การประยุกต์ใช้งาน สมาร์ตการ์ดในระบบรักษาความปลอดภัยและอำนวยความสะดวก

SMART CARDS APPLICATION IN SECURITY AND FACILITATE SYSTEM

โดย

นายกิตติศักดิ์ ศรีม่วง 43010030

นายธัญญะ คณาพรพงศ์ 43010182

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การประยุกต์ใช้งาน สมาร์ตการ์ดในระบบรักษาความปลอดภัยและอำนวยความสะดวก

SMART CARDS APPLICATION IN SECURITY AND FACILITATE SYSTEM

โดย

นายกิตติศักดิ์ ศรีม่วง รหัส 43010030

นายธัญญะ คณาพรพงศ์ รหัส 43010182

อาจารย์ที่ปรึกษา

รศ. สุชาติ คุณทวีเทพ

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน สมาร์ตการ์ดในระบบรักษาความปลอดภัยและอำนวยความสะดวก

Smart cards application in security and facilitate system

ผู้จัดทำ

นายกิตติศักดิ์ ศรีม่วง รหัส 43010030

นายธัญญะ คณาพรพงศ์ รหัส 43010182

อาจารย์ที่ปรึกษา

(รศ. สุชาติ คูณทวีเทพ)

การประยุกต์ใช้งาน สมาร์ตการ์ดในระบบรักษาความปลอดภัยและอำนวยความสะดวก

นายกิตติศักดิ์ ศรีม่วง รหัส 43010030
นายธัญญะ คณาพรพงศ์ รหัส 43010182
รศ.สุชาติ คูณทวีเทพ อาจารย์ที่ปรึกษา
ภาคการศึกษาที่ 1 ปีการศึกษา 2546

บทคัดย่อ

ในปัจจุบันเทคโนโลยีทางอิเล็กทรอนิกส์ได้ก้าวไปไกลซึ่งระบบรักษาความปลอดภัยและอำนวยความสะดวกสามารถนำเอาสมาร์ตการ์ดมาประยุกต์ใช้ได้ ในโครงงานนี้ใช้บัตรโทรศัพท์สาธารณะที่ใช้หมดแล้วมาทำเป็นบัตรของระบบรักษาความปลอดภัยและอำนวยความสะดวก บัตรโทรศัพท์สาธารณะเป็นบัตรสมาร์ตการ์ดแบบหน่วยความจำชนิด SLE4436 เครื่องอ่านบัตรใช้ไมโครคอนโทรลเลอร์ MCS-51 โดยจะได้ข้อมูลในส่วนที่เป็นหมายเลขบัตร และควบคุมการส่งข้อมูลจากเครื่องอ่านสมาร์ตการ์ดมายังคอมพิวเตอร์ด้วยการสื่อสารแบบอนุกรมตามมาตรฐาน RS-232 ข้อมูลที่ได้จะนำมาวิเคราะห์ในฐานข้อมูลที่มีอยู่โดยใช้โปรแกรม Microsoft Visual Basic Version 6.0 Enterprise และโปรแกรม Microsoft Access ในการออกแบบและสร้างโปรแกรมที่ทำการวิเคราะห์ฐานข้อมูลดังกล่าวรวมทั้งการแสดงรายละเอียดผลการวิเคราะห์ผ่านจอคอมพิวเตอร์

Smart cards application in security and facilitate system

Mr.Kittisak Srimuang 43010030

Mr.Thanya Kanapompong 43010182

Assoc.Suchart Khuntaweetep

Academic year 2003

Abstract

Electronics technologies have been made progresses every day now we can apply smart cards for security and facilitate system. This project utilizes discarded TOT cards to make security and facilitate system card. TOT card is SLE 4436 memory smart card. Smart card reader was used micro controller MCS-51 to get identity card number and control data transfer from smart card reader to computer by using RS-232 serial port standard .The data analysis is using Microsoft Visual Basic Version 6.0 Enterprise and Microsoft Access data base and show the analysis result detail through computer monitor .

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี ก็ด้วยความเมตตาของ รศ. สุชาติ คูณทวีเทพ ที่ท่านได้ให้คำปรึกษาอย่างเป็นกันเอง เอื้ออำนวยเรื่องอุปกรณ์การทดลอง ห้องทำงานอันสะดวกสบาย

โดยส่วนตัวของผม นาย กิตติศักดิ์ ศรีม่วง ขออุทิศ การทำงานทั้งหมดของผมให้ คุณย่าของผม คุณย่า ละออง ศรีม่วง ผู้ที่เลี้ยงผมมา น้องวิ นาย วิทิต ศรีม่วง น้องชายอันเป็นที่รักผู้ล่วงลับ ผมหวังที่จะเห็นปริญญานิพนธ์ฉบับเต็มของเค้า หวังที่จะเห็นเค้าประสบความสำเร็จในชีวิตแต่ผมไม่มีโอกาสได้เห็น หวังเพียงให้มารับรู้ในสิ่งที่ผมตั้งใจทำมากที่สุดสิ่งหนึ่งในชีวิต และขอขอบคุณ พ่อ สุธัมม์ ศรีม่วง และ คุณ แม่ นารีรัตน์ ศรีม่วง ที่ ให้การสนับสนุนกำลังทรัพย์ที่ไม่จำกัดอีกทั้งยัง คอยเป็นกำลังใจให้ยามอ่อนล้าจนปริญญานิพนธ์ ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

สารบัญรูป

หน้า

รูปที่ 6.21 รูปโปรแกรมส่วนห้องพักเมื่อทำการเสียบบัตร

สารบัญตาราง

หน้า

บทที่ 1 บทนำ

1.1 ความเป็นมาของโครงงาน

เนื่องจากในปัจจุบัน การพัฒนาในด้านเทคโนโลยีการเก็บข้อมูล และการเชื่อมต่อข้อมูลมีความก้าวหน้าไปอย่างมาก และเป็นที่น่าสนใจในการจะนำมาประยุกต์ใช้ในชีวิตประจำวันให้มากขึ้น โดยเฉพาะอย่างยิ่งเทคโนโลยีการเก็บข้อมูลแบบหนึ่งที่กำลังเป็นที่สนใจเป็นอย่างมากในปัจจุบันคือ สมาร์ตการ์ด ซึ่งเริ่มจะมีการใช้มากขึ้นในปัจจุบัน จากเหตุผลนี้จึงเป็นที่น่าสนใจในการที่จะศึกษาและนำมาประยุกต์ใช้สมาร์ตการ์ดให้มีบทบาทในชีวิตประจำวันมากขึ้น โดยในโครงงานนี้ได้มุ่งเน้นการประยุกต์ไปในงานด้านระบบความปลอดภัยและในอำนวยความสะดวกให้แก่ผู้ใช้ ซึ่งทั้ง2ด้านนี้เป็นก็ถือมีความสำคัญกับชีวิตประจำวันอย่างมาก สำหรับสมาร์ตการ์ดที่ใช้ในโครงงานนี้ได้ประยุกต์มาจากบัตรโทรศัพท์ TOT Card ที่ไม่สามารถใช้ชำระค่าโทรศัพท์ได้แล้ว ซึ่งบัตร TOT Card นี้ก็เป็นบัตรสมาร์ตการ์ดชนิดหนึ่งที่เราหาได้ง่ายและถือว่าเป็นการนำของที่หมดค่าแล้วมาประยุกต์ให้เกิดประโยชน์อีกทางหนึ่ง

1.2 วัตถุประสงค์ของโครงงาน

1. ศึกษาโครงสร้างและการอินเตอร์เฟสกับบัตรสมาร์ตการ์ด โดยเฉพาะอย่างยิ่งบัตร TOT Card
2. ศึกษาการสื่อสารแบบอนุกรมมาตรฐาน RS-232 เพื่อส่งข้อมูลที่ได้จากบัตรสมาร์ตการ์ดสู่คอมพิวเตอร์เพื่อวิเคราะห์ผลต่อไป
3. ศึกษาการเขียนโปรแกรมประยุกต์สำหรับงานด้านการจัดการฐานข้อมูลและการแสดงผล

1.3 ขอบเขตของโครงงาน

ทำการสร้างเครื่องอ่านข้อมูลจากบัตร TOT Card และวงจรที่ช่วยจัดการการส่งข้อมูลที่ได้จากเครื่องอ่านข้อมูลแต่ละเครื่องเข้าสู่คอมพิวเตอร์ ในส่วนของโปรแกรมประยุกต์ที่จัดการด้านฐานข้อมูลและแสดงผลจะสามารถแสดงข้อมูลต่างๆของเจ้าของบัตรแต่ละใบที่ทำการลงทะเบียนไว้กับระบบและทำการฟ้องเมื่อมีการนำบัตรที่ไม่ได้ลงทะเบียนกับระบบมาใช้งาน

1.4 วิธีการดำเนินงาน

1. ศึกษาการอินเตอร์เฟสเพื่ออ่านข้อมูลภายในบัตร TOT Card พร้อมออกแบบและสร้างเป็นเครื่องอ่านข้อมูลจากบัตร TOT Card
2. ศึกษาการเชื่อมโยงข้อมูลระหว่างเครื่องอ่านข้อมูลบัตรTOT Card กับคอมพิวเตอร์

3. เขียนโปรแกรมประยุกต์ที่มีความสามารถในการรับข้อมูลจากเครื่องอ่านบัตร TOT Card พร้อมทั้งนำข้อมูลที่ได้มาใช้ในการค้นหาข้อมูลที่สอดคล้องกันในฐานข้อมูลที่มีอยู่ และแสดงผลข้อมูลที่สอดคล้องในฐานข้อมูลบนจอคอมพิวเตอร์

ประโยชน์ที่ได้รับ

1. เพิ่มประสิทธิภาพความปลอดภัย
2. สร้างความสะดวกสบายแก่ผู้ใช้ในการทำกิจกรรมต่างๆในชีวิตประจำวันมากขึ้น
3. นำของที่หมดค่ามาประยุกต์ใช้ให้เกิดประโยชน์มากขึ้น

บทที่ 2 สมาร์ตการ์ด (Smart card)

2.1 ความหมายของสมาร์ตการ์ด

Smart Card หมายถึง การ์ดที่มีหน่วยความจำ หรือ มี Microprocessor ฝังอยู่ในการ์ดอาจจะเป็น Chip หน่วยความจำชนิดที่ถูกโปรแกรมเรียบร้อยแล้วมาจากโรงงาน แต่ถ้าเป็นแบบ Microprocessor นั้นก็จะสามารถเพิ่มข้อมูล หรือลบข้อมูล หรือไม่เช่นนั้นก็จะสามารถปรับปรุงเปลี่ยนแปลง ข้อมูลบนตัวการ์ดนี้ อีกประเภทคือ การ์ดที่มีหน่วยความจำคงที่ หรือที่เรียกว่า memory-chip card เช่น การ์ดโทรศัพท์ เป็นต้น ลักษณะตัวการ์ด Smart card นั้นจะเป็นแผ่นพลาสติก ขนาดเท่ากับ บัตรเครดิต หรือ ขนาดใกล้เคียงกับนามบัตร ซึ่งเป็นขนาดมาตรฐานทั่วโลก ภายในการ์ดนี้ก็จะมีเนื้อที่ส่วนที่เป็นหน่วยความจำอัดอยู่บนการ์ด ซึ่งในส่วนนี้นี่เองที่จะเป็นส่วนบรรจุข้อมูลอยู่ภายใน หากจะเพิ่มเติมข้อมูล หรือ อ่านข้อมูลจากบัตรก็จะต้องมีเครื่องอ่านบัตร ที่เรียกว่า Smart Card Reader

Smart Card นั้นจะไม่เหมือนกับการ์ดชนิดใช้แถบแม่เหล็ก เพราะ Smart Card นั้นจะมีข้อมูลที่จำเป็น และข้อมูลข่าวสารอื่นๆอยู่บนการ์ด ดังนั้นการอ่านข้อมูลจึงไม่ต้องย้อนกลับไปค้นข้อมูลจากศูนย์ข้อมูลอันเป็นการเสียเวลาเช่นเดียวกับการ์ดแถบแม่เหล็ก(เช่น บัตรเอทีเอ็ม) นี่ก็จะเป็นข้อดีของ Smart Card

2.2 ประวัติความเป็นมาของสมาร์ตการ์ด

Smart Card นั้นประดิษฐ์ขึ้น ในปี ค.ศ. 1974 จวบกระทั่งทุกวันนี้ มี Smart Card ใช้อยู่ทั่วโลกราว 1000 ล้านใบ(ชิ้น) จากหลายๆผู้ผลิต 95 เปอร์เซ็นต์ ใช้ในประเทศในแถบยุโรป, อเมริกาใต้, และประเทศในแถบเอเชีย คาดว่าสิ้นปีนี้อาจจะใช้งานถึง 3000 ล้านใบ(ชิ้น) กว่า 15 เปอร์เซ็นต์ก็จะใช้งานในประเทศสหรัฐอเมริกาและคานาดา ในจำนวนนี้ประมาณว่า จำนวน 900 ล้านใบ(ชิ้น) จะใช้ในกิจการทางด้านการให้บริการการเงิน การ์ดธนาคาร หรือ การ์ด ATM, การ์ดด้านรักษาความปลอดภัย, การ์ดที่เกี่ยวกับโทรศัพท์ไร้สาย(มือถือ), การ์ดที่เกี่ยวกับการสื่อสารดาวเทียม, และการ์ดที่เกี่ยวกับการให้บริการเคเบิลทีวี TV Set-Top Boxes เป็นต้น

ปัจจุบันในบางประเทศใกล้บ้านเรานี้ ปัจจุบันเขาก็นำเอามาใช้เต็มรูปแบบแล้ว บ้านเราก็คงจะต้องอีกระยะหนึ่ง หรืออีกไม่นานท่านก็จะคุ้นหูกับศัพท์อีกคำหนึ่งก็คือคำว่า E-Money ที่อาจจะต้องมีการนำเอา Smart Card มาใช้งานด้วยอย่างแน่นอน ปัจจุบันราคาเครื่องผลิต Smart Card ราคายังสูงมาก นัยว่าเป็น 10 ล้าน ต่อเครื่อง ถ้าได้ใช้งานอย่างคุ้มค่า ก็ต้องให้บริการแก่หน่วยงานใหญ่ๆที่ส่วนตัวหน้าตาเป็นอย่างไร มีตำหนิรูปพรรณที่ไหน บ้านอยู่ไหน เป็นลูกใคร กรุ๊ปเลือดใด เคยไปบริจาคเลือดที่ไหน กี่ครั้ง รวมไปถึงประวัติ การเรียน การทำงาน ความสามารถพิเศษ ได้กระทำความดี หรือ มีประวัติ

เสียส่วนใด เช่น อุบัติเหตุ ก่อคดีวิวาท เป็นโจทย์ เป็นจำเลย คดีแพ่ง คดีอาญา จ่ายภาษี จ่ายประกันสังคม มีรถยนต์กี่คัน มีจำเป็นที่จะต้องใช้งานจำนวนมากก็อาจจะคุ้ม นั่นก็ต้องย่อมขึ้นอยู่กับจำนวนการผลิตเท่านั้น ว่าไปแล้วอีกไม่นานนักหากการจัดระบบการรักษาความปลอดภัย ระบบการจัดเก็บ ระบบการเช็คชื่อไม่ว่าจะเป็นบริษัท ห้างร้าน โรงงาน หรือ ตามโรงเรียนต่างๆ ในอนาคตย่อมจะหลีกเลี่ยงการนำเอา Smart Card มาประยุกต์ใช้อย่างแน่นอน รวมไปถึงระบบการรักษาความปลอดภัยของประชาชน เช่นบัตรประจำตัวประชาชนก็อาจจะเปลี่ยนเป็นแบบ Smart Card มีข้อมูลบ้านกี่หลัง มีลูกกี่คน มีทรัพย์สมบัติอะไรบ้าง ตั้งแต่อดีตจนกระทั่งปัจจุบัน ก็จะถูกบันทึกอยู่ในแผ่นการ์ดนี้แผ่นเดียว เข้าโรงพยาบาลเข้าเครื่องอ่านบัตรก็รู้ทันทีว่าแพ้ยาอะไร ใครจะเป็นคนจ่ายเงินค่ารักษาพยาบาล แพทย์ก็ทำการรักษาอย่างถูกต้องทันต่อการช่วยชีวิตทันทีทันควัน หรือ ในบางธุรกิจ เช่น บริษัทจำหน่ายรถยนต์ หรือ อู่ซ่อมรถยนต์ อาจจะนำ Smart Card นี้ไปประยุกต์ใช้ เช่นติดชิ้นส่วนของ Smart Card ไว้ที่เครื่องยนต์ เมื่อรถยนต์เข้าอู่ ก็จะอ่านข้อมูลทราบรายละเอียดของรถได้ทันที ใครเป็นเจ้าของ ซื้อเมื่อไหร่ หมดประกันเมื่อใด ซ่อมอะไรไปบ้าง ชิ้นส่วนใดจะต้องเปลี่ยน รสนิยมของเจ้าของเป็นอย่างไร จ่ายเงินช้า เร็ว ชอบของแท้ หรือ ของเทียมเป็นต้น

เทคโนโลยี	Magnetic Card	Optical Card	Smart Card
อัตราการบกพร่อง	2 %	2%	0.03 %
จำนวนครั้งที่เข้าไปยุ่งเกี่ยว	400	400	100
จำนวนครั้งที่บกพร่องต่อปี	1000	800	100
จำนวนเครื่องที่สามารถบำรุงรักษาโดยใช้ช่างเทคนิค 1 คน	20	15	100
ค่าใช้จ่ายอะไหล่	400	500	100

ตารางที่ 2.1 การเปรียบเทียบการใช้งานการ์ดชนิดต่างๆของ France Telecom

2.3 ข้อดีของสมาร์ตการ์ด

1. มีความไว้วางใจได้ดีกว่าบัตรที่ใช้แถบแม่เหล็ก
2. สามารถเก็บสะสมข้อมูลได้มากกว่าบัตรที่ใช้แถบแม่เหล็กเป็นร้อย ๆ เท่า
3. ลดโอกาสที่จะเข้าไปยุ่งเกี่ยวและป้องกันการปลอมแปลงด้วยระบบป้องกันที่ซับซ้อน
4. สามารถเปลี่ยนมือและนำกลับมาใช้ใหม่ได้

5. ทำหน้าที่ต่าง ๆ ได้มากมาย

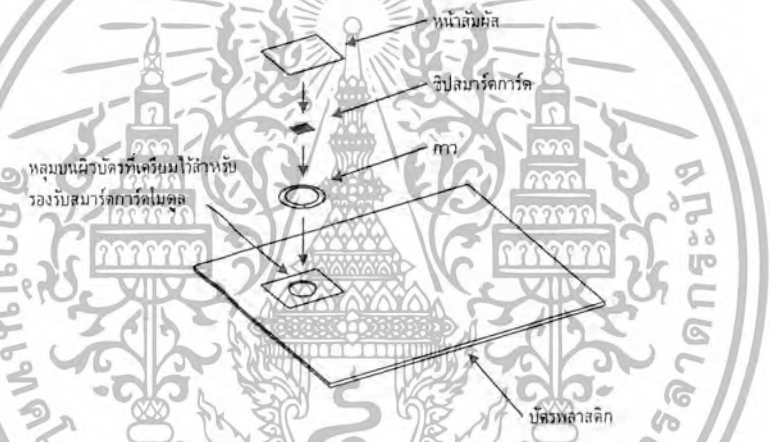
6. สามารถนำไปใช้ในงานต่าง ๆ ได้อย่างกว้างขวาง เช่น การขนส่ง ธนาคาร และการรักษาสุขภาพ เป็นต้น

7. สามารถประยุกต์ใช้กับอุปกรณ์อิเล็กทรอนิกส์แบบพกพาต่าง ๆ ได้ เช่น เครื่องโทรศัพท์และเครื่องคอมพิวเตอร์กระเป๋าหิ้ว

8. ทำงานด้วยเทคโนโลยีเซมิคอนดัคเตอร์ที่มีการพัฒนาอย่างรวดเร็ว

2.4 ส่วนประกอบและโครงสร้างของสมาร์ตการ์ด

สมาร์ตการ์ดประกอบด้วยบัตรพลาสติก กาวหรือวัสดุที่ใช้เชื่อมต่อ และหน้าสัมผัสที่บรรจุชิปสมาร์ตการ์ดเรียบร้อยแล้ว ซึ่งส่วนประกอบต่างๆแสดงดังรูป

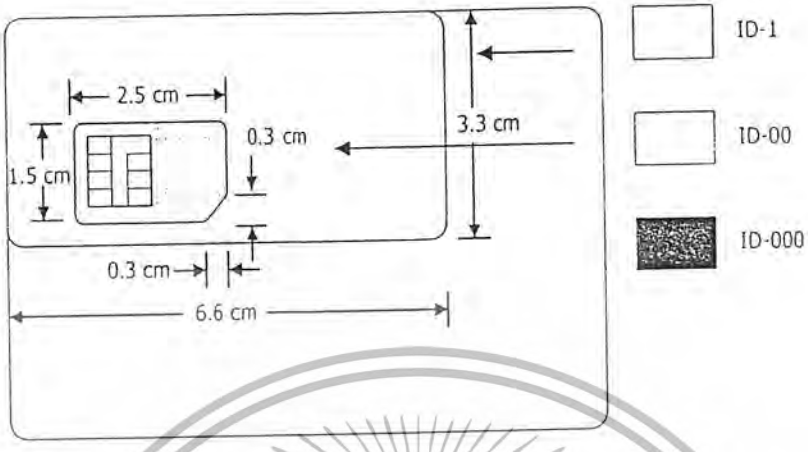


รูปที่ 2.1 ส่วนประกอบของสมาร์ตการ์ด

2.4.1 ตัวบัตรพลาสติค (Plastic Card)

ขนาดของบัตรพลาสติคที่นำมาทำสามาร์ตการ์ดกำหนดโดยมาตรฐานระหว่างประเทศ คือ ISO 7810 โดยมาตรฐานนี้ยังได้กำหนดถึงคุณลักษณะทางกายภาพของพลาสติคที่นำมาใช้ทำบัตรด้วย เช่น ความคลาดเคลื่อนของอุณหภูมิ และความยืดหยุ่นตัวในการใช้งาน ตำแหน่งของหน้าสัมผัสทางไฟฟ้า และการทำงานของมัน ตลอดจนกำหนดว่าการติดต่อระหว่างวงจรร่วม (Integrated Circuit) หรือ IC กับโลกภายนอกเป็นอย่างไรอีกด้วย มีพลาสติกอยู่4ชนิดที่นำมาใช้ผลิตสมาร์ตการ์ดได้แก่PVC (Polyethylene Terephthalate), ABS (Acrylonitrile Butadiene Styrene), PC (Polycarbonate), และ PET (Polyethylene Terephthalate) แต่ที่นิยมใช้กันมากในประเทศไทยคือ พีวีซี (PVC - Polyvinyl Chloride) และเอบีเอส (ABS - Acrylonitrile Butadiene Styrene) อย่างไรก็ดี การใช้พีวีซีมีข้อดีคือสามารถพิมพ์

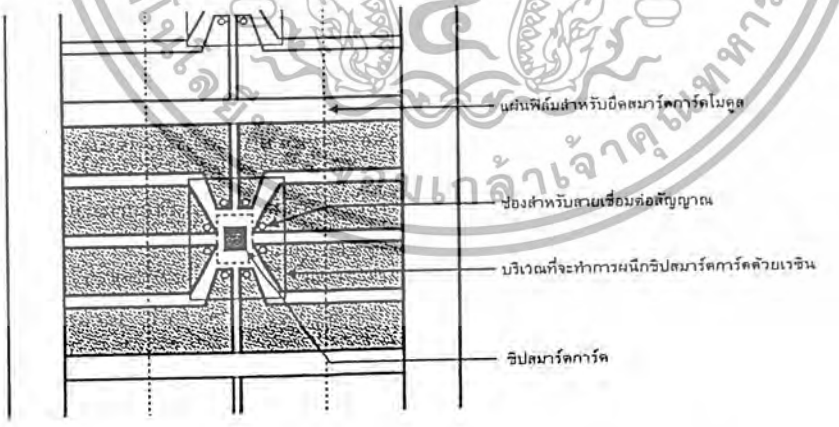
ลายนูนได้ แต่ไม่สามารถนำกลับมาใช้ใหม่ไม่ย่อยสลายในธรรมชาติได้ ส่วนเอบีเอสไม่สามารถพิมพ์นูนได้แต่นำกลับมาใช้งานใหม่ได้



รูปที่ 2.2 การแบ่งชนิดของบัตรตามรูปร่างที่นำไปใช้งาน และขนาด

2.4.2 หน้าสัมผัสและชิปสมาร์ตการ์ด (Smart Card Module)

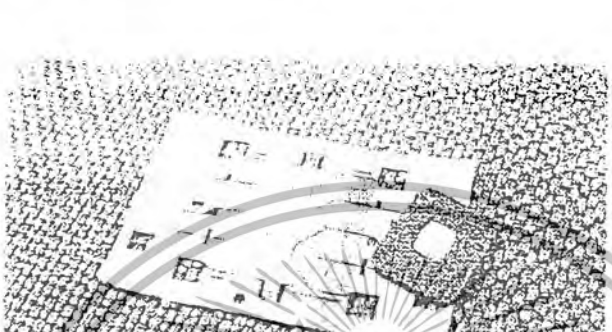
สมาร์ตการ์ดโมดูลหรือหน้าสัมผัสและชิปสมาร์ตการ์ดคือ ส่วนที่แสดงความเป็นตัวตนของสมาร์ตการ์ดที่สุด สมาร์ตการ์ดบางชนิดเมื่อหยิบขึ้นมาแรงอาจไม่อาจทราบได้เลยว่ามันคือ สมาร์ตการ์ดที่มีการฝังชิปไว้ในบัตรโดยส่วนที่จะแสดงภาพลักษณ์ที่ชัดเจนของสมาร์ตการ์ดคือสมาร์ตการ์ดโมดูล



รูปที่ 2.3 ส่วนประกอบของสมาร์ตการ์ดโมดูลในสายการผลิตสมาร์ตการ์ด

ในการผลิตสมาร์ตการ์ดโมดูล ส่วนที่เป็นหน้าสัมผัสของสมาร์ตการ์ดประกอบด้วยโลหะหลายชนิดประกอบกัน แต่ละส่วนจะถูกยึดด้วยฟิล์มบางๆทางด้านหลังของหน้าสัมผัสเพื่อให้คงรูปอยู่ได้

