStateMachine::SystemStateStackInterface);*

สร้าง Association ไปยัง SystemStateStack Instance ซึ่งเป็น object จัดการและเก็บข้อมูลตัวอักษรระหว่างการประมวลผล

void clearWritingPad();

ลบลายมือเขียนบน Writing Pad

4.2.1.5. Class GUIControl คลาสจัดการ user interface ทั้งหมดของหน้าต่างหลักของโปรแกรม ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
public ref class MyUI : public
System::Windows::Forms::UserControl
{
public:
      property WriterManagement::WriterInterface* WriterIn {
            WriterManagement::WriterInterface* get()
            {
                  return writerIn;
            }

            void set(WriterManagement::WriterInterface* wrt)
            {
                  writerIn = wrt;
            }
      }
private:
      PrototypeManagement* prototypeMan;
      UnicodeTransform::UnicodeToWChartMap*  unicodeWchMap;
      std::string* recogString;
      CharPrototypeList* charVec;
      CharNumPrototypeList* charNumVec;
```

```
        RecognitionProcess::PrototypeList* prototypeList;

        SystemStateMachine::ProcessingMachine* coreMan;
        int mode;
        WriterManagement::WriterInterface* writerIn;
        LineAbstraction* disLine;
        PrototypeControl^ protoControl;
        Array^ disPrototypeLst;
public:
        int getScriptMode();
        System::Windows::Forms::TableLayoutPanel^
getMyControlPanel();
        System::Windows::Forms::TableLayoutPanel^
getPrototypeControlPanel();
        void setScriptTextBoxString(const std::string& str);
        void createDisPrototypeItem();
        char setDisplayTextBoxPrototype();
        void setPrototypeControl(PrototypeControl^ pttCon);
        void setRecogTextBoxString(std::string recText);
        void setProcessingMachine
(SystemStateMachine::ProcessingMachine* coreman);
        void
setWriterInterface(WriterManagement::WriterInterface*
writer);
        void setPrototypeList(RecognitionProcess::PrototypeList
pptLst);
        char setPointListPrototypeSelected(unsigned
pntLstDisplay);
        void setDistanceBetweenWrittenLine(LineAbstraction*
line);

};
```

int getScriptMode();

แสดงโหมดการประมวลผลที่ทำงานอยู่ในปัจจุบัน

void setScriptTextBoxString(const std::string&);

กำหนดข้อความที่จะแสดงออกที่ Script Window

void createDisPrototypeItem();

กำหนดตัวอักษรโปรโทไทป์ 5 ตัวที่จะแสดงที่ Text Box

char setDisplayTextBoxPrototype();

แสดงตัวอักษรโปรโทไทป์ทั้งหมด 5 ตัวอักษรที่ Text Box และให้ตัวอักษรโปรโทไทป์ที่มีลักษณะเหมือนอักษรลายมือเขียนมากที่สุดออกมา

void setPrototypeControl(PrototypeControl^);

สร้าง Associationไปยัง PrototypeControl Instance ซึ่งเป็น object แสดงตัวอักษรโปรโทไทป์

void setRecogTextBoxString(std::string);

กำหนดข้อความที่จะแสดงออกที่ Recognition Window

```
void setProcessingMachine
(SystemStateMachine::ProcessingMachine*);
```

สร้าง Association ไปยัง SystemStateStack Instance ซึ่งเป็น object จัดการและเก็บข้อมูลตัวอักษรระหว่างการประมวลผล

```
void setWriterInterface
(WriterManagement::WriterInterface*);
```

สร้าง Association ไปยัง WriterInterface Instance ซึ่งเป็น object จัดการผู้ใช้ระบบ

```
void setPrototypeList
(RecognitionProcess::PrototypeList);
```

กำหนดค่า data member ชื่อ `&prototypeMan->prototypeList` ให้มีค่าเท่ากับตัวอักษรโปรโทไทป์ทั้งหมด 5 ตัวที่ได้จากการรู้จำ

```
char setPointListPrototypeSelected
(unsigned index);
```

เลือกตัวอักษรโปรโทไทป์ที่จะแสดงที่ Prototype Window โดย index คือค่าที่ใช้อ้างตำแหน่งในการเข้าถึงข้อมูลใน `&prototypeMan->prototypeList`

```
void setDistanceBetweenWrittenLine
(LineAbstraction* line);
```

สร้าง Association ไปยัง LineAbstraction Instance ซึ่งเป็น object จัดการการตีเส้นบรรทัดบน Writing Pad

```
property WriterManagement::WriterInterface*
WriterIn {
      WriterManagement::WriterInterface* get()
      {
            return writerIn;
      }

      void set(WriterManagement::WriterInterface*
      wrt)
      {
                  writerIn = wrt;
      }
}
```

เป็นคุณสมบัติของ Class ซึ่งจัดการกับข้อมูลของผู้ใช้โปรแกรมขณะนั้น

4.2.1.6. Class CharacterDialogBox จัดการกับ user interface ของ Dialog box แสดงข้อมูลตัวอักษรและข้อมูลผู้ใช้จากฐานข้อมูล โดยคลาสประกอบด้วยฟังก์ชันต่างๆดังนี้

```
public ref class CharacterDialogBox : public
```

```
                        System::Windows::Forms::Form
{
private:
      DbUI^ dbUI;
      WriterUI^ wrtUI;
public:
      CharacterDialogBox(void)
      {
            InitializeComponent();
            dbUI = gcnew DbUI;
            dbUI->Dock =
                  System::Windows::Forms::DockStyle::Fill;
            this->Controls->Add(dbUI);
      }

      CharacterDialogBox(const std::string& ui )
      {
            InitializeComponent();
            if (ui == ConstantIdentifier::CHARACTER){
                  dbUI = gcnew DbUI;
                  dbUI->Dock =
                  System::Windows::Forms::DockStyle::Fill;

                  this->Controls->Add(dbUI);
            }
            else if (ui == ConstantIdentifier::WRITER){
                  wrtUI = gcnew WriterUI;
                  wrtUI->Dock =
                  System::Windows::Forms::DockStyle::Fill;

                  this->Controls->Add(wrtUI);
            }
      }
};
```

4.2.1.7. Class DialogBoxView (class DbView) จัดการ user interface ในการแสดงผลลายมือเขียนที่โหลดมาจากฐานข้อมูล ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
public ref class DbView : publi
System::Windows::Forms::UserControl
{
public:
      DbView(void): disPntLst(new DisplayPointList)
      {
            InitializeComponent();
            pointLst = &disPntLst->pointLst;
      }

private:
      DisplayPointList* disPntLst;
      MyPoint::PointList* pointLst;
      System::ComponentModel::Container ^components;
public:
      void setDisplayPointList(const std::string& pntStr);
};
```

void setDisplayPointList(const std::string&);

แสดงตัวอักษรลายมือเขียนในฐานข้อมูลทั้งหมด

4.2.1.8. Class DialogBoxUI จัดการฟังก์ชันการทำงานของ Control ต่างๆใน Dialog box ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
public ref class DbUI : public
System::Windows::Forms::UserControl
{
private:
      CharacterSetStruct* charSet;
      CharacterInterface::CharacterObjectSet* charInSet;
      UnicodeTransform::UnicodeToWChartMap*  unicodeWchMap;
      unsigned index;
      DbView^ dbView;
private:
      void exitButtonClick(System::Object^  sender,
      System::EventArgs^  e);
      void updateButtonClick(System::Object^  sender,
      System::EventArgs^  e);
      void deleteButtonClick(System::Object^  sender,
      System::EventArgs^  e);
      void loadButtonClick(System::Object^  sender,
      System::EventArgs^  e);
      void nextButtonClick(System::Object^  sender,
      System::EventArgs^  e);
      void setTextBoxOutput(unsigned curIndex );
      void  setCharacterObjectValue
      (CharacterInterface::CharacterInterface& charIn);
      void setDisplayView();
      void setTextBoxOutNext();
      void setTextBoxOutBack();
      void backButtonClick(System::Object^  sender,
      System::EventArgs^  e);
};
```

void setTextBoxOutput(unsigned curIndex);

กำหนดข้อมูลตัวอักษรที่จะแสดงที่ Text Box

void setCharacterObjectValue
(CharacterInterface::CharacterInterface& charIn);

กำหนดข้อมูลตัวอักษรใหม่ที่จะแก้ไขในฐานข้อมูล

void setDisplayView();

แสดงตัวอักษรลายมือเขียนในฐานข้อมูลทั้งหมด

void setTextBoxOutNext();

กำหนดข้อมูลตัวอักษรที่จะแสดงที่ Text Box ตัวถัดไป

void setTextBoxOutBack();

กำหนดข้อมูลตัวอักษรที่จะแสดงที่ Text Box ตัวก่อนหน้านี้

4.2.2. ScriptManagement: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดกอักษรต่างๆที่ต้องการให้ผู้ใช้งานระบบเขียนตามตัวพิมพ์

4.2.2.1. Class ScriptProcess คลาสจัดการกับตัวอักษรที่ต้องการให้ผู้ใช้งานระบบเขียนตาม ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class ScriptProcess
{
public:
      ScriptProcess();
      const std::string* beginWrdLst();
      const std::string* endWrdLst();
      WOrdLen getCurrCharPosition() const;
      ScriptWordListSize getCurrWordPosition() const;
      char nextChar();
      std::string getScriptString();
private:
      ScriptWordList scriptWordLst;
      ScriptWordListSize wordNum;
      ScriptWordListSize currWord;
      WOrdLen currCh;

};
```

```
char nextChar();
```

ไปยังตัวอักษรสคริปต์ตัวถัดไป

```
std::string getScriptString();
```

ให้สคริปต์กลับออกมา

4.2.3. RecognotionProcess: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดการติดต่อกับระบบของโครงงานเดิมเพื่อแปลงตัวอักษรลายมือเขียนเป็นอักษร

4.2.3.1. Class RecognitionInterface เป็น abstract class ซึ่งประกอบด้วย method ต่างๆ เพื่อติดต่อกับโครงงานเดิมในการแปลงอินพุทลายมือเขียนเป็นอักษรตัวพิมพ์ คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class RecognitionInterface
{
public:
      virtual void recognitionMain(const MyPoint::PointList&
            pnt,PrototypeList& prototypeList, int yMinLevel,
            int yMaxLevel) = 0;
};
```

```
virtual void recognitionMain
(const MyPoint::PointList& pnt,
PrototypeList& prototypeList,
int yMinLevel, int yMaxLevel) = 0;
```

จัดการการ Recognition โดยให้ผลกลับมาเป็นลำดับของอักษรโปรโทไทป์ที่ใกล้เคียงกับอักษรลายมือเขียนมากที่สุด 5 อันดับแรก

4.2.3.2. Class RecognitionProcess สืบทอด (inherit) มาจากClass RecognitionInterface คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class RecognitionProcess : public RecognitionInterface
{
public:
      RecognitionProcess();
      void recognitionMain(const MyPoint::PointList& pnt,
                        PrototypeList& prototypeList,
                        int yMinLevel, int yMaxLevel) ;
      char createSelectedPrototypeLst
                        (ThaiRecogInterface* thaiRec,
                        PrototypeList& prototypeList,
                        std::string pntLstStr);
      char createNewPrototypeLst(const MyPoint::PointList&
                              wrt_pntLst , PrototypeList&
                              prototypeList, char c);
      virtual ~RecognitionProcess();
private:
      ThaiRecogInterface* inLineRecog;
      ThaiRecogInterface* upLineRecog;
      ThaiRecogInterface* underLineRecog;
};
```

```
char createSelectedPrototypeLst
    (ThaiRecogInterface* thaiRec,
    PrototypeList& prototypeList, std::string
pntLstStr);
```

สร้าง prototypeList ซึ่งเป็นลำดับของอักษรโปรโทไทป์ 5 อันดับแรกที่มีลักษณะใกล้เคียงกับอินพุทลายมือเขียนมากที่สุด

```
char createNewPrototypeLst(const
MyPoint::PointList&
          wrt_pntLst , PrototypeList&
    prototypeList, char c);
```

สร้าง prototypeList ของอักษรที่มีลักษณะใกล้เคียงกับอินพุทลายมือเขียนมากที่สุด

4.2.3.3. Class PrototypeCharater เก็บข้อมูลอักษรโปรโทไทป์ที่ ได้จากการรู้จำ (Recognition) คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class PrototypeCharacter
{
public:
      PrototypeCharacter():pc_score(0), pc_char(0), p
                              c_charNum(0), pc_pntLst(){}
      PrototypeCharacter(const PrototypePoint* begin, const
                  PrototypePoint* end,char ch, int num,
                  float score);
      PrototypeCharacter(const MyPoint::PointList& pnt,
                        char ch, int num);
```

```
        void setPrototypeCharPntLst(const PrototypePoint*
begin,
                                    const PrototypePoint*
                                end);
        void setPrototypeChar(char ch);
        void setPrototypeCharNum(int num);
        void setPrototypeScore(float score);
        float getPrototypeScore() const;
        const MyPoint::PointList& getPrototypeCharPntLst()
const;
        char getPrototypeChar() const;
        int getPrototypeCharNum() const;
        const PrototypeCharacter& getPrototypeCharacter();
        void setPrototypeCharacter(const MyPoint::PointList&
pnt, char ch, int num);

private:
        float pc_score;
        char pc_char;
        int  pc_charNum;
        MyPoint::PointList pc_pntLst;

};
```

4.2.4. SystemStateMachine: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดการกับตัวอักษรที่ไม่เป็นอักษรโดด เป็นส่วนหลักของระบบในการติดต่อกับเนมสเปช (namespace) อื่นๆคล้ายเป็นmain function ของระบบ

4.2.4.1. Class RecogItem เป็นโครงสร้างข้อมูล โดยเก็บข้อมูลตัวอักษรโดด ซึ่งเป็นเอาท์พุทจากการรู้จำ (Recognition) ในแต่ละครั้ง ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class RecogItem
{
private:
        int state;
        char recog;
        MyPoint::PointList pointList;
public:
        RecogItem(): state(ConstantIdentifier::s_init_state),
                     recog(0), pointList(){}
        RecogItem(int st, char rec = 0, const
MyPoint::PointList&
                     pntLst = MyPoint::PointList());
        void setState(int st);
        void setRecogChar(char rec);
        void setWrittenPntLst(const MyPoint::PointList&
pntLst);
        int getState() const;
        char getRecogChar() const;
        MyPoint::PointList getWrittenPntLst()const;
};
```

void setState(int st);

กำหนดState ID

void setRecogChar(char rec);

กำหนดอักษรโดด ที่ได้จากการ Recognition

void setWrittenPntLst
(const MyPoint::PointList&);
กำหนดลายมือเขียนของอักษรโดด

int getState() const;
ให้ State ID

char getRecogChar() const;
ให้ อักษรโดด ที่ได้จากการ Recognition

MyPoint::PointList getWrittenPntLst()const;
ให้อักษรโดด ที่ได้จากการ Recognition

4.2.4.2. Class RecogList เนื่องจากตัวอักษรบางตัวไม่เป็นอักษรโดด ซึ่งต้องผ่านการประมวลหลายครั้ง ทำให้ได้ instance ของ Class RecogItem มากกว่า 1 instance คลาสนี้จึงสร้างขึ้นเพื่อเก็บ instance เหล่านั้น