แถบฟิล์มตัวนี้จะมีการเจาะช่องเล็กๆสำหรับการเชื่อมต่อสายนำสัญญาณกับสมาร์ตชิปกับหน้าสัมผัส หลังจากที่วางชิปสมาร์ตการ์ดลงในตำแหน่งที่ต้องการ และทำการเชื่อมต่อสายนำสัญญาณจากชิปสมาร์ตการ์ดเข้ากับหน้าสัมผัสเรียบร้อยแล้ว ขั้นสุดท้ายจะเป็นการผนึกชิปเพื่อป้องกันตัวชิป และสายนำสัญญาณต่างๆจากสิ่งแวดล้อมภายนอกขั้นสุดท้ายจะเป็นการนำหน้าสัมผัสและชิปใส่ลงในบัตรพลาสติกและทดสอบการทำงานของชิปขั้นสุดท้าย



รูปที่ 2.4 ตัวอย่างสมาร์ตการ์ดโมดูล

2.5 องค์ประกอบในการใช้งานสมาร์ตการ์ด

2.5.1 ตัวบัตรและตัวชิป

บัตรและชิปสมาร์ตการ์ดเป็นส่วนแรกที่จะกล่าวถึงเพราะสมาร์ตการ์ดมีหลากหลายรูปแบบ หลากหลายการใช้งาน โดยหลักการแล้วสมาร์ตการ์ดเป็นเพียงบัตรฝังชิป IC ที่สามารถเก็บข้อมูลได้เท่านั้นผู้ออกแบบระบบมีหน้าที่นำสมาร์ตการ์ดมาใช้งานอย่างชาญฉลาดเหมาะสมตามประเภทงาน และบริหารข้อมูลภายในสมาร์ตการ์ดให้เกิดความปลอดภัยสูงสุด

สมาร์ตการ์ดที่นำมาใช้งานมีตั้งแต่ราคาใบละไม่กี่ร้อยบาท ถึงใบละหลายพันบาน โดยในปัจจุบันเราสามารถเห็นการใช้งานสมาร์ตการ์ดในหลายรูปแบบเช่น บัตรโทรศัพท์ ซิมการ์ดในโทรศัพท์มือถือ , บัตรเข้าออกที่อยู่อาศัย (คอนโดมิเนียมบางแห่ง) , บัตรนักศึกษา , บัตรพนักงาน, บัตรเติมน้ำมันแบบเครดิต (Fleet Card), บัตรแทนเงินสด, ซิมการ์ดในโทรศัพท์มือถือซึ่งมีการกำหนดเป็นมาตรฐาน GSM โดยผู้ผลิตสมาร์ตการ์ดต้องผลิตสมาร์ตการ์ดที่มีโครงสร้างที่มีโครงสร้างข้อมูลภายในตามที่มาตรฐาน GSM กำหนด

2.5.2 สมาร์ตการ์ดรีดเดอร์

สมาร์ตการ์ดรีดเดอร์จะประกอบด้วยขาสำหรับเชื่อมสัญญาณกับหน้าสัมผัสบนชิปสมาร์ตการ์ด (Card Contact) หรือเป็นเสาอากาศรับส่งคลื่นวิทยุสำหรับสมาร์ตการ์ดแบบไม่มีหน้าสัมผัส (Contact less) และหน่วยประมวลผลพร้อมหน่วยความจำสำหรับติดต่อสื่อสารกับชิปสมาร์ตการ์ดโดยตรง การสร้างสมาร์ตการ์ดรีดเดอร์ขึ้นใช้เองสามารถทำโดยการนำไมโครโปรเซสเซอร์หรือไมโครคอนโทลเลอร์ มาประยุกต์ใช้ในการเชื่อมต่อกับสมาร์ตการ์ด

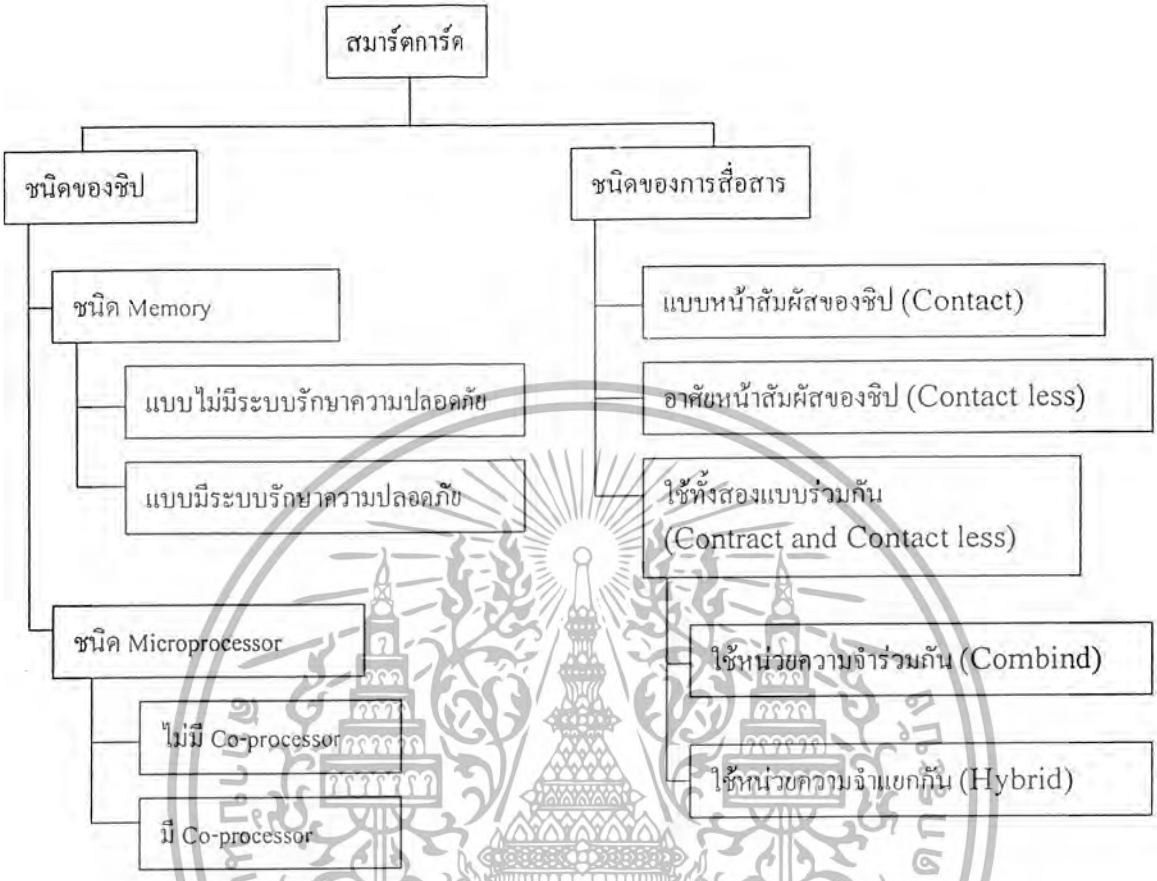
2.5.3 ซอฟต์แวร์

ในการใช้งานสมาร์ตการ์ดนอกจากตัวบัตรสมาร์ตการ์ด สมาร์ตการ์ดรีดเดอร์แล้วยังมีส่วนประกอบอีกส่วนที่สำคัญคือ ซอฟแวร์สำหรับจัดการข้อมูลในสมาร์ตการ์ด และซอฟแวร์สำหรับบริหารงานด้านบัตร หรืออาจเรียกได้ว่าระบบ Front-End (เหมือนกับระบบในบัตรเครดิต) ซึ่งระบบ Front – End ของสมาร์ตการ์ดจะแตกต่างจากระบบบัตรแถบแม่เหล็ก เนื่องจากสมาร์ตการ์ดไม่จำเป็นต้องมีการติดต่อสื่อสารกับ Front-End ทุกครั้งที่ทำรายการเหมือนในระบบบัตรเครดิต ทำให้ระบบ Front-End ของสมาร์ตการ์ดมีเวลามากพอในการบริหารงานด้านอื่นๆหากต้องการติดต่อสื่อสารกับระบบ Front-End ของสมาร์ตการ์ดจำเป็นต้องใช้สมาร์ตการ์ดรีดเดอร์ที่มีส่วนสำหรับการติดต่อสื่อสารไม่ว่าจะเป็น MODEM , Ethernet , Local Area Network ,ระบบสื่อสารด้วยเครื่องวิทยุ, ระบบสื่อสารอนุกรม RS-485/422 สำหรับการสื่อสารในบริเวณพื้นที่ให้บริการที่ไม่กว้างใหญ่นัก เพื่อใช้รับ-ส่งข้อมูล ระหว่าง Front-End เมื่อจำเป็น

2.5.4 Back Office

เป็นซอฟแวร์สำหรับควบคุมดูแลระบบทั้งหมด ประกอบด้วยซอฟต์แวร์สำหรับป้อนข้อมูลเกี่ยวกับบัตร และผู้ถือบัตรเพื่อออกบัตรใหม่ (Card Issuer), ซอฟต์แวร์สำหรับออกรายงานต่างๆ (Report) และซอฟแวร์ส่วนสุดท้ายคือสำหรับให้บริการผู้ถือบัตร เช่น ซอฟแวร์สำหรับเติมเงินลงในชิป (สมาร์ตการ์ดที่ใช้แทนเงินสด), ซอฟแวร์สำหรับเติม-แลกแต้มในบัตรสะสมแต้ม (Royalty Card) ปกติแล้วซอฟแวร์ในส่วนของ Back-End และ BackOffice ที่กล่าวมาต้องทำร่วมกับสมาร์ตการ์ดรีดเดอรเสมอ เพราะเพียงสมาร์ตการ์ดไม่สามารถทำรายการใดๆได้เอง

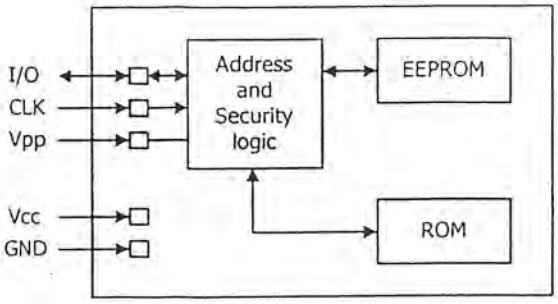
2.6ประเภทของสมาร์ตการ์ด



รูปที่ 2.5 การแบ่งสมาร์ตการ์ดตามชนิดของหน่วยความจำและประเภทของหน้าสัมผัส

2.6.1 Memory Card (Synchronous Card)

สมาร์ตการ์ด แบบ Memory หรืออีกชื่อหนึ่งคือ Synchronous Card เนื่องจากสมาร์ตการ์ดชนิดนี้มีการรับ-ส่งข้อมูลตามสัญญาณนาฬิกาที่ป้อนให้แก่ชิป (ข้อมูลแต่ละบิตที่ส่งให้แก่ชิปต้องสัมพันธ์กับสัญญาณนาฬิกา สมาร์ตการ์ดชนิดนี้มีโครงสร้างที่ประกอบไปด้วย วงจรสำหรับติดต่อสื่อสารภายนอก,หน่วยความจำข้อมูล,และหน่วยความจำสำหรับเก็บชุดคำสั่งของสมาร์ตการ์ดดังรูปที่ 2.6

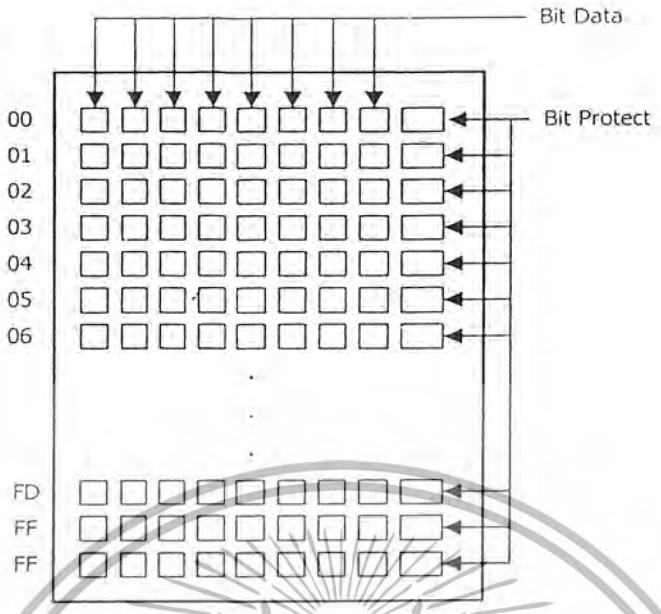


รูปที่ 2.6 บล็อกไดอะแกรมโครงสร้างในชิปสมาร์ตการ์ดชนิด Memory (Synchronous Card)

สมาร์ตการ์ดที่เป็นพื้นฐานในปัจจุบัน ก็คือสมาร์ตการ์ดชนิด Free Access Memory สมาร์ตการ์ดชนิดนี้เปิดโอกาสให้อ่านหรือเขียนข้อมูลในแอดเดรสใดๆ ก็ได้ตามชื่อของสมาร์ตการ์ดชนิดนี้ ไม่มีการป้องกันข้อมูลใดๆภายในการ์ดชนิดนี้เป็นการ์ดที่มีความปลอดภัยต่ำสุด แต่การอ่านข้อมูลก็จะไม่สามารถอ่านได้ง่ายนักเนื่องจากเมื่อมีการออกแบบหน่วยความจำให้มีการสลับตำแหน่งของบิตข้อมูล โดยมีวงจรควบคุมการสลับตำแหน่งของบิตเป็นส่วนป้องกันข้อมูลอีกต่อหนึ่ง ดังนั้นการอ่านข้อมูลแบบธรรมดา จะไม่ได้ข้อมูลที่ถูกต้องหากไม่ได้ติดต่อกับวงจรควบคุมการสลับตำแหน่งของบิตโดยตรง

นอกจากสมาร์ตการ์ดแบบชนิด Memory ธรรมดายังมีการใส่วงจรกำหนดเงื่อนไขการเข้าถึงข้อมูลลงไปด้วย ทำให้สามารถกำหนดเงื่อนไขการอ่าน-เขียนข้อมูลได้ทุกไบต์ โดยสมาร์ตการ์ดที่มีวงจรป้องกันการอ่าน-เขียนชนิดนี้ถูกเรียกว่า PIN Protect Memory เนื่องจากการเข้าถึงข้อมูลจะต้องแสดงรหัสผ่านให้บัตรรับทราบก่อนจึงจะสามารถเข้าถึงข้อมูลได้ วงจรกำหนดเงื่อนไขการเข้าถึงข้อมูลจะมีบิตพิเศษที่มีชื่อว่า Bit Protect ซึ่งเป็นบิตข้อมูลที่ฝากไว้กับข้อมูลให้เป็นบิตที่ 9 แต่ไม่สามารถแก้ไขด้วยคำสั่งเขียนข้อมูลธรรมดา เพราะ Bit Protect ไม่ได้เป็นส่วนหนึ่งของข้อมูลจริงๆในการแก้ไข Bit Protect นี้จะสามารถ ทำการเปลี่ยนแปลงได้เพียงครั้งเดียวด้วยคำสั่งเฉพาะเท่านั้นเช่น หากต้องการบังคับไม่ให้ข้อมูลไบต์ใดไม่สามารถแก้ไขได้ก็ให้ทำการเคลียร์บิตที่ 9 ของข้อมูลไบต์นั้นๆ แต่สำหรับรหัสผ่านในการเข้าถึงข้อมูลสามารถเปลี่ยนแปลงได้แต่ต้องแสดงรหัสผ่านชุดเก่าให้บัตรได้รับทราบก่อนจึงจะสามารถเปลี่ยนแปลงรหัสผ่านได้

สมาร์ตการ์ดชนิด Memory เป็นสมาร์ตการ์ดที่เป็นพื้นฐานของสมาร์ตการ์ดรุ่นใหม่ๆในปัจจุบัน ด้วยโครงสร้างและการทำงานที่ง่ายต่อการทำความข้อใจ ราคาถูก สามารถเก็บข้อมูลได้จำนวนมาก และความเร็วในการประมวลผลไม่มากเกินไป จึงทำให้สมาร์ตการ์ดชนิดนี้เหมาะที่จะนำไปประยุกต์ใช้งานกับข้อมูลที่ไม่มีความสำคัญมากนักเช่น บัตรลงเวลา บัตรผ่านประตู บัตรโทรศัพท์ ปัจจุบันสมาร์ตการ์ดชนิด Memory มีขนาดหน่วยความจำสูงสุด 64 กิโลไบต์ และอนาคตความจุของสมาร์ตการ์ดจะเพิ่มขึ้นไปอีก



รูปที่ 2.7 ตัวอย่างหน่วยความจำของ Memory Card ชนิด PIN Protect

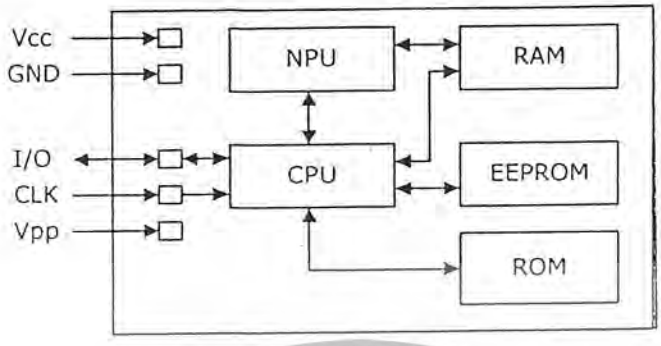
2.6.2 การ์ดแบบออพทิคอลเมมโมรี่ (Optical Memory Cards)

เป็น Smart Card แบบที่ใช้ระบบส่งข้อมูลทางแสง (Optical) ในส่วนที่มีข้อมูลนั้นก็จะมีลักษณะเหมือนแผ่นพลาสติก หรือ เศษของแผ่น CD อัดอยู่บนการ์ดนี้ ซึ่งในขณะนี้ก็มีการนำมาใช้บ้างแล้ว Optical Memory Card นี้ สามารถเก็บข้อมูลได้ถึง 4MB แต่ลง หรือ บรรจุข้อมูลได้ครั้งเดียว เปลี่ยนแปลง หรือ โยกย้ายข้อมูลออกจากการ์ดนี้ไม่ได้ ดังนั้นการ์ด Optical Memory Card นี้ก็มักจะถูกนำไปใช้ในวงการ การเก็บรักษาข้อมูล เช่น ข้อมูลทางการแพทย์ ข้อมูลเกี่ยวกับบุคคล ข้อมูลการท่องเที่ยว เป็นต้น ปัจจุบัน ตัวการ์ดเองก็ไม่มี Microprocessor อยู่ในตัวการ์ด แต่ในอนาคตก็อาจจะมีการปรับปรุงให้ทันสมัยขึ้นมากกว่าเวลานี้ อีกประการหนึ่งก็คือ เครื่องอ่านข้อมูลของบัตร Optical Memory Card นี้ ราคายังค่อนข้างแพง และ ที่สำคัญก็ยังไม่มีมาตรฐานมาบังคับใช้

2.6.3 Processor Card (Asynchronous Card)

สมาร์ตการ์ดชนิดนี้เป็นสมาร์ตการ์ดที่ได้รับการพัฒนาปรับปรุงจากสมาร์ตการ์ดชนิด Memory ด้วยการใส่เทคโนโลยีไมโครโปรเซสเซอร์เข้าไปในชิป เพื่อให้ชิปสามารถประมวลผลข้อมูลและเพิ่มความปลอดภัยให้แก่ข้อมูลให้สูงขึ้น การที่ใส่ไมโครโปรเซสเซอร์ลงในชิปทำให้จำเป็นต้องมีการเพิ่มส่วนของหน่วยความจำสำหรับจัดการเก็บระบบปฏิบัติการของไมโครโปรเซสเซอร์ และหน่วยความจำชั่วคราสำหรับการประมวลผลนอกจากนี้การใส่ชิปประมวลผลทางคณิตศาสตร์ลงในชิป

สมาร์ตการ์ดเพื่อช่วยในการประมวลผลข้อมูลด้วยอัลกอริทึมสำหรับเข้า-ถอดรหัส ทำให้สมาร์ตการ์ดชนิด Processor มีความเร็วสูงกว่าสมาร์ตการ์ดชนิด Memory หลายเท่า



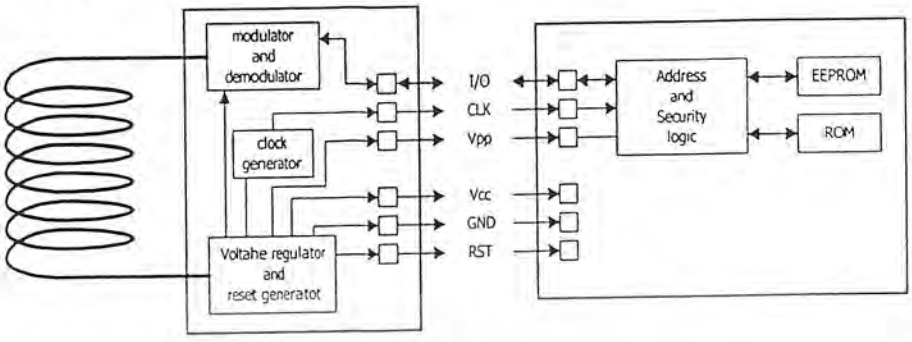
รูปที่ 2.8 บล็อกไดอะแกรมโครงสร้างภายในชิปสมาร์ตการ์ดชนิด Processor (Asynchronous Card)

ในการรับส่งข้อมูลของสมาร์ตการ์ดชนิดนี้ จะใช้หน้าสัมผัสชนิดเดียวกันกับสมาร์ตการ์ดชนิด Memory โดยสัญญาณนาฬิกาที่ป้อน จะถูกใช้เป็นสัญญาณนาฬิกาให้แก่โปรเซสเซอร์ภายในสมาร์ตการ์ด ข้อมูลที่รับ-ส่งจึงไม่จำเป็นต้องสัมพันธ์กับสัญญาณนาฬิกาที่ป้อนให้แก่ชิปเพียงกำหนดอัตราการรับ- ส่งข้อมูลเป็น 9600 บิต/วินาที ก็จะสามารถติดต่อกับโปรเซสเซอร์ของชิปได้แล้ว แต่การเข้าถึงข้อมูลจะไม่สามารถทำได้เหมือนกับการ์ดชนิด Memory การเข้าถึงข้อมูลต้องกระทำผ่านโปรเซสเซอร์ของสมาร์ตการ์ดเท่านั้น ไม่ว่าจะเป็นการอ่านหรือการเขียนข้อมูลก็ตามเพราะหน่วยความจำจะอยู่ภายในความควบคุมของโปรเซสเซอร์เพียงอย่างเดียว ข้อดีอีกอย่างที่ไม่สามารถติดต่อกับหน่วยความจำในชิปโดยตรงก็คือ การลอบเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตแทบเป็นไปไม่ได้ ยกเว้นมีความบกพร่องในการกำหนดเงื่อนไขในการเข้าถึงข้อมูลที่เป็นความลับ

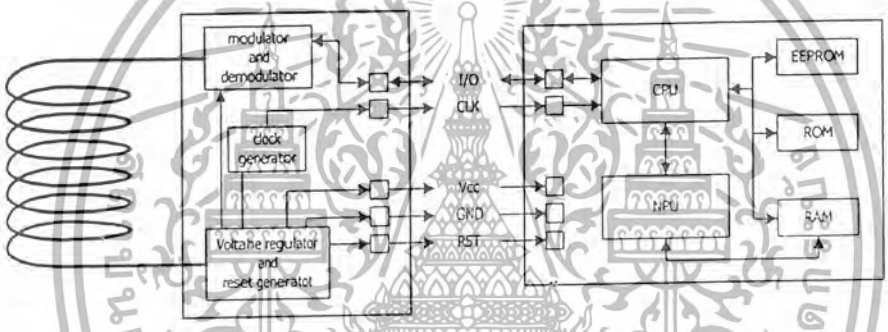
2.6.4 Contact less Smart Card

สมาร์ตการ์ดชนิด Contact less ใช้เทคโนโลยีการสื่อสารผ่านคลื่นวิทยุ โดยการส่งคลื่นวิทยุความถี่ 13.56 เมกะเฮิรตซ์ ที่ได้รับการมอดดูเลทข้อมูลแล้วส่งให้กับชิปสมาร์ตการ์ด ทางด้านชิปสมาร์ตการ์ดจะใช้ขดลวด เป็นเสารับส่งสัญญาณ โดยเสารับส่งสัญญาณนี้จะเป็นขดลวดขนาดเล็กที่ถูกฝังอยู่ในตัวบัตร ภายนอกบัตรชนิดนี้แทบดูไม่ออกว่าเป็นบัตรสมาร์ตการ์ด จากรูป 2.9 จะเห็นว่าส่วนที่เพิ่มเข้ามาเป็นส่วนที่รับสัญญาณคลื่นวิทยุมาแบ่งเป็นสองส่วน โดยส่วนแรกจะถูกแปลงเป็นกระแสไฟฟ้าสำหรับป้อนชิป และวงจรสร้างสัญญาณนาฬิกาให้สามารถทำงานได้ อีกส่วนหนึ่งจะถูก Demodulate เอาข้อมูลออกจากคลื่นวิทยุ และส่งให้แก่ชิปสมาร์ตการ์ดอีกต่อหนึ่ง ส่วนการส่งข้อมูล

กลับก็จะใช้กระแสไฟฟ้าที่ได้จากคลื่นวิทยุมาใช้ในการ Modulate ข้อมูลและส่งกลับไปยังเสารับ- ส่งสัญญาณภายในบัตร



รูปที่ 2.9 สมาร์ตการ์ด ชนิด Memory แบบ Contact less

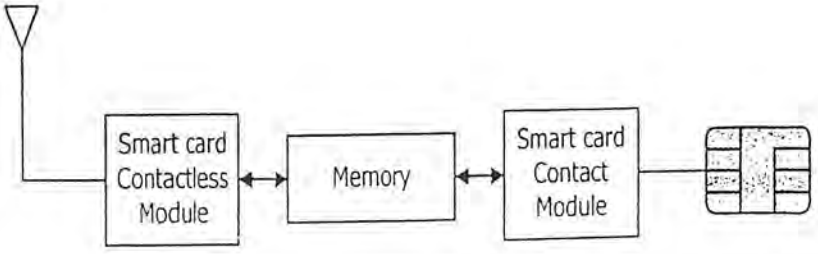


รูปที่ 2.10 สมาร์ตการ์ดชนิดProcessor แบบ Contact less

สมาร์ตการ์ดชนิด Contact less ถูกออกแบบให้ใช้กระแสไฟฟ้าต่ำเพราะกระแสไฟฟ้าที่ได้จากคลื่นวิทยุนั้นมีปริมาณเพียงเล็กน้อยไม่เพียงพอที่จะทำให้สมาร์ตการ์ดแบบธรรมดาสามารถทำงานได้ สมาร์ตการ์ดชนิดนี้รุ่นแรกๆจะไม่สามารถทำคำสั่งที่ซับซ้อนมากๆเช่นคำสั่งในการเข้ารหัสข้อมูล หรือคำสั่งที่ต้องใช้เวลาในการประมวลผลมากๆและระยะในการรับ-ส่งสัญญาณก็ไม่มากนัก แต่ปัจจุบันสมาร์ตการ์ด Contact less สามารถทำการเข้ารหัสที่ยุ่งยากได้แล้วด้วยการเพิ่มวงจรสำหรับเข้ารหัสภายในชิป

2.6.5 Com-Bi Card

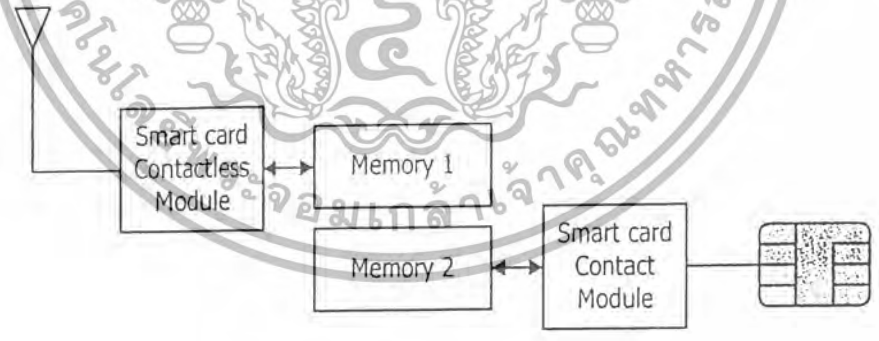
สมาร์ตการ์ดชนิดนี้เป็นการรวมเอาสมาร์ตการ์ดแบบ Contact และสมาร์ตการ์ดชนิด Contact less เข้าด้วยกัน โดยใช้หน่วยความจำข้อมูลร่วมกันเพื่อให้การทำรายการที่จำเป็นต้องอยู่ภายใต้การควบคุมอยู่ และสามารถใช้งานทั่วไปได้อย่างสะดวกสบาย (Speed Pass) ผ่านทางคลื่นวิทยุ



รูปที่ 2.11โครงสร้างภายในของสมาร์ตการ์ดชนิด Contact แบบ Com-bi card

2.6.6 Hybrid Smart Card

สมาร์ตการ์ดชนิดนี้มีลักษณะโครงสร้างภายในคล้ายกับประเภท Com-Bi Card แต่จะแตกต่างกันในเรื่องของหน่วยความจำข้อมูล โดยหน่วยความจำข้อมูลจะถูกแยกจากกันอย่างสิ้นเชิงระหว่าง Contact และ Contact less เพื่อความสะดวกในการแยกใช้งาน ซึ่งปัจจุบัน Hybrid Card จะมีความหมายรวมถึงบัตรที่มีคุณสมบัติในการใช้งานตั้งแต่สองอย่างขึ้นไป บัตรสมาร์ตการ์ดที่มีทั้งชิปสมาร์ตการ์ดและแถบแม่เหล็ก, บัตรสมาร์ตการ์ด ที่เป็นทั้ง Contact และ Contact less



รูปที่ 2.12 โครงสร้างภายในของสมาร์ตการ์ดชนิด Hybrid Smart Card

2.7 รูปแบบของสมาร์ตการ์ดที่นำมาใช้ในโครงงาน

ในโครงงานได้นำบัตรโทรศัพท์สาธารณะ TOT Card ที่ใช้งานหมดแล้วมาใช้ เป็นบัตรโทรศัพท์รูปแบบใหม่ที่ใช้เทคโนโลยีใหม่ล่าสุดของระบบโทรศัพท์สาธารณะที่ใช้อยู่ในปัจจุบัน โดยบัตรโทรศัพท์สาธารณะ TOT Card โดย TOT Card เป็นรูปแบบหนึ่งของ Memory Card เครื่องโทรศัพท์จะมีอุปกรณ์รักษาความปลอดภัย หรือ SAM (Security Access Module) ติดตั้งในเครื่อง เพื่อทำการตรวจสอบบัตร และมูลค่าบัตร จึงเป็นการป้องกันการนำบัตรปลอมมาใช้งาน

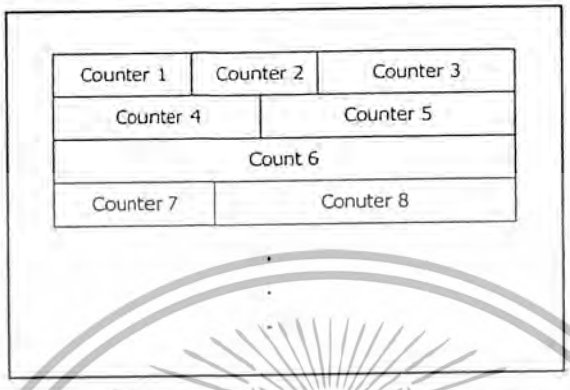
2.7.1 บัตรโทรศัพท์สาธารณะ TOT CARD

รูปที่ 2.13 ตัวอย่างบัตรโทรศัพท์ TOT

เป็นบัตรพลาสติกที่มีขนาดเท่ากับขนาดมาตรฐาน ของบัตรเครดิต ที่ใช้ในปัจจุบันซึ่งบรรจุ CHIP บันทึกข้อมูล ติดไว้บนหน้าบัตร สำหรับเป็นสื่อสัญญาณ ระหว่างบัตรกับ เครื่องโทรศัพท์ที่มีหัวอ่าน พร้อมระบบรักษาความปลอดภัย ซึ่งใช้ในการตรวจสอบความถูกต้องของข้อมูลในบัตร องค์การโทรศัพท์ฯ ใช้ มาตรฐาน EURO CHIP นำมาให้บริการ และเพื่อป้องกัน การลอกเลียนหรือปลอมแปลงบัตร องค์การโทรศัพท์จึงนำระบบ การสร้างรหัสหลัก Master Key อันเป็น ลิขสิทธิ์ของ องค์การโทรศัพท์ เป็นองค์ประกอบหลักในการเข้ารหัสข้อมูล เพื่อผลิตบัตร ซึ่ง สามารถป้องกันการปลอมแปลงบัตรโทรศัพท์ ได้เป็นอย่างดี คุณสมบัติ ทนทานในทุกสภาพ อายุการใช้งาน 3 ปีโดยระบุที่ด้านหลังบัตรทุกใบ ไอซี (CHIP) เป็นแบบ MEMORY CHIP ซึ่งเมื่อใช้จนมูลค่าในบัตรหมดไปแล้ว ไม่สามารถเติม มูลค่าเงินลงในบัตรได้อีก ทั้งนี้ในอนาคตอันใกล้ องค์การโทรศัพท์ จะพัฒนาบริการ ไปสู่บริการกระเป๋าเงินสดอิเลคทรอนิคส์ คือบัตรใบเดียวสามารถจับจ่ายใช้สอย หลายๆ บริการทั่วประเทศ

TOT Card เป็น Token Memory Card ภายในสมาร์ตการ์ดชนิดนี้จะมีการเก็บข้อมูลในลักษณะของการนับจำนวน (Counter) ซึ่งจำนวนนับนี้จะเป็นตัวเลขแทนมูลค่าของเงินที่ระบุบนบัตร การนับเลขเป็นการนัยถอยหลังเพื่อเป็นการนับมูลค่าคงเหลือในบัตร หมายความว่าหากใช้บัตรในการโทรศัพท์ไปเรื่อยๆ มูลค่าในบัตรก็จะถูกลดลงตามไปด้วยเช่นกัน ในการเข้าถึงข้อมูลของสมาร์ตการ์ดชนิดนี้ต้องมีการแสดงรหัสผ่านให้บัตรรับทราบเหมือนกัน Memory Card แต่ไม่มีBit Protect เท่านั้น

การเข้าถึงข้อมูลในหน่วยความจำของสมาร์ตการ์ดชนิดนี้ต้องอ้างอิงกับแอดเดสเสมอ ทั้งนี้ต้องขึ้นอยู่กับข้อกำหนดของสมาร์ตการ์ดแต่ละรุ่น แต่ใน Token Memory Card นี้จะสามารถเข้าถึงและสามารถเปลี่ยนแปลงข้อมูลได้เพียงบางส่วนเท่านั้น โดยส่วนที่สามารถทำการเข้าถึงได้จะเป็นส่วนหมายเลขประจำของแต่ละบัตร (Serial Number) ซึ่งบัตร TOT แต่ละใบจะมีเลขประจำบัตรที่ไม่ซ้ำกัน

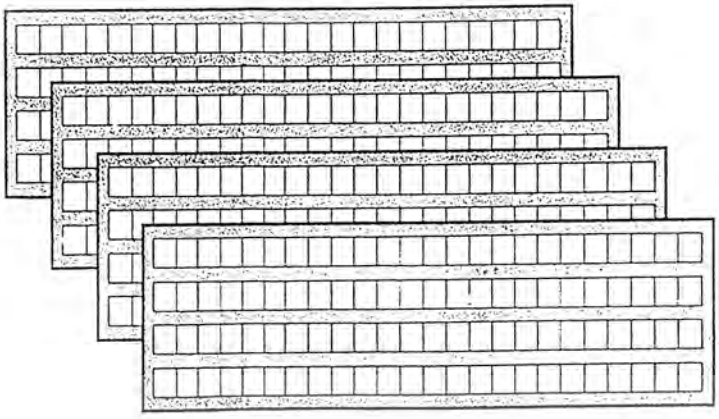


รูปที่ 2.14 ตัวอย่างหน่วยความจำของสมาร์ตการ์ดชนิด Memory แบบ Token

2.7.1.1 การจัดการหน่วยความจำภายใน

ในสมาร์ตการ์ดชนิด Memory มีการแบ่งวิธีการจัดการหน่วยความจำเป็นสองแบบ คือ bitwise และ bytewise การจัดการหน่วยความจำแบบ bitwise เป็น การจัดการหน่วยความจำที่ใช้ในสมาร์ตการ์ดรุ่นแรกๆ การจัดการหน่วยความจำแบบนี้มักใช้บอกขนาดของหน่วยความจำของสมาร์ตการ์ดเป็นหน่วยบิตเช่น 1 กิโลบิต (128 ไบต์), 8 กิโลบิต(1 กิโลไบต์) สาเหตุที่ bitwise จัดการหน่วยความจำข้อมูลเป็นบิตเนื่องมาจากข้อมูลที่ใช้รับ-ส่งในสมาร์ตการ์ดชนิดนี้ทำกันในระดับบิตเท่านั้น หมายความว่าการรับส่งข้อมูลไม่จำเป็นต้องทำให้ครบทั้ง 8 บิตหรือ 1 ไบต์ การจัดการหน่วยความจำแบบนี้สามารถอ่านข้อมูลที่บิตใดก็ได้ ซึ่งมีใช้ในสมาร์ตการ์ดที่มีหน่วยความจำข้อมูลไม่มากนัก

สำหรับการจัดการหน่วยความจำแบบ bytewise เป็นการจัดการหน่วยความจำที่อ้างถึงข้อมูลขนาด 8 บิตหรือ 1 ไบต์เต็ม การรับ-ส่งข้อมูลกับชิปต้องทำการรับ-ส่งข้อมูลทั้ง 8 บิตจนครบจึงจะทำให้การรับ-ส่งข้อมูลเสร็จสมบูรณ์ (หากทำไม่เสร็จสมบูรณ์ ชิปสมาร์ตการ์ดจะยกเลิกการรับส่งข้อมูลครั้งนั้นๆ) นอกจากนี้การอ้างอิงหน่วยความจำยังมีความแตกต่างกันเช่น บางผู้ผลิตกำหนดให้หน่วยความจำเป็นแอดเกศที่ต่อเนื่องกันตั้งแต่แอดเดสที่ 0 ถึงแอดเดสสุดท้ายบางผู้ผลิตแบ่งหน่วยความจำออกเป็นเพจ (Page) แต่ละเพจมีขนาดแตกต่างกันตามแต่รุ่นที่ผลิต ทำให้การอ้างถึงข้อมูลใดๆในหน่วยความจำของสมาร์ตการ์ดแต่ละแบบไม่เหมือนกันตามแต่ผู้ผลิตจะออกแบบ



รูปที่ 2.15 หน่วยความจำข้อมูลของสมาร์ตการ์ดชนิด Memory

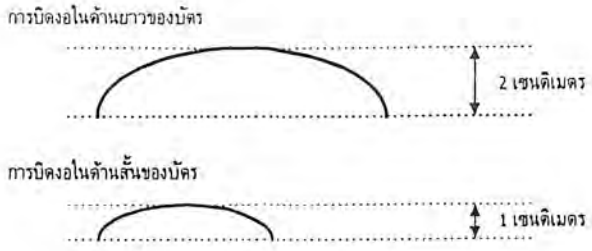
2.7.1.2 มาตรฐานของสมาร์ตการ์ดที่เกี่ยวข้อง

เพื่อให้เกิดความเข้าใจกันได้ของสมาร์ตการ์ด จึงมีการกำหนดมาตรฐานของสมาร์ตการ์ดคือ ISO7816 เป็นข้อกำหนดในเรื่องของคุณสมบัติของบัตรพลาสติกที่จะนำมาใช้ทำเป็นสมาร์ตการ์ด โดยมีหัวข้อย่อยแบ่งเป็น ISO7816-1, ISO7816-2, ISO7816-3, ISO7816-4, ISO7816-5, ISO7816-6 ในที่นี้จะกล่าวในรายละเอียดของ3มาตรฐานแรกเท่านั้นเนื่องจากมีความสำคัญต่อการใช้งานในโครงงาน

1) มาตรฐานISO7816-1

-เป็นมาตรฐานที่กำหนดด้วยเรื่องของคุณสมบัติทางกายภาพเบื้องต้นของสมาร์ตการ์ด ประกอบด้วย

- ความคงทนต่อแสงและรังสีต่างๆ
- ขนาดความหนาของชิปสมาร์ตการ์ด
- ความทนต่อแรงกดของหน้าสัมผัส (ทนต่อแรงกด 1.5 นิวตันได้โดยไม่เสียหาย)
- ค่าความต้านทานของหน้าสัมผัส (ไม่เกิน 0.5 โอห์ม ที่กระแส 0.5 ไมโครแอมป์ – 300 มิลลิแอมป์)
- ความทนต่อสนามแม่เหล็ก
- ความทนต่อไฟฟ้าสถิต (1500 โวลต์ ประจุ 100 พิโกฟารัด ที่ 1500 โอห์ม)
- ความทนทานต่อการบิดงอ เป็นจำนวน 30 ครั้งต่อนาที โดยที่บัตรและชิปต้องไม่เกิดความเสียหาย

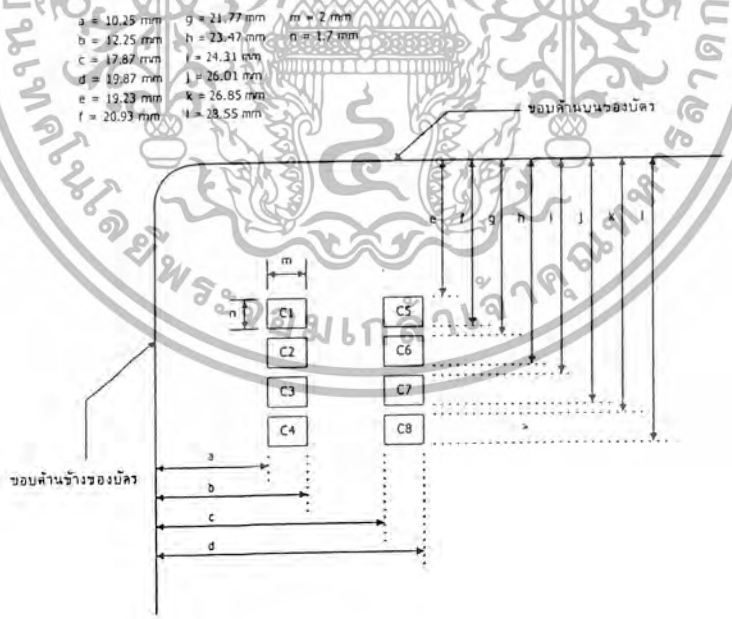


รูปที่ 2.16 วิธีทดสอบการบิดงอสมาร์ตการ์ด

2) มาตรฐาน ISO7816-2

เป็นมาตรฐานที่กำหนดขนาดของหน้าสัมผัส และตำแหน่งของหน้าสัมผัสชิปสมาร์ตการ์ดบนบัตร ประกอบด้วย

- ขนาดของหน้าสัมผัสชิปสมาร์ตการ์ด
- ตำแหน่งของหน้าสัมผัสบนบัตร ดังรูป



รูปที่ 2.17 ตำแหน่งหน้าสัมผัสของชิปสมาร์ตการ์ด

3) มาตรฐาน ISO7816-3

มาตรฐานที่กำหนดคุณสมบัติทางไฟฟ้าและ Protocol ที่ใช้ในการสื่อสารกับชิปสมาร์ตการ์ด จะเป็นการบรรยายเกี่ยวกับหน้าสัมผัสมีดังนี้

Vcc แรงดันไฟบวกของแหล่งจ่ายไฟฟ้าที่ป้อนให้แก่ชิป

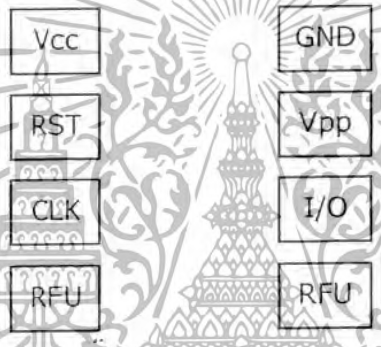
Vpp แรงดันไฟฟ้าสำหรับการเขียนข้อมูลลงในชิปสมาร์ตการ์ด

GND กราวด์ของแหล่งจ่ายกระแสไฟฟ้าที่ป้อนให้แก่ชิป

RST แรงดันไฟฟ้าสำหรับรีเซตชิปสมาร์ตการ์ด

I/O Input – Output สำหรับการรับส่งข้อมูลแบบอนุกรม

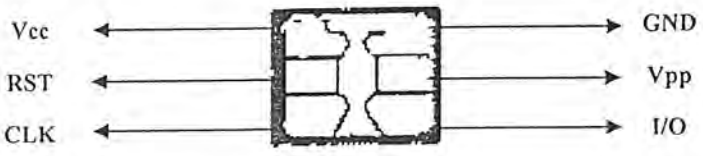
CLK สัญญาณนาฬิกาสำหรับกำหนดจังหวะการรับ-ส่งข้อมูล



รูปที่ 2.18 หน้าที่การทำงานของแต่ละหน้าสัมผัส

2.7.1.3 รูปแบบของสมาร์ตชิปในบัตร TOT

ในบัตร TOT การ์ด จะมีหน้าสัมผัสอยู่ 6 หน้าสัมผัส โดยจะไม่มี ขา RFU ทั้งสองขา

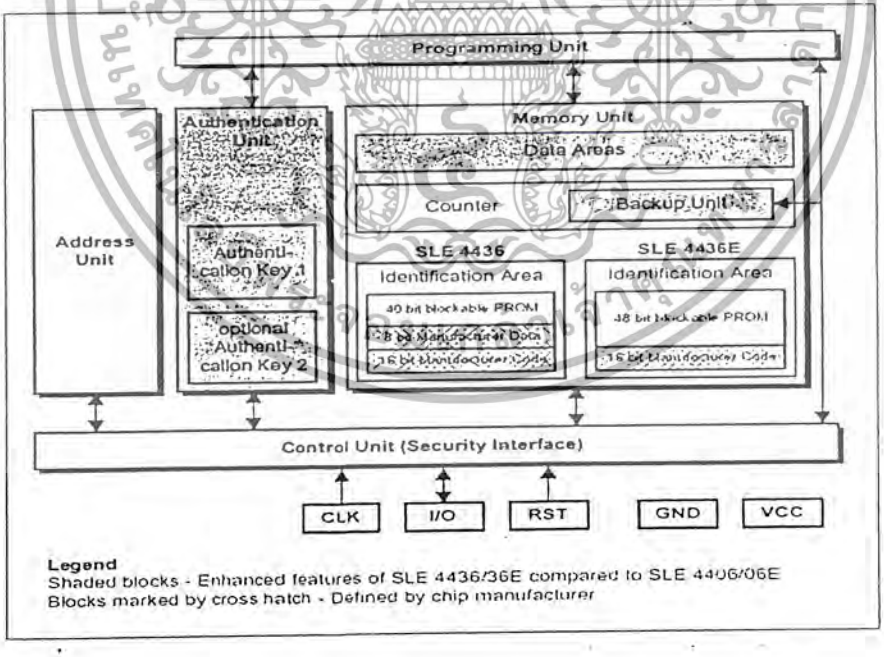


รูปที่ 2.19 สมาร์ตชิปในบัตร TOT

บัตรโทรศัพท์สาธารณะ TOT Card เป็นสมาร์ตการ์ดแบบ Synchronous Card เบอร์ SLE4436 โดยจะเป็นรูปแบบของเดบิตการ์ดหรือบัตรชนิดพื้นฐานที่นำไปใช้เพื่อเป็นมูลค่าแทนเงินสดในการใช้งานโทรศัพท์สาธารณะภายในบัตรจะมีการบันทึกข้อมูลในรูปของ Unit Counter อยู่ภายในส่วนของหน่วยบันทำข้อมูลในบัตร (Memory Unit) โดยหลังจากการบันทึกข้อมูลลงยังบัตรโดยผู้ผลิตแล้วข้อมูลส่วนหนึ่งเช่น Customer Code, หมายเลขบัตร, วันหมดอายุของบัตร, จะถูกเขียนลงไปอย่างถาวรไม่สามารถแก้ไขได้ ในขณะที่ข้อมูลอีกส่วนหนึ่งคือ Unit Counter จะสามารถลดค่าลงได้เพียงอย่างเดียวไม่สามารถเพิ่มค่าได้ โครงสร้างของส่วนที่เก็บข้อมูล Unit Counter เป็นฟิวส์ขนาด 40 บิต (Logical Fuse) จากการใช้งานข้อมูลของ Unit Counter จะถูกลดค่าลงจนเป็นศูนย์แต่ข้อมูลอื่นๆในบัตรยังคงอยู่ในโครงนี้จะใช้ข้อมูลที่ยังคงอยู่นี้ประยุกต์ใช้ในงานระบบรักษาความปลอดภัย

2.7.1.4 โครงสร้างภายในบัตร SLE4436

บัตร SLE4436 บัตรชนิดนี้ถูกออกแบบมาเพื่อการใช้บริการพรีเพด (Pre-Paid) หรือบริการจ่ายเงินก่อนค่อยใช้บริการ โครงสร้างภายในชิปไอซีของบัตร SLE4436 โดยทั่วไปจะประกอบด้วยหน่วยความจำแบบ EEPROM ขนาด 221 บิต, หน่วยความจำ ROM ขนาด 16 บิต, ส่วนควบคุมและรักษาความปลอดภัยให้ข้อมูล (Control Security Unit), ส่วนประมวลผลเฉพาะสำหรับการรับรอง (Authentication)



รูปที่ 2.20 ไดอะแกรมแสดงส่วนการทำงานภายในสมาร์ตการ์ดตระกูล SLE4436

สำหรับบัตร SLE4436 ที่ถูกนำมาผ่านกระบวนการ เพื่อใช้เป็นบัตร TOT Card ภายในหน่วยบันทึกข้อมูล (Memory Unit) จะถูกบรรจุไว้เป็นข้อมูลขนาด 48 ไบต์ ข้อมูลจะถูกบันทึกตั้งแต่ในกระบวนการผลิตของโรงงาน ไม่สามารถเข้าไปเปลี่ยนแปลงหรือแก้ไขค่าข้อมูลได้ ที่สามารถแบ่งออกได้เป็น 5 ส่วน อันได้แก่

- **ข้อมูลชุดที่ 1** มีขนาด 3 ไบต์ เป็นข้อมูลที่ระบุถึง Factory Code มีไว้สำหรับระบุข้อมูลของบัตรไปสร้างแอพพิเคชันเพื่อให้บริการ เช่น ถ้าต้องการใช้เกี่ยวกับการทำบัตรนี้ทำบริการเครื่องขายสินค้าอัตโนมัติก็จะมีการระบุรหัสประจำตัวของผู้ให้บริการ เพื่อให้สามารถแยกแยะตัวผู้ให้บริการได้
- ข้อมูลชุดที่ 2 มีขนาด 5 ไบต์ เป็นข้อมูลหมายเลขบัตรซึ่งถูกเก็บอยู่ในรูปของรหัส BCD โดยเป็นตัวเลขขนาด 10 หลัก
- ข้อมูลชุดที่ 3 มีขนาด 5 ไบต์ เป็น Balance Unit หรือมูลค่าตัวเงินของบัตร การนับมูลค่าเงินหรือ Balance Counter จะมีวิธีการคำนวณตามการใช้งาน
- ข้อมูลชุดที่ 4 มีขนาด 32 ไบต์ เป็นข้อมูลลับของทางผู้ผลิตสมาร์ตการ์ด
- **ข้อมูลชุดที่ 5** มีขนาด 1 ไบต์ เป็นข้อมูลของปีและเดือนที่บัตรจะหมดอายุ

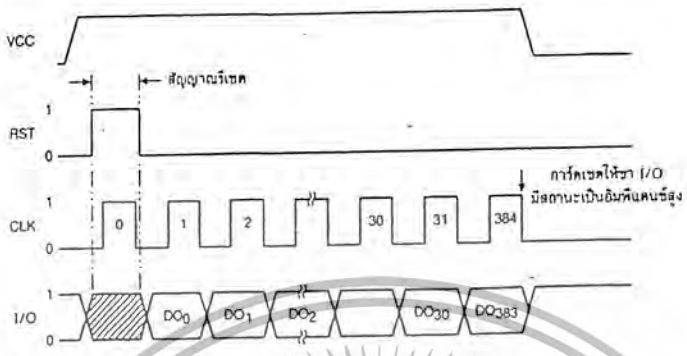
การจัดเก็บข้อมูลภายในหน่วยบันทึกข้อมูลของบัตร SLE4436				
ข้อมูลไบต์ที่ 0-2	ข้อมูลไบต์ที่ 3-7	ข้อมูลไบต์ที่ 8-12	ข้อมูลไบต์ที่ 13-46	ข้อมูลไบต์ที่ 47
Customer Code	Card Number	Balance Counter	ข้อมูลที่เป็นความลับของผู้ผลิต	วันหมดอายุของบัตร

ตารางที่ 2.2 โครงสร้าง และรายละเอียดที่เกี่ยวกับข้อมูลทั้ง 48 ไบต์ ในบัตร SLE4436