4.2.4.3 Class StateContent เนื่องจากโปรแกรมจัดการกับอักษรที่ไม่เป็นอักษรโดดโดยใช้ หลักการของ State Machine อักษรแต่ละตัวจึงถูกเก็บไว้ใน State คลาสนี้แสดงฟังก์ชันต่างๆในการจัดการกับแต่ละ State

```
class StateContent
{
public:
      StateContent():script(0), scriptLabel(0), recogRes(0),

      strokeCountConstant(ConstantIdentifier::IGNORE_STROKE),
            strokeCount(0), recogList(){}
      StateContent(const RecogItem& item);
      void addRecogItem(const RecogItem& item);
      void addRecogItem(int st, char rec = 0,
            const MyPoint::PointList& pntLst =
MyPoint::PointList());

      void setScriptChar(char ch);
      void setScriptLabel(char ch);
      void setRecogResult(char ch);
      void setStrokeCountConstant(unsigned stroke);
      void setStrokeCount(unsigned stroke);
      void increaseStrokeCount();

      char getRecogResult() const;
      char getScriptChar() const;
      char getScriptLabel() const;
      unsigned getStrokeCount() const;
      unsigned getStrokeCountConstatnt() const;
      int getState() const;
      MyPoint::PointList getPointList();
      MyPoint::PointList getPrevLastRecogItemPntLst();
      const RecogList& getRecogLst() const;
      RecogListSize getRecItemNumRemove();
```

```
        RecogItem getLastRecogItem();

        int deleteRecogItem(RecogList::iterator pos);
        bool isRecogListEmpty() const;
        RecogList::iterator  endRecogList();
        RecogListSize recogItemAmount();
        bool isRecogCharLikeScript();
private:
        char script;
        char scriptLabel;
        char recogRes;
        unsigned strokeCountConstant;
        unsigned strokeCount;
        RecogList recogList;
};
```

void addRecogItem(const RecogItem& item);

กำหนดข้อมูลอักษรที่ได้หลังจากการ Recognition ลง State

void addRecogItem(int st, char rec = 0,
const MyPoint::PointList& pntLst =
MyPoint::PointList());

กำหนดข้อมูลอักษรที่ได้หลังจากการ Recognition ลง State

void setScriptChar(char ch);

กำหนดตัวอักษรสคริปต์ของ State

void setScriptLabel(char ch);

กำหนดตัวอักษรสคริปต์ของ State

void setRecogResult(char ch);

กำหนดตัวอักษรเอาท์พุทของ State

void setStrokeCountConstant(unsigned stroke);

กำหนด Stroke การเขียนตัวอักษรของ State

void increaseStrokeCount()

เพิ่มจำนวน Stroke การเขียนตัวอักษรของ State

char getRecogResult() const;

ให้ตัวอักษรเอาท์พุทของ State

char getScriptChar() const;

ให้ตัวอักษรสคริปต์ของ State

char getScriptLabel() const;

ให้ตัวอักษรสคริปต์ของ State

unsigned getStrokeCount() const;

ให้ Stroke การเขียนตัวอักษรว่าเขียนไป Stroke ที่เท่าใดแล้ว

unsigned getStrokeCountConstatnt() const;

ให้ Stroke การเขียนตัวอักษรของ State

int getState() const;

ให้ State ID

MyPoint::PointList getPointList();

ให้ลายมือเขียนตัวอักษรโดด

bool isRecogListEmpty() const;

ตรวจสอบว่ามีข้อมูลอยู่ในState หรือไม่

4.2.4.4. Class SystemStackStateInterface เป็น abstract class ซึ่งประกอบด้วยฟังก์ชันต่างๆในการจัดการ Stack ซึ่งเก็บ State ต่างๆ ของตัวอักษรไว้ คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class SystemStateStackInterface
{
public:
      virtual void pushRecogModeState() = 0;
      virtual void pushScriptModeState(char ch_script) = 0;
      virtual void pushRecState(int state, char ch_reg,
            char ch_output, const MyPoint::PointList& pntLst)
= 0;
virtual void addRecogList(int state, char ch_reg = 0,
char ch_outPut = 0, const MyPoint::PointList& pntLst =
MyPoint::PointList()) = 0;
      virtual void addRecogList(int state, char ch_reg,
            const MyPoint::PointList& pntLst) = 0;

      virtual const StateContent& getPrevCurStateContent() =
0;
      virtual int getCurrentState() = 0;
      virtual char getCurrRecogChar() = 0;
      virtual char getCurrScriptChar() = 0;
      virtual char getCurrScriptLabel() = 0;
      virtual char getCurrCharFromRecLst() = 0;
      virtual unsigned getCurrStrokeCount() = 0;
      virtual unsigned getCurrStrokeCountConstant() = 0;
      virtual MyPoint::PointList getPrevCurrPointList() = 0;
//    virtual MyPoint::PointList getCurrPointList() = 0;
      virtual std::string getOutputRecogText() = 0;
      virtual int getStateNumber() = 0;

      virtual bool isStackEmpty() = 0;
      virtual bool isCurrRecogedListEmpty() = 0;
      virtual bool isFinishScriptState() = 0;
      virtual void increaseCurrStrokeCount() = 0;
      virtual void resetScriptChar(char ch) = 0;
      virtual void clearStack() = 0;
```

```
        virtual void setCurrScriptLabel(char ch) = 0;
        virtual void setWriterInterface
(WriterManagement::WriterInterface* writer) = 0;

        virtual void saveStateChangeRecogChar(char ch = 0) = 0;
        virtual void saveState() = 0;
        virtual void popStack() = 0;

};
```

virtual void pushRecogModeState() = 0;

เพิ่ม Recognition State เข้า State Stack

virtual void pushScriptModeState(char ch_script) = 0;

เพิ่ม Script State เข้า State Stack

virtual void pushRecState(int, char ,char, const MyPoint::PointList&) = 0;

เพิ่ม Recognition State เข้า State Stack

virtual void addRecogList (int state, char ch_reg = 0, char ch_outPut = 0, const MyPoint::PointList& pntLst = MyPoint::PointList()) = 0;

เพิ่มผลการ Recognition อักษรโดดเข้า State

virtual void addRecogList(int state, char ch_reg, const MyPoint::PointList& pntLst) = 0;

เพิ่มผลการ Recognition อักษรโดดเข้า State

virtual int getCurrentState() = 0;

ให้ State ปัจจุบันใน Stack

virtual char getCurrRecogChar() = 0;

ให้ อักษรโดดที่ Reconition ได้ล่าสุดใน Stack

virtual char getCurrScriptChar() = 0;

ให้ อักษรสคริปต์ล่าสุดใน Stack

virtual char getCurrScriptLabel() = 0;

ให้ อักษรสคริปต์ล่าสุดใน Stack

virtual unsigned getCurrStrokeCount() = 0;

ให้ Stroke การเขียนตัวอักษรว่าเขียนไป Stroke ที่เท่าใดแล้ว

virtual unsigned getCurrStrokeCountConstant() = 0;

ให้ Stroke การเขียนตัวอักษรของ State

```
virtual MyPoint::PointList getCurrPointList() = 0;
```
ให้ อักษรลายมือเขียนล่าสุดใน Stack

```
virtual bool isStackEmpty() = 0;
```
ตรวจสอบว่า เป็น Stack เปล่าหรือไม่

```
virtual bool isCurrRecogedListEmpty() = 0;
```
ตรวจสอบว่ามีข้อมูลผลการ Recognition อยู่ใน State หรือไม่

```
virtual std::string getOutputRecogText() = 0;
```
ให้ String เอาท์พุท

```
virtual void clearStack() = 0;
```
ลบ State ทั้งหมดใน Stack

```
virtual void increaseCurrStrokeCount() = 0;
```
เพิ่มค่า Stroke การเขียนตัวอักษร

```
virtual int getStateNumber() = 0;
```
ให้ State ID ได้ล่าสุดใน Stack

```
virtual void setCurrScriptLabel(char ch) = 0;
```
กำหนดตัวอักษรสคริปต์ของ State

```
virtual void setWriterInterface
(WriterManagement::WriterInterface* writer) = 0;
```
กำหนด ความสัมพันธ์ไปยัง WriterInterface ซึ่งจัดการข้อมูลของผู้เขียน

```
virtual void saveStateChangeRecogChar(char ch = 0)
 = 0;
```
แก้ไขข้อมูลในฐานข้อมูลและบันทึกการแก้ไขนั้น

```
virtual void saveState() = 0;
```
บันทึกข้อมูลลงฐานข้อมูล

```
virtual void popStack() = 0;
```
ลบ State ล่าสุดออกจาก Stack

4.2.4.5. Class SystemStackState สืบทอด (inherit) มาจาก Class SystemStackStateInterface คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class SystemStateStack :public SystemStateStackInterface
{
public:
      SystemStateStack();
      void pushRecogModeState();
      void pushScriptModeState(char ch_script);
      void pushRecState(int state, char ch_reg, char
ch_output,
            const MyPoint::PointList& pntLst);
      void reduceStateNumber();
      void addRecogList(int state, char ch_reg = 0,
char ch_outPut = 0, const MyPoint::PointList&
pntLst =  MyPoint::PointList());
      void addRecogList(int state, char ch_reg, const
MyPoint::PointList& pntLst);

      const StateContent& getPrevCurStateContent();
      int getCurrentState();
      char getCurrRecogChar();
      char getCurrScriptChar();
      char getCurrScriptLabel();
      char getCurrCharFromRecLst();
      unsigned getCurrStrokeCount();
      unsigned getCurrStrokeCountConstant();

      char getTrickCurScriptChar();

      MyPoint::PointList getCurrPointList();
      bool isStackEmpty();
      bool isCurrRecogedListEmpty();
      std::string getOutputRecogText();
      void resetStateValue(int st, char recogRes);
      void resetScriptChar(char ch);
      void clearStack();
      MyPoint::PointList getPrevCurrPointList();

      void increaseCurrStrokeCount();
      bool isFinishScriptState();
      int getStateNumber();
      void setCurrScriptLabel(char ch);
      void setCurrRecogLabel(char ch);

      void
setWriterInterface(WriterManagement::WriterInterface*
writer);
      void saveStateToDB(char recCH, const
MyPoint::PointList&
pntLst);
      void saveStateChangeRecogChar(char ch = 0);
      void saveState();
      void popStack();
private:
      unsigned charID;
      WriterManagement::WriterInterface* writerIn;
      std::string outputTextRecog;
      StateStack stateStack;
      ScriptStrokeCountMap scriptStrokeMap;
      ASCIIScriptCharMap asciiScpMap;
};
```

4.2.4.6. Class ProcessingMachine ดูแลและควบคุมขั้นตอนการประมวลโปรแกรมทั้งหมด ทำหน้าที่คล้าย main function ของโปรแกรม คลาสประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class ProcessingMachine
{
public:
      ProcessingMachine();
      void setRunningMode(int rmode);

      void setMyUI(HandWriting::MyUI^ myui);
      void setSystemStateStack(SystemStateStackInterface*
stack);
      void setCurveListBuilder(CurveListBuilder* curveLstBD);
      void setRecognitionInterface
(RecognitionProcess::RecognitionInterface* recog);
      void setScriptProcess
(ScriptManagement::ScriptProcess* scptMan);

      void machineMainRoutine(int yMinLevel, int yMaxLevel);

      void coreProcess(int yMinLevel, int yMaxLevel);

      void machineTranspose(const MyPoint::PointList&
inputPntLst, char input);
      std::string activatedScriptMode();
      std::string pushScriptState();
      void setSelectedPrototypeChar(char ch);
      void setMyUIRecogTextBox();
      void clearStack();
private:
      int mode;
      unsigned selectedPrototypeChar;
      gcroot<HandWriting::MyUI^> myUI;
      RecognitionProcess::RecognitionInterface* recogMan;
      ScriptManagement::ScriptProcess* scriptMan;
      SystemStateStackInterface* stateStack;
      CurveListBuilder* curveLst;
      RecognitionProcess::PrototypeList prototypeList;
      MachineStateMap machineMap;
      ScriptCharMap scriptCharMap;
};
```

void setRunningMode(int rmode);

กำหนดโหมดการประมวลผลโปรแกรม

void setMyUI(HandWriting::MyUI^ myui);

กำหนด ความสัมพันธ์ไปยัง MyUI Instant

void setSystemStateStack
(SystemStateStackInterface stack);*

กำหนด ความสัมพันธ์ไปยัง SystemStateStack Instant

```
void setCurveListBuilder
(CurveListBuilder* curveLstBD);
```
กำหนด ความสัมพันธ์ไปยัง CurveListBuilder Instant

```
void setRecognitionInterface
(RecognitionProcess::RecognitionInterface*);
```
กำหนด ความสัมพันธ์ไปยัง Recognition Instant

```
void setScriptProcess
(ScriptManagement::ScriptProcess* scptMan);
```
กำหนด ความสัมพันธ์ไปยัง Script Instant

```
void machineMainRoutine(int yMinLevel, int
yMaxLevel);
```
คล้าย main function ดูแลการประมวลผลโปรแกรมโดยรวม

```
void coreProcess(int yMinLevel, int yMaxLevel);
```
ฟังก์ชันดูแลการประมวลผลโปรแกรมส่วนหาอักษรโดดจากการ Recognition

```
void machineTranspose(const MyPoint::PointList&
inputPntLst, char input);
```
ฟังก์ชันดูแลการประมวลผลโปรแกรมส่วนหาอักษรเอาท์พุทโดยอาศัย State Machine

```
std::string activatedScriptMode();
```
ฟังก์ชันสั่งเริ่มการประมวลผลโปรแกรมแบบโหมดสคริปต์ และให้ Script String ออกมา

```
void cleanWrittingPad(const RemovedStatedMessage&
removedMss);
```
ลบลายมือเขียนบน Writing Pad

```
std::string pushScriptState();
```
เพิ่ม Script State เข้า State Stack และให้ Script String ออกมา

```
void setSelectedPrototypeChar(char ch);
```
กำหนดอักษรโปรโทไทป์ที่ถือว่ามีคุณลักษณะใกล้เคียงลายมือเขียนมากที่สุด

```
void setMyUIRecogTextBox();
```
กำหนด String ที่จะแสดงออกที่ Recognition Window

```
void clearStack();
```
ลบ State ทั้งหมดใน Stack

4.2.4.7. Class PrototypeCharacter คลาสซึ่งระบุและแสดงโครงสร้างข้อมูล โดยเก็บข้อมูลตัวอักษรเอาท์พุท 5 ตัวที่ได้มาจากการเรียกโครงงานเดิมในแต่ละครั้ง

4.2.5. WriterManagement: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดการเกี่ยวกับผู้ใช้งานระบบ โดยหากมีผู้ใช้งานระบบใหม่เข้ามาจะเก็บข้อมูลของผู้ใช้งานระบบเข้าสู่ฐานข้อมูล

4.2.5.1. Class WriterInterface เป็น abstract class ซึ่งประกอบด้วยฟังก์ชันต่างๆ ในการจัดการกับข้อมูลของผู้ใช้ระบบ (Writer) ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class WriterInterface
{
public:
      virtual unsigned saveWriter() =0;
      virtual const DBManagement::Writer& getWriter()
                                          const = 0;
      virtual void deleteWriterFromDB(const char* db, const
                  char* host,const char* usr) = 0;
      virtual void updateWriterTODB(const char* db, const
char*
                                    host,const char* usr) =
                                0;
      virtual void updateLastTimeWritten (WriterInterface*
                                    wrtIn, const char* db,
                                    const char* host,const
                                    char* usr) = 0;
      virtual void updateLastTimeWritten (const char* db,
const
                                    char* host,const char*
                                    usr) = 0;
      virtual void updateWriterNameAndStdID(const
std::string&
                              name, const std::string& stdId,
                              const char* db, const char*
                              host,const char* usr) = 0;

      virtual void setWriterID(unsigned id) = 0;
      virtual void setName(const std::string& nm) = 0;
      virtual void setFirstWrtTime
                        (const std::string& fstTime) = 0;
      virtual void setLastWrtTime
                       (const std::string& lstTime) = 0;
      virtual void setStudentID
                   (const std::string& stdID) = 0;

      virtual void setName(System::String^ nm) = 0;
      virtual void setFirstWrtTime
                         (System::String^ fstTime) = 0;
      virtual void setLastWrtTime
                     (System::String^ lstTime) = 0;
      virtual void setStudentID(System::String^ stdID) = 0;

      virtual unsigned getWriterID() const = 0;
      virtual const std::string& getName() const = 0 ;
      virtual const std::string& getFirstWrtTime() const = 0;
      virtual const std::string& getLastWrtTime() const = 0;
      virtual const std::string& getStudentID() const = 0;
```

```
        virtual System::String^ getSysStringWriterID() = 0;
        virtual System::String^ getSysStringName() = 0;
        virtual System::String^ getSysStringFirstWrtTime() = 0;
        virtual System::String^ getSysStringLastWrtTime() = 0;
        virtual System::String^ getSysStringStudentID() = 0;
};
```

virtual unsigned saveWriter() =0;