2.7.1.5 การติดต่อกับการ์ด SLE4436

จ่ายไฟตรงให้แก่บัตร กำหนดให้สัญญาณที่ขา Clock เป็นลอจิกสูง กำหนดให้สัญญาณที่ขา RST เป็นลอจิกสูงประมาณ 10 uS แล้วให้ลอจิกต่ำที่ขาเป็นเวลาประมาณ 10 uS จากนั้น กำหนดให้สัญญาณที่ขา Clock เป็นลอจิกสูงค้างไว้นานประมาณ 5 uS ให้อ่านข้อมูลจากขา I/O โดยเลื่อนบิตข้อมูลที่อ่านได้ไปทีละ 1 บิต ซึ่งหมายความว่าเมื่อป้อนสัญญาณ Clock ครบ 8 ลูก อ่านข้อมูลจากขา I/O เลื่อนข้อมูลและเก็บค่าจนครบ 8 บิต ข้อมูลที่ได้มาสุดท้ายก็คือข้อมูลขนาด 1 ไบต์ (ข้อมูลบิตแรกที่ได้ เป็น MSB) ใช้รูปแบบของสัญญาณตามที่ได้กล่าวมาจนได้สัญญาณที่ขา Clock ครบ 384 ลูก หรือเทียบได้เป็นข้อมูล 48 ไบต์ (เนื่องจากเป็นการอ่านข้อมูลแบบอนุกรมจึงต้องอ่านข้อมูลออกมาทั้ง 48 ไบต์)

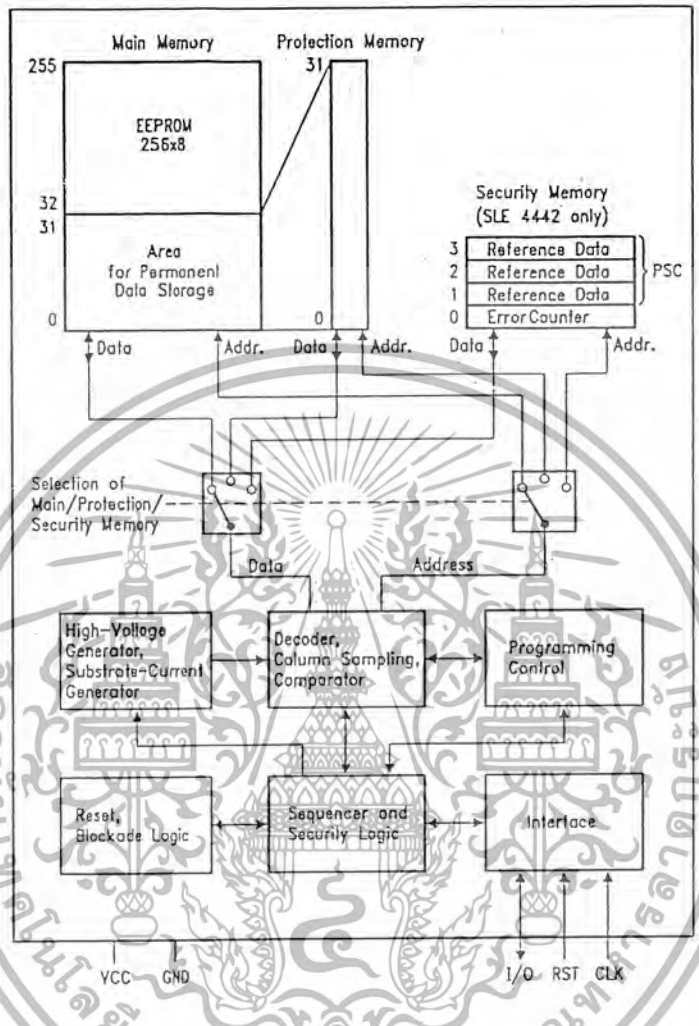
หลังจากนั้นจึงนำข้อมูลที่ได้มาแยกแยะว่าส่วนใดคือข้อมูลอะไรโดยอ้างอิงจากตารางที่ 1.2



รูปที่ 2.21 แผนผังทางเวลาของสัญญาณที่เกี่ยวข้องสำหรับการอ่านข้อมูลจากบัตร SLE4436

2.7.2 การ์ดที่มีระบบป้องกันข้อมูล SLE4442

การ์ดที่มีระบบป้องกันความปลอดภัยข้อมูลหรือ Security Memory Card คือสมาร์ตการ์ดที่การอ่านข้อมูลสามารถทำได้อย่างอิสระ แต่การเขียนข้อมูลจะไม่สามารถทำได้หากไม่มีรหัสผ่านหรือรหัส PSC ที่ถูกต้อง วิธีการในลักษณะนี้ช่วยให้ข้อมูลภายในสมาร์ตการ์ดได้รับการปกป้องและมีความน่าเชื่อถือ จุดนี้เองเป็นส่วนที่ทำให้ Security Memory Card แตกต่างไปจาก Free Access Memory Card อย่างชัดเจน รูปแบบการสื่อสารข้อมูลของการ์ดชนิดนี้เป็นการสื่อสารข้อมูลแบบซิงโครนัส (Synchronous) ตามมาตรฐาน ISO7816 ซึ่งรูปแบบคำสั่งที่ใช้ในการติดต่อและควบคุมการ์ดจะแตกต่างกันไปในผู้ผลิตการ์ดแต่ละราย (ข้อมูลส่วนนี้สามารถดูได้จากเอกสารของผู้ผลิตการ์ด) ซึ่งภายในหัวข้อนี้จะอ้างอิงจากสมาร์ตการ์ด SLE4442 ของบริษัทSiemens เนื่องจากเป็นการ์ดที่มีคุณสมบัติในการรักษาความปลอดภัยข้อมูลอย่างครบถ้วนและสามารถหามาใช้งานได้ง่ายในบ้านเรา



รูปที่ 2.22 บล็อกไดอะแกรมแสดงโครงสร้างภายในของ SLE4442

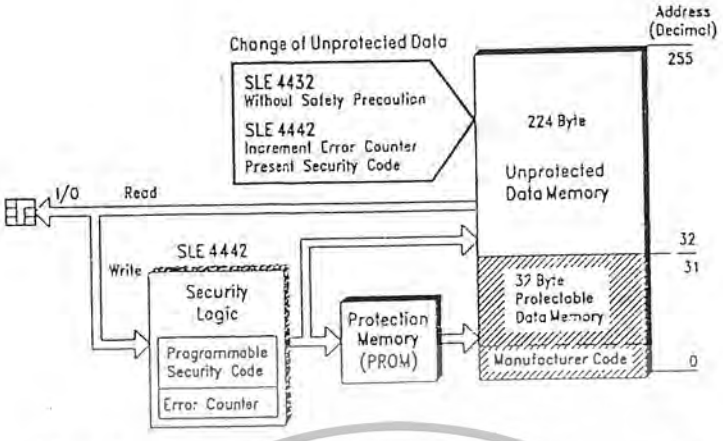
คุณสมบัติโดยทั่วไปของ SLE4442

- ใช้หน่วยความจำ EEPROM 8บิต ความจุข้อมูล 256 ไบต์
- ใช้รูปแบบของ ATR (Answer To Reset) ตามมาตรฐาน ISO7816-3
- อินเตอร์เฟสแบบซิงโครนัส(Synchronous) ตามมาตรฐาน ISO7816
- ป้องกันการเขียนข้อมูลด้วยรหัสผ่าน PSC (Programmable Security Code)
- การลบและเขียนข้อมูลในแต่ละไบต์ใช้เวลาเพียง 2.5 มิลลิวินาที
- มีฟังค์ชันป้องกันข้อมูลในพื้นที่หน่วยความจำ 32 ไบต์แรก โดยสามารถจะกำหนดให้ข้อมูลที่เขียนลงไปยังพื้นที่ช่วงดังกล่าวถูกเขียนลงไปอย่างถาวรได้

จากไดอะแกรมในรูปที่ 2.22 จะเห็นได้ว่าหน่วยความจำขนาด 256 ไบต์ ที่อยู่ภายใน SLE4442 จะถูกแบ่งออกเป็น 2 ส่วนด้วยกัน ได้แก่ข้อมูลในช่วง 32 ไบต์แรกซึ่งเป็นพื้นที่ที่มีระบบป้องกันการเขียนข้อมูลทับ และหน่วยความจำส่วนถัดมาซึ่งเป็นอีอีพรอม (EEPROM) ที่สามารถทั้งเขียนและอ่านได้ กลไกในการป้องกันข้อมูลของ SLE4442 มาจากส่วนที่เป็น Security Memory (ในรูปที่ 2.22) ที่ได้รับการปกป้องโดยข้อมูลสำคัญ 2 ส่วน คือ

- Reference Data(PSC) เป็นข้อมูลขนาด 3 ไบต์ที่เก็บค่าของรหัสผ่านสำหรับการเข้าไปแก้ไขข้อมูลในหน่วยความจำเอาไว้ (รหัส PSC ไม่สามารถถูกอ่านออกมาได้) รหัส PSC จะถูกกำหนดเป็นค่าหนึ่งมาโดยผู้ผลิตก่อนซึ่งสามารถจะมาปรับเปลี่ยนเองได้ภายหลังเมื่อใช้งาน
- Error Counter Byte เป็นข้อมูลที่บอกถึงจำนวนครั้งที่ป้อนรหัส PSC ผิดซึ่งจะถูกกำหนดเอาไว้ตายตัวว่าจะผิดได้ไม่เกิน 3 ครั้ง หากเกินกว่านั้นการ์ดจะล็อกตัวเองอย่างถาวรทันทีและไม่มีทางปลดล็อกได้ แม้จะป้อนรหัส PSC ที่ถูกต้องไปแล้วก็ตาม การเขียนข้อมูลยังหน่วยความจำก็จะไม่สามารถทำได้อีกต่อไป แต่ยังคงอ่านข้อมูลออกมาได้ตามปกติ การป้อนรหัส PSC ผิดแต่ละครั้ง Error Counter จะถูกลดลงไป 1 ค่าทันที ถ้าหากค่า Error Counter ถูกลดจนมีค่าเป็น 0 เมื่อไรก็แสดงว่าการ์ดได้ถูกล็อกไปเรียบร้อยแล้ว (ในกรณีที่ป้อนรหัส PSC ผิดมาแล้ว 2 ครั้ง แต่ป้อนรหัสถูกในครั้งที่ 3 ค่าของError Counter จะถูกรีเซตกลับไปเป็น 3 ครั้งเหมือนอย่างตอนแรกเริ่ม)

รูปที่ 2.23 เป็นไดอะแกรมที่แสดงภาพรวมของ SLE4442 จะเห็นได้ว่าการอ่านข้อมูลจากหน่วยความจำนั้น เราสามารถจะอ่านข้อมูลออกมาได้โดยไม่ต้องผ่านขั้นตอนการป้อนรหัส PSC แต่สำหรับการเขียนข้อมูลแล้ว เราจะต้องป้อนรหัส PSC ที่ถูกต้องเสียก่อน เพื่อเปิดลอจิกในการเขียนข้อมูลลงยังหน่วยความจำ นอกจากนั้นก็จะเห็นได้ว่าข้อมูล 4 ไบต์แรก เป็นข้อมูลของผู้ผลิต หรือ Manufacturer Code มีขนาด 4 ไบต์ พื้นที่ส่วนนี้ใช้เก็บข้อมูลของATR โดยความหมายของข้อมูลที่อยู่ในพื้นที่ส่วนนี้แต่ละไบต์จะถูกกำหนดโดยบริษัทผู้ผลิตการ์ดแต่ละราย



รูปที่ 2.23 ไดอะแกรมแสดงภาพรวมของ Security Memory Card

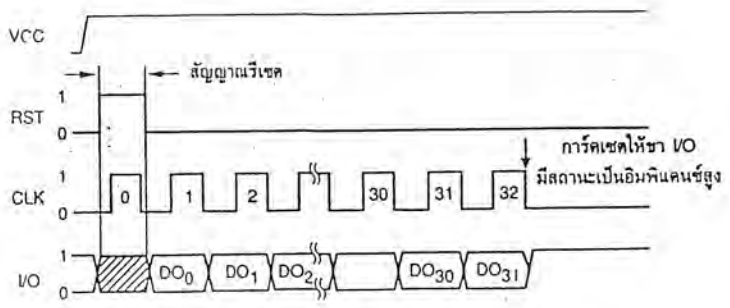
รูปแบบการสื่อสารข้อมูลของสมาร์ตการ์ด SLE4442

รูปแบบการสื่อสารข้อมูลของสมาร์ตเบอร์ SLE4442 เป็นการรับส่งข้อมูลระหว่างเครื่องอ่านและสมาร์ตการ์ดแบบ 2 ทิศทาง (ข้อมูลบนสาย I/O จะถูกอ่านค่าที่ขอบขาลงของสัญญาณนาฬิกา) โดยรูปแบบการสื่อสารที่ว่านี้ประกอบด้วย 4 โหมดการทำงาน ได้แก่

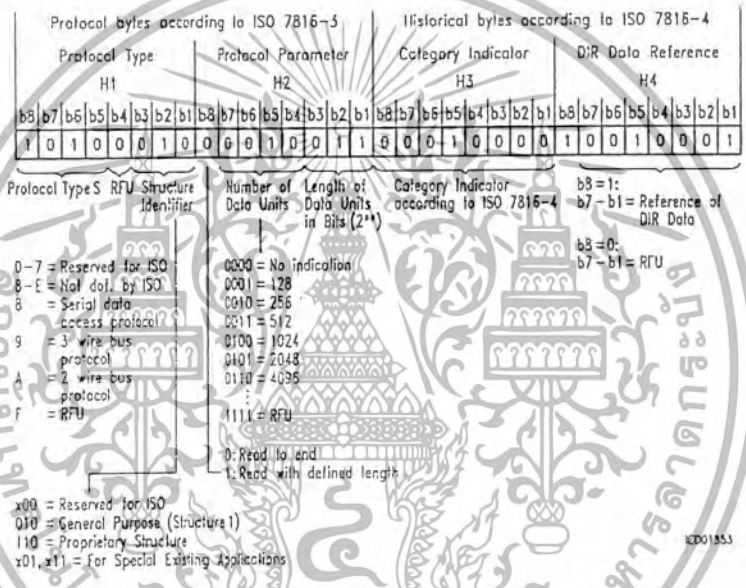
- การรีเซตและการตอบกลับด้วย ATR (Answer To Reset)
- โหมดการส่งคำสั่ง (Command Mode)
- โหมดการอ่านข้อมูล (Out-going Data Mode)
- โหมดการดำเนินการ (Processing Mode)

การรีเซตและการตอบกลับ

การอินเตอร์เฟสเข้ากับ Security Memory Card ทั่วๆไป รวมทั้ง SLE4442 จะสอดคล้องกับมาตรฐานในการอินเตอร์เฟสแบบซิงโครนัสภายในมาตรฐาน ISO7816 โดยเมื่อรีเซตการทำงานของการ์ดจะทำให้การ์ดมีการตอบกลับด้วยข้อมูล ATR สำหรับข้อมูลที่ตอบกลับมาจาก SLE4442 จะประกอบด้วยข้อมูล 4 ไบต์ การอ่านข้อมูลที่ว่านี้สามารถทำได้โดยอ้างอิงจากสัญญาณในรูปที่ 2.24 สำหรับโครงสร้างข้อมูล ATR ของสมาร์ตการ์ด SLE4442 ถูกแสดงอยู่ในรูปที่ 2.25 (ในกรณีของ Free Access Memory จะไม่มีข้อมูลของ ATR เก็บอยู่ภายใน)



รูปที่ 2.24 รูปสัญญาณของการรีเซตและการตอบกลับ ATR



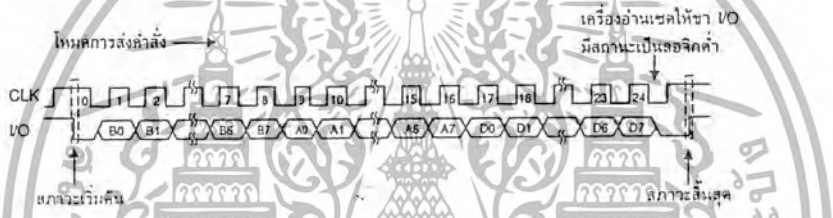
รูปที่2.25 โครงสร้างของข้อมูล ATR ในหน่วยความจำ 4 ไบต์แรกของ SLE4442

Byte 1 Control								Byte 2 Address	Byte 3 Data	Operation	Mode
B7	B6	B5	B4	B3	B2	B1	B0	A7-A0	D7-D0		
0	0	1	1	0	0	0	0	address	no effect	READ MAIN MEMORY	outgoing data
0	0	1	1	1	0	0	0	address	input data	UPDATE MAIN MEMORY	processing
0	0	1	1	0	1	0	0	no effect	no effect	READ PROTECTION MEMORY	outgoing data
0	0	1	1	1	1	0	0	address	input data	WRITE PROTECTION MEMORY	processing
0	0	1	1	0	0	0	1	no effect	no effect	READ SECURITY MEMORY	outgoing data
0	0	1	1	1	0	0	1	address	input data	UPDATE SECURITY MEMORY	processing
0	0	1	1	0	0	1	1	address	input data	COMPARE VERIFICATION DATA	processing

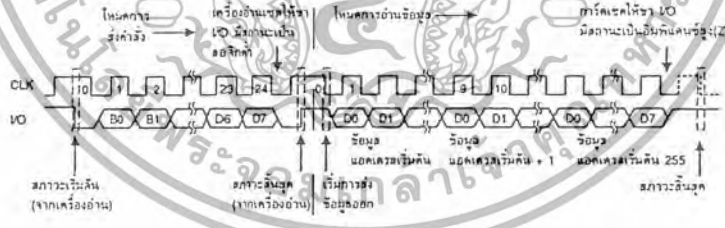
รูปที่ 2.26 โครงสร้างและความหมายของชุดคำสั่งที่ SLE4442 รองรับ

การทำงานของโหมดการส่งคำสั่ง

การส่งคำสั่งไปยังสมาร์ตการ์ดหรือการทำงานในโหมดการส่งคำสั่ง ก็คือกระบวนการต่อเนื่องหลังจากการ์ดฟถูกรีเซตไปเรียบร้อยแล้ว โดยการ์ดจะรอรับคำสั่งที่ส่งมาจากเครื่องอ่านซึ่งมีรูปแบบเป็นข้อมูลความยาว 3 ไบต์ โครงสร้างของข้อมูลดังกล่าวประกอบด้วยคำสั่ง(Command), แอดเดรส(Address) และข้อมูล(Data) โดยคำสั่งทั้งหมดที่การ์ดSLE4442 รองรับถูกแสดงอยู่ในรูปที่ 2.26 ส่วนรูปสัญญาณที่จะเกิดขึ้นระหว่างการทำงานของโหมดการส่งคำสั่งก็เป็นดังรูป2.27 จะเห็นได้ว่าการส่งข้อมูลแต่ละครั้งจะต้องมีการส่งสภาวะเริ่มต้นและสภาวะสิ้นสุดกำกับไปกับตัวข้อมูลด้วย ในที่นี้สภาวะเริ่มต้นก็คือการเปลี่ยนระดับจากลอจิกค่าสูงเป็นต่ำที่ขาI/O ในขณะที่ระดับลอจิกที่ขา CLK เป็นค่าสูง , ส่วนภาวะสิ้นสุดก็คือการเปลี่ยนระดับจากลอจิกค่าต่ำเป็นสูงที่ขา I/O ในขณะที่ระดับลอจิกที่ขา I/O ในขณะที่ระดับลอจิกที่ขา CLK เป็นค่าสูง ต่อไปคือความหมายและวิธีการทำงานของแต่ละคำสั่ง



รูปที่ 2.27 รูปสัญญาณของการส่งคำสั่งไปยังการ์ด

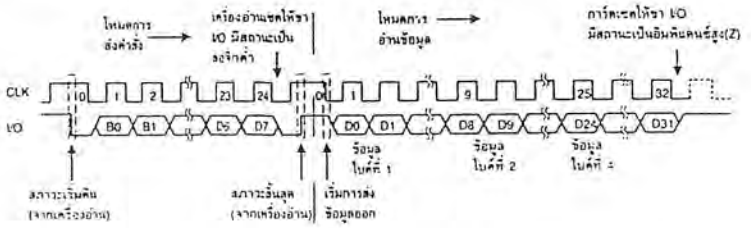


รูปที่ 2.28 รูปสัญญาณของคำสั่ง Read Main Memory

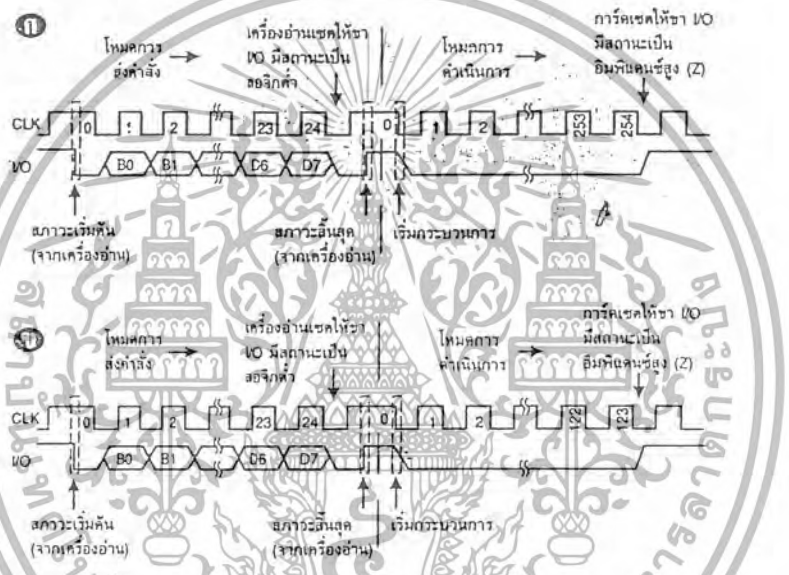
- Read Main Memory คือคำสั่งที่ใช้ในการอ่านข้อมูลทั้งหมดออกมาจากหน่วยความจำของการ์ด ทั้งจากพื้นที่ส่วนที่ได้รับการป้องกัน (หน่วยความจำ 32 ไบต์แรก) และส่วนที่ไม่ได้รับการป้องกัน (หน่วยความจำ 224 ไบต์หลัง) โดยจะเป็นการอ่านค่าโดยเริ่มต้นจากแอดเดรสที่ส่งไปจนถึงแอดเดรสสุดท้าย (0ffh) ของพื้นที่หน่วยความจำ
- Read Protection Memory คือคำสั่งที่ใช้ในการอ่านข้อมูลทั้งหมดออกมา จากหน่วยความจำ 32 ไบต์แรก

- Update Main Memory คือคำสั่งที่ใช้ในการเขียนข้อมูลยังแอดเดรสใดๆ ของหน่วยความจำทั้ง 256 ไบต์ ในกรณีที่ใช้คำสั่งนี้ในการเขียนข้อมูลลงยังหน่วยความจำ 32 ไบต์แรก ข้อมูลจะยังคงแก้ไขเปลี่ยนแปลงได้ภายหลัง สำหรับการเขียนข้อมูลจะประกอบด้วย 3 เงื่อนไข คือ
  - ✓ การลบข้อมูลที่แอดเดรสของหน่วยความจำที่กำหนดให้เป็น 0FFH แล้วทำการเขียนข้อมูลซ้ำลงยังแอดเดรสเดิม กระบวนการนี้จะต้องใช้สัญญาณนาฬิกา 255 ลูก
  - ✓ การเขียนข้อมูลที่แอดเดรสของหน่วยความจำที่กำหนดโดยไม่ต้องลบข้อมูลออก สำหรับแอดเดรสดังกล่าวจะต้องเป็นที่ว่าง (มีข้อมูลเป็น 0FFH) อยู่ก่อนหน้านี้แล้วเท่านั้น กระบวนการนี้จะใช้สัญญาณนาฬิกา 124 ลูก
  - ✓ การลบข้อมูลที่แอดเดรสของหน่วยความจำที่กำหนด (มีค่าข้อมูลเป็น0FFH) โดยไม่มีการเขียนข้อมูลต่อ สำหรับกระบวนการนี้จะใช้สัญญาณนาฬิกา 124 ลูก เช่นกัน
- Write Protection Memory คือการเขียนข้อมูลลงยังแอดเดรสของหน่วยความจำใดๆใน 32 ไบต์แรก คำสั่งนี้มีเงื่อนไขว่าข้อมูลที่เขียนลงไปจะถูกเขียนลงยังแอดเดรสของหน่วยความจำที่กำหนดอย่างถาวร ไม่สามารถแก้ไขเปลี่ยนแปลงอะไรได้อีก สำหรับรูปสัญญาณของกระบวนการนี้อ้างอิงได้จากรูปสัญญาณของคำสั่ง Update Main Memory
- Read Security Memory คือการอ่านค่าของ Error Counter เพื่อตรวจดูว่าการ์ดใบนั้นๆ ได้ถูกล็อกไปแล้วหรือยัง โดยค่าภายในบิต D2,D1 และ D0 ของError Counter จะเป็นส่วนที่บอกถึงสถานะของการ์ดในขณะนั้น หากค่าของบิต D2,D1 และ D0 เป็น 0 ทั้งหมดก็แสดงว่าการ์ดได้ถูกล็อคไปแล้ว ซึ่งจะไม่สามารถแก้ไขอะไรได้และจะไม่สามารถเขียนข้อมูลลงยังการ์ดนั้นได้อีกต่อไป (แต่ว่าการอ่านข้อมูลการ์ดจะยังคงทำได้อย่างตามปกติ)
- Update Security Memory คือการเข้าไปแก้ไขข้อมูลของรหัส PSC ภายในการ์ด หรือพูดง่ายๆ ก็คือการเข้าไปเปลี่ยนรหัสป้องกันของการ์ดนั่นเอง คำสั่งจะถูกกระทำต่อเมื่อมีการส่งรหัส PSC ที่ถูกต้องไปยังการ์ดเสียก่อน โดยในกรณีที่ป้อนรหัสผิด ค่าของบิต D2,D1 และ D0 ใน Error Counter จะค่อยๆถูกเปลี่ยนจากค่า "1" เป็น "0" ไล่ไปทีละบิตตามจำนวนครั้งที่ป้อนรหัสผิด หากทั้งหมดกลายเป็นศูนย์เมื่อไรการ์ดก็จะถูกล็อกทันทีซึ่งนั่นหมายความว่าโอกาสป้อนรหัสผิดจะมีเพียง 3 ครั้งเท่านั้น สำหรับรูปสัญญาณของกระบวนการนี้จะเหมือนกับรูปสัญญาณของคำสั่ง Update Main Memory
- Compare Verification Data คือการสั่งให้การ์ดทำการเปรียบเทียบรหัส PSC กับรหัสผ่านที่เราได้ส่งไปยังการ์ด ในการเปรียบเทียบที่ว่านี้ ข้อมูลที่การ์ดจะส่งกลับมาคือค่าของ Error

Counter ที่จะบอกว่ารหัสที่เราป้อนไปนั้นถูกต้องหรือไม่ และยังเหลือโอกาสพลาดอีกกี่ครั้งเท่านั้น (เราไม่สามารถจะเข้าไปอ่านค่ารหัส PSC ของการ์ดออกมาได้)