เก็บข้อมูลผู้ใช้ระบบลงฐานข้อมูล

virtual const DBManagement::Writer& getWriter() const = 0;

ให้ข้อมูลของผู้ใช้ระบบในปัจจุบัน

virtual void deleteWriterFromDB(const char db, const char* host,const char* usr) = 0;*

ลบเก็บข้อมูลผู้ใช้ระบบจากฐานข้อมูล

virtual void updateWriterTODB(const char db, const char* host,const char* usr) = 0;*

แก้ไขข้อมูลผู้ใช้ระบบในฐานข้อมูล

virtual void updateLastTimeWritten (WriterInterface wrtIn, const char* db, const char* host,const char* usr) = 0;*

แก้ไขข้อมูลเวลาการใช้ระบบครั้งล่าสุดของผู้ใช้ในฐานข้อมูล

virtual void updateLastTimeWritten (const char* db, const char* host,const char* usr) = 0;

แก้ไขข้อมูลเวลาการใช้ระบบครั้งล่าสุดของผู้ใช้ในฐานข้อมูล

virtual void updateWriterNameAndStdID(const std::string& name, const std::string& stdId, const char db, const char* host,const char* usr) = 0;*

แก้ไขข้อมูลชื่อและรหัสนักศึกษาของผู้ใช้ระบบในฐานข้อมูล

4.2.5.2. Class WriterProfile สืบทอด (inherit) มาจาก Class WriterInterface คลาส ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class WriterProfile : public WriterInterface
{
public:
        WriterProfile();
        WriterProfile(const std::string& fstWrtTime,
                const std::string& lstWrtTime,
                const std::string& stdID, const std::string&
nme);
        WriterProfile(const std::string& nme,
```

```
            const std::string& stdID);

        WriterProfile(const DBManagement::Writer& writer);
        WriterProfile(const WriterProfile& writer);
        WriterProfile& operator=
              (const DBManagement::Writer& writer);
        unsigned saveWriter();
        const DBManagement::Writer& getWriter() const;
        void deleteWriterFromDB(const char* db,
              const char* host,const char* usr);
        void updateWriterTODB(const char* db,
              const char* host,const char* usr);
        void updateLastTimeWritten (WriterInterface* wrtIn,
              const char* db, const char* host,const char*
        usr);
        void updateLastTimeWritten (const char* db,
              const char* host,const char* usr);
        void updateWriterNameAndStdID(const std::string& name,
              const std::string& stdId,const char* db, const
              char* host,const char* usr);

        void setWriterID(unsigned id);
        void setName(const std::string& nm);
        void setFirstWrtTime(const std::string& fstTime);
        void setLastWrtTime(const std::string& lstTime);
        void setStudentID(const std::string& stdID);

        void setName(System::String^ nm);
        void setFirstWrtTime(System::String^ fstTime);
        void setLastWrtTime(System::String^ lstTime);
        void setStudentID(System::String^ stdID);

        unsigned getWriterID() const;
        const std::string& getName() const;
        const std::string& getFirstWrtTime() const;
        const std::string& getLastWrtTime() const;
        const std::string& getStudentID() const;

        System::String^ getSysStringWriterID();
        System::String^ getSysStringName();
        System::String^ getSysStringFirstWrtTime();
        System::String^ getSysStringLastWrtTime();
        System::String^ getSysStringStudentID();

    private:
          unsigned  writerID;
          std::string  firstWrtTime;
          std::string   lastWrtTime;
          std::string  studentID;
          std::string  name;
    };
```

4.2.6. CharacterInterface: แหล่งรวมของโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่เป็นสื่อกลางในการติดต่อกันระหว่างข้อมูลจากฐานข้อมูลและ การแสดงผลข้อมูลออกทางหน้าจอ

4.2.6.1. Class CharacterInterface ประกอบด้วยฟังก์ชันในการติดต่อกับ CharacterDB Instant เพื่อแปลงอักษรระหว่างรหัส ASCII และ รหัส UNICODE ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class CharacterInterface
{
public:
      CharacterInterface();
      CharacterInterface
            (const DBManagement::CharacterDB& chDB);
      CharacterInterface& operator=
            (const DBManagement::CharacterDB& chDB);

      void setCharID(const System::String^ chId);
      void setWriterID(const System::String^ writerId);
      void setScriptChar(const System::String^ scpCh);
      void setRecogChar(const System::String^ recCh);
      void setEditChar(const System::String^ edCh);
      void setWriteTime(const System::String^ wrtTime);
      void setWrtPointList(const System::String^ pntLst);
      void setEditPointList(const System::String^ pntLst);
      void setStartPoint(const System::String^ strPnt);
      void setEndPoint(const System::String^ nPnt);


      void setCharID(const std::string& chId);
      void setWriterID(const std::string& writerId);
      void setScriptChar(const std::string& scpCh);
      void setRecogChar(const std::string& recCh);
      void setEditChar(const std::string& edCh);
      void setWriteTime(const std::string& wrtTime);
      void setWrtPointList(const std::string& pntLst);
      void setEditPointList(const std::string& pntLst);
      void setStartPoint(const std::string& strPnt);
      void setEndPoint(const std::string& nPnt);

      System::String^ getCharID();
      System::String^ getWriterID();
      System::String^ getScriptChar();
      System::String^ getRecogChar();
      System::String^ getEditChar();
      System::String^ getWriteTime();
      System::String^ getWrtPointList();
      System::String^ getEditPointList();
      System::String^ getStartPoint();
      System::String^ getEndPoint();

      unsigned getUnsignedCharID() const;
      unsigned getUnsignedWriterID() const;
      const std::string& getStringScriptCh() const;
      const std::string& getStringRecogCh() const;
      const std::string& getStringEditCh() const;
      const std::string& getStringWriteTime() const;
      const std::string& getStringWrtPntLst() const;
      const std::string& getStringEditPntLst() const;
      int getIntStartPoint() const;
      int getIntEndPoint() const;

      void deleteCharFromDB(const char* db, const char*
                        host,const char* usr);
      void updateCharTODB(const char* db, const char*
                        host,const char* usr);
private:
```

```
        unsigned charID;
        unsigned writerID;
        std::string scriptCh;
        std::string recogCh;
        std::string editCh;
        std::string writeTime;
        std::string wrtPntLst;
        std::string editPntLst;
        int startPnt;
        int endPnt;
};
```

void deleteCharFromDB(const char db, const char* host,const char* usr);*

ลบข้อมูลตัวอักษรจากฐานข้อมูล

void updateCharTODB(const char db, const char* host,const char usr);*

แก้ไขข้อมูลตัวอักษรในฐานข้อมูล

4.2.6.2. Class CharacterObjectSet จัดการกับลำดับของ CharacterInterface Instance

4.2.7. DBManagement: รวบรวมโมดูล คลาส และ ฟังก์ชันต่างๆ ที่ทำหน้าที่จัดการติดต่อกับฐานข้อมูล

4.2.7.1. Class CharacterDB เก็บข้อมูลของตัวอักษรต่างๆที่โหลดมาจากฐานข้อมูล ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class CharacterDB
{
private:
        typedef mysqlpp::Null< int , mysqlpp::NullisZero >
                                                    IndexPntList;
        Writer writer;
        unsigned int charID;
        unsigned int writerID;
        MyString writeLabel;
        MyString recogLabel;
        MyString editLabel;
        mysqlpp::DateTime writtenTime;
        std::string writtenPointList;
        MyString editPointList;
        IndexPntList startPoint;
        IndexPntList endPoint;
public:
        CharacterDB(): charID(0),writerID(0),writeLabel(),
                    recogLabel(),editLabel(),writtenTime(),
                    writtenPointList(),editPointList(),
                    startPoint(0),endPoint(0){}
        CharacterDB(const Writer& wrt, std::string wrtLB,
                    std::string regLB, const std::string&
                    wrtPntList, const char* db, const char*
                    host, const char* usr, const char* pass="",
```

```
            std::string editLB="", std::string
            editPntLst="",int start=0, int end=0);

CharacterDB(unsigned wrtID, std::string wrtLB,
            std::string regLB, const std::string&
            wrtPntList, const char* db, const char*
            host, const char* usr, const char* pass="",
            std::string editLB="", std::string
            editPntLst="",int start=0, int end=0);

CharacterDB(const mysqlpp::Row& row):
            charID(row[unsigned(0)]),
            writerID(row[1]),
            writeLabel(std::string(row[2])),
            recogLabel(std::string(row[3])),
            editLabel(std::string(row[4])),
            writtenTime(row[5]),
            writtenPointList(row[6]),
            editPointList(std::string(row[7])),
            startPoint(int (row[8])),
            endPoint(int (row[9]))   {}

CharacterDB& operator=(const mysqlpp::Row& row)
{
      charID = row[unsigned(0)];
      writerID = row[1];
      writeLabel = std::string(row[2]);
      recogLabel = std::string(row[3]);
      editLabel = std::string(row[4]);
      writtenTime = row[5];
      writtenPointList = row[6];
      editPointList = std::string(row[7]);
      startPoint = int (row[8]);
      endPoint = int (row[9]);
      return *this;
}

CharacterDB
(const CharacterInterface::CharacterInterface& chIn);
CharacterDB& operator=
(const CharacterInterface::CharacterInterface& chIn);

mysqlpp::SQLString getCharID() const;
mysqlpp::SQLString getWriterID() const;
mysqlpp::SQLString getWrtLabel() const;
mysqlpp::SQLString getRecogLabel() const;
mysqlpp::SQLString getEditLabel() const;
mysqlpp::SQLString getWrtTime() const;
mysqlpp::SQLString getWrtPntList() const;
mysqlpp::SQLString getEditPntList() const;
mysqlpp::SQLString getStartPnt() const;
mysqlpp::SQLString getEndPnt() const;

unsigned getCurrWrtCharID() const;


void updateCharacter
      const CharacterInterface::CharacterInterface&
      chIn,const char* db, const char* host,
      const char* usr, const char* pass = "");

void updateRecogChar(unsigned charId, std::string
```

```
            recogChar,const char* db, const char* host,const
            char* usr, const char* pass = "");
      void deleteRecCharRow(unsigned charId,const char* db,
            const char* host,const char* usr,
            const char* pass = "");
      unsigned getCharIDUnsigned() const;
      unsigned getWriterIDUnsigned() const;
      std::string getScriptLabelString() const;
      std::string getRecogLabelString() const;
      std::string getEditLabelString() const;
      std::string getWrtTimeString() const;
      std::string getWrtPntListString() const;
      std::string getEditPntListString() const;
      int getStartPntInt() const;
      int getEndPntInt() const;

};
```

void updateCharacter(const CharacterInterface::CharacterInterface& chIn,const char db, const char* host,const char* usr, const char* pass = "");*

แก้ไขข้อมูลตัวอักษรในฐานข้อมูล

void updateRecogChar(unsigned charId, std::string recogChar, const char db, const char* host,const char* usr, const char* pass = "");*

แก้ไขผลการ Recognition ของลายมือเขียนในฐานข้อมูล

void deleteRecCharRow(unsigned charId, const char db, const char* host,const char* usr, const char* pass = "");*

ลบข้อมูลตัวอักษรในฐานข้อมูล

4.2.7.2. Class Writer เก็บข้อมูลของผู้เขียนหรือผู้ใช้ระบบซึ่งโหลดจากฐานข้อมูล ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class Writer
{
private:
      typedef mysqlpp::Null<mysqlpp::DateTime,
                        mysqlpp::NullisBlank> WrittenTime;
      unsigned int writerID;
      WrittenTime  firstWrtTime;
      WrittenTime  lastWrtTime;
      MyString     studentID;
      MyString     name;
public:
      Writer():writerID(0),firstWrtTime(),lastWrtTime(),
            studentID(),name(){}
      Writer(const mysqlpp::Row& row):
            writerID(row[unsigned (0)]),
            firstWrtTime(mysqlpp::DateTime(row[1])),
            lastWrtTime(mysqlpp::DateTime (row[2])),
```

```
      studentID(std::string(row[3])),
      name(std::string (row[4])){}
Writer(const char* db, const char* host,
      const char* usr, const char* pass="",
      std::string& stdID="");
Writer& operator=(const mysqlpp::Row& row)
{
      writerID = row[unsigned(0)];
      firstWrtTime = std::string(row[1]);
      lastWrtTime = std::string(row[2]);
      studentID = std::string(row[3]);
      name = std::string(row[4]);
      return *this;
}
Writer(const WriterManagement::WriterInterface& wrtIn);
Writer& operator=
      (const WriterManagement::WriterInterface& wrtIn);

mysqlpp::SQLString getStringWriterID() const;
mysqlpp::SQLString getWriterName
      (const char* db, const char* host,
      const char* usr, const char* pass="") const;
mysqlpp::SQLString getWriterName() const;
mysqlpp::SQLString getFstWrtTime() const;
mysqlpp::SQLString getLstWrtTime() const;
mysqlpp::SQLString getStdID() const;

unsigned    getWriterID() const;
std::string getStringWriterName() const;
std::string getStringFstWrtTime() const;
std::string getStringLstWrtTime() const;
std::string getStringStdID() const;


void setWriterID(unsigned wrtID);
void setName(const std::string& nm );
void setFirstWritingTime(const std::string& fstTime);
void setLastWritingTime(const std::string& lstTime);
void setStudentID(const std::string& stdID);

unsigned insertWriter
      (const char* db, const char* host,const char*
usr);
void updateLastWrtTime
      (const std::string& userName, const char* db,
       const char* host,const char* usr, const char*
       pass="");
unsigned int getMaxWriterID
      (const char* db, const char* host,const char*
usr,
       const char* pass="")const;
void updateWriter
      (const WriterManagement::WriterInterface& wrtIn,
       const char* db,const char* host,const char* usr,
       const char* pass = "");
void deleteWriter
      (const WriterManagement::WriterInterface& wrtIn,
       const char* db,const char* host,const char* usr,
       const char* pass = "");
void updateWriterNameAndStdID
      (unsigned wrtID, const std::string& name,
       const std::string& stdId,const char* db,
       const char* host,const char* usr);
```

```
};
```

```
unsigned insertWriter(const char* db, const char*
host,const char* usr);
```

เพิ่มผู้ใช้ระบบเข้าฐานข้อมูล

```
void updateLastWrtTime(const std::string& userName,
const char* db, const char* host,const char* usr,
const char* pass="");
```

แก้ไขเวลาการเข้าใช้งานครั้งล่าสุดของผู้ใช้ในฐานข้อมูล

```
unsigned int getMaxWriterID(const char* db, const
char* host,const char* usr, const char*
pass="")const;
```

ให้รหัส Writer ID สูงสุดที่มีในฐานข้อมูล

```
void updateWriter(const
WriterManagement::WriterInterface& wrtIn, const
char* db,const char* host,const char* usr, const
char* pass = "");
```

แก้ไขข้อมูลของผู้ใช้ในฐานข้อมูล

```
void deleteWriter(const
WriterManagement::WriterInterface& wrtIn, const
char* db, const char* host,const char* usr, const
char* pass = "");
```

ลบข้อมูลของผู้ใช้ในฐานข้อมูล

```
void updateWriterNameAndStdID(unsigned wrtID, const
std::string& name, const std::string& stdId, const
char* db, const char* host,const char* usr);
```

แก้ไขข้อมูลชื่อ และรหัสนักศึกษาของผู้ใช้ ในฐานข้อมูล

4.2.7.3. Class DBConnection ติดต่อฐานข้อมูล ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
class DBConnection
{
private:
      const char* dbName;
      const char* hostAddress;
      const char* userName;
      const char* passWord;
      mysqlpp::Connection con;

public:
      DBConnection():con(){}
      DBConnection(const char* db, const char* host = "",
                  const char* user= "", const char*
```

```
                password=""):dbName(db),hostAddress(host),
                userName(user),passWord(password),con(db,
                host, user, password)
{
      mysqlpp::Query query = con.query();
      std::string setName = "set names tis620";
      query.exec(setName);
      query.reset();
}

bool isWriterExist(std::string name);
bool isThereAnyWriter();

bool selectWriterByRow(std::string name,Writer& wrt);
bool selectWriterByRow(std::string name);
unsigned int selectWriterID(std::string name);
std::string selectWrittenTime(std::string wrtTime,
                              unsigned int wrt_id);

bool insertWriter(const Writer& wrt);
unsigned int insertCharacter(const CharacterDB& chDB );
unsigned int insertNewWriter(const Writer& wrt);

void updateLstWrtTime(unsigned int wrtId,
                      std::string lstTime);
void updateRecogLabel(unsigned int charId,
                      std::string recogChar);
void updateCharacter(const CharacterDB& chDB);
void updateWriter(const Writer& wrt);
void updateWriterNameAndStdID(unsigned wrtID, const
      std::string& name, const std::string&
   stdId);

void deleteCharRow(unsigned chID);
void deleteWriter(unsigned wrtID);

unsigned int getMaxWriterID();
const char* getDBName() const;
const char* getHostAddress() const;
const char* getUserName() const;
const char* getPassWord() const;
bool getCharacterSet
    (std::vector<CharacterDB>& charSet);
void getCharacterByWriter(unsigned wrtID,
      std::vector<CharacterDB>& charSet);
void getWriterSet(std::vector<Writer>& wrtSet);
void setNullWriterToCharacter(const
      std::vector<CharacterDB>& charSet);
};
```

bool isWriterExist(std::string name);

ตรวจสอบว่ามีชื่อผู้ใช้ในระบบหรือไม่ในฐานข้อมูล

bool selectWriterByRow(std::string name,Writer& wrt);

เลือกดึงข้อมูลของผู้ใช้จากฐานข้อมูล

bool selectWriterByRow(std::string name);

เลือกดึงข้อมูลของผู้ใช้จากฐานข้อมูล

```
unsigned int selectWriterID(std::string name);
```
ให้รหัส Writer IDของผู้ใช้จากฐานข้อมูล

```
bool insertWriter(const Writer& wrt);
```
เพิ่มผู้ใช้ระบบในฐานข้อมูล

```
unsigned int insertNewWriter(const Writer& wrt);
void updateLstWrtTime(unsigned int wrtId,
std::string lstTime);
```
เพิ่มผู้ใช้ระบบในฐานข้อมูล

```
unsigned int insertCharacter(const CharacterDB& );
```
เพิ่มข้อมูลตัวอักษรในฐานข้อมูล

```
unsigned int getMaxWriterID();
```
ให้รหัส Writer ID สูงสุดของผู้ใช้จากฐานข้อมูล

```
void updateRecogLabel(unsigned int charId,
std::string recogChar);
```
แก้ไขผลการ Recognition ของลายมือเขียนในฐานข้อมูล

```
void updateCharacter(const CharacterDB& chDB);
```
แก้ไขข้อมูลตัวอักษรในฐานข้อมูล

```
bool getCharacterSet(std::vector<CharacterDB>&);
```
ให้ข้อมูลตัวอักษรทั้งหมดในฐานข้อมูล

```
void deleteCharRow(unsigned chID);
```
ลบข้อมูลตัวอักษรในฐานข้อมูล

```
void updateWriter(const Writer& wrt);
```
แก้ไขผู้ใช้ระบบในฐานข้อมูล

```
void deleteWriter(unsigned wrtID);
```
ลบผู้ใช้ระบบในฐานข้อมูล

```
void getWriterSet(std::vector<Writer>& wrtSet);
```
ให้ผู้ใช้ระบบทั้งหมดในฐานข้อมูล

```
void updateWriterNameAndStdID(unsigned wrtID, const
std::string& name, const std::string& stdId);
```

แก้ไขข้อมูลชื่อ และรหัสนักศึกษาของผู้ใช้ ในฐานข้อมูล

4.2.8. ไม่มี Namespace

4.2.8.1. Class CurveListBuilder เก็บตัวอักษรลายมือเขียนเป็นลำดับของพิกัด (x, y) ที่เขียนบน Writing Pad ประกอบด้วยฟังก์ชั่นต่างๆ ดังนี้