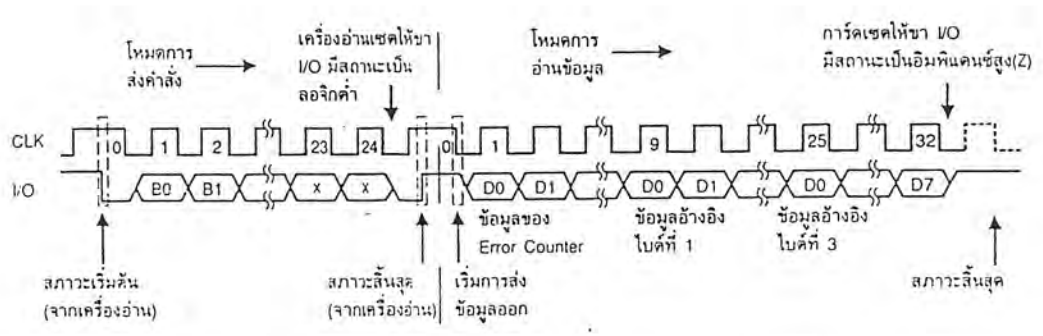
รูปที่ 2.29 รูปสัญญาณของคำสั่ง Read Protect Memory



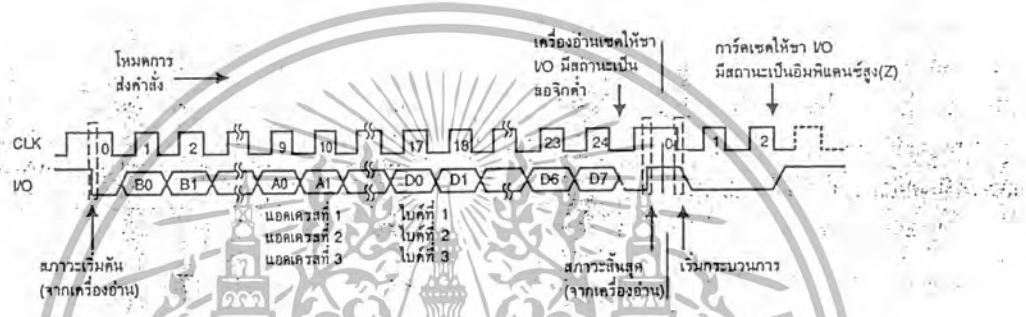
รูปที่ 2.30 รูปสัญญาณของคำสั่ง Update Main Memory (ก) การลบข้อมูลแล้วเขียนข้อมูลซ้ำ (ข) การลบหรือเขียนข้อมูล (อย่างใดอย่างหนึ่ง)

โหมดการอ่านข้อมูล(Outgoing Data Mode)

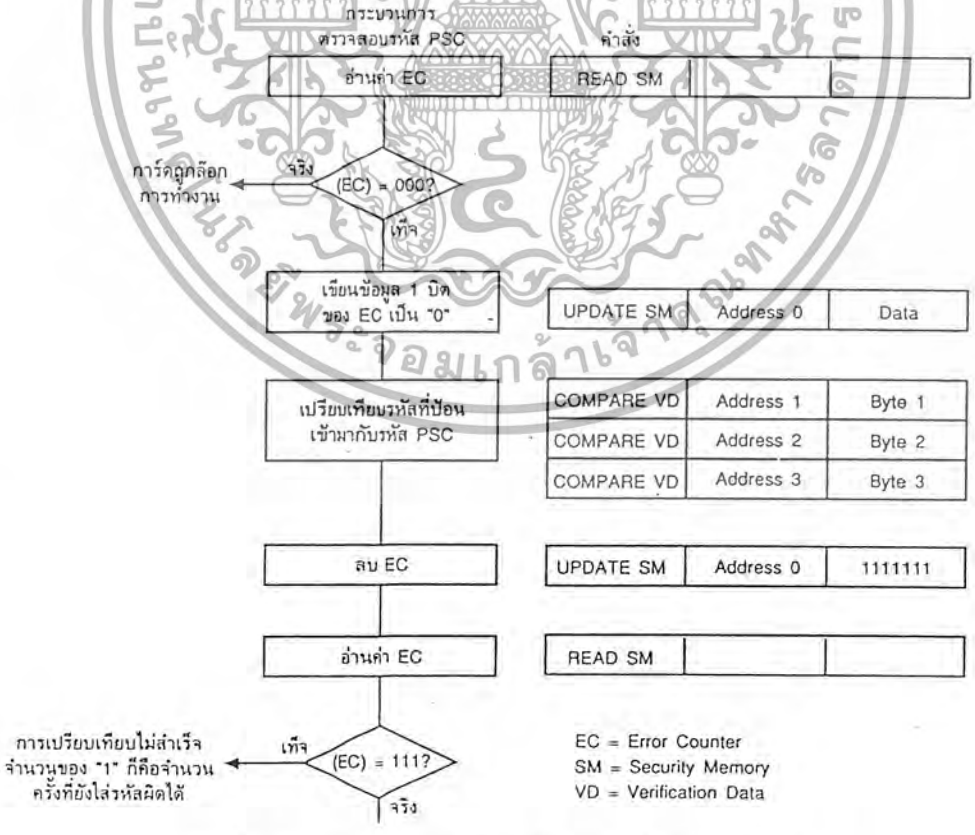
โหมดการทำงานนี้จะเกิดขึ้นหลังจากที่มีการส่งคำสั่งในกลุ่มของการขออ่านข้อมูล (เช่น Read Main Memory,Read Protection Memory และ Read Security Memory) ไปยังสมาร์ตการ์ดเพื่อขออ่านข้อมูลจากพื้นที่ใดๆในหน่วยความจำหลังจากได้รับคำสั่งดังกล่าวสมาร์ตการ์ดจะส่งข้อมูลที่ถูกร้องขอกลับมายังเครื่องอ่านซึ่งก็เท่ากับว่าเครื่องอ่านจะสามารถอ่านข้อมูลที่ต้องการออกมาได้สำเร็จจากโหมดการทำงานนี้



รูปที่ 2.31 รูปสัญญาณของคำสั่ง Read Security Memory



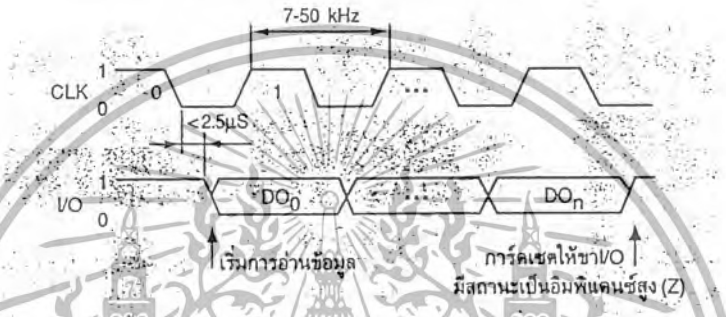
รูปที่ 2.32 รูปสัญญาณของคำสั่ง Compare Verification Data



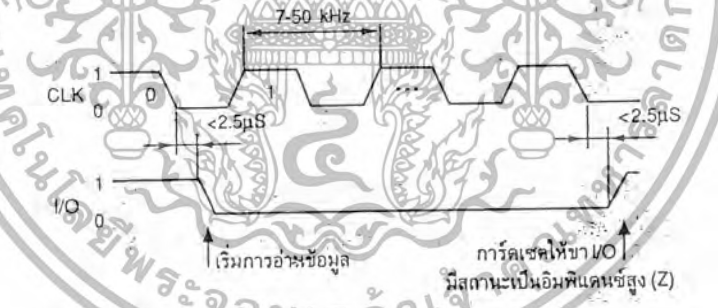
รูปที่ 2.33 กระบวนการเปรียบเทียบรหัสผ่านกับรหัส PSC

โหมดดำเนินการ(Processing Mode)

โหมดการดำเนินการจะเกิดขึ้นหลังจากที่มีการส่งคำสั่งในกลุ่มของการขอเขียนหรือลบข้อมูลออกจากพื้นทใดๆ ในหน่วยความจำ (เช่น Update Main Memory ,Write Protection Memory,Update Security Memory และ Compare Verification Data) โดยหลังจากได้รับคำสั่งดังกล่าว สมาร์ตการ์ดจะเริ่มดำเนินกระบวนการตามที่ได้รับการสั่งมาในระหว่างโหมดการทำงานนี้จะสังเกตว่าข้อมูลจากขา I/O จะไม่ถูกนำมาใช้ร่วมในการทำงานเลย (เนื่องจากมีสถานะเป็นลอจิกต่ำตลอดทั้งช่วง)



รูปที่ 2.34 รูปสัญญาณที่จะเกิดขึ้นในระหว่างโหมดการอ่านข้อมูล

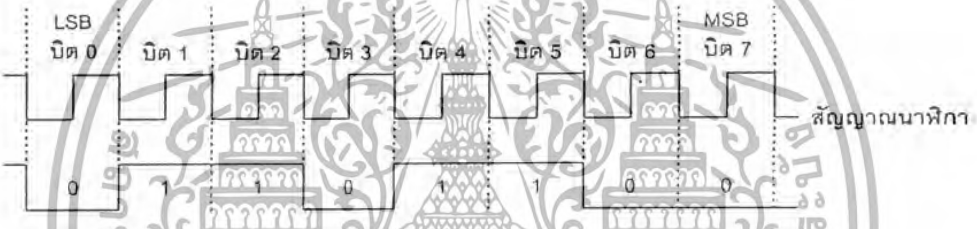


รูปที่ 2.35 รูปสัญญาณที่เกิดขึ้นในระหว่างโหมดดำเนินการ

บทที่ 3 การสื่อสารแบบอนุกรม

3.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นจะแบ่งออกได้เป็น 2 แบบคือการสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้ต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา , ข้อมูล และ กราวด์ รูปที่ 3.1 แสดงให้เห็นถึงไทมิ่งค์ไดอะแกรมของการส่งข้อมูล แบบ ซิงโครนัส



รูปที่ 3.1 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรม

3.2 การสื่อสารข้อมูลแบบอะซิงโครนัส

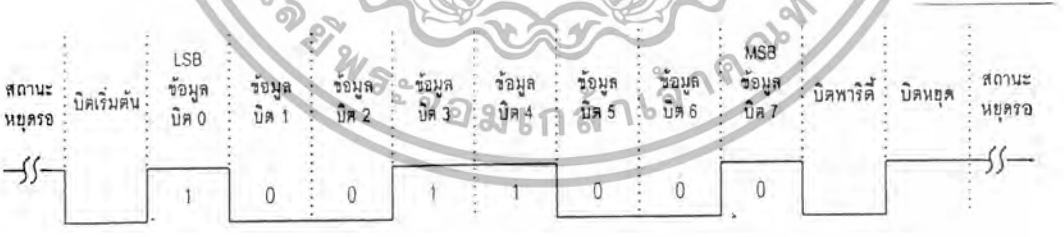
การสื่อสารข้อมูลแบบอะซิงโครนัส คือ การรับและส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนแบบการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บอดเรต (Baudrate) มีหน่วยเป็น บิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5 , 6 , 7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิต หรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1 , 1.5 หรือ 2 บิต

3.2 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีข้อมูลที่จะส่ง ขา Data จะมีสถานะลอจิก "1" ซึ่งจะเรียกสถานะนี้ว่าสถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา Data มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด (LSB) ก่อน ซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5 , 6 , 7 หรือ 8 บิตก็ได้ จากนั้นตามด้วย บิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่ส่งคือ บิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต , 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัส คือ บอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232 ได้แก่ 110 , 150 , 300 , 600 , 1200 , 2400 , 4800 , 9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามารถกำหนดค่าบอดเรตได้สูงถึง 115200 บิตต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่าง ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตเปิดปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็สามารถรับส่งข้อมูลได้ด้วยอัตราเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งจะเหลือ 872 ไบต์แต่วินาที



รูป 3.2 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่(odd) , แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิกสูง ภายในข้อมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิต และมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นว่า

ข้อมูลในไบต์นี้มีจำนวนลอจิก "1" จำนวน 4 ตัว ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดค่าพาริตี้เป็นคู่ ค่าในบิตพาริตี้ จะต้องมีลอจิกเป็น "0" แต่ถ้าพาริตี้เป็นคี่ ค่าที่บิตพาริตี้จะต้องเป็น "1" เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้มีจำนวนบิตที่ลอจิก "1" รวมกันเป็นเลขคี่ ในตาราง 3.1 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART โดยภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ให้ตรงกันว่า จะตรวจสอบพาริตี้คี่หรือคู่ จากนั้นภาครับของ UART จะตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือ เป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแจ้งข้อผิดพลาดให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดของการถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	1	0
11111111	0	1

ตารางที่ 3.1 แสดงบิตพาริตี้ของข้อมูล

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่นXT ใช้ UART เบอร์ 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5 V) แต่เพื่อให้แรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้นโดยลอจิก "0" มีระดับแรงดัน +3V ถึง +12 V ในขณะที่ลอจิก "1" มีระดับแรงดัน -3 V ถึง -12 V

3.3 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่า สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ -3 ถึง -12 V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12 V แสดงว่าเป็นช่องว่าง(Space)

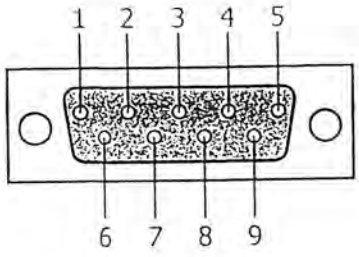
มาตรฐานRS-232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment: DTE) กับวงจรข้อมูลปลายทาง(Data Circuit Terminating: DCE) ไว้ว่า อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์ หรือ ไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสองจะกระทำผ่านมาตรฐาน RS-232

ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็คเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็คเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็คเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE.

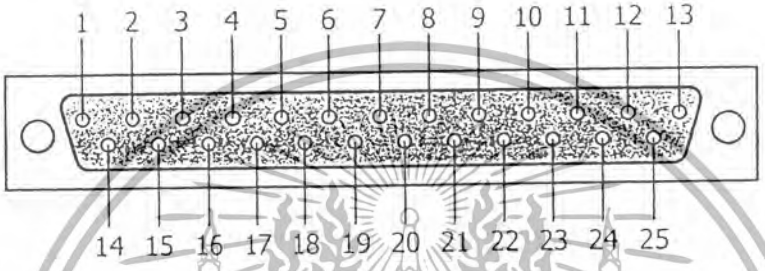
สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS-232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของสายสัญญาณสูงสุด ถึง 20 เมตร

3.3.1คอนเน็คเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็คเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ซึ่งคอนเน็คเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็คเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 3.3 และตาราง 3.2



รูปที่ 3.3 ก



รูปที่ 3.3 ข

รูปที่ 3.3 คอนเนคเตอร์อนุกรม

รูป3.3 ก คอนเนคเตอร์อนุกรม 9 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)

รูป3.3 ข คอนเนคเตอร์อนุกรม 25 ขาหรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์

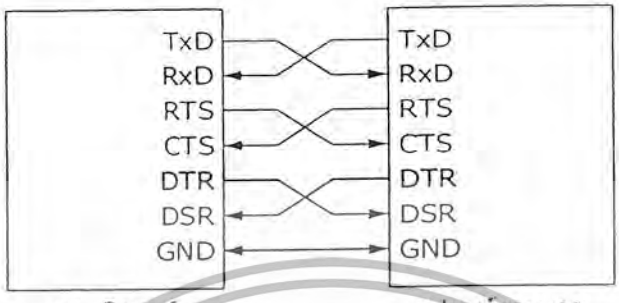
คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Signal Ground : GND	-
6	6	Data Set Ready : DSR	อินพุต
7	4	Request To Send : RTS	เอาต์พุต
8	5	Clear To Send : CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

ตารางที่ 3.2 หน้าที่การทำงานของขาอนุกรม

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 3.4 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 3.4 ก เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็คเต็มรูปแบบ ส่วนในรูปที่ 3.4 ข เป็นการเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้น โดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

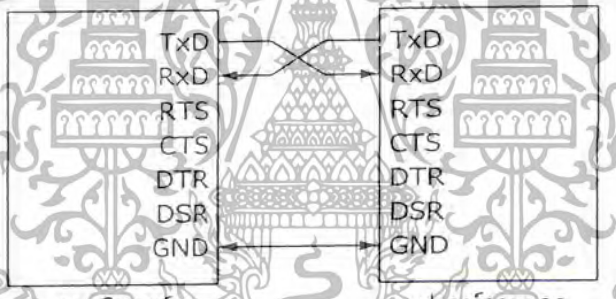
- Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect :CD ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก
- Receive Data :RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์ บัฟเฟอร์
- Transmitted Data :TD หรือ TxD ใช้ส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป
- Data Terminal Ready : DTR เป็นขาสัญญาณที่ส่งออกมาจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ Null modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์
- Signal Ground : GND กราวด์ระบบ
- Data Set Ready :DSR ขานี้จะใช้คู่กับขาDTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขาDSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขาDTR
- Request to Send :RTS เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณRTS ก็คือขาCTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null modem 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- Clear To Send :CTS ขานี้จะคอยรับสัญญาณจากขาRTS เมื่อรับสัญญาณได้ ข้อมูลที่ขาTxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

- Ring Indicator :RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น



การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์
แบบ Null modem

รูปที่ 3.4 ก



การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์
แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

รูปที่ 3.4 ข

รูปที่ 3.4 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่างๆ

3.3.2 UART (Universal Asynchronous Receiver Transmitter)

เป็นอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของUART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะ

ซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (บอดเรต) , รูปแบบการส่งข้อมูล , ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้ , เฟรมข้อมูล , โอเวอร์รัน) เป็นต้น

ภายใน UART จะมีส่วนของวงจรสร้างบอดเรตแบบโปรแกรมได้ (Programmable buadrate generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกา UART โดยตัวหารนี้มีขนาด 16 บิต ดังนั้นจึงสามารถกำหนดตัวหารอยู่ในช่วง 1 - 65,535 UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (half duplex) และฟูลดูเพล็กซ์ (full duplex) โดยการส่งแบบฮาล์ฟดูเพล็กซ์เป็นการส่งแบบทิศทางเดียว ส่วนการส่งแบบฟลูดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน

3.3.3ชนิดของ UART

ในเครื่องคอมพิวเตอร์ทั่วไปมีUART ที่ใช้งานกันอยู่ 2 เบอร์ คือ 8250 ซึ่งเป็นUART มาตรฐานที่มีใช้กันมายาวนาน UARTเบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตัวแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่UART เบอร์นี้ก็ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุกๆรุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของUART เบอร์นี้

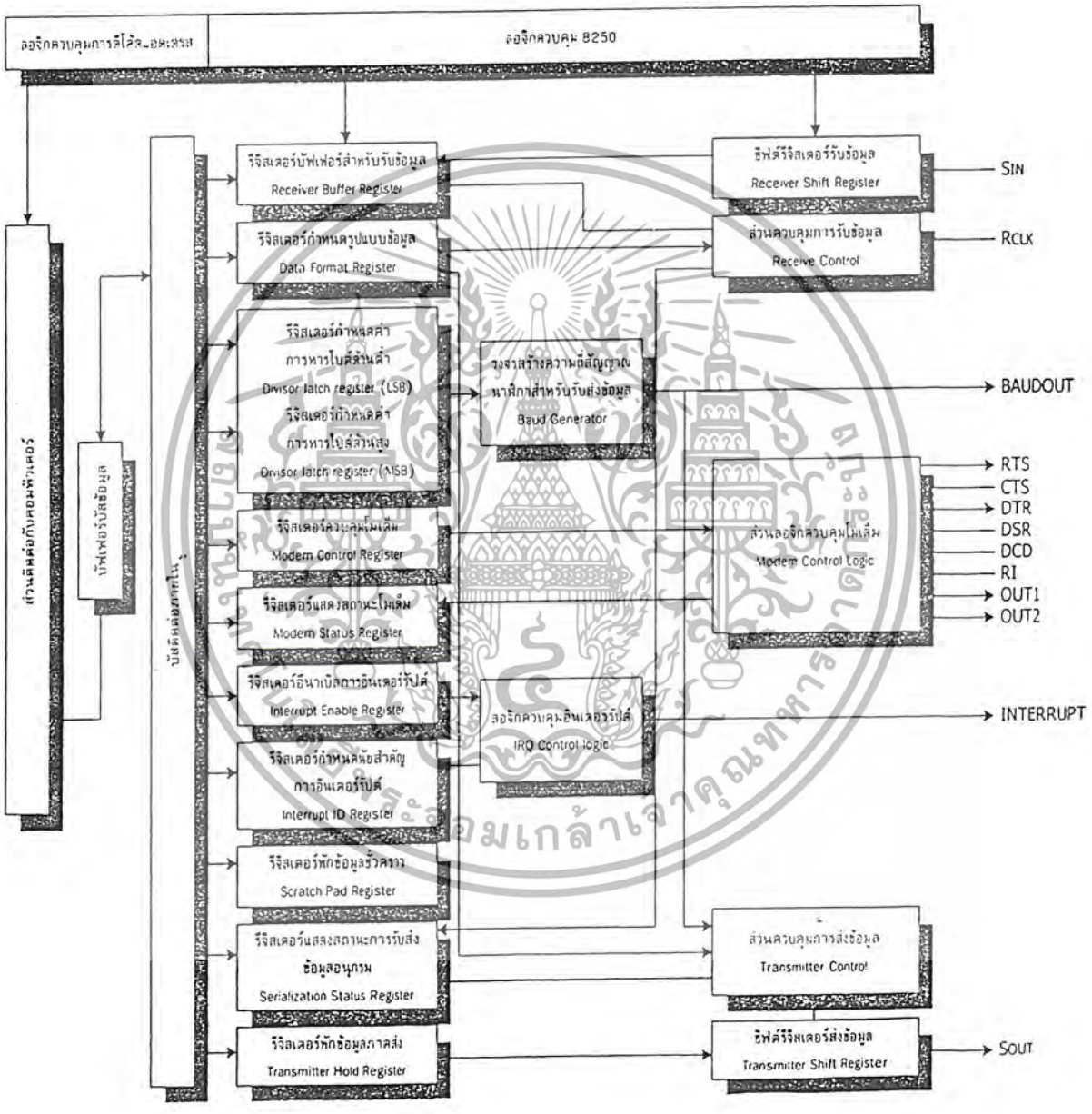
UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 115,200 บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO (First IN First Out) ขนาด 16 ไบต์เข้าไปทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบFIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5V และ +3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อมูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาทีเมื่อใช้สัญญาณนาฬิกา 16 MHz

อย่างไรก็ตาม ความเร็วในการส่งข้อมูลที่มากมายของ UART เบอร์ใหม่ ๆ ก็ไม่ได้ช่วยให้การรับส่งข้อมูลของคอมพิวเตอร์เร็วขึ้น เนื่องจากว่าคอมพิวเตอร์ยังใช้ความถี่ของสัญญาณนาฬิกาในการแปลงข้อมูลเพียง 1.8432 MHZ เท่านั้น

3.4วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรมสูงสุดได้ 4 พอร์ต มีชื่อเรียกเป็น COM1,COM2,COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน ในรูปที่ 3.5 แสดงผังการทำงานภายในของพอร์ตอนุกรม

ประกอบด้วยรีจิสเตอร์ 8 บิต 8 ตัว ที่ใช้งานร่วมกับ UART แอดเดรสของรีจิสเตอร์ภายในพอร์ตอนุกรม สามารถคำนวณได้จากค่ารีจิสเตอร์พื้นฐานของพอร์ตอนุกรม ยกตัวอย่าง พอร์ตอนุกรม COM1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจิสเตอร์ต่างๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้



รูปที่ 3.5 ไดอะแกรมการทำงานภายในของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

00H เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้าหรือเตรียมข้อมูลก่อนที่จะส่งออกไป

01H รีจิสเตอร์เอ็นเนเบิลการอินเตอร์รัปต์ ใช้เซตโหมดการอินเตอร์รัปต์ของพอร์ตอนุกรม

02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดของการอินเตอร์รัปต์

03H รีจิสเตอร์กำหนดรูปแบบขอข้อมูล

04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR

05H รีจิสเตอร์แสดงสถานการณ์รับและการส่งข้อมูลแบบอนุกรม

06H รีจิสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขา DCD ,RI,DSR และ CTS

07H รีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราว

3.4.1 รีจิสเตอร์ตำแหน่ง 00H (รีจิสเตอร์บัฟเฟอร์)

เป็นรีจิสเตอร์เก็บข้อมูลที่รับเข้ามาและส่งออก โดยการติดต่อกับรีจิสเตอร์นี้เพื่อเก็บข้อมูลจะต้องกำหนดให้บิตDLAB ในรีจิสเตอร์กำหนดรูปแบบข้อมูล(03H) มีสถานะเป็น "0" ซึ่งการเขียนข้อมูลมายังแอดเดรสนี้เป็นการส่งข้อมูลไปยังรีจิสเตอร์ส่งข้อมูลและข้อมูลจะถูกส่งออกไปแบบอนุกรม สำหรับการรับข้อมูล เมื่อรับเข้ามาแล้ว จะส่งต่อไปยังรีจิสเตอร์เก็บข้อมูล หลังจากอ่านค่าจากรีจิสเตอร์นี้ออกไป รีจิสเตอร์นี้จะถูกเคลียร์ และเตรียมพร้อมสำหรับรับข้อมูลในไบต์ต่อไป

3.4.2 รีจิสเตอร์ตำแหน่ง 01H (รีจิสเตอร์เอ็นเนเบิลการอินเตอร์รัปต์)

เป็นรีจิสเตอร์สำหรับการเอ็นเนเบิลการอินเตอร์รัปต์ ซึ่งเป็นการกำหนดให้UART สร้างสัญญาณอินเตอร์รัปต์ขึ้นมา ฟังก์ชันการทำงานในแต่ละบิตของรีจิสเตอร์นี้มีดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	SINP	ERBK	TBE	RxRD

ตารางที่ 3.3 ฟังก์ชั่นการทำงานของรีจิสเตอร์ตำแหน่ง 01H

บิต 4-7 บิตเหล่านี้ไม่ถูกใช้งาน กำหนดให้เท่ากับ"0"

SINP เอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากเกิดจากเปลี่ยนสถานะที่ขาอินพุต CTS,DSR,DCD หรือขา RI

"1" เอ็นเอเบิลการอินเตอร์รัปต์

"0" ดิสเอเบิล

ERBK เอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากเกิดความผิดพลาดขึ้นด้วยสาเหตุจากพาริตี้ , โอเวอร์รัน , เฟรมข้อมูล หรือการเบรกข้อมูล