```
void setPointList(const MyPoint::PointList& pntLst)
```

กำหนดค่าลำดับของพิกัด(x, y) ซึ่งเป็นอินพุทลายมือเขียน

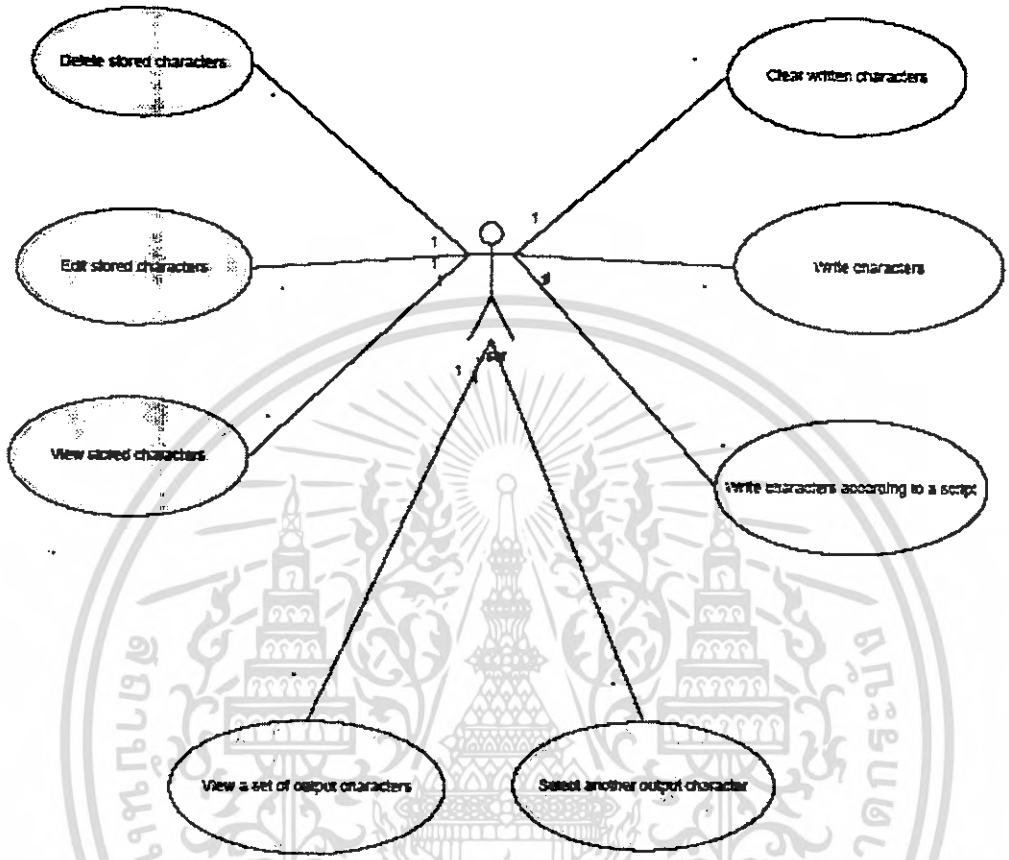
```
const MyPoint::PointList& getPointList() const;
```

ให้ค่าลำดับของพิกัด(x, y) ซึ่งเป็นอินพุทลายมือเขียน

4.2.8.2. Class ThaiRecognition คลาสติดต่อกับ โครงงานเดิม โดยเป็น dynamic link library ค่าต่างๆที่ได้ออกมาส่งผ่านทาง pointer

4.3 UML ของระบบ

4.3.1 การออกแบบ Use Case Diagram ของระบบ



รูปที่ 4.2 แสดง Usecase Diagram ของระบบ

1. Delete Stored Characters: ลบข้อมูลตัวอักษรลายมือเขียนที่ละตัวอักษรในฐานข้อมูล

2. Edit Stored Characters: แก้ไขข้อมูลตัวอักษรลายมือเขียนที่ละตัวอักษรในฐานข้อมูล

3. View Stored Characters: ดูข้อมูลตัวอักษรลายมือเขียนที่ละตัวอักษรในฐานข้อมูล

4. Clear Written Characters: ตัวอักษรลายมือเขียนทั้งหมดในWriting Pad

5. Write Characters: เขียนตัวอักษรลงในWriting Pad

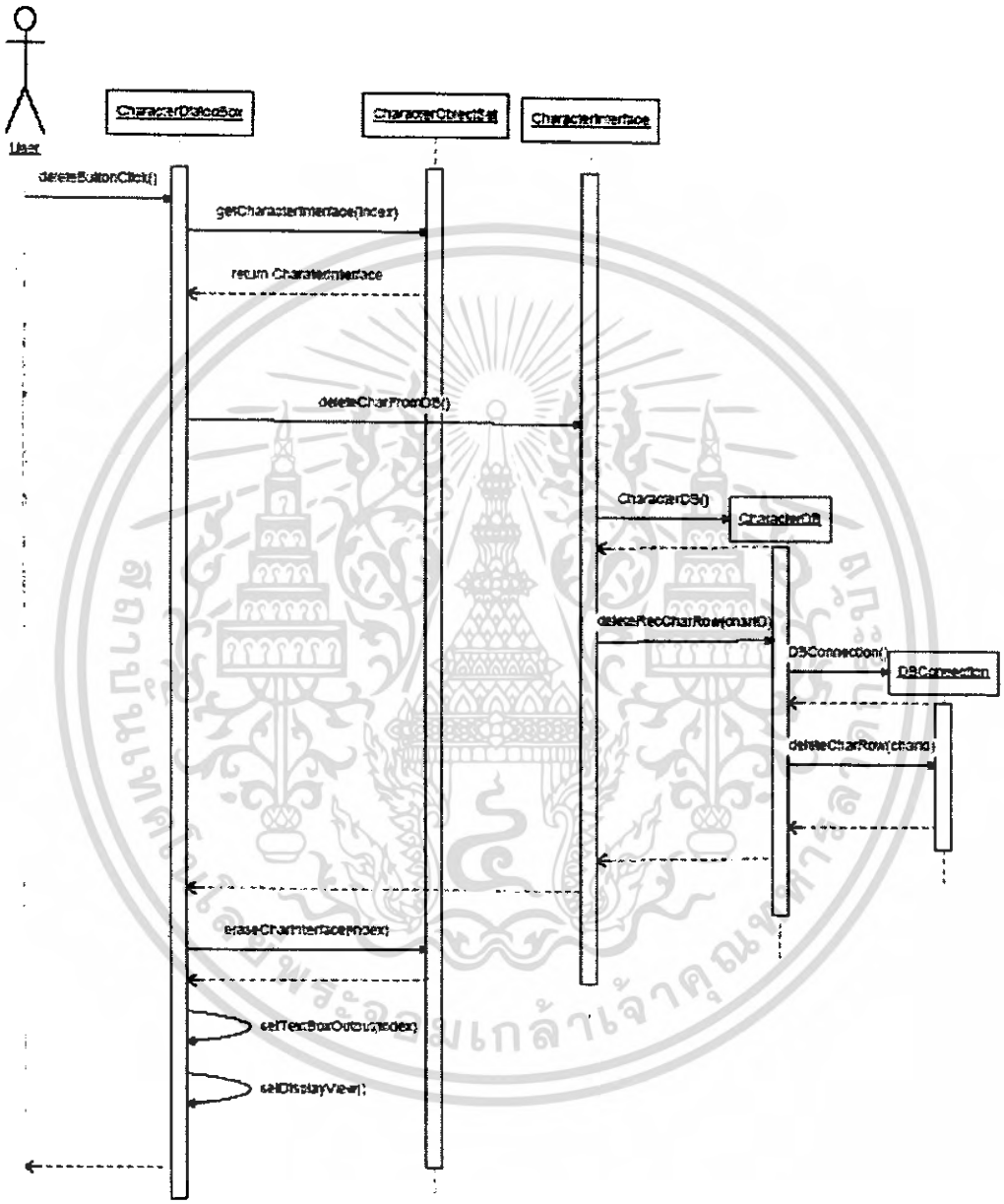
6. Write Characters according to scripts: เขียนตัวอักษรตามที่แสดงใน Script Window ลงในWriting Pad

7. View a set of output characters: แสดงตัวอักษรทั้งหมดที่มีลักษณะใกล้เคียงกับตัวอักษรลายมือเขียน

8. Select another output character: เลือกตัวอักษรเอาท์พุทที่มีลักษณะใกล้เคียงกับตัวอักษรลายมือเขียน

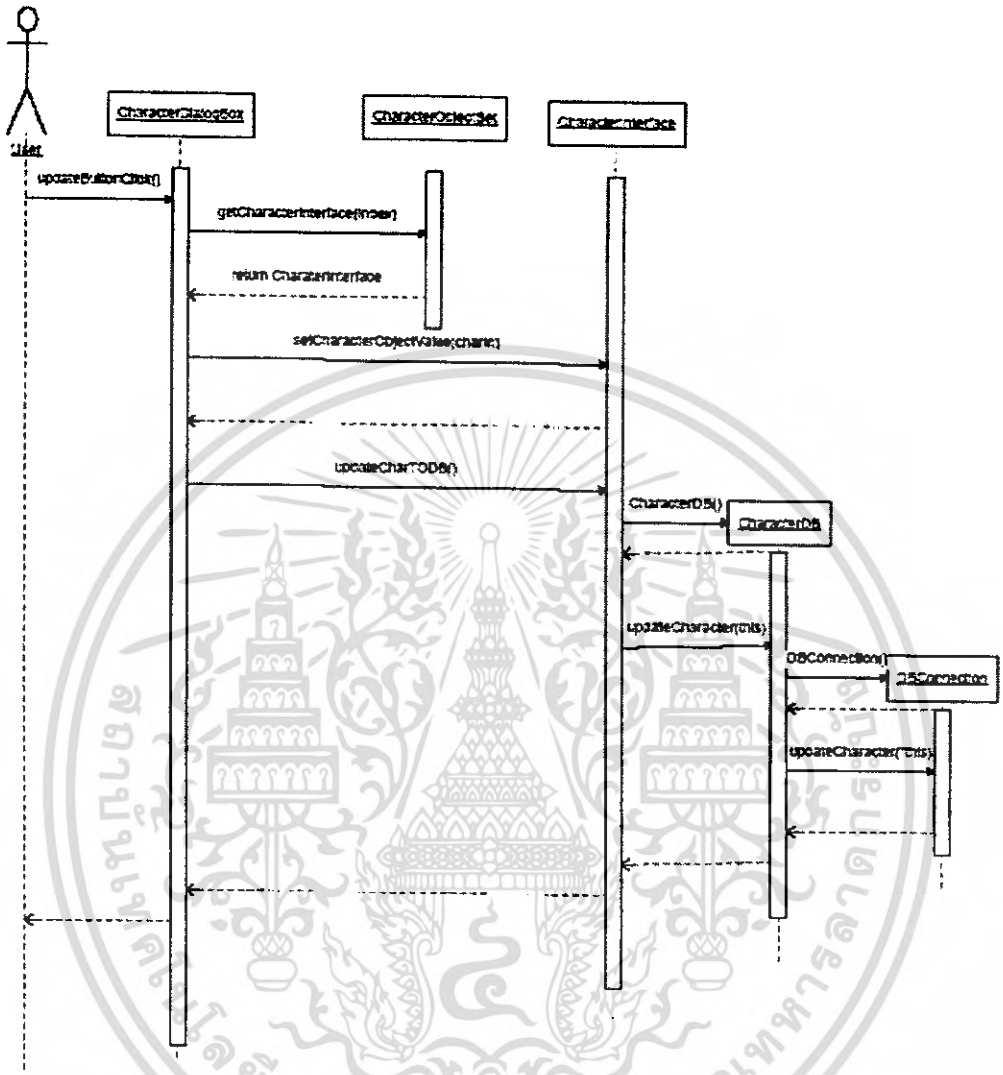
4.3.2 การออกแบบ Sequence Diagram ของระบบ

4.3.2.1. Delete Stored Characters



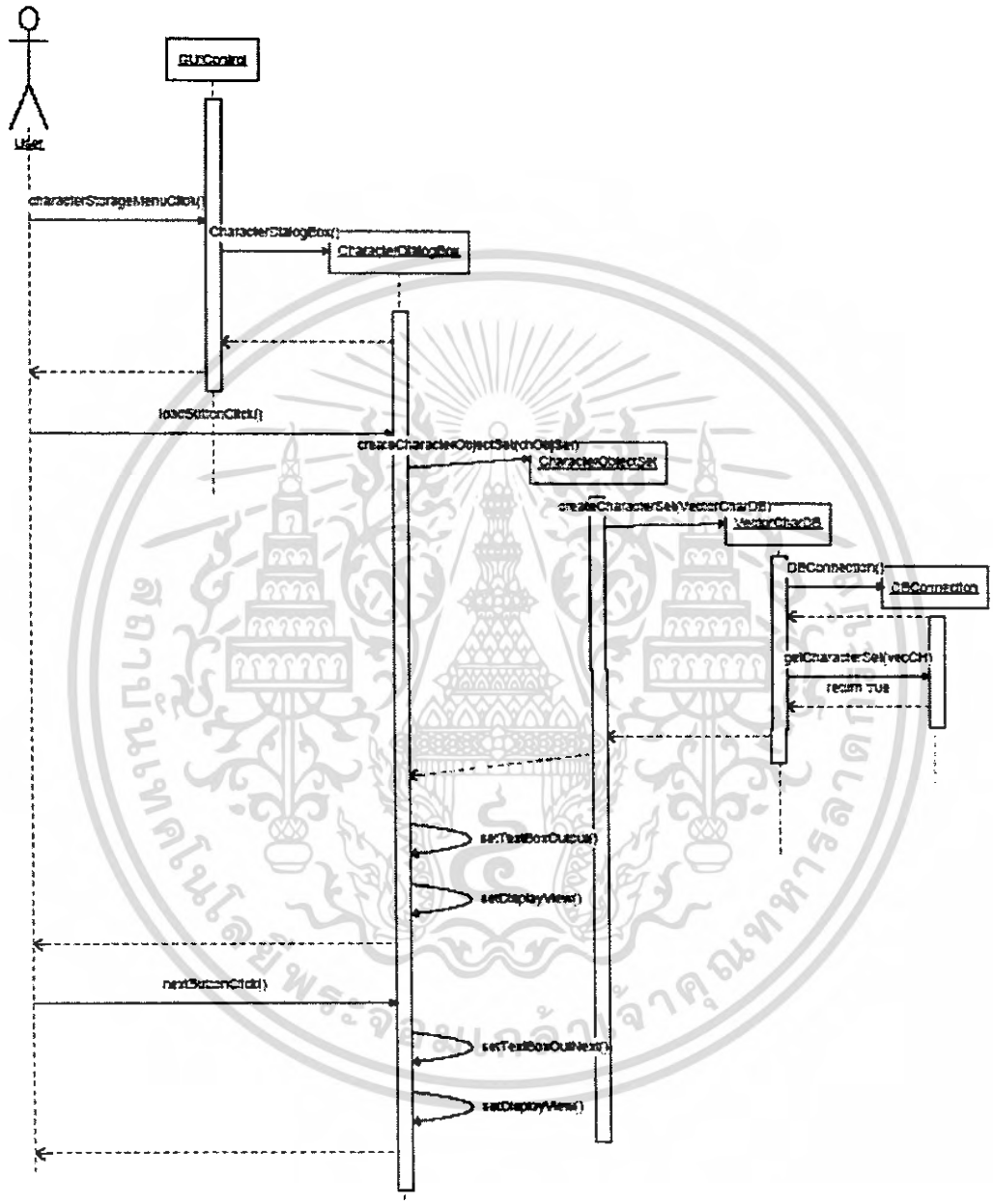
รูปที่ 4.3 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้ลบตัวอักษรออกจากฐานข้อมูล

4.3.2.2. Edit Stored Characters



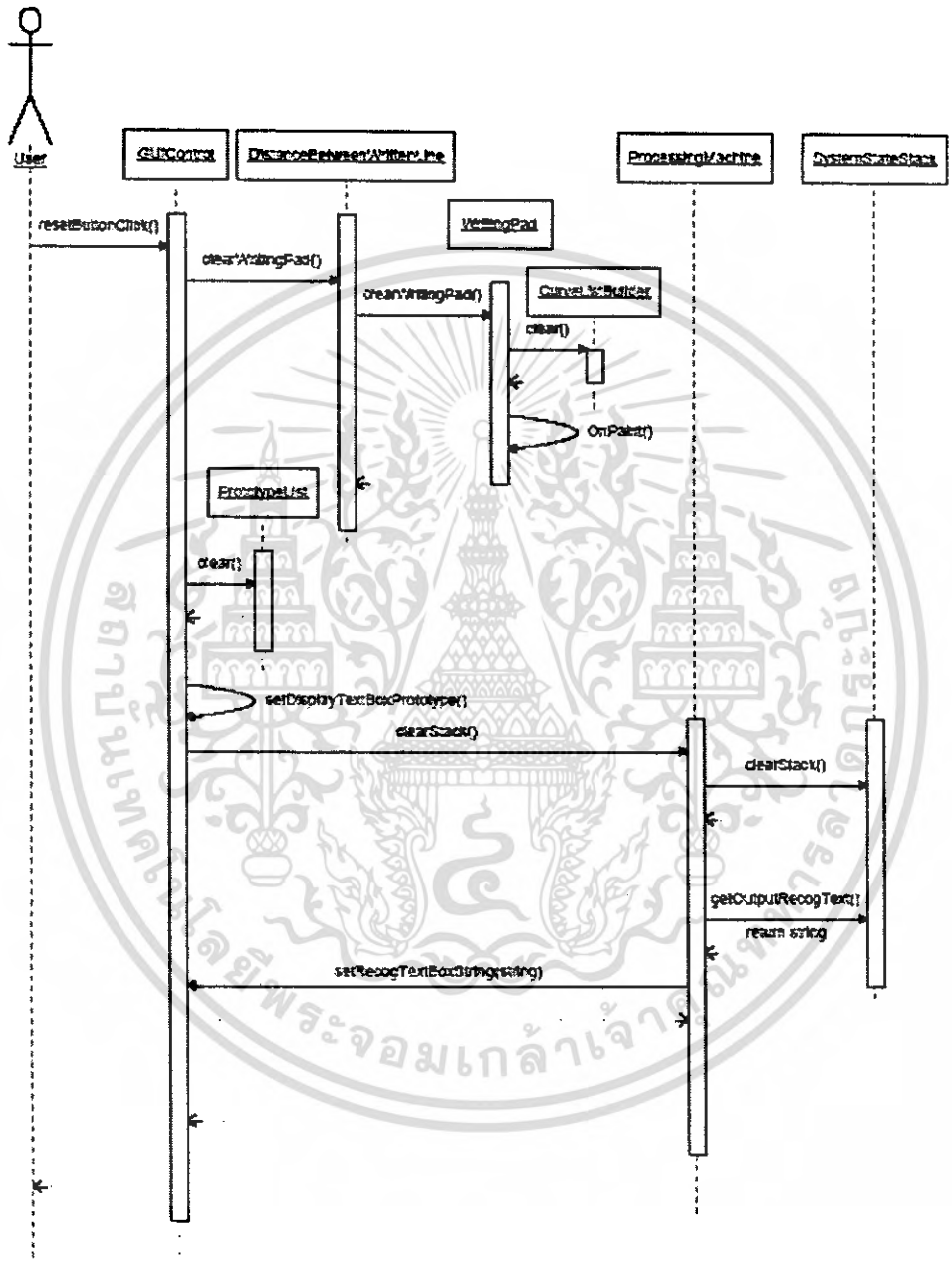
รูปที่ 4.4 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้แก้ไขข้อมูลตัวอักษรในฐานข้อมูล

4.3.2.3. View Stored Characters



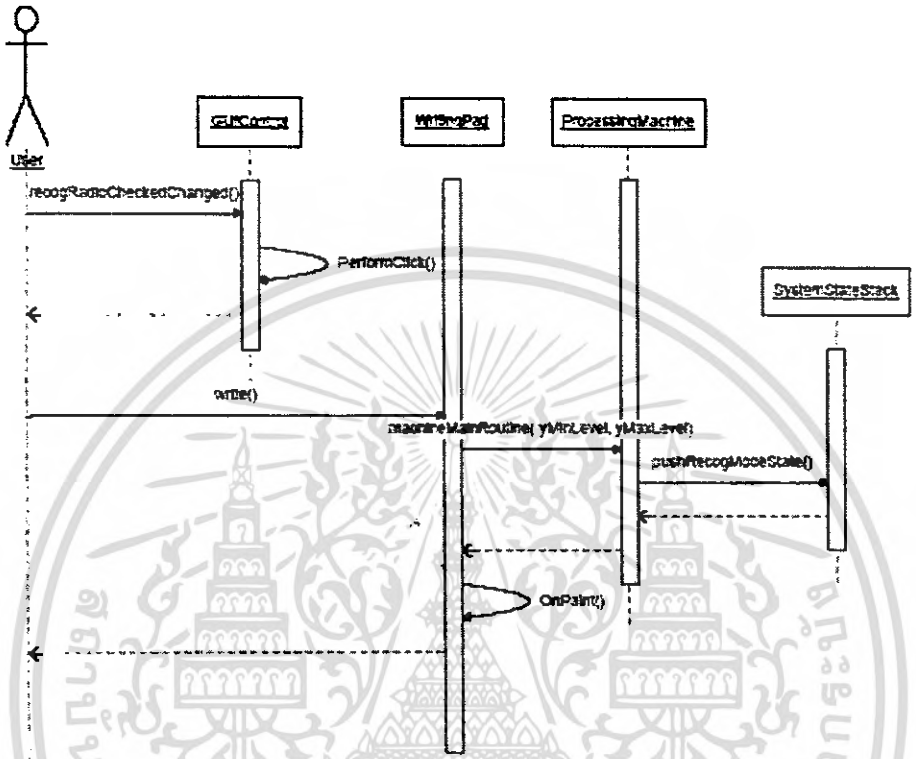
รูปที่ 4.5 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้เรียกดูข้อมูลตัวอักษรในฐานข้อมูล

4.3.2.4. Clear Written Characters



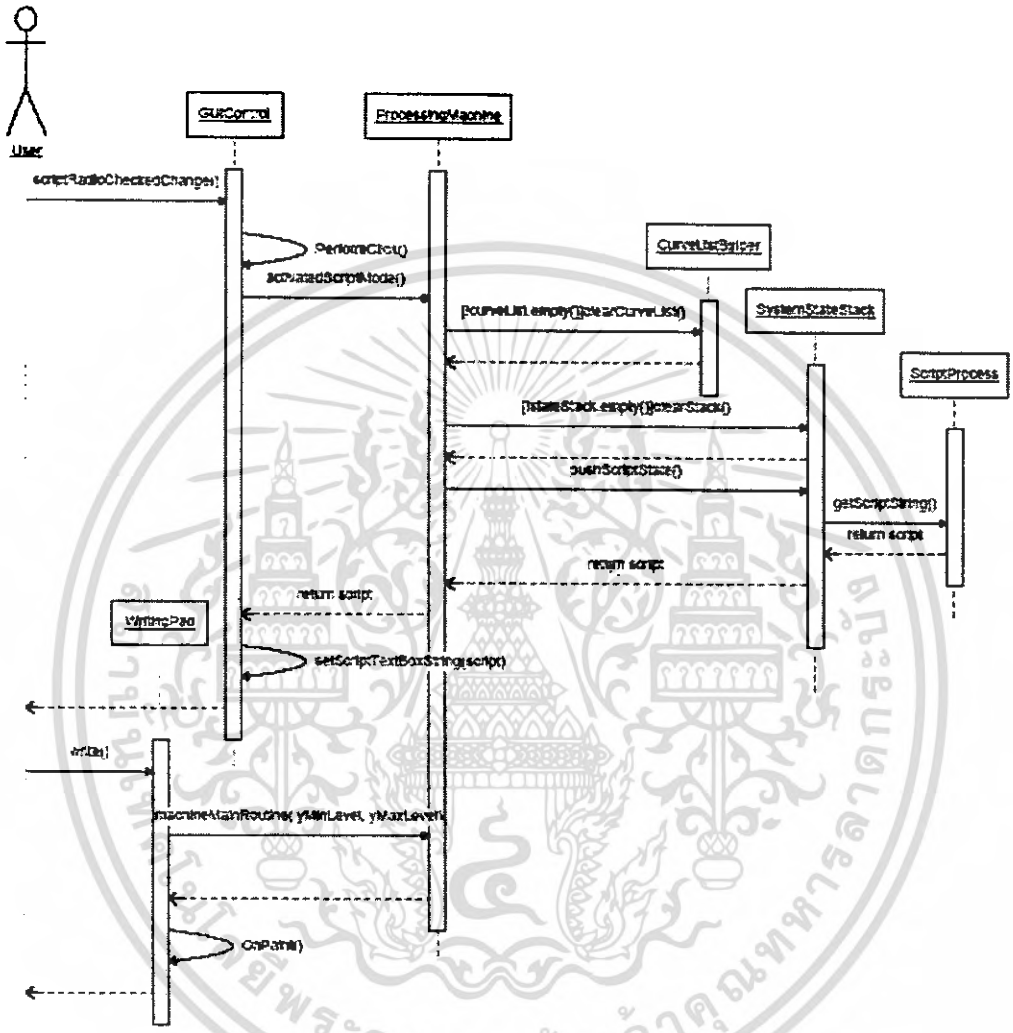
รูปที่ 4.6 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้ต้องการรีเซตระบบใหม่

4.3.2.5. Write Characters



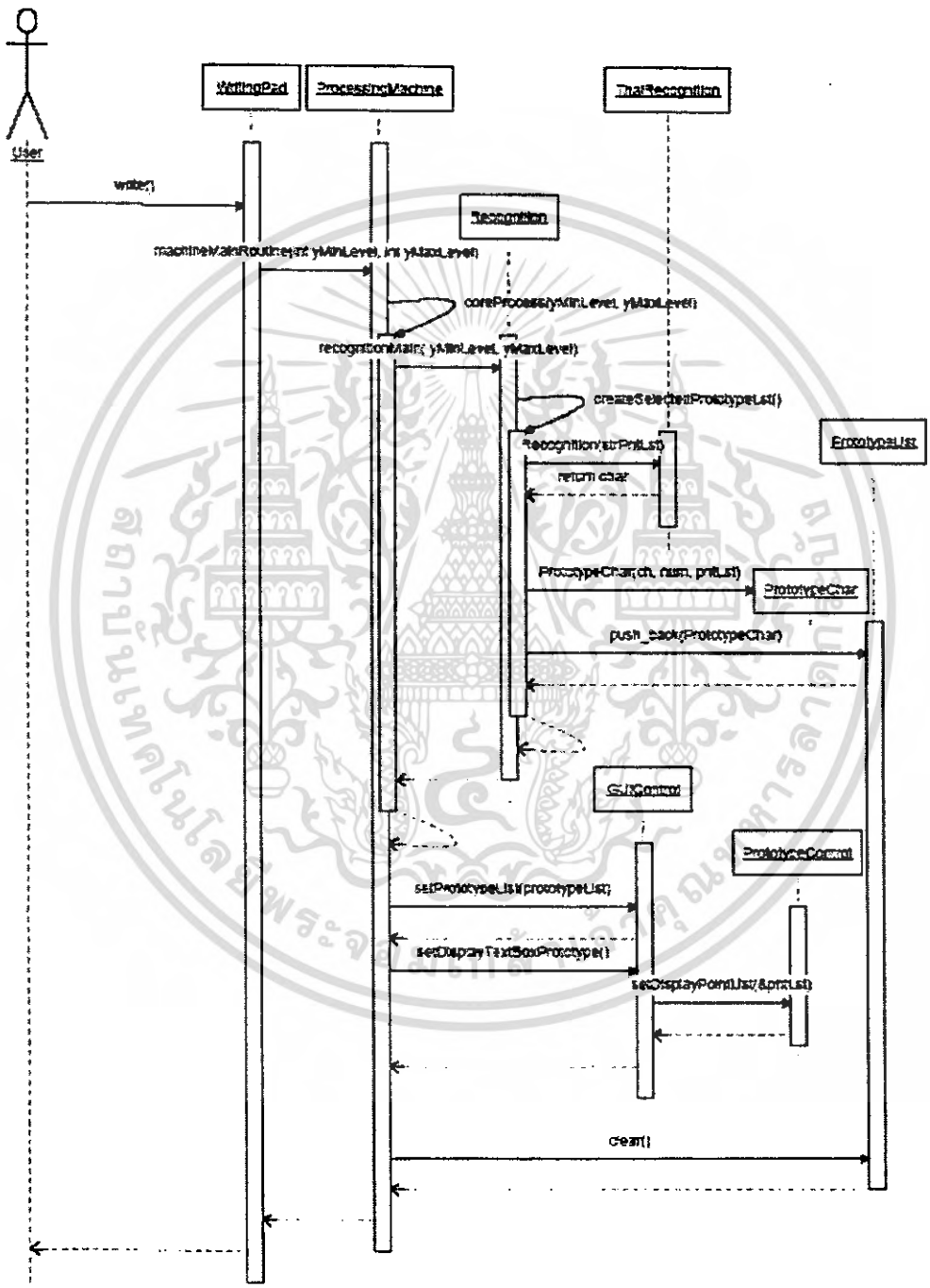
รูปที่ 4.7 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้ต้องการเขียนตัวอักษรใดๆ

4.3.2.6. Write Characters according to scripts:



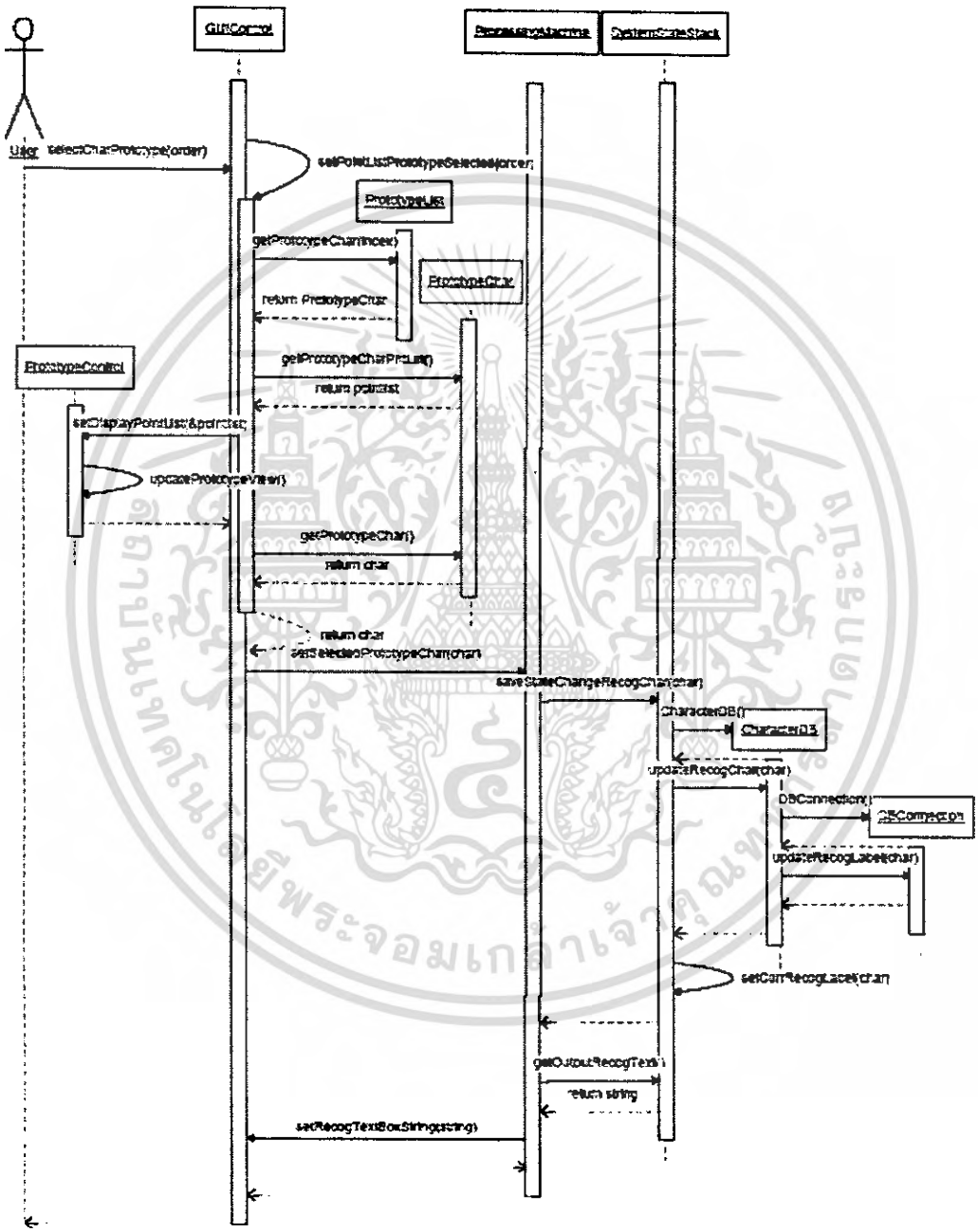
รูปที่ 4.8 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้ต้องการเขียนตัวอักษรตามสคริปต์

4.3.2.7. View a set of output characters



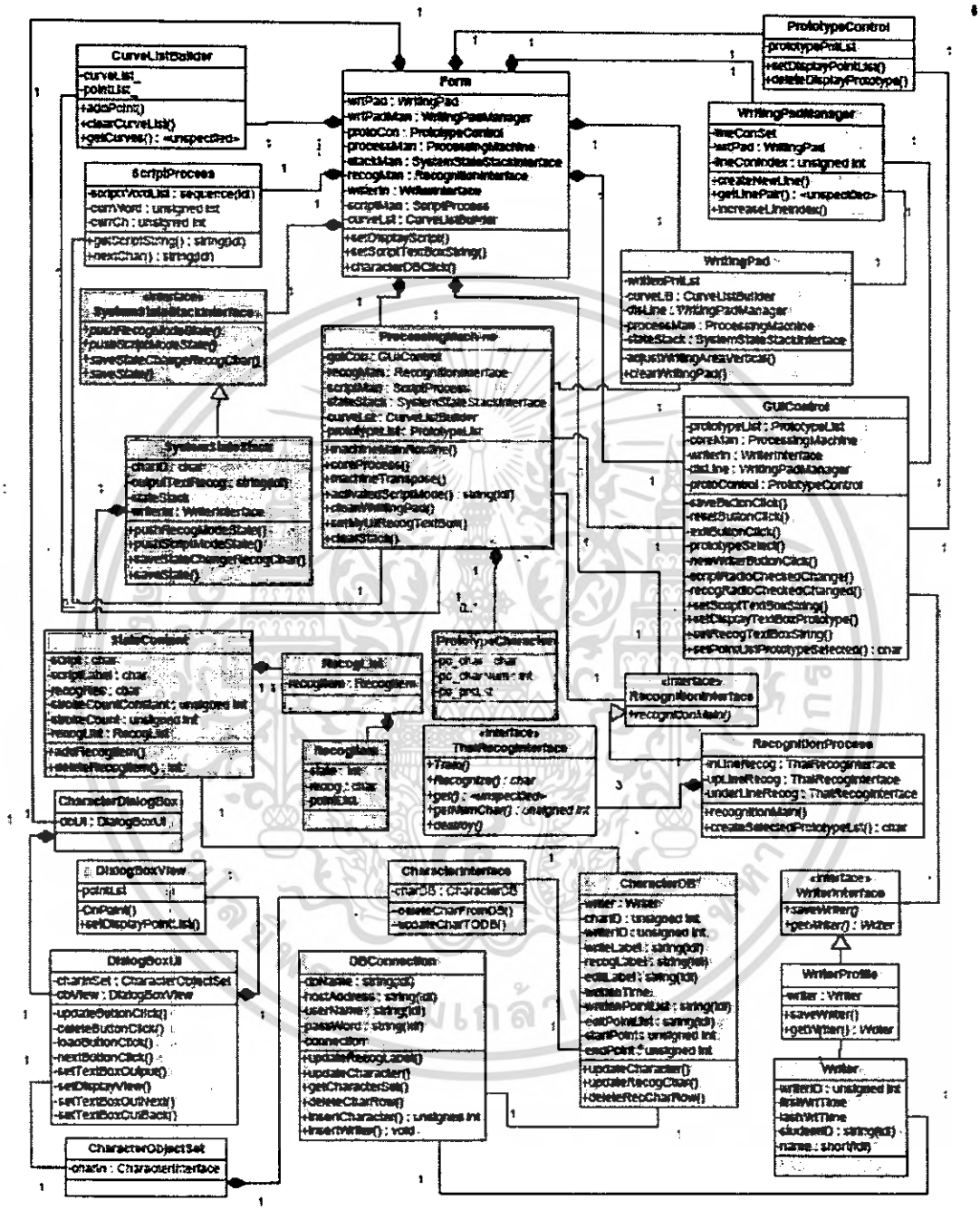
รูปที่ 4.9 แสดง Sequence Diagram ของระบบซึ่งแสดงผลตัวอักษรตัวพิมพ์ที่ใกล้เคียงกับลายมือเขียนมากที่สุด 5 อันดับแรก

4.3.2.8. Select another output character



รูปที่ 4.10 แสดง Sequence Diagram ของระบบเมื่อผู้ใช้เลือกตัวอักษรตัวพิมพ์ตัวอื่นๆ

4.3.3 การออกแบบ Class Diagram ของระบบ



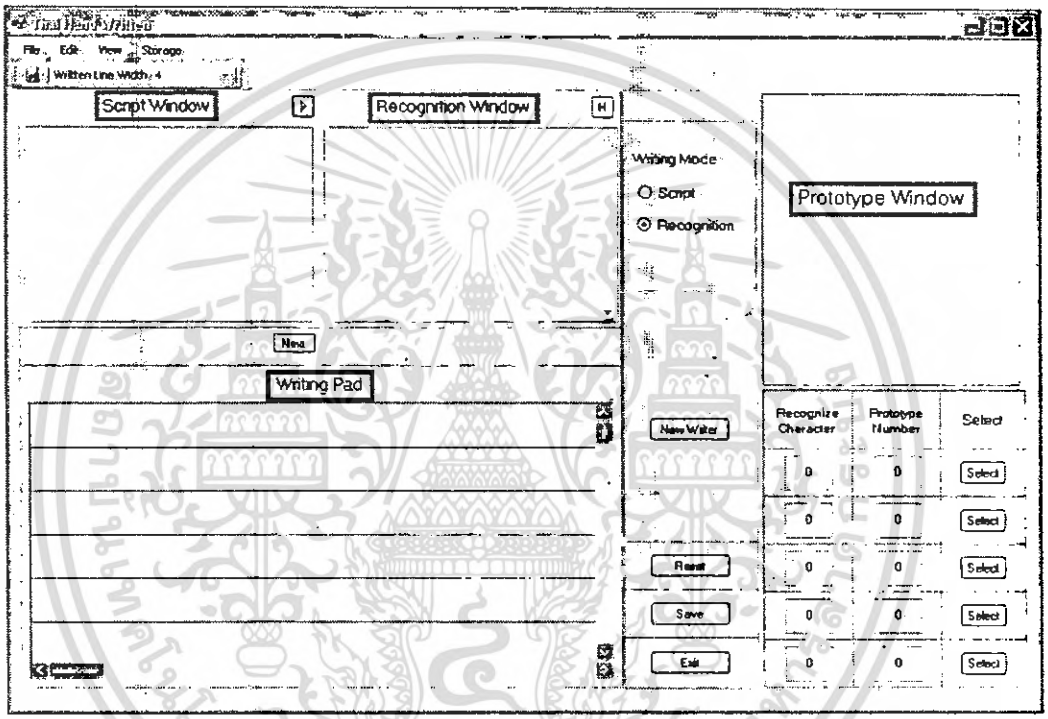
รูปที่ 4.11 แสดง Class Diagram ของระบบ

4.4 การออกแบบหน้าจอส่วนติดต่อกับผู้ใช้งานระบบ

เนื่องจากระบบอ่านลายมือเขียนประกอบด้วยการติดต่อกับผู้ใช้งานเป็น 3รูปแบบหลัก ดังนี้

1. การเขียนตัวอักษรตามที่ระบบแสดง(เขียนตาม script)
2. การเขียนตัวอักษรใดๆ
3. การดูข้อมูลในฐานข้อมูล

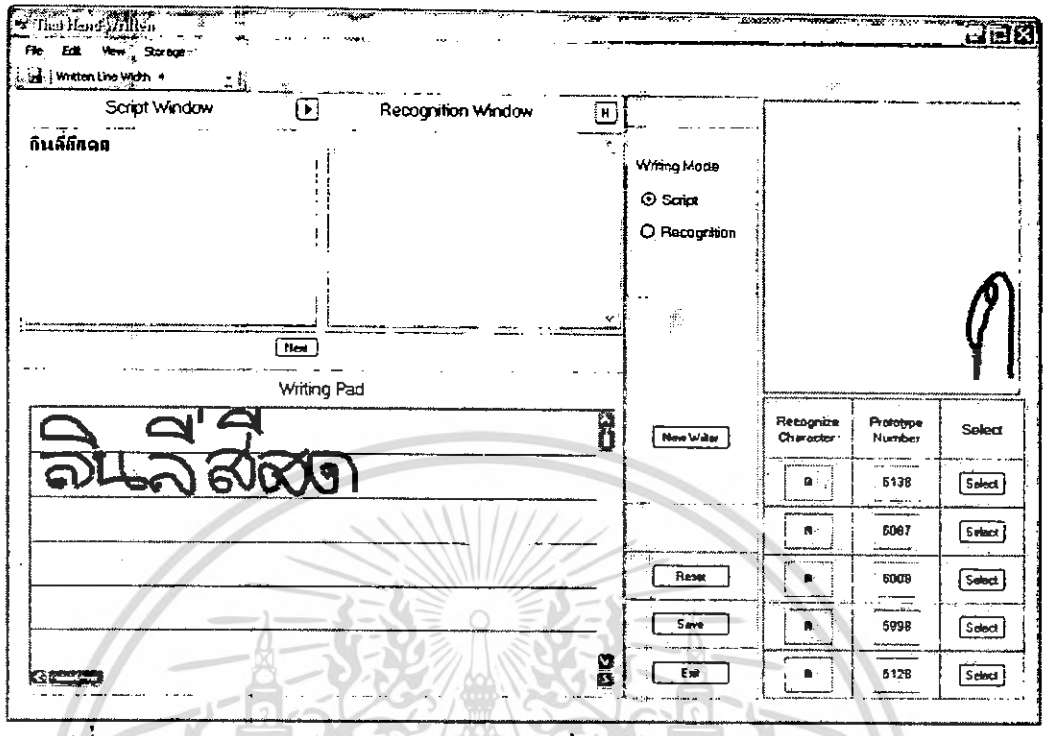
หน้าจอติดต่อกับผู้ใช้งานระบบจึงถูกออกแบบให้มีลักษณะดังนี้



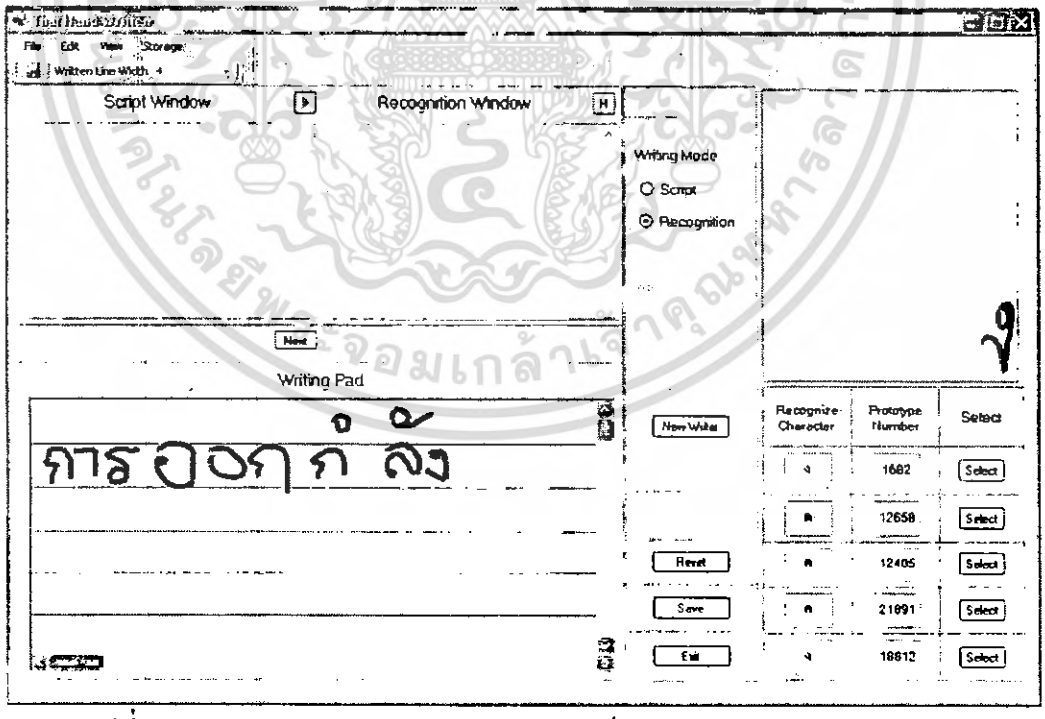
รูปที่ 4.12 แสดง User Interface หลักของระบบ

ในหน้าต่างประกอบด้วยหน้าต่างย่อยซึ่งทำหน้าที่ต่างๆกันดังนี้

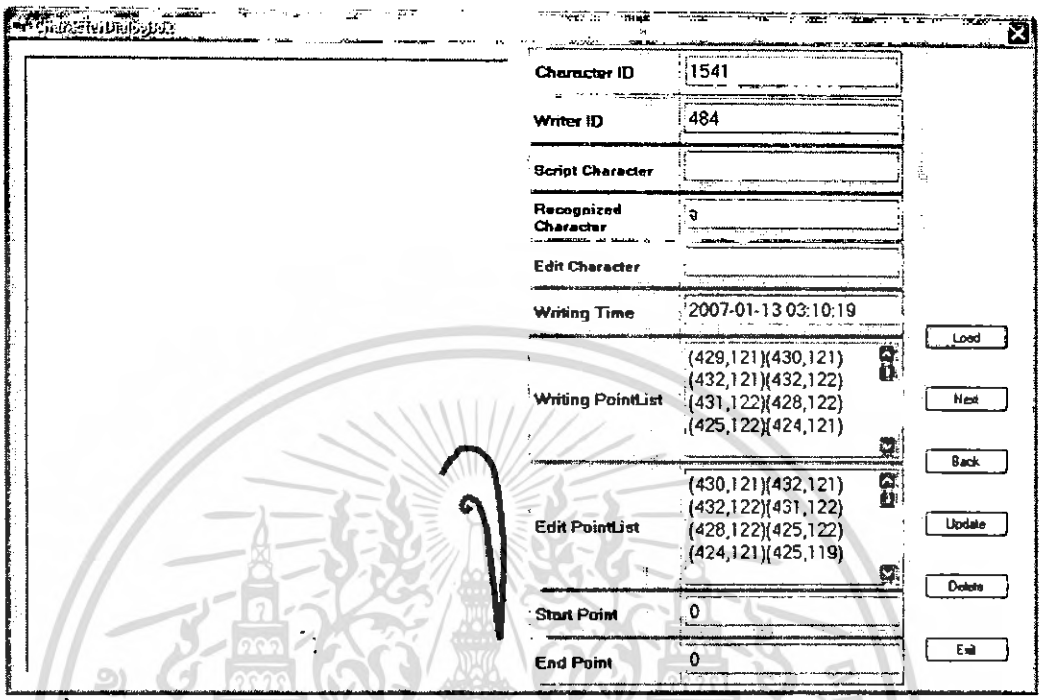
1. Script Window แสดงตัวอักษรที่ต้องการให้ผู้ใช้งานระบบเขียนตาม
2. Recognition Window แสดงตัวอักษรตัวพิมพ์ซึ่งเป็นเอาพุทของระบบ
3. Prototype Window แสดงตัวอักษรลายมือเขียนที่ใช้เป็นตัวอักษรตัวอย่างให้ระบบ



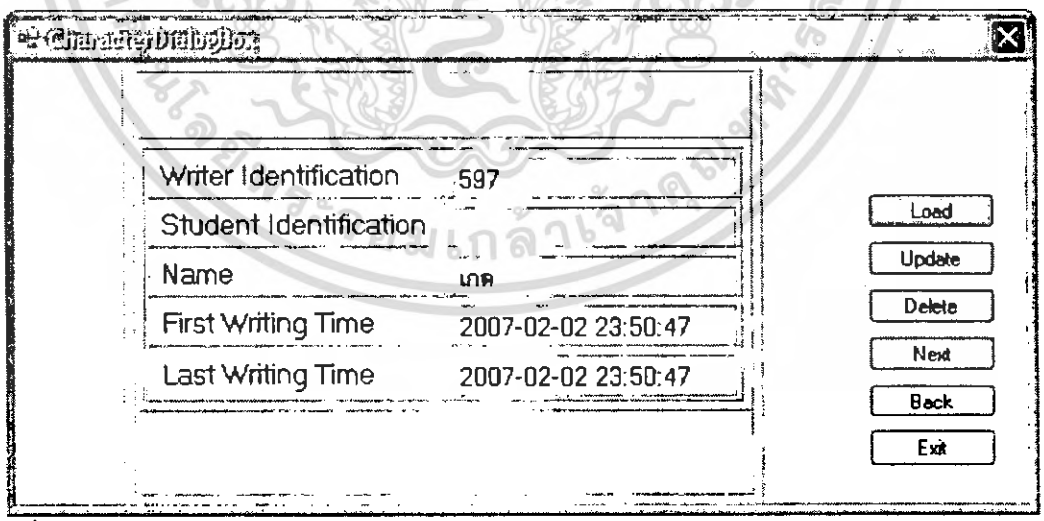
รูปที่ 4.13 แสดง User Interface หลักของระบบเมื่อผู้ใช้เขียนตัวอักษรตามสคริปต์



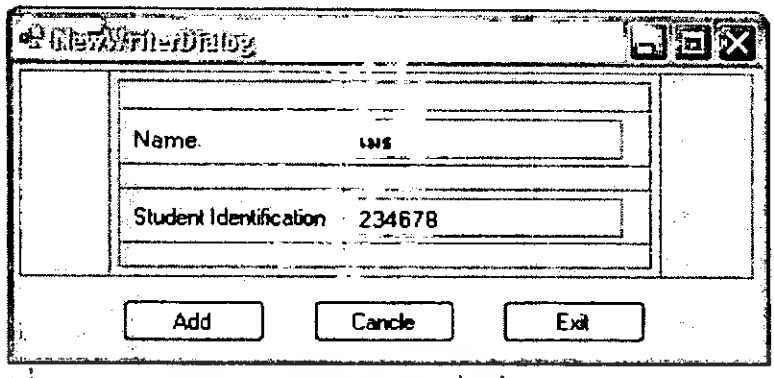
รูปที่ 4.14 แสดง User Interface หลักของระบบเมื่อผู้ใช้เขียนตัวอักษรใดๆ



รูปที่ 4.15 แสดง User Interface ของระบบเมื่อผู้ใช้เรียกดูข้อมูลตัวอักษรจากฐานข้อมูล



รูปที่ 4.16 แสดง User Interface ของระบบเมื่อผู้ใช้เรียกดูข้อมูลผู้ใช้งานระบบจากฐานข้อมูล



รูปที่ 4.17 แสดง User Interface ของระบบเมื่อเพิ่มผู้ใช้ระบบเข้าสู่ฐานข้อมูล

4.5 การพัฒนาระบบอ่านลายมือเขียนภาษาไทย

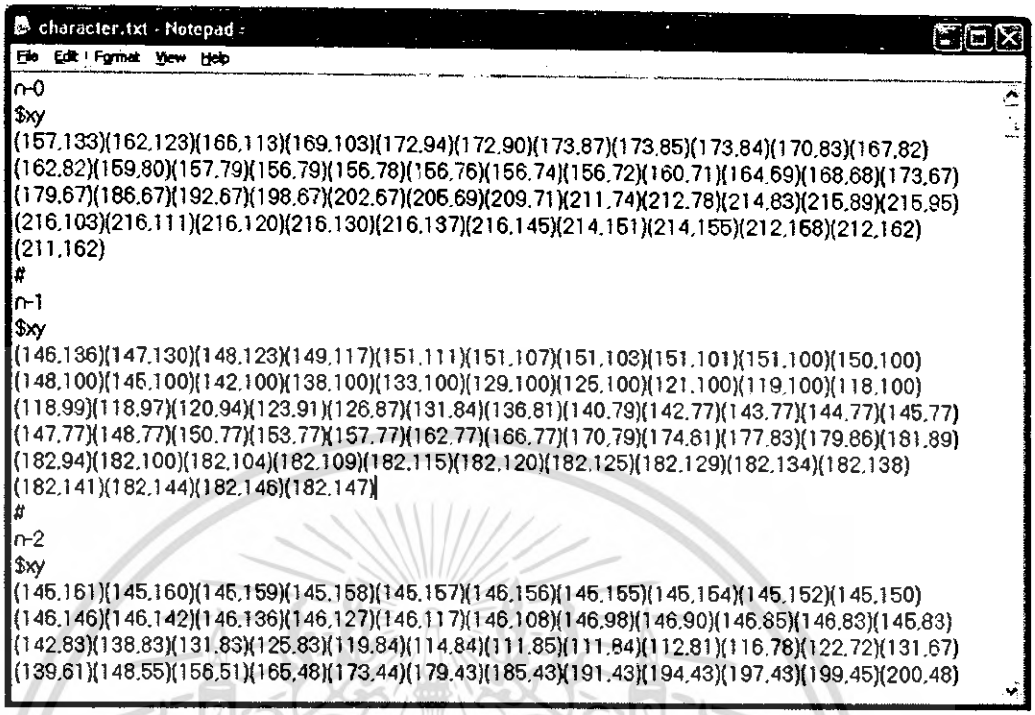
เนื่องจากการพัฒนาใช้เครื่องมือ .Net Framework โดยใช้ภาษา CLI ซึ่งทำให้สามารถเรียกใช้ library ได้ทั้งของ C++ และ C# ทำให้ส่วน user interface พัฒนาได้อย่างรวดเร็ว แต่ C++ และ C# จัดการกับหน่วยความจำต่างกัน ข้อมูลของ C++ จะถูกเก็บไว้ในหน่วยความจำที่เรียกว่า unmanaged memory ซึ่งก็คือระบบจะไม่จัดการคืน memory ให้นอกจากผู้ใช้ภาษาจะต้องรับผิดชอบการคืน memory เอง ต่างจากภาษา C# ซึ่งข้อมูลของ C# จะถูกเก็บไว้ในหน่วยความจำที่เรียกว่า managed memory ที่ผู้ใช้ภาษาไม่รับผิดชอบการคืนหน่วยความจำเพราะระบบจะจัดการคืนหน่วยความจำให้เมื่อข้อมูลดังกล่าวถูกเลิกใช้

การจัดการกับหน่วยความจำต่างกันดังกล่าวของ C++ และ C# ทำให้เกิดปัญหาในการอ้างอิงถึงข้อมูลระหว่างกัน เพราะข้อมูลถูกเก็บไว้ในหน่วยความจำที่ต่างกันและมองไม่เห็นกัน ทำให้ต้องศึกษาเพิ่มเติมในส่วนนี้ค่อนข้างมาก

นอกจากนี้ ภาษาทั้งสอง C++ และ C# ก็ใช้รหัสตัวอักษรที่ต่างกันโดย C++ ใช้อักษร ASCII ในขณะที่ C# ใช้อักษร UNICODE จึงต้องมีการแปลงรหัสอักษรไปมาระหว่างการประมวลผลโปรแกรม นั้นคือ ส่วนที่ติดต่อกับผู้ใช้งานระบบใช้รหัส UNICODE ขณะที่ส่วน business process และ storage ใช้อักษร ASCII

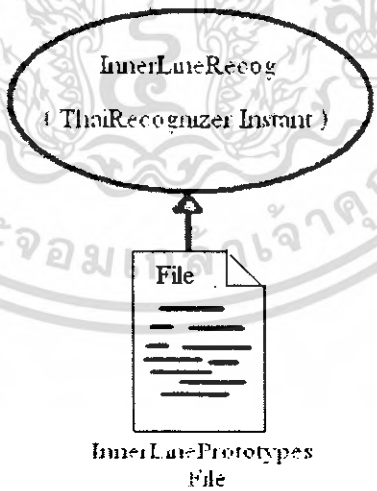
4.5.1 การพัฒนาโปรแกรมเพื่อจัดการกับลายมือเขียนที่บรรทัดต่างๆ

จัดการกับลายมือเขียนที่บรรทัดต่างๆ โดยสร้างไฟล์เก็บตัวอักษรตัวอย่างที่ตำแหน่งกัน (โปรโตไทป์ของตัวอักษรที่ตำแหน่งกัน) ไว้ต่างไฟล์กัน และสร้าง instant ของ class ThaiRecognizer สาม instant โหลดข้อมูลตัวอักษรตัวอย่างจากไฟล์ที่ต่างกันดังกล่าว เพื่อจัดการกับลายมือเขียนที่ตำแหน่งต่างๆ



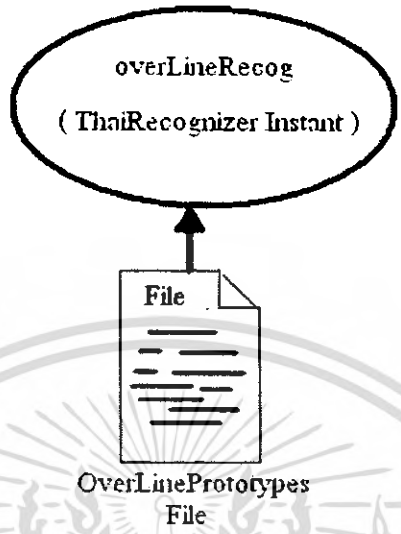
รูปที่ 4.18 แสดงไฟล์สำหรับเก็บตัวอักษรตัวอย่าง (ตัวอักษรprototype)

1. ตัวอักษรที่อยู่ในบรรทัด สร้าง innerLineRecog instant : โหลดตัวอักษรจากไฟล์ตัวอักษรตัวอย่างในบรรทัด (โปรโตไทป์ของอักษรในบรรทัด)



รูปที่ 4.19 แสดงการโหลดตัวอักษรตัวอย่าง (ตัวอักษรprototype) จากไฟล์สำหรับเก็บตัวอักษรตัวอย่างในบรรทัด

2. ตัวอักษรบนบรรทัด สร้าง overLineRecog instant : : โหลดตัวอักษรจากไฟล์ตัวอักษรตัวอย่างบนบรรทัด (โปรโตไทป์ของอักษรบนบรรทัด)

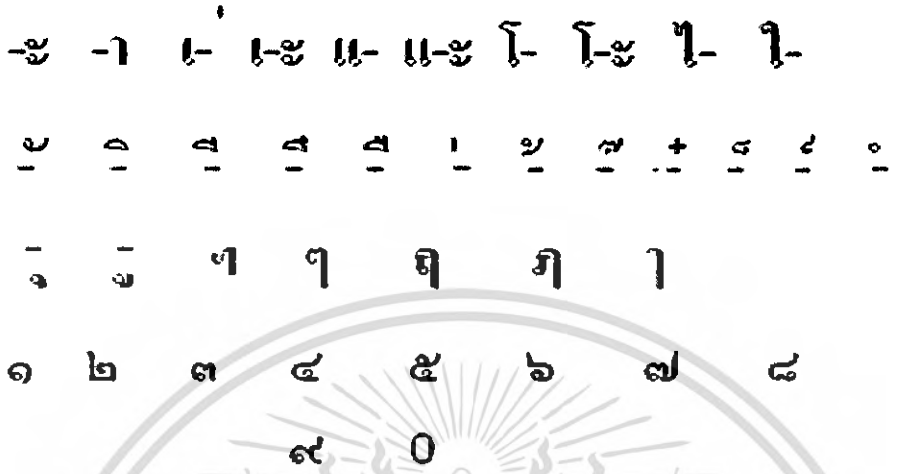


รูปที่ 4.20 แสดงการโหลดตัวอักษรตัวอย่าง (ตัวอักษรprototype) จากไฟล์สำหรับเก็บตัวอักษรตัวอย่างเหนือเส้นบรรทัด

3. ตัวอักษรใต้บรรทัด สร้าง underLineRecog instant : : โหลดตัวอักษรจากไฟล์ตัวอักษรตัวอย่างใต้บรรทัด (โปรโตไทป์ของอักษรใต้บรรทัด)

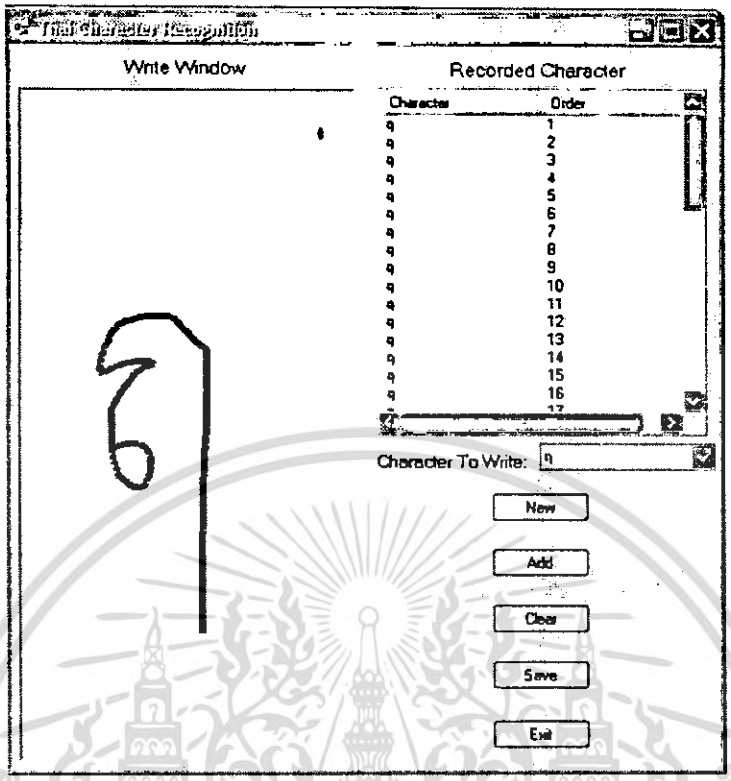
รูปที่ 4.21 แสดงการโหลดตัวอักษรตัวอย่าง (ตัวอักษรprototype) จากไฟล์สำหรับเก็บตัวอักษรตัวอย่างใต้เส้นบรรทัด

4.5.2 การพัฒนาโปรแกรมเพื่อเก็บตัวอักษรลายมือเขียนตัวอย่าง (โปรโตไทป์) เพิ่มเติม

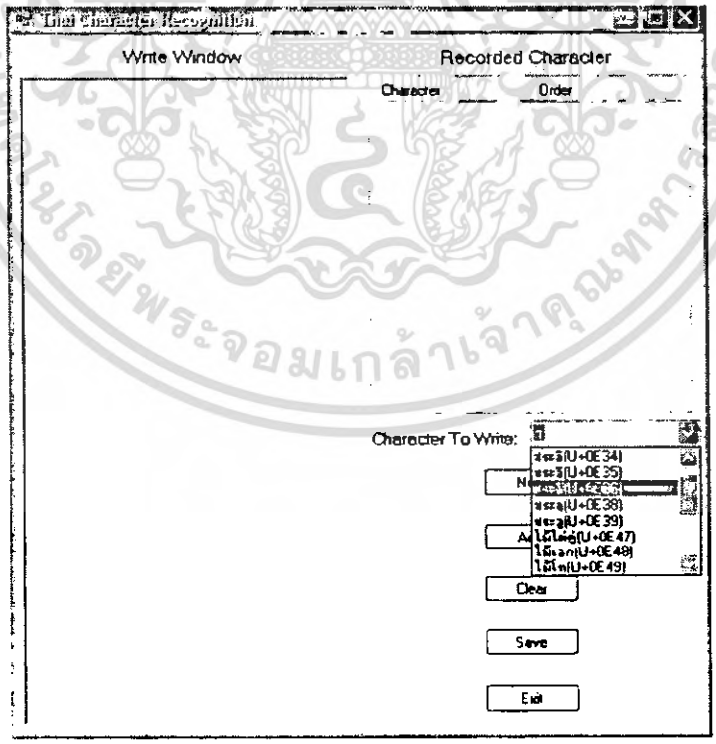


รูปที่ 4.22 แสดงตัวอักษรตัวอย่าง (ตัวอักษรprototype) ที่เก็บเพิ่มเติม

แอพพลิเคชั่นหรือโปรแกรมที่พัฒนาขึ้นเพื่อเก็บตัวอักษรลายมือเขียนตัวอย่าง (โปรโตไทป์) เพิ่มเติมมีลักษณะดังนี้



รูปที่ 4.23 แสดงโปรแกรมรวบรวมตัวอักษรลายมือเขียนตัวอย่างในบรรทัดเพิ่มเติม



รูปที่ 4.24 แสดงโปรแกรมรวบรวมตัวอักษรลายมือเขียนตัวอย่างบนและใต้บรรทัดเพิ่มเติม

4.5.3 พัฒนาฐานข้อมูล สำหรับเก็บรวบรวมข้อมูลลายมือเขียน โดยแบ่งออกเป็น 2 ตารางซึ่งประกอบด้วยคอลัมน์ต่างๆดังนี้

4.5.3.1 ตาราง CharacterDB

CHARACTER_ID: รหัสประจำตัวอักษร

WRITER_ID: รหัสประจำตัวผู้เขียนตัวอักษร

WRITE_LABEL: ตัวอักษรที่ผู้ใช้งานโปรแกรมต้องการจะเขียนหรือตัวอักษรที่กำหนดใน Script Window

RECOG_LABEL: ตัวอักษรตัวพิมพ์เอาพุทที่ได้จากการรู้จำตัวอักษร

EDIT_LABEL: ตัวอักษรตัวพิมพ์เอาพุทจริงควรจะเป็น

WRITTEN_TIME: เวลาที่เขียนตัวอักษร

WRITTEN_POINTLIST: เซตของพิกัด (x, y) ในการเขียนตัวอักษร

EDIT_POINTLIST: เซตของพิกัด (x, y) ในการเขียนตัวอักษรที่ได้รับการแก้ไขแล้ว

START_POINT: จุดเริ่มต้นการเขียนเป็น index ชี้ไปยังพิกัดต่างๆในเซต

END_POINT: จุดเสิ้นสุดการเขียนเป็น index ชี้ไปยังพิกัดต่างๆในเซต

4.5.3.2 ตาราง Writer

WRITER_ID: รหัสประจำตัวผู้เขียนตัวอักษร

FIRST_WRITTEN_TIME: เวลาที่ผู้เขียนเริ่มเขียนตัวอักษรตัวแรก

LAST_WRITTEN_TIME: เวลาที่ผู้เขียนเขียนตัวอักษรตัวสุดท้าย

STUDENT_ID: รหัสนักศึกษา

NAME: ชื่อ

←T→	WRITER_ID	FIRST_WRITTEN_TIME	LAST_WRITTEN_TIME	NAME
☐ ✎ ✕	1	2006-09-17 21:21:25	2006-09-17 21:24:58	anonymous1
☐ ✎ ✕	2	2006-09-17 21:25:14	2006-09-17 21:30:37	anonymous2
☐ ✎ ✕	3	2006-09-17 21:31:32	2006-09-17 21:32:11	anonymous3
☐ ✎ ✕	4	2006-09-17 21:32:32	2006-09-17 21:33:15	anonymous4
☐ ✎ ✕	5	2006-09-17 21:33:24	2006-09-17 21:34:11	anonymous5
☐ ✎ ✕	6	2006-09-17 21:34:16	2006-09-17 21:34:51	anonymous6

รูปที่ 4.25 แสดงตารางเก็บข้อมูลเกี่ยวผู้ใช้งานระบบ

CHARACTER_ID	WRITER_ID	WRITTEN_TIME	WRITTEN_POINTLIST
1	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
2	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
3	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
4	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
5	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
6	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
7	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
8	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
9	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
10	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
11	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
12	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
13	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
14	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
15	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
16	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
17	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
18	1	2006-09-17 21:24:41	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
19	1	2006-09-17 21:24:42	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
20	1	2006-09-17 21:24:42	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
21	1	2006-09-17 21:24:42	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
22	1	2006-09-17 21:24:42	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...
23	1	2006-09-17 21:24:42	(93,264)(93,261)(94,257)(96,254)(98,250)(101,248)(...

รูปที่ 4.26 แสดงตารางเก็บข้อมูลลายมือเขียน

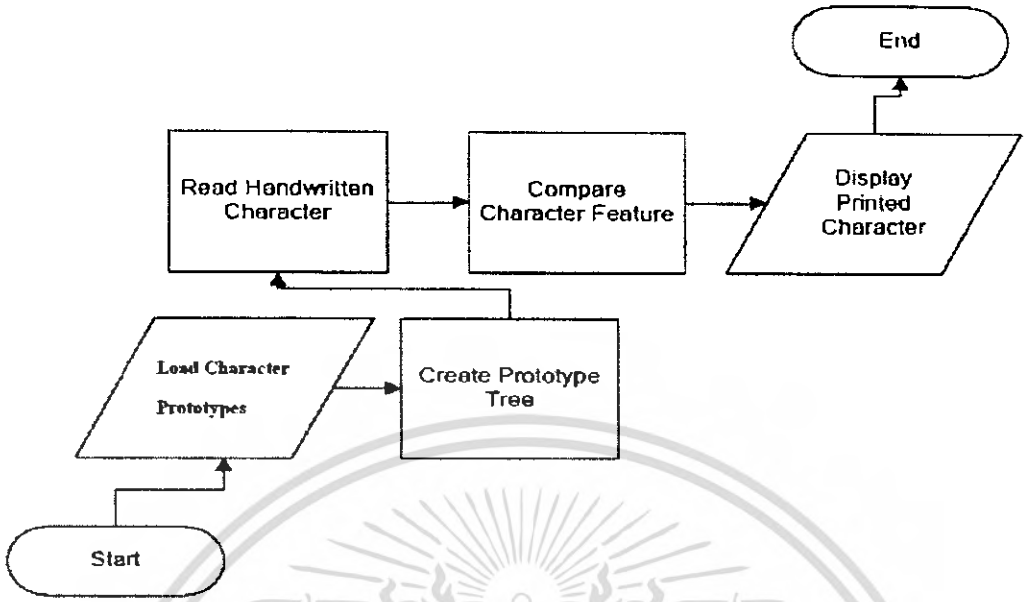
4.5.4 ขั้นตอนการประมวลผลของโปรแกรม

1. โหลดตัวอักษรตัวอย่างขึ้นมาจากไฟล์ (โปรโตไทป์ของตัวอักษร)

2. สร้าง tree ซึ่งเป็น data structure สำหรับเก็บข้อมูลคุณลักษณะเด่นของตัวอักษรแต่ละตัวที่โหลดมาจากไฟล์

3. นำลักษณะเด่นของตัวอักษรอินพุท เปรียบเทียบกับลักษณะเด่นของตัวอักษรตัวอย่าง (โปรโตไทป์ของตัวอักษร)ทั้งหลาย เพื่อหาตัวอักษรที่มีลักษณะเด่นใกล้เคียงกัน

(ทฤษฎีและกลไกการทำงานของระบบส่วนนี้ อาจารย์คำเพ็ด บุญนะดี เป็นผู้ศึกษาและพัฒนา)



รูปที่ 4.27 แสดงขั้นตอนการประมวลผลของโปรแกรม

บทที่ 5
บทสรุป

ทฤษฎีที่ใช้ในการรู้จำตัวอักษรซึ่งใช้ในโครงงานเดิม แบ่งแยกความแตกต่างของตัวอักษรแต่ละตัวตามคุณลักษณะเด่นของตัวอักษรโดยพิจารณาจำนวนเซ็กเมนต์ของตัวอักษรและทิศทางการเขียนเซ็กเมนต์แต่ละเซ็กเมนต์ หากตัวอักษรใดที่มีคุณลักษณะดังกล่าวคล้ายคลึงหรือใกล้เคียงกัน จะใช้ค่าความยาวในการลากตามแกน X จากซ้ายไปขวา จากขวามาซ้าย และค่าความยาวในการลากตามแกน y จากบนลงล่าง และจากล่างขึ้นบน คิดเป็นเปอร์เซ็นต์เทียบกับความยาวของเซ็กเมนต์แต่ละเซ็กเมนต์ ในการพิจารณาแยกตัวอักษรเหล่านี้ออกจากกัน ซึ่งตามทฤษฎีหรืออัลกอริธึมนี้ ทำให้สามารถรู้จำเฉพาะตัวอักษร โคด และตัวอักษรที่มีการเปลี่ยนทิศทางการเขียนหรือมีส่วนโค้งเท่านั้น

สำหรับส่วนของโครงงานที่เพิ่มเติมเข้ามานั้นประกอบด้วย การจัดการอักษรที่ไม่เป็นอักษรโคดซึ่งก็คืออักษรที่ต้องเขียนมากกว่า 1 ครั้ง เช่น สระอี สระอือ ไม้จัตวา และ ฐ.ฐาน เป็นต้น เนื่องจากการจัดการกับข้อมูลลักษณะดังกล่าวต้องอาศัยการทดสอบเงื่อนไขหลายเงื่อนไขซึ่งจะทำให้โปรแกรมมีความซับซ้อนสูงและเข้าขาก จึงนำ state machine มาใช้แทนการทดสอบเงื่อนไข (if else) ทำให้ลดความซับซ้อนของโปรแกรมลงไปได้ระดับหนึ่งและ ทำให้สมารถตรวจสอบหาข้อผิดพลาดของโปรแกรมได้สะดวกยิ่งขึ้นอีกด้วย

และเนื่องจากขอบข่ายของโครงงานต้องการจัดการกับตัวอักษรที่ตำแหน่งต่างๆ ของบรรทัด เพื่อลดเวลาในการประมวลผลและลดโอกาสผิดพลาดเนื่องจากการรู้จำตัวอักษรบางตัวที่อยู่คนละตำแหน่งแต่มีลักษณะใกล้เคียงกัน จึงเก็บตัวอักษรตัวอย่างที่ตำแหน่งต่างๆ ของบรรทัดไว้ต่างไฟล์กัน และนำหลักการของ object oriented เข้ามาใช้โดยสร้าง instance ของ class Thairecognition ขึ้นมา 3 ตัว โดยแต่ละตัวโหลดตัวอักษรตัวอย่างจากต่างไฟล์กันตามตำแหน่งที่แต่ละ instance รับผิดชอบ

ระบบที่พัฒนาขึ้นมานี้คาดว่าจะสามารถนำไปใช้ได้จริงและสามารถอำนวยความสะดวกในการทดสอบอัลกอริทึมในการรู้จำลายมือเขียนได้ เก็บข้อมูลลายมือเพิ่มเติม และนำไปใช้ในการพัฒนาต่อไปเพื่อเป็นโปรแกรมที่อำนวยความสะดวกให้ผู้ใช้งานได้ใช้งานอย่างสมบูรณ์ในอนาคต

บรรณานุกรม

[1] Kevin Atkinson(original author), Sinisa Milivojevic,Monty Widenius, Warren Young, "MySQL++ User Manual", [Online] URL: http://tangentsoft.net/mysql++/doc/userman/html/

[2] Kevin Atkinson(original author), Sinisa Milivojevic,Monty Widenius, Warren Young, "MySQL++ Reference Manual 2.2.0", [Online] URL: http://tangentsoft.net/mysql++/doc/refman/html/

[3] คำเพ็ด บุญนะดี "การรู้จำลายมือเขียนตัวอักษรลาวแบบออนไลน์โดยใช้ลักษณะเด่นทางโครงสร้างแบบยืดหยุ่น" วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2546 บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[4] Microsoft, "Microsoft Visual Studio 2005 Document" , 2005 , Microsoft Corporation

[5] Peter Linz, 2000, " An Introduction to Formal Languages and Automata" , Jones And Bartlett Publishers