"1" เอ็นเอเบิลการอินเตอร์รัปต์

"0" ดิสเอเบิล

TBE เอ็นเอเบิลการอินเตอร์รัปต์เมื่อรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"1" เอ็นเอเบิลการอินเตอร์รัปต์

"0" ดิสเอเบิล

RxRD เอ็นเอเบิลการอินเตอร์รัปต์เนื่องจากรีจิสเตอร์บัฟเฟอร์ได้รับข้อมูลแล้ว

"1" เอ็นเอเบิลการอินเตอร์รัปต์

"0" ดิสเอเบิล

3.4.3 รีจิสเตอร์ตำแหน่ง 02H (รีจิสเตอร์แสดงโหมดและสถานะการอินเตอร์รัปต์)

มีรายละเอียดของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	ID1	ID0	PND

ตารางที่ 3.4 ฟังก์ชั่นการทำงานของรีจิสเตอร์ 02H

บิต 3-7 ไม่ได้ใช้งาน อ่านค่าได้เท่ากับ "0"

ID1,ID0 ใช้งานร่วมกันเพื่อแจ้งสาเหตุของการเกิดอินเตอร์รัปต์

"00" เกิดการอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงของขาอินพุตขึ้น การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 4

"01" เกิดการอินเตอร์รัปต์เนื่องจากรีจิสเตอร์บัฟเฟอร์ส่งข้อมูลว่างขึ้น การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 3

"10" เกิดการอินเตอร์รัปต์เนื่องจากข้อมูลถูกเก็บลงในรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเรียบร้อยแล้ว การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 2

"11" เกิดการอินเตอร์รัปต์เนื่องจากความผิดพลาดในการถ่ายทอดข้อมูล หรือเกิดการเบรก (Break :เกิดการหยุดถ่ายทอดข้อมูลกระทันหัน) การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 1 หรือมีนัยสำคัญสูงสุด

PND ใช้แสดงสถานะของการเกิดอินเตอร์รัปต์

"1" แสดงว่าไม่มีการอินเตอร์รัปต์

"0" แสดงว่ามีการอินเตอร์รัปต์เกิดขึ้น

เมื่อมีการสร้างสัญญาณอินเตอร์รัปต์ขึ้น จะต้องมีการเคลียร์ค่าก่อนที่จะให้เกิดอินเตอร์รัปต์ครั้งต่อไป โดยสามารถทำได้ดังนี้ คือ

- ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงของขาอินพุตจะต้องอ่านค่าจากรีจิสเตอร์แสดงสถานะของโมเด็ม (รีจิสเตอร์ตำแหน่ง 06H) เพื่อเคลียร์ค่าการอินเตอร์รัปต์
- ถ้าเกิดการอินเตอร์รัปต์เนื่องจากบัฟเฟอร์ส่งข้อมูลว่าง จะต้องเขียนข้อมูลไปยังรีจิสเตอร์บัฟเฟอร์ข้อมูล (รีจิสเตอร์ตำแหน่ง 00H) หรืออ่านค่ารีจิสเตอร์แสดงสถานะอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง 02H)เพื่อเคลียร์ค่าการอินเตอร์รัปต์
- ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเก็บข้อมูลลงในรีจิสเตอร์บัฟเฟอร์สำหรับข้อมูลเรียบร้อย จะต้องเคลียร์ค่าอินเตอร์รัปต์โดยการอ่านข้อมูลจากรีจิสเตอร์บัฟเฟอร์
- ถ้าเกิดอินเตอร์รัปต์เนื่องจากความผิดพลาดในการรับส่งข้อมูลหรือเกิดการเบรก จะต้องเคลียร์ค่าอินเตอร์รัปต์โดยการอ่านค่ารีจิสเตอร์แสดงสถานการณ์รับและส่งข้อมูลแบบอนุกรม

3.4.4 รีจิสเตอร์ตำแหน่ง 03H (รีจิสเตอร์กำหนดรูปแบบของข้อมูล)

มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DLAB	BRK	PAR2	PAR1	PAR0	STOP	DAB1	DAB0

ตารางที่ 3.5 ฟังก์ชั่นการทำงานของรีจิสเตอร์ 03H

DLAB ใช้ในการกำหนดหน้าที่การทำงานของรีจิสเตอร์บัฟเฟอร์ (00H)

"1" เป็นการเข้าสู่โหมดการหารค่าบอดเรต

"0" เป็นการเข้าถึงรีจิสเตอร์บัฟเฟอร์(รีจิสเตอร์ตำแหน่ง 00H) และรีจิสเตอร์สำหรับเอ็นเอเบิลการอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง01H) เมื่อบิต DLAB เป็น "1" รีจิสเตอร์บัฟเฟอร์ (00H) และรีจิสเตอร์เอ็นเอเบิลการอินเตอร์รัปต์ (01H) จะใช้สำหรับโหลดค่าการหารความถี่สำหรับกำหนดค่าบอดเรต

โดยรีจิสเตอร์ 00H เก็บค่าตัวหารไบต์ต่ำ ส่วนรีจิสเตอร์ 01H ใช้เก็บค่าตัวหารไบต์สูง การหาค่าบอดเรตสามารถเขียนเป็นสมการได้ดังนี้

บอดเรต = 115200/ ค่าตัวหาร16 บิต

ค่าตัวเลข 115200 มาจากความถี่คริสตอลในวงจร UART ภายในเครื่องคอมพิวเตอร์ โดยคริสตอลที่ใช้มีความถี่ 1.8432 MHz วงจรภายใน UART จะหารค่าความถี่นี้ด้วย 16 ทำให้ได้ค่าความถี่ 115200 Hz ออกมา

ค่าตัวหาร16บิต = ข้อมูลในรีจิสเตอร์00H + (256 * ข้อมูลในรีจิสเตอร์01H)

ถ้าต้องการบอดเรตเท่ากับ 9600 ค่าตัวหารที่ใช้จะต้องมีค่าเท่ากับ 12 ซึ่งค่านี้จะถูกเขียนลงในรีจิสเตอร์ 00H และเขียนค่า 0 ลงไปในรีจิสเตอร์ 01H ค่าตัวหารที่ทำให้เกิดค่าบอดเรตสูงสุดที่ 115200 บิตต่อวินาที คือ ค่า 0001 นั่นคือ รีจิสเตอร์ 00H มีค่าเท่ากับ 1 และรีจิสเตอร์ 01H มีค่าเท่ากับ 0

BRK ใช้ควบคุมการหยุดถ่ายทอดข้อมูล

"1" สามารถหยุดหรือเบรกได้

"0" ไม่มีการหยุดหรือเบรกได้

PAR2,PAR1,PAR0 ใช้เพื่อกำหนดบิตพาริตี้ได้

"000" ไม่ใช้บิตพาริตี้

"001" กำหนดพาริตี้คี่

"011" กำหนดพาริตี้คู่

"101" มาร์ก (Mark)

"111" ช่องว่าง(Space)

STOP ใช้กำหนดจำนวนบิตปิดท้าย

"1" มีบิตปิดท้าย 2 บิต

"0" มีบิตปิดท้าย 1 บิต

DAB1,DAB0 ใช้ร่วมกันในการกำหนดจำนวนบิตของข้อมูลที่ต้องการถ่ายทอด

"00" จำนวนบิตข้อมูลเท่ากับ 5 บิต

"01" จำนวนบิตข้อมูลเท่ากับ 6 บิต

"10" จำนวนบิตข้อมูลเท่ากับ 7 บิต

"11" จำนวนบิตข้อมูลเท่ากับ 8 บิต

3.4.5 รีจิสเตอร์ตำแหน่ง 04H (รีจิสเตอร์ควบคุมโมเด็ม)

มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	LOOP	OUT2	OUT1	RTS	DTR

ตารางที่ 3.6 ฟังก์ชั่นการทำงานของรีจีสเตอร์ 04H

บิต 5-7 ไม่มีการใช้งาน อ่านค่าได้เท่ากับ 0

LOOP "1" เอ็นเอเบิลการส่งค่ากลับ

"0" ดิสเอเบิล

OUT1,OUT2 "1" เอ็นเอเบิลการใช้งานภายใน

"0" ดิสเอเบิล

RTS ใช้ควบคุมการทำงานของขาRTS(Ready TO Send)

"1" เอ็นเอเบิล

"0" ดิสเอเบิล

DTR ใช้ควบคุมการทำงานของขาDTR (Data Terminal Ready)

"1" เอ็นเอเบิล

"0" ดิสเอเบิล

3.4.6 รีจิสเตอร์ตำแหน่ง 05H (รีจิสเตอร์แสดงสถานะรอรับและส่งข้อมูลอนุกรมของUART)

ใช้งานร่วมกับรีจิสเตอร์แสดงโหมดและสถานะของการอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง02H) เพื่อแสดงสาเหตุของการเกิดอินเตอร์รัปต์ มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	TXE	TBE	BREK	FRME	PARE	OVFE	RxRD

ตารางที่ 3.7 ฟังก์ชั่นการทำงานของรีจีสเตอร์ 05H

TXE(Transmitter Empty)

"1" แสดงว่ารีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"0" แสดงว่ายังคงมีข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

TBE(Transmitter Buffer Empty)

"1" รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"0" ยังคงมีข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

BREK(Break)

"1" UART ตรวจพบการเบรก

"0" ไม่มีการเบรก

FRME (Frame Error)

"1" UART ตรวจพบความผิดพลาดด้านเฟรมข้อมูล

"0" ไม่พบความผิดพลาดทางพาริตี้

OVRE(Overrun Error)

"1" UART ตรวจพบความผิดพลาดแบบโอเวอร์รัน

"0" ไม่พบความผิดพลาดแบบโอเวอร์รัน

RxRD(Received Data Ready)

"1" มีการรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์

"0" ไม่มีข้อมูล

3.4.7 รีจิสเตอร์ตำแหน่ง 06H (รีจิสเตอร์แสดงสถานะของโมเด็ม)

ใช้เพื่อกำหนดสถานะสัญญาณอินพุต ของพอร์ตอนุกรม RS-232 ซึ่ง ได้แก่ สัญญาณ DCD,DSR,CTS และ RI สำหรับการเชื่อมต่อใช้งานแบบอเนกประสงค์ ดังมีรายละเอียดหน้าที่ของแต่ละบิตต่อไปนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

ตารางที่ 3.8 ฟังก์ชั่นการทำงานของรีจีสเตอร์ 06H

DCD ใช้แสดงสภาวะของขา DCD

"1" แสดงว่าที่ขา DCD เป็นลอจิก "1"

"0" แสดงว่าที่ขา DCD เป็นลอจิก "0"

RI ใช้แสดงสภาวะของขา RI

"1" แสดงว่าที่ขา RI เป็นลอจิก "1"

"0" แสดงว่าที่ขา RI เป็นลอจิก "0"

DSR ใช้แสดงสภาวะของขา DSR

"1" แสดงว่าที่ขา DSR เป็นลอจิก "1"

"0" แสดงว่าที่ขา DSR เป็นลอจิก "0"

DCTS(Delta Clear To Send) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต CTS

"1" แสดงว่าบิตCTS เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าบิตCTS ไม่เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

DRI (Delta Ring Indicator)ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต RI

"1" แสดงว่าบิตRI เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าบิตRI ไม่ เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

DDCD(Delta Data Carrier Detect)ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นในบิตDDCD

"1" แสดงว่าบิตDDCD เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่า ไม่ เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

DCTS(Delta Clear to send) ใช้แสดงสภาวะของ CTS

"1" แสดงว่าที่ขา CTS มีลอจิกเป็น "1"

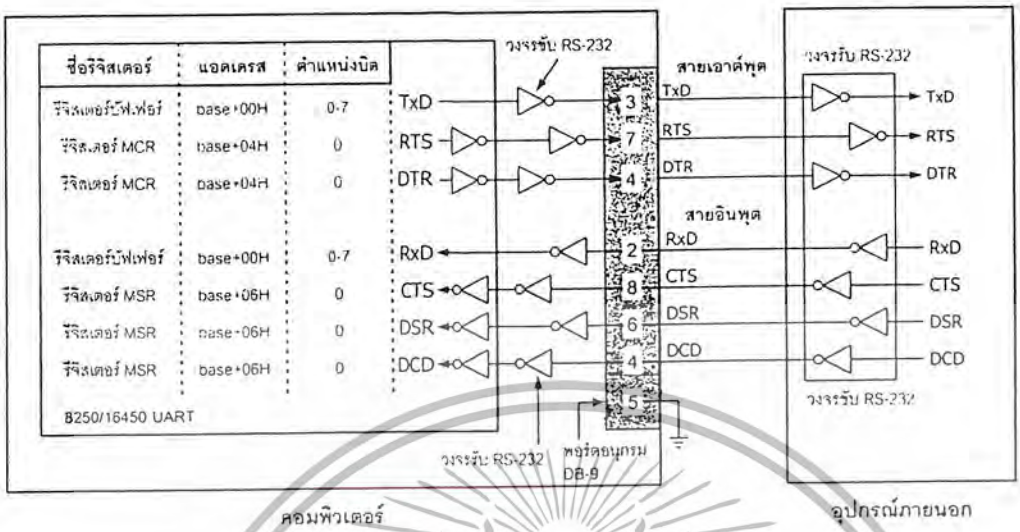
"0" แสดงว่าที่ขา CTS มีลอจิกเป็น "0"

3.4.8 รีจิสเตอร์ตำแหน่ง07H (รีจิสเตอร์สำหรับเก็บข้อมูลชั่วคราว)

ทำหน้าที่เป็นหน่วยความจำแรมขนาด 1 ไบต์ การอ่านและเขียนข้อมูลที่รีจิสเตอร์ตัวนี้ไม่ส่งผลใดๆ ต่อการใช้งานUART

3.5 ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ตRS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTS) และสัญญาณแสดงสถานะอินพุต(CTS,CSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจากUART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับแรงดันให้ได้ระดับแรงดันเป็นไปตามมาตรฐานRS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะดังแสดงเป็นบล็อกไดอะแกรมในรูปที่ 3.6



รูปที่ 3.6 ไดอะแกรมแสดงโครงสร้างทางฮาร์ดแวร์ของพอร์ตอนุกรม

3.6 แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

บิต 3	บิต 2	บิต1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 1 พอร์ต
0	1	1	มีพอร์ตอนุกรม 2 พอร์ต
1	0	0	มีพอร์ตอนุกรม

ตาราง 3.9 ข้อมูลในแอดเดส 0000:0411H ที่ใช้แจ้งจำนวนพอร์ตอนุกรม

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรมไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2ไบต์ สำหรับพอร์ตอนุกรม COM1จะเก็บไว้ที่แอดเดรส 0000:0400H และ 0000:0401H ส่วนตำแหน่งอื่นๆมีรายละเอียดดังนี้

COM2 = 0000:0402H – 0000:0403H

COM3 = 0000:0404H – 0000:0405H

COM4 = 0000:0406H – 0000:0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส 0000:0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในเครื่องคอมพิวเตอร์อีกด้วย โดยรายละเอียดดังแสดงในตารางที่ 3.8

บทที่ 4 ไมโครคอนโทรลเลอร์ MCS-51

4.1 ไมโครคอนโทรลเลอร์ MCS-51

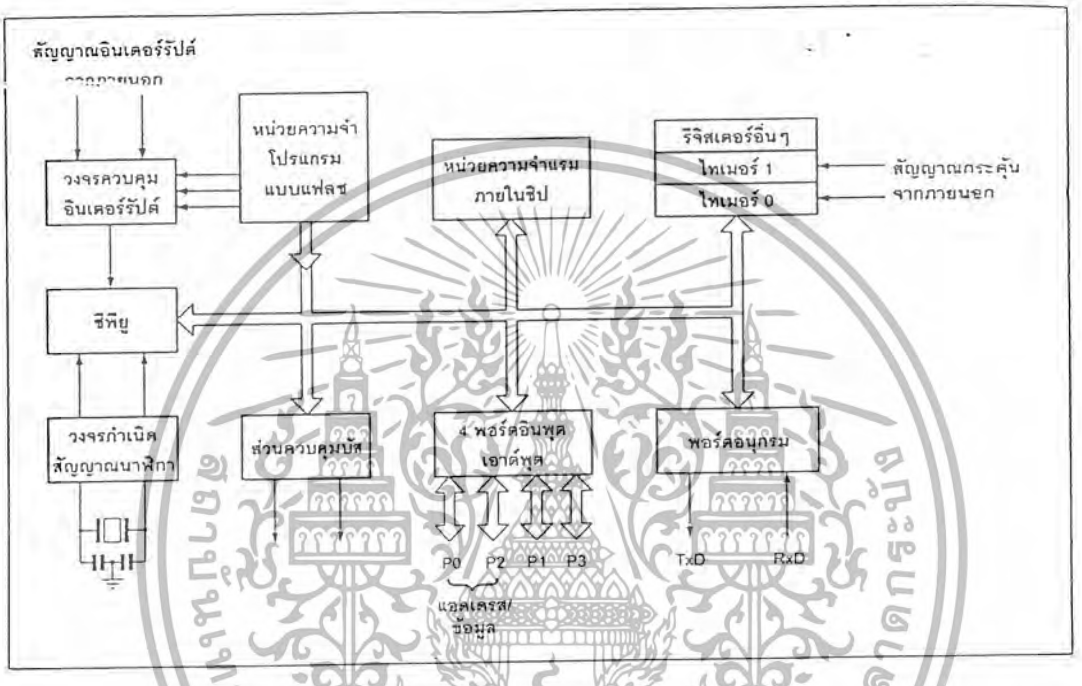
ไมโครคอนโทรลเลอร์ MCS-51 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิตที่มีอุปกรณ์สนับสนุนประกอบอยู่ภายในหลายอย่างได้แก่ หน่วยความจำสำหรับเก็บข้อมูล หน่วยความจำสำหรับเก็บโปรแกรม ตัวตั้งเวลา/ตัวนับ อุปกรณ์รับส่งข้อมูลแบบอนุกรม เนื่องจากโครงสร้างของไมโครคอนโทรลเลอร์มีอุปกรณ์สนับสนุนประกอบอยู่ภายในนี้เอง ทำให้การใช้งานง่ายขึ้นและมีประสิทธิภาพมากขึ้นโดยไม่ต้องมีการเชื่อมต่ออุปกรณ์ภายนอกเพิ่มเติมมากเหมือนกับ ตัวไมโครโปรเซสเซอร์ทั่วไป นอกจากนี้หากเราต้องการใช้งานไมโครคอนโทรลเลอร์ร่วมกับ อุปกรณ์อื่นเพิ่มเติมเช่น ไอซี 8255 หรือหน่วยความจำภายนอก เรายังสามารถนำมาเชื่อมต่อเพิ่มเติมเข้ากับไมโครคอนโทรลเลอร์ได้อีกด้วย

4.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แสดงในรูปที่ 4.1 ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (BOOLEAN PROCESSOR)
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์
- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบ อีพรอม (เบอร์ 8451)
- หน่วยความจำแบบ แรม ภายในจำนวน 128 ไบต์
- พอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบดูเพล็กเต็ม(FULL DUPLEX)
- วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนด ลำดับ

วงจรผลิตสัญญาณนาฬิกาภายในซึ่งโครงสร้างการทำงานทั้งหมดของไมโครคอนโทรลเลอร์จะอาศัยหลักการทำงานที่เกี่ยวข้องกัน โดยอาศัยหลักการทำงานที่เป็นไป ตามโครงสร้างเสมอ



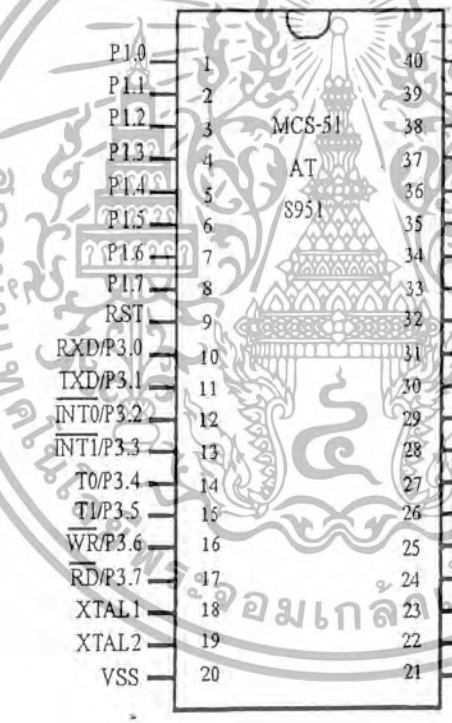
รูปที่ 4.1 แสดงโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51

โดยมากแล้วไมโครคอนโทรลเลอร์ตระกูลนี้มักจะมีรูปร่างของไอซีเป็นแบบขนาด 40 ขา ดังแสดงในรูปที่ 4.2 ซึ่งแต่ละขาสัญญาณจะมีหน้าที่ที่ระบุชัดเจนตามสัญลักษณ์ชื่อย่อ ที่กำกับในแต่ละขา อย่างไรก็ตามจะมีบางขาสัญญาณที่อาจจะมีหน้าที่ได้มากกว่าหนึ่งอย่าง ซึ่งจะไม่สามารถใช้งานในเวลาเดียวกันได้ ตัวอย่างเช่นขาสัญญาณบิต 0 ของพอร์ต 3 (ใช้ตัวย่อเป็น P3.0) อาจจะใช้เป็นขาสัญญาณเอาต์พุต หรืออินพุตตามปกติ ภายในไมโครคอนโทรลเลอร์ MCS-51 ซึ่งประกอบด้วยหน่วยการทำงานต่างๆ ภายในไอซี MCS-51 จำนวนมาก โดยแต่ละบล็อกซึ่งเป็นวงจรควบคุมรีจิสเตอร์ (REGISTER) หรือหน่วยความจำภายในของไอซี MCS-51 จะถูกเชื่อมต่อเข้าด้วยกันผ่านทางเส้นสัญญาณที่เรียกว่า

บัสข้อมูลภายใน รีจิสเตอร์และหน่วยความจำเหล่านี้จะถูกนำไปใช้ระหว่างการประมวลผลคำสั่ง หน้าที่ของโปรแกรมที่ผู้ใช้สร้างขึ้นมาก็เป็นการควบคุมการรับหรือส่งข้อมูลระหว่างรีจิสเตอร์เหล่านี้ ซึ่งอาจจะมีการดำเนินการร่วมกับหน่วยการดำเนินงานประมวลผลทางคณิตศาสตร์และลอจิก(Arithmatic and Logic Unit (ALU)

4.3 โครงสร้างหน่วยความจำภายในไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 แยกการจัดการหน่วยความจำออกเป็นสองส่วนอย่างชัดเจน คือ หน่วยความจำโปรแกรม (PROGRAM MEMORY) และหน่วยความจำข้อมูล (DATA MEMORY) หน่วยความจำทั้งสองนี้ มีหน้าที่แตกต่างไปจากกัน และใช้วิธีการอ้างแอดเดรสสัญญาณการติดต่อแยกออกจากกัน



รูปที่ 4.2 แสดงรูปร่างและการจัดวางขาต่างๆ ของไมโครคอนโทรลเลอร์ MCS-51

4.4 หน่วยความจำโปรแกรม

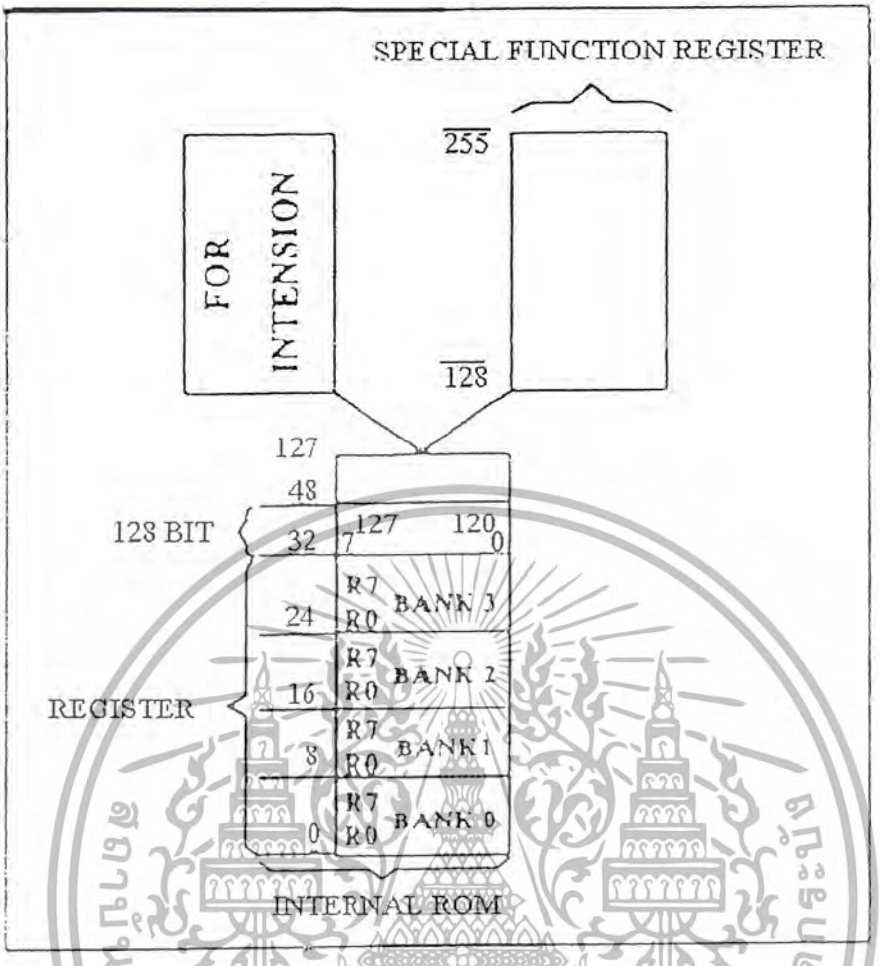
หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 เป็นบริเวณหน่วยความจำสำหรับเก็บข้อมูลและคำสั่งใช้งานต่างๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบข้อมูลเหล่านี้ก็ยังคง

อยู่ไม่สูญหายโครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไมโครคอนโทรลเลอร์ MCS-51 ของหน่วยความจำ ประเภทต่างๆ เช่น หน่วยความจำแบบรอม (READ ONLY MEMORY) หรือ อีพรอม (ERASABLE PROGRAMABLE READ ONLY MEMORY) ในไมโครคอนโทรลเลอร์ MCS-51 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (INTERNAL PROGRAM MEMORY) ซึ่งเป็นหน่วยความจำรอม หรือ อีพรอม ที่อยู่ภายในตัวไอซีของไมโครคอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก (EXTERNAL PROGRAM MEMORY) ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ

ไมโครคอนโทรลเลอร์เบอร์ต่างๆ ของตระกูล 8051 นี้สามารถขยายให้ใช้งาน ในหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มีทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้นจะต้องทำการควบคุมระดับลอจิกของสัญญาณในขณะนั้นด้วย ขนาดหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์เบอร์ต่างๆ ภายในตระกูล 8051 จะแตกต่างกันออกไป เพื่อความเหมาะสมกับการนำไปใช้งานลักษณะต่างๆ

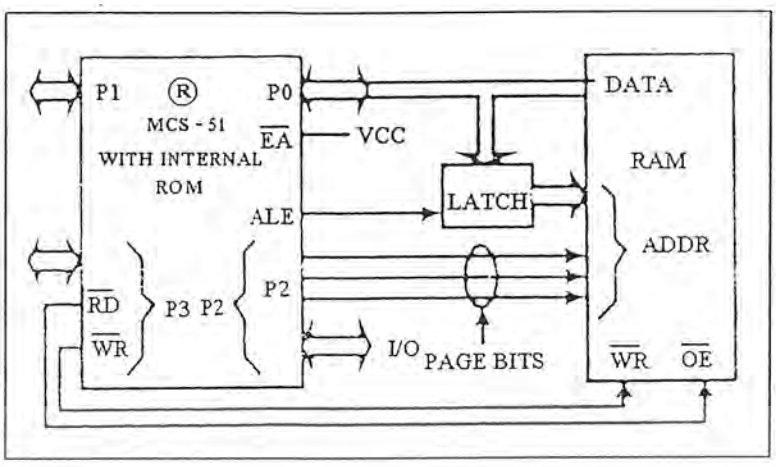
4.5 หน่วยความจำข้อมูล

หน่วยความจำข้อมูล (DATA MEMORY) ซึ่งโดยพื้นฐานแล้วเป็นหน่วยความจำแรมสามารถเขียนหรืออ่านข้อมูลได้ (READ OR WRITE MEMORY) ใช้สำหรับเก็บข้อมูลหรือตัวแปร ที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว ซึ่งโดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำแรมแบบสแตติกดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญไป พื้นที่ของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 มีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ตามลักษณะของหน่วยความจำโปรแกรมภายในซึ่งก็เป็นแรมที่อยู่ภายในตัวไอซีในตระกูลของไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลภายนอกซึ่งเป็นการใช้ไอซีหน่วยความจำแรมมาเพิ่มเติมเข้าไปในวงจรลักษณะเดียวกับ การนำไอซีอีพรอมมาใช้งานเป็นหน่วยความจำโปรแกรมนั่นเอง



รูปที่ 4.3 แสดงการจัดหน่วยความจำข้อมูล

โดยที่หน่วยความจำสำหรับเก็บข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 นี้สามารถแบ่งออกเป็น 2 ส่วนคือ ในส่วนที่เป็นหน่วยความจำสำหรับเก็บข้อมูลภายในไอซี และหน่วยความจำสำหรับเก็บข้อมูลภายนอกไมโครคอนโทรลเลอร์ MCS-51 ทุกๆ เบอร์จะมีหน่วยความจำเก็บข้อมูลทั่วๆ ไปภายในไอซีอย่างน้อยคือ 128 ไบต์ ไปจนถึง 256 ไบต์ทั้งนี้ ขึ้นกับเบอร์ของไอซี หน่วยความจำสำหรับเก็บ ข้อมูลภายในไอซีในบริเวณ 128 ไบต์เรียกว่า LOWER 128 และในบริเวณ 128 ไบต์หลัง ที่มีเพิ่มในบางเบอร์มีชื่อเรียกว่า UPPER



รูปที่ 4.4 แสดงการต่อกับหน่วยความจำข้อมูลภายนอกไอซี

4.6รีจิสเตอร์ที่เกี่ยวกับไมโครคอนโทรลเลอร์MCS-51

รีจิสเตอร์ในกลุ่มนี้จะเป็นรีจิสเตอร์ขนาด 16บิตที่ใช้งานเพื่อเก็บข้อมูลของตัวแอดเดรสเป็นสำคัญโดยค่าที่อยู่ภายในแอดเดรสนี้จะนำไปเป็นค่าของข้อมูลที่ส่งออกไปทางบัสแอดเดรส ในส่วนของไมโครคอนโทรลเลอร์ เพื่อบอกตำแหน่งที่ต้องการติดต่อ รีจิสเตอร์ที่จัดในกลุ่มนี้ประกอบด้วยรีจิสเตอร์ใช้งานทั่วไป (GENERAL-PURPOSE REGISTERS) รีจิสเตอร์ในกลุ่มนี้จัดเป็นพื้นที่หน่วยความจำที่ใช้ในการสนับสนุนในการประมวลผล การทำงานจากหน่วยประมวลผลทางคณิตศาสตร์และลอจิก (ALU) เพื่อให้สามารถจัดการข้อมูลให้เร็วที่สุด นอกจากนี้โปรแกรมที่ไม่ได้ใช้คำสั่งเหล่านี้ก็ยังใช้เป็นการเก็บข้อมูลตัวแปรภายในโปรแกรม จะเห็นได้ว่าชื่อของรีจิสเตอร์ไม่ว่าจะอยู่ในรีจิสเตอร์แบงก์ใด ก็จะมีชื่อว่า R0 ถึง R7 เหมือนกันทั้งสิ้น ดังนั้นในการใช้งานผู้ใช้จะต้องให้ความระมัดระวังว่า ต้องการรีจิสเตอร์นั้นๆ จากแบงก์ใดๆ ซึ่งการกำหนดเลือกแต่ละกลุ่มของรีจิสเตอร์นี้ก็ทำได้ง่าย เพียงการกำหนดค่าของบิตที่อยู่ภายในแฟลก (PSW) เท่านั้นอย่างไรก็ตามโดยทั่วไปก็มักจะมีการใช้งานรีจิสเตอร์ R0 ถึง R7 เฉพาะในแบงก์ 0 เท่านั้น ดังนั้นพื้นที่ของแบงก์อื่นๆ ที่เหลือก็สามารถนำมาใช้ในลักษณะของหน่วยความจำแรม

4.6.1 รีจิสเตอร์หน้าที่พิเศษ

เป็นรีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับ การควบคุมหน้าที่ และการทำงานของอุปกรณ์หรือพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 ทั้งหมด ตำแหน่งของรีจิสเตอร์เหล่านี้จะจัดอยู่ในบริเวณแอดเดรส 80H - FFH การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้สามารถทำได้ทั้งการระบุชื่อของรีจิสเตอร์ หรือตำแหน่งแอดเดรส ที่เป็นของรีจิสเตอร์นั้นก็ได้ การจัดพื้นที่หน่วยความจำสำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้ โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดรสที่มีจำนวนเป็นทวีคูณ

ของค่า 8 จะสามารถอ้างถึงในระดับบิตได้ด้วย (นั่นคือแอดเดรส 80H 88H 90H A0H A8H B0H B8H D0H E0H และ F0H)

4.6.2แอกคิวมูเลเตอร์ (ACCUMULATOR)

หรือ ACC เป็นรีจิสเตอร์ขนาด 8 บิต ทำหน้าที่ในการเก็บข้อมูลที่จะส่งให้กับหน่วยทำงานภายในหน่วยประมวลผลกลาง และเก็บผลลัพธ์ที่ได้จากการทำงานเท่านั้น การทำงานของรีจิสเตอร์นั้นมีลักษณะเช่นเดียวกับตัวแอกคิวมูเลเตอร์ของโปรเซสเซอร์ทั่วไป การใช้งานในโปรแกรมซึ่งใช้เรียกเป็นรีจิสเตอร์ A

4.7 ชุดคำสั่งของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ประกอบด้วยคำสั่งทั้งหมดจำนวนมาก ซึ่งสามารถจะจัดกลุ่มคำสั่งเหล่านี้ตามลักษณะและหน้าที่การทำงานที่คล้ายคลึงกัน เพื่อความสะดวกต่อการศึกษา ทำความเข้าใจและใช้งาน ดังนี้

4.7.1 กลุ่มการถ่ายเทข้อมูล คือ กลุ่มคำสั่งในการโอนย้ายข้อมูล ทำหน้าที่ในโอนย้ายข้อมูลระหว่างรีจิสเตอร์ หรือหน่วยความจำภายในแรม โดยมีรายละเอียดดังนี้ ชุดคำสั่งในการถ่ายเทแรมภายในนั้น ซึ่งเวลาที่ใช้ในหนึ่งคำสั่งนั้น จะเป็นเวลาเมื่อขณะที่ความถี่ในการทำงานของหน่วยประมวลผลกลาง ที่ความถี่ 12 เมกะเฮิรตซ์ และรายละเอียดของแต่ละคำสั่งมีดังนี้

MOV :จะทำงานในลักษณะเป็นการถ่ายเทข้อมูลที่มีขนาดเป็นไบต์ หรือ บิตก็ได้ จากแหล่งกำเนิดเข้าสู่ตัวรับข้อมูลในฟิลด์โอเปอร์แรนด์

PUSH:จะทำงาน โดยเพิ่มค่ารีจิสเตอร์ SP ก่อนแล้วจึงทำการถ่ายเทข้อมูล 1 ไบต์จากแหล่งกำเนิดไปบริเวณสแต็กตามตำแหน่งที่รีจิสเตอร์ SP กำหนด POP:การถ่ายเทข้อมูลขนาด 1 ไบต์จากบริเวณตำแหน่งที่รีจิสเตอร์ SP กำหนดไปยังรีจิสเตอร์ที่โอเปอร์แรนด์ กำหนดและหลังจากนั้นรีจิสเตอร์ SP จะลดค่าลง

XCH:คำสั่งแลกเปลี่ยนไบต์ระหว่างแหล่งกำเนิดโอเปอร์แรนด์กับรีจิสเตอร์ AXCHD คำสั่งในการแลกเปลี่ยนขนาดนิบเบิลทางอันดับต่ำของแหล่งกำเนิดโอเปอร์แรนด์กับนิบเบิลอันดับต่ำลงของแอกคิวมูเลเตอร์ ตัวอย่างเช่นทำการเลื่อนข้อมูลไป 2ไบต์ทางขวามือซึ่งจะมี 2 วิธีคือใช้คำสั่ง MOV หรือใช้คำสั่ง XCH

4.7.2 กลุ่มคำสั่งทางคณิตศาสตร์ เช่น การบวก ลบ คูณ และหารข้อมูลภายในตัว รีจิสเตอร์ต่างๆ ช่วงเวลาการทำงาน ของแต่ละคำสั่งนั้นจะกำหนดที่ความถี่ของสัญญาณนาฬิกาที่ 12 เมกะเฮิรตซ์ คำสั่งทางคณิตศาสตร์ส่วนใหญ่ใช้เวลา 1 ms ยกเว้นคำสั่ง INC DPTR ซึ่งใช้เวลา 2 ms โดยที่คำสั่งการคูณและหารใช้เวลา 4 ms โดยมีรายละเอียดการใช้คำสั่งมีดังนี้

INC: เป็นการบวกหนึ่งกับโอเปอร์แรนด์และใส่ค่าใหม่กลับเข้าที่ตัวโอเปอร์แรนด์นั้นๆ

DEC: เป็นการลบออกจากตัวเลขที่อยู่ในแหล่งกำเนิดโอเปอร์แรนด์ และนำผลลัพธ์ที่ได้มาเก็บไว้ที่ตัวโอเปอร์แรนด์นั้น

ADD: เป็นการบวกในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์

ADDC: เป็นการบวกค่าต่างๆ ในแอกคิวมูเลเตอร์เข้ากับค่าในแหล่งกำเนิดโอเปอร์แรนด์และบวกกับบิตทดด้วย

SUBB: เป็นการนำเลขที่แหล่งกำเนิดโอเปอร์แรนด์ ลบออกจากตัวเลขใน A และนำค่าบิตตัวทดมาลบออกอีกและผลลัพธ์ที่ได้นำมาใส่ลงในแอกคิวมูเลเตอร์ A

MUL: เป็นการคูณแบบไม่คิดตัวเครื่องหมายของตัวเลขที่อยู่ใน แอกคิวมูเลเตอร์กับเลขในรีจิสเตอร์ B แล้วได้ผลลัพธ์ 2 ไบต์ นำมาเก็บไว้ที่ AB โดย A จะรับอันดับต่ำส่วน B จะรับอันดับสูง

DIV: เป็นคำสั่งในการหารแบบไม่คิดเครื่องหมายที่อยู่ในแอกคิวมูเลเตอร์แล้วหารตัวเลขในรีจิสเตอร์ B แล้วนำผลลัพธ์ไปเก็บในแอกคิวมูเลเตอร์และเศษของการหารตัวเลข จะเก็บไว้ในรีจิสเตอร์ B

DA: สำหรับการบวกกันทางตัวเลข BCD เป็นการปรับค่ารวม ซึ่งเป็นผลมาจากการบวกกันทางไบนารี่ของระบบตัวเลข BCD ขนาด 2 หลักสองจำนวน การปรับค่าตัวเลขผลรวมด้วยการใช้คำสั่ง DA จะได้ผลลัพธ์กลับมาที่แอกคิวมูเลเตอร์

4.7.3 กลุ่มคำสั่งทางตรรกศาสตร์หรือ แบบลอจิก ทำหน้าที่เกี่ยวกับการประมวลผลแบบ ลอจิกต่างๆ เช่น การ AND OR หรือ EX-OR ระหว่างข้อมูลในรีจิสเตอร์ A นั่นเอง โดยมีการใช้คำสั่งดังนี้

CPL: เป็นการใช้คำสั่งกลับค่าหรือคอมพลีเมนต์ ข้อมูลในแอกคิวมูเลเตอร์จะไม่มีผลใดๆ ต่อค่าของแฟลก หรือการอ้างถึงตำแหน่งแอดเดรสนั้นตามบิตนั้นๆ

RL, RLC, RR, RRC, SWAP: ทั้ง 5 คำสั่งนี้เป็นคำสั่งในการทำงานการวนบิตบนตัวของแอกคิวมูเลเตอร์ซึ่ง RL เป็นการวนบิตทางขวา, RLC เป็นการทำการวนทางซ้ายผ่านบิตทด, RRC เป็นการวนขวาผ่านบิตทด และ SWAP เป็นการวนซ้ายสี่ครั้ง

ANL: เป็นการ ADD กันทางตรรกศาสตร์ ระหว่างแหล่งกำเนิดสองโอเปอร์แรนด์ ซึ่งจะสั่งให้ทำงานในรูปแบบของตรรกศาสตร์ทางข้อมูลขนาดเป็นไบต์หรือบิต กลุ่มคำสั่งแบบบูลีนหรือแบบบิต ซึ่งเป็นความสามารถของไมโครคอนโทรลเลอร์ MCS-51 ที่จะดำเนินการประมวลผลแบบบิต แทนที่จะ

เป็นข้อมูลทั้งไบต์เช่นปกติ โดยมีชุดคำสั่งที่จัดการโดยตรง ทุกคำสั่งจะเข้าถึงข้อมูลโดยตรงในระดับบิต โดยมีการบิตแอดเดรสได้ตั้งแต่ 00H - 7FH ในพื้นที่ 128 บิต หน่วยความจำข้อมูลภายในและบิตแอดเดรส 80H - FFH ในบริเวณกลุ่มรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR)

4.7.4 กลุ่มคำสั่งในการกระโดดไปยังตำแหน่งต่างๆภายในโปรแกรม ซึ่งจะเปลี่ยนลำดับของการประมวลผลภายในโปรแกรมไปยังส่วนต่างๆแทนที่จะดำเนินการไปเป็นลำดับ ต่อเนื่องโดยที่คำสั่ง JMP จะแบ่งเป็น 3 ลักษณะ คือ SJMP, LJMP, AJMP ซึ่งในแต่ละคำสั่ง จะมีข้อแตกต่างของการกระโดดไปยังแอดเดรสไกลสุดที่ต่างกัน คำสั่ง JMP ซึ่งเป็นแบบโมนีชิก ที่สามารถจะใช้ได้โดยมีรายละเอียดการใช้งานของคำสั่งดังต่อไปนี้

SJMP: เป็นการกระโดดแบบการย้ายอันดับตำแหน่งของแอดเดรสตำแหน่งเดิมซึ่งจะสามารถกระโดดได้ -128 ถึง +127 ไบต์

AJMP: ลักษณะแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 2 กิโลไบต์ ซึ่งจะใช้หน่วยความจำเพียง 2 ไบต์เท่านั้นในการกำหนด

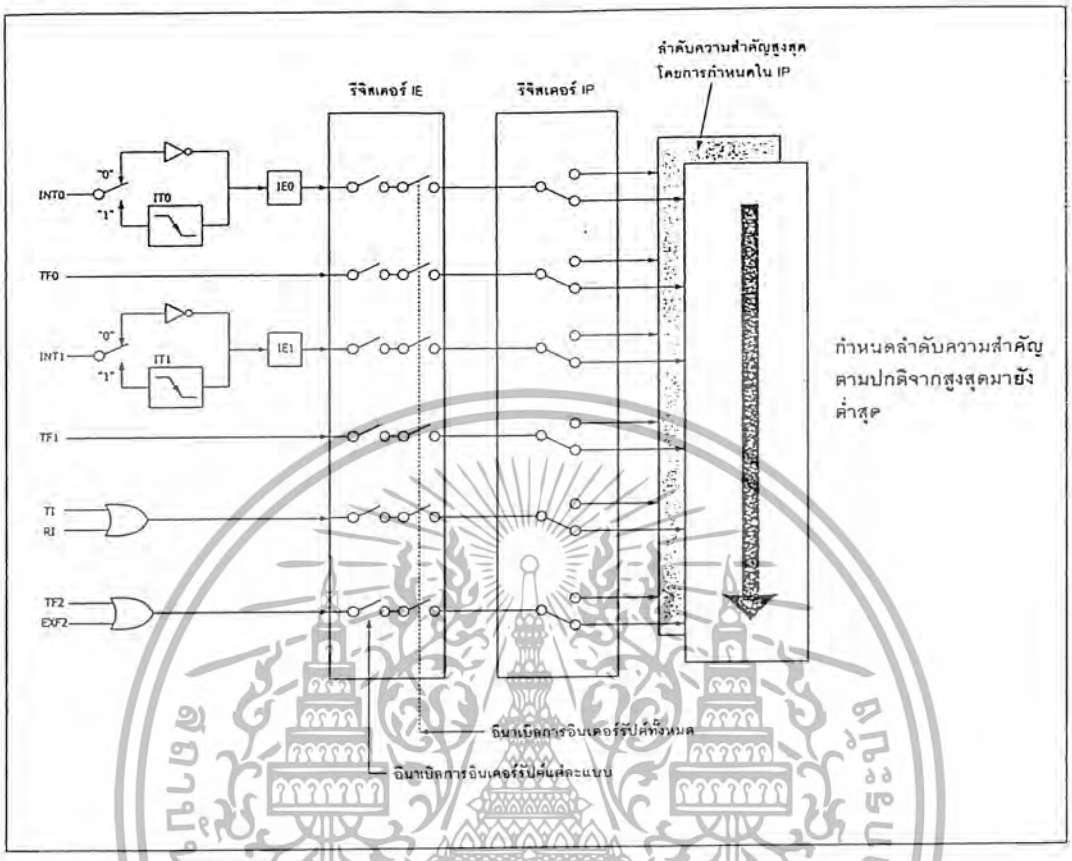
LJMP: ลักษณะแบบนี้จะสามารถกระโดดได้ไกลสุดประมาณ 64 กิโลไบต์ ซึ่งจะใช้หน่วยความจำเพียง 3 ไบต์เท่านั้นในการกำหนด

JMP @A+DPTR:เป็นการควบคุมการกระโดดไปยังโปรแกรมที่ต้องการเฉพาะภายในส่วนต่างๆ

4.8 โครงสร้างการอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์ MCS-51

จากรูปที่ 4.8 โครงสร้างระบบอินเตอร์รัปต์ของ ไมโครคอนโทรลเลอร์ MCS-51สัญญาณที่เข้ามาทำการอินเตอร์รัปต์ MCS-51 นั้นเกิดขึ้นได้ 5 ลักษณะตามตารางข้อมูล ในรูปที่ .....โดยจะเห็นได้ว่าสามารถที่จะกำหนดเลือกเพื่อยินยอม (หรืออีนาเบิล : ENABLE) และห้าม (หรือดิสเอเบิล : DISABLE) ไม่ให้มีการอินเตอร์รัปต์แต่ละประเภทได้ โดยการกำหนดบิตของข้อมูลที่เกี่ยวข้องซึ่งมักจะอยู่ภายในรีจิสเตอร์ TCON และ SCON นอกจากนี้ยังมีตำแหน่งบิตภายในรีจิสเตอร์ IE (INTERRUPT ENABLE REGISTER) ซึ่งทำหน้าที่เสมือนกับเป็นสวิตซ์หลักที่เกี่ยวข้องกับสัญญาณอินเตอร์รัปต์ทั้งหมด หากว่ากำหนดไม่ให้เกิดการอินเตอร์รัปต์แล้วการกำหนดบิตเพื่อห้ามหรือยินยอมของแต่ละอินเตอร์รัปต์ก็จะไม่มีผลใดๆเกิดขึ้น ยังแสดงให้เห็นว่าสัญญาณอินเตอร์รัปต์แต่ละประเภทยังสามารถกำหนดระดับความสำคัญ (PRIORITY) ของการอินเตอร์รัปต์ได้สองลักษณะ คือ ระดับความสำคัญสูงหรือต่ำ (HIGH OR LOW PRIORITY) กล่าวคือขณะที่กำลังประมวลผลอยู่ภายในส่วนของโปรแกรมย่อยบริการอินเตอร์รัปต์ของสัญญาณที่มีระดับความสำคัญต่ำอยู่ ก็อาจจะถูกขัดจังหวะให้ไปประมวลผล

ของสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญสูงกว่า แต่หากว่าเป็นสัญญาณอินเตอร์รัปต์ที่มีระดับความสำคัญต่ำเช่นเดียวกันแล้ว ก็ต้องรอให้เสร็จสิ้นการประมวลผลที่ ดำเนินการอยู่ก่อน



รูปที่ 4.5 โครงสร้างของระบบอินเตอร์รัปต์ภายใน MCS-51

4.9 ไทเมอร์/ เคาน์เตอร์ของไมโครคอนโทรลเลอร์ MCS-51

ไทเมอร์ / เคาน์เตอร์ เป็นอีกส่วนหนึ่งที่สำคัญของไมโครคอนโทรลเลอร์ เนื่องจากในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บ และตรวจสอบค่าของเวลาและจำนวนสัญญาณนาฬิกาอยู่ตลอดเวลา ทั้งนี้เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณพัลส์ เปรียบเทียบค่าเวลาหรือ ค่าการนับ รวมไปถึงการกำหนดอัตราเร็วในการสื่อสารข้อมูลของพอร์ตอนุกรมด้วย

ในการใช้งานไทเมอร์ / เคาน์เตอร์นั้น จะต้องมีการกำหนดหรือควบคุมการทำงานของไทเมอร์ / เคาน์เตอร์ โดยใช้รีจิสเตอร์ที่สำคัญ 2 ตัว คือ

TCON (Timer / Counter Control Register) : มีแอดเดรสอยู่ที่ 88H เป็นรีจิสเตอร์ที่ใช้ควบคุมการเปิด / ปิดไทเมอร์แต่ละตัว , และแสดงถึงการเกิดโอเวอร์โฟรวที่เกิดจากไทเมอร์ / เคาน์เตอร์ อีกทั้งเป็นตัวกำหนดลักษณะการเกิดอินเตอร์รัปต์ของสัญญาณภายนอกด้วย

TMOD (Timer / Counter Mode Control Register) : มีแอดเดรสอยู่ที่ 89H เป็นรีจิสเตอร์ที่ใช้กำหนดการทำงานว่าเป็นไทเมอร์ หรือ เคาน์เตอร์ อีกทั้งเป็นตัวกำหนดโหมดการทำงานของไทเมอร์ / เคาน์เตอร์ ซึ่งมีอยู่ 4 โหมดด้วยกัน คือ

โหมด 0 : ไทเมอร์ / เคาน์เตอร์ 13 บิต

โหมด 1 : ไทเมอร์ / เคาน์เตอร์ 16 บิต

โหมด 2 : ไทเมอร์ / เคาน์เตอร์ 8 บิต แบบตั้งค่าอัตโนมัติ

โหมด 3 : ไทเมอร์ / เคาน์เตอร์แยกส่วน

4.10 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบ ฟลูดูเพล็กซ์ 1 ชุด โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขาP3.0 เป็นขารับข้อมูลหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของ MCS-51 เป็นแบบอะซิงโครนัส ปกติแล้ว พอร์ตอนุกรมจะใช้ติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบัน สามารถติดต่อกับ RS-422 หรือ RS-485 ได้แล้ว โดยใช้ IC พิเศษ ทำหน้าที่ในการแปลงสัญญาณการสื่อสาร

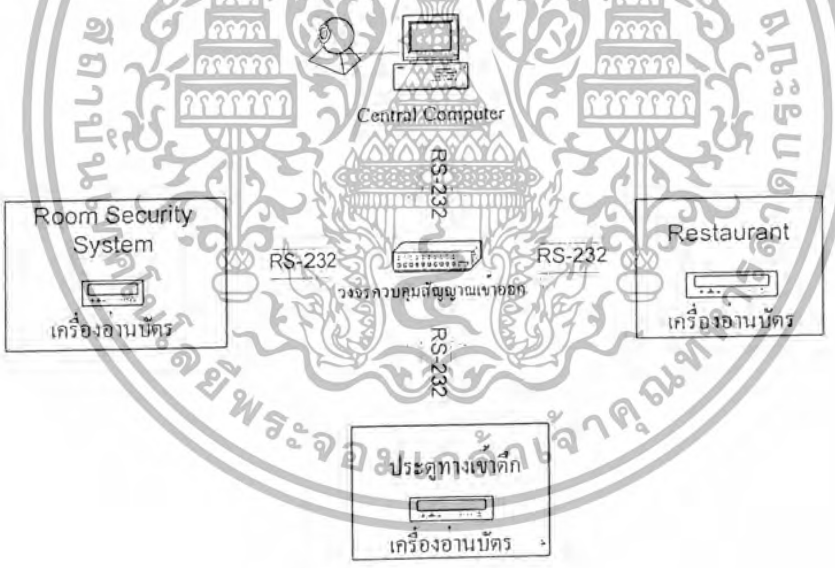
รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมในMCS -51 ได้แก่ SBUF (Serial data buffer register) ซึ่งเป็นบัฟเฟอร์สำหรับ ส่งข้อมูล และรับข้อมูล และรีจิสเตอร์อีกตัวที่สำคัญคือ SCON (Serial Port Control) ซึ่งใช้ในการกำหนด โหมดการทำงานของพอร์ตอนุกรม ซึ่งมี 4 โหมดด้วยกัน คือ

1. โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟรีจิสเตอร์
2. โหมด 1 เป็นการกำหนดให้UART ขนาด 8 บิต สามารถเลือกอัตรา บอร์ดได้
3. โหมด 2 UART ขนาด 8 บิต สามารถเลือกอัตรา บอร์ดได้
4. โหมด 3 เป็นตัวกำหนดให้เป็น UART ขนาด9 บิต สามารถเลือกอัตราบอดได้

บทที่ 5 การออกแบบและหลักการทำงานของวงจร

5.1 องค์ประกอบของระบบโดยรวม

การประยุกต์ใช้สมาร์ตการ์ดในโครงงานนี้ได้ถูกออกแบบมาเพื่อใช้ในงานด้านการรักษาความปลอดภัยและการอำนวยความสะดวก นั่นคือ ในส่วนของการรักษาความปลอดภัยนั้น จะเป็นการนำสมาร์ตการ์ดนี้ใช้ในการเข้าออกสถานที่ต่างๆ เช่นห้องพักส่วนตัว หรือ สถานบริการที่เจ้าของบัตรเป็นสมาชิกอยู่ เป็นต้น โดยมีการให้ใส่รหัสผ่านด้วยเพื่อยืนยันการเป็นเจ้าของบัตร ซึ่งนอกจากจะเป็นการป้องกันการเข้าออกสถานที่นั้นๆแล้ว ยังสามารถตรวจสอบการใช้สถานที่นั้นๆของเจ้าของบัตร ไม่ว่าจะเป็นในเรื่องของจำนวนครั้งที่ใช้ เวลาที่ใช้ และข้อมูลอื่นๆ ที่เป็นส่วนในการเพิ่มความปลอดภัยมากขึ้น สำหรับในส่วนของการอำนวยความสะดวกนั้นจะเป็นการประยุกต์ใช้ในลักษณะเป็นที่ให้บัตรสมาร์ตการ์ดใช้เป็นบัตรที่สามารถชำระค่าใช้บริการต่างๆในบริเวณที่เปิดให้บริการ เพื่ออำนวยความสะดวกต่อผู้ใช้ในการใช้บริการมากขึ้น จากที่กล่าวมาข้างต้นทั้งหมดนี้คือระบบโดยรวมที่ได้ออกแบบไว้ สามารถเขียนเป็นภาพโครงสร้างโดยรวมอย่างง่ายๆ ได้ดังรูปที่ 5.1



รูปที่ 5.1 โครงสร้างของระบบทั้งหมด

จากข้างต้น สิ่งที่สำคัญที่สุดสำหรับการสร้างระบบนี้มี 2 ส่วนคือ ส่วนของวงจรที่ทำหน้าที่ในการอ่านข้อมูลที่เก็บอยู่ในสมาร์ตการ์ดซึ่งในที่นี้หมายถึงบัตร TOT Card ที่ไม่สามารถชำระค่าโทรศัพท์ได้แล้วนั่นเอง และส่วนของโปรแกรมประยุกต์บนคอมพิวเตอร์ที่ทำการวิเคราะห์ข้อมูลที่เครื่องอ่านสามารถอ่านได้ เพื่อใช้ในงานรักษาความปลอดภัยและอำนวยความสะดวกต่อผู้ใช้นั่นเอง

5.2 หลักการทำงานเครื่องอ่านรหัสบัตร TOT Card

ในรูปที่ 5.2 คือวงจรที่สมบูรณ์ของเครื่องอ่านรหัสบัตร TOT Card สำหรับดูแลความเรียบร้อยในการเข้าออกสถานที่ต่างๆ ซึ่งการทำงานของวงจรนี้ถูกควบคุมโดยไมโครคอนโทรลเลอร์ตระกูล MCS51 เบอร์ AT89C51 หน้าที่หลักโดยทั่วไปของไมโครคอนโทรลเลอร์ตัวนี้คือ ควบคุมการรับส่งข้อมูลกับเครื่องคอมพิวเตอร์ และสามารถจัดการทางด้านข้อมูลในกรณีที่มีการจัดเก็บหรือบันทึกข้อมูลบางอย่างไว้เป็นฐานข้อมูล กลไกในการที่AT89C51 จะสื่อสารกับสมาร์ตการ์ดซึ่งในที่นี้คือบัตรTOT Card นั้น เกิดจากรูปแบบสัญญาณที่พอร์ต P0.2 – P0.6 (ขาที่ 39-35 ของ IC ตามลำดับ) ซึ่งเป็นไปตามที่ได้กล่าวมาแล้วในบทที่แล้ว

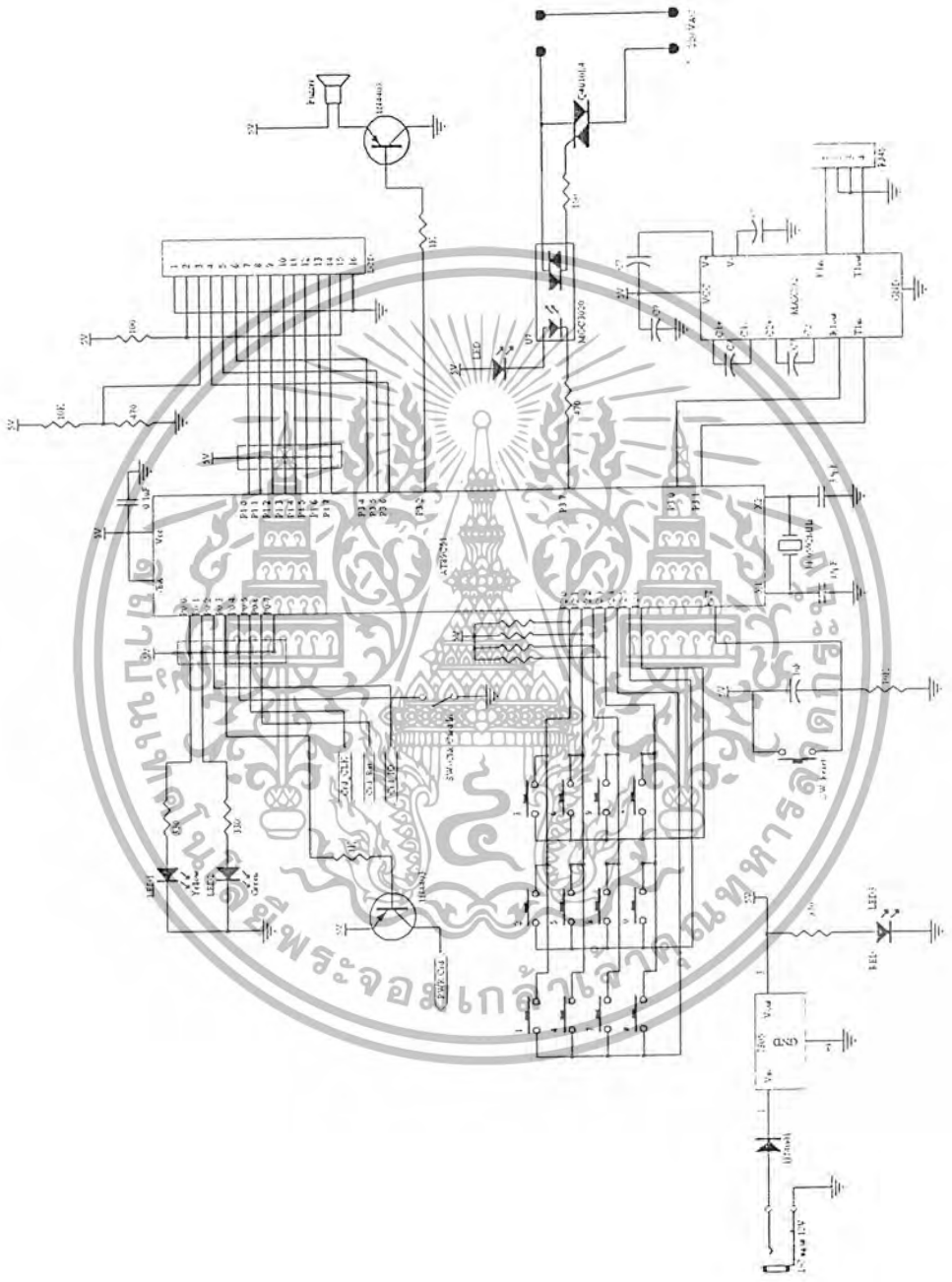
สำหรับการสื่อสารข้อมูลกับสมาร์ตการ์ดนั้น จะมีAT89C51 ควบคุมการจ่ายไฟให้กับบัตรทางบิตP0.3 สัญญาณจากบิตนี้จะถูกส่งไปทริกให้ทรานซิสเตอร์ 1N4403 ซึ่งทำงานเป็นสวิตช์ตัดต่อแหล่งจ่ายไฟให้บัตรทำงานหรือหยุดทำงาน บิตP0.6 และ บิต P0.5 จะทำงานเป็นขาสัญญาณรีเซตและสัญญาณนาฬิกา ตามลำดับ ส่วนบิต P0.4 ถูกใช้ในการตรวจสอบว่าบัตรถูกเสียบเข้ามายังช่องรับแล้วหรือไม่ จากสถานะของสวิตช์ที่ถูกซ่อนอยู่ภายในซ็อกเกตรับบัตรสมาร์ตการ์ด

การสื่อสารระหว่าง AT89C51 กับพอร์ตอนุกรม RS-232 ของเครื่องคอมพิวเตอร์ในแง่ของกระบวนการใช้จะถูกจัดการโดยโปรแกรมที่เขียนและบันทึกไว้ในAT89C51 ส่วนในแง่ของระดับสัญญาณที่ใช้ในการสื่อสารจะถูกจัดการโดยไอซี MAX232 ซึ่งจะทำการแปลงให้ระดับแรงดันจากที่ระดับ 0 และ 5 โวลต์ ไปเป็น +12 และ -12 โวลต์ เพื่อช่วยให้คอมพิวเตอร์สามารถเข้าใจกับข้อมูลที่ AT89C51 ส่งไปได้

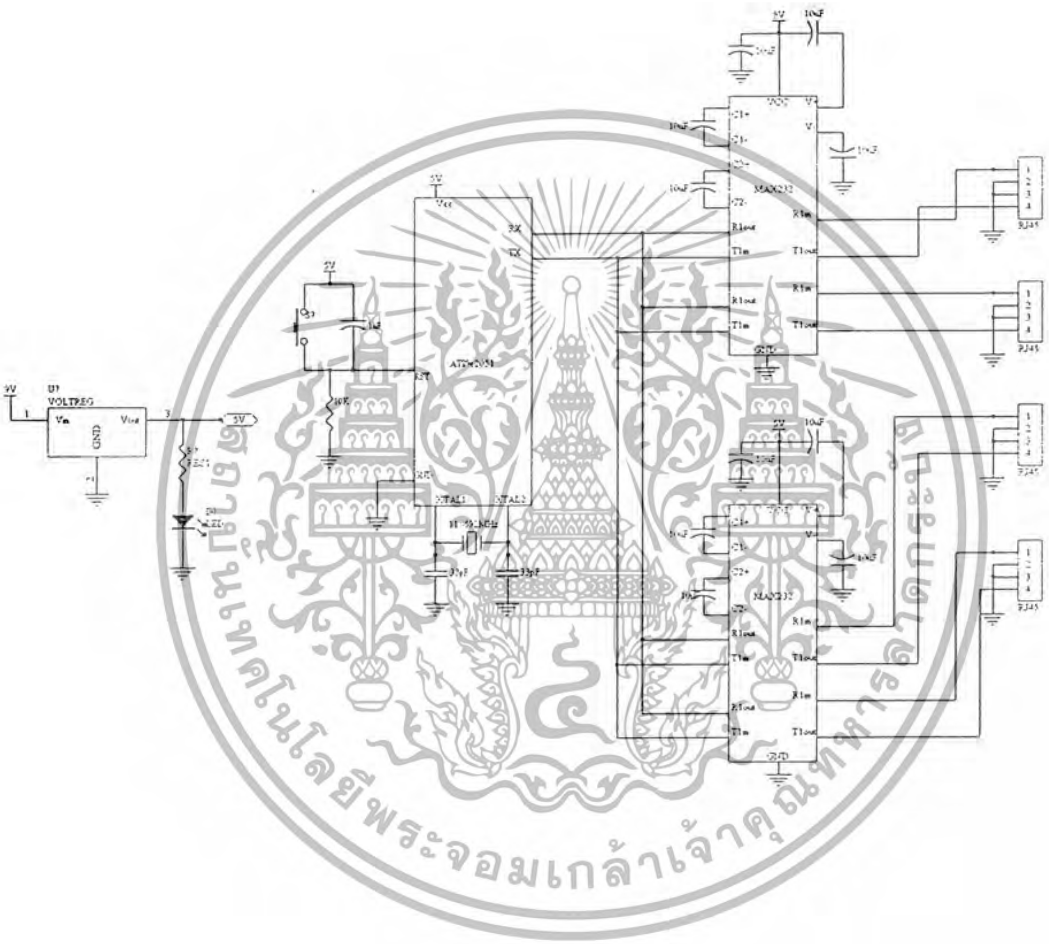
หน้าที่อีกอย่างของ AT89C51 ก็คือการควบคุมอุปกรณ์ภายนอก อย่างเช่น บัซเซอร์ , หลอดแอลอีดี ,จอแอลซีดี , คีย์แพด และ ออปโตไอโซเลเตอร์แบบไดแอค นั่นคือ บัซเซอร์จะถูกควบคุมจากบิต P3.2 บิตนี้จะจ่ายสัญญาณเป็นชุดของพัลส์ความถี่ ออกมายังทรานซิสเตอร์ 1N4403 เพื่อขับให้บัซเซอร์กำเนิดเสียง ส่วนหลอดแอลอีดี (LED1-LED2) จะถูกขับโดยตรงจาก AT89C51 โดยมีการฟูลอัพกระแสด้วยความต้านทาน 10K (R pack) และจำกัดกระแสที่จ่ายให้แอลอีดีแต่ละตัวด้วยความต้านทาน 330 Ω นอกจากนั้นยังใช้พอร์ต P1 ในการส่งข้อมูลแสดงผลออกจอแอลซีดี โดยมีการใช้ขา P3.4 - P3.6 ช่วยในการควบคุมการแสดงผลดังกล่าว และใช้ขา P2.0 - P2.6 ในการควบคุมและรับค่าที่ได้ป้อนเข้ามาทางคีย์แพดเพื่อประโยชน์ในการกรอกรหัสผ่าน และอื่นๆ อีกส่วนหนึ่งคือ ออปโตไอโซเลเตอร์แบบไดแอค ควบคุมโดย P3.7 เพื่อที่จะนำไปทริกให้ขาเกทของไตรแอค TR1 แล้วนำไปควบคุม อุปกรณ์ไฟฟ้าที่จุดที่ได้ตามต้องการ

สำหรับในส่วนของแหล่งจ่ายไฟของทั้งวงจรนั้น ได้ใช้วงจรเรกูเลตแรงดันให้มีค่าเท่ากับ 5 V ซึ่งในที่นี้ได้ใช้ไอซีเบอร์ LM7805 มาช่วยในการทำงานในส่วนนี้

วงจรข้างต้นนั้นเป็นวงจรที่ใช้สำหรับระบบความปลอดภัยในการเข้าของส่วนบุคคล สำหรับวงจรในส่วนของการป้องกันประตูทางเข้าออกอาคาร และในส่วนของการอำนวยความสะดวกในการคำนวณและบันทึกค่าบริการในร้านอาหาร นั้นจะมีความคล้ายคลึงกันมากแตกต่างกันในส่วนของโปรแกรมการควบคุมภายใน MCS51 และรูปแบบข้อมูลที่ส่งให้แก่คอมพิวเตอร์ส่วนกลางเพื่อบันทึกไว้เท่านั้น จึงไม่ได้แสดงไว้ ณ ที่นี้

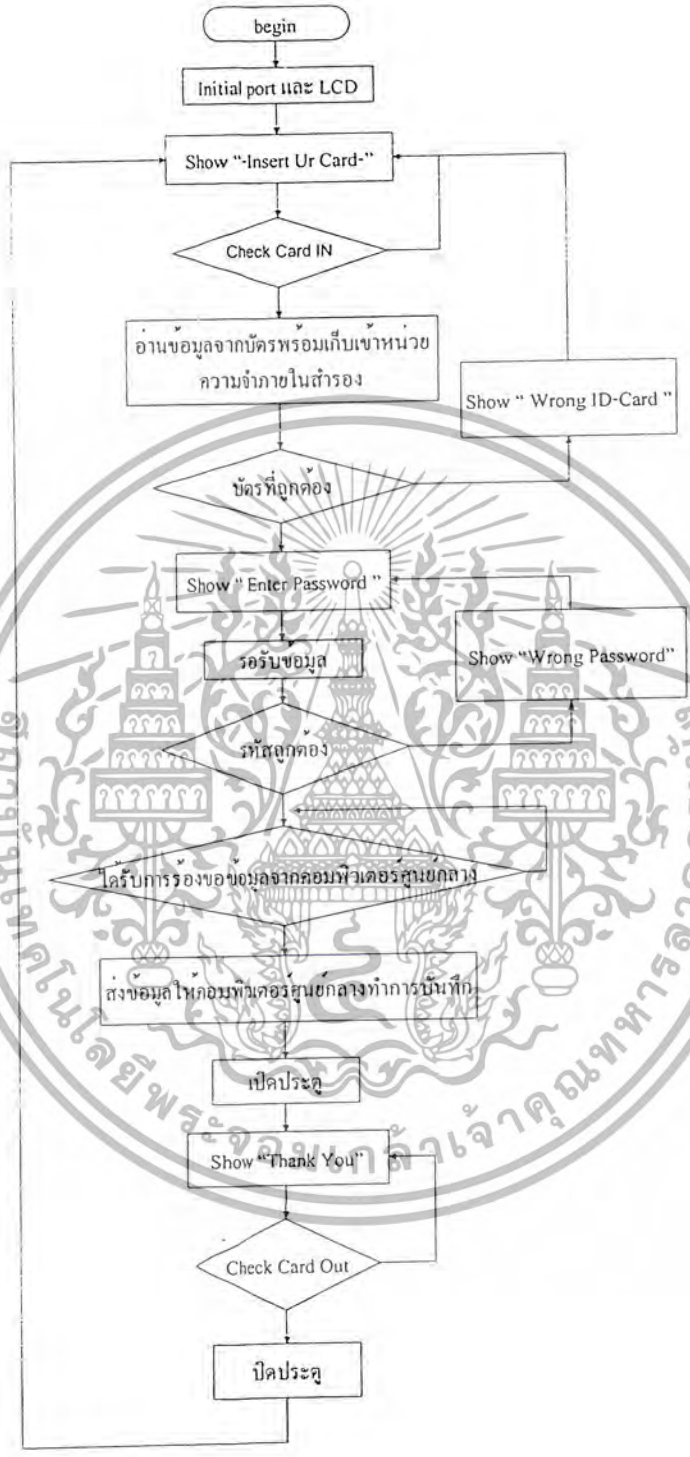


รูปที่ 5.2 วงจรเครื่องอ่านบัตรสมาร์ตการ์ด



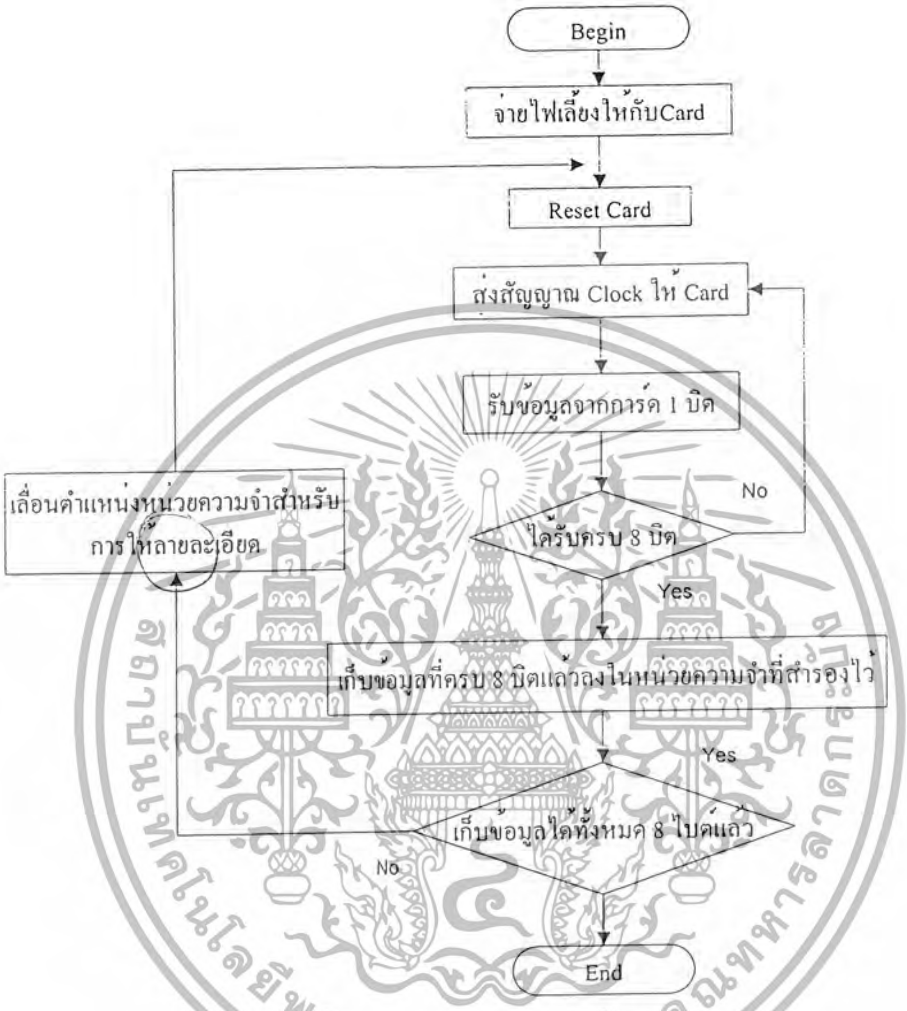
รูปที่ 5.3 วงจรจัดการการส่งรับส่งข้อมูลระหว่างเครื่องอ่านทุกเครื่องและคอมพิวเตอร์ส่วนกลาง

โฟลชาร์ตการทำงานของวงจรเครื่องอ่านหมายเลขบัตร TOT Card



รูปที่ 5.4 โฟลชาร์ตการทำงานของวงจรเครื่องอ่านหมายเลขบัตร TOT Card

โฟลชาร์ตการทำงานของโปรแกรมในส่วนของการอินเตอร์เฟสกับบัตร TOT Card

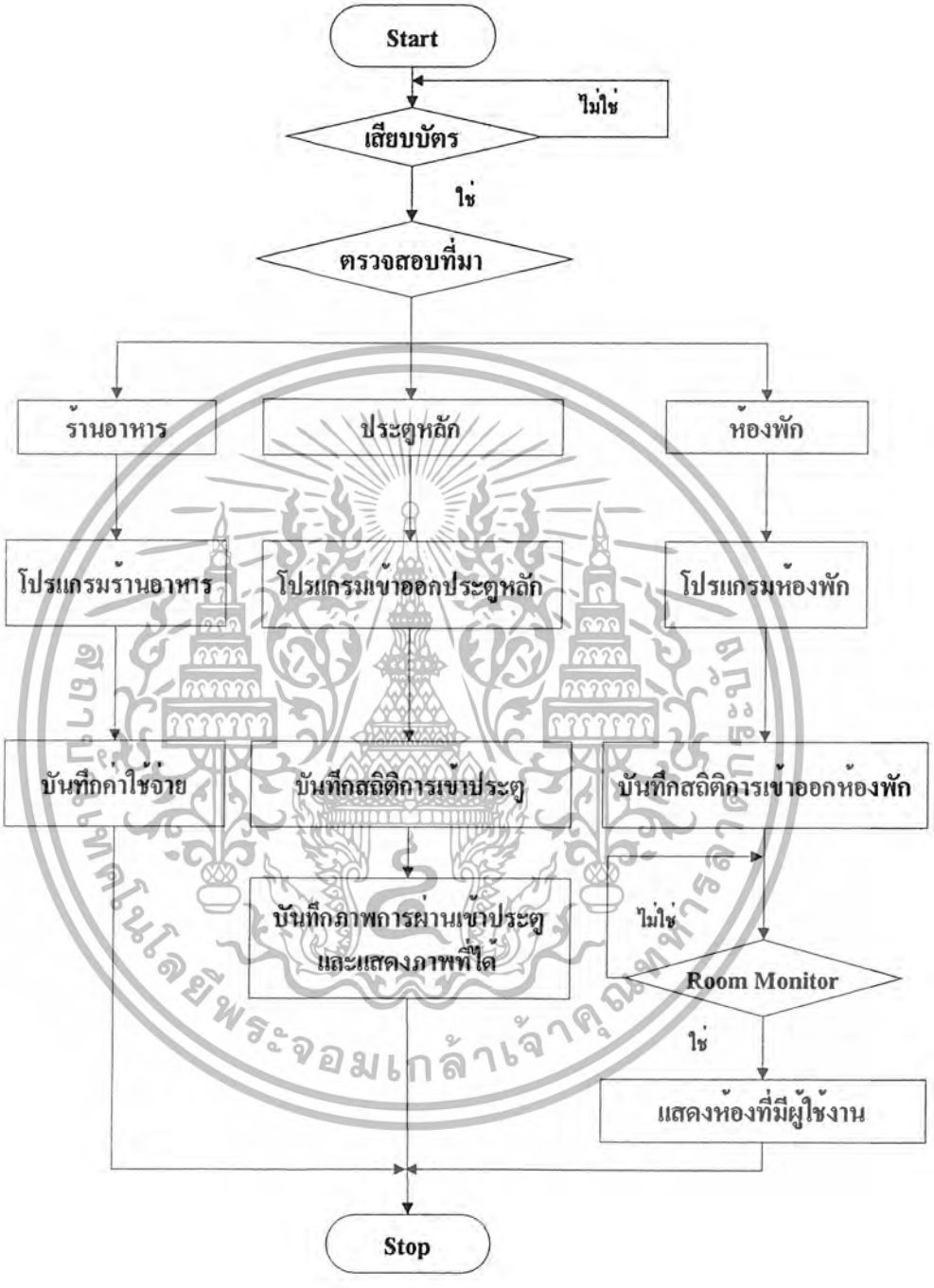


รูปที่ 5.5 โฟลชาร์ตการทำงานของโปรแกรมในส่วนของการอินเตอร์เฟสกับบัตร TOT Card

โฟล์วชาร์ตการทำงานของวงจรจัดการการส่งรับส่งข้อมูลระหว่างเครื่องอ่านทุกเครื่องและคอมพิวเตอร์ส่วนกลาง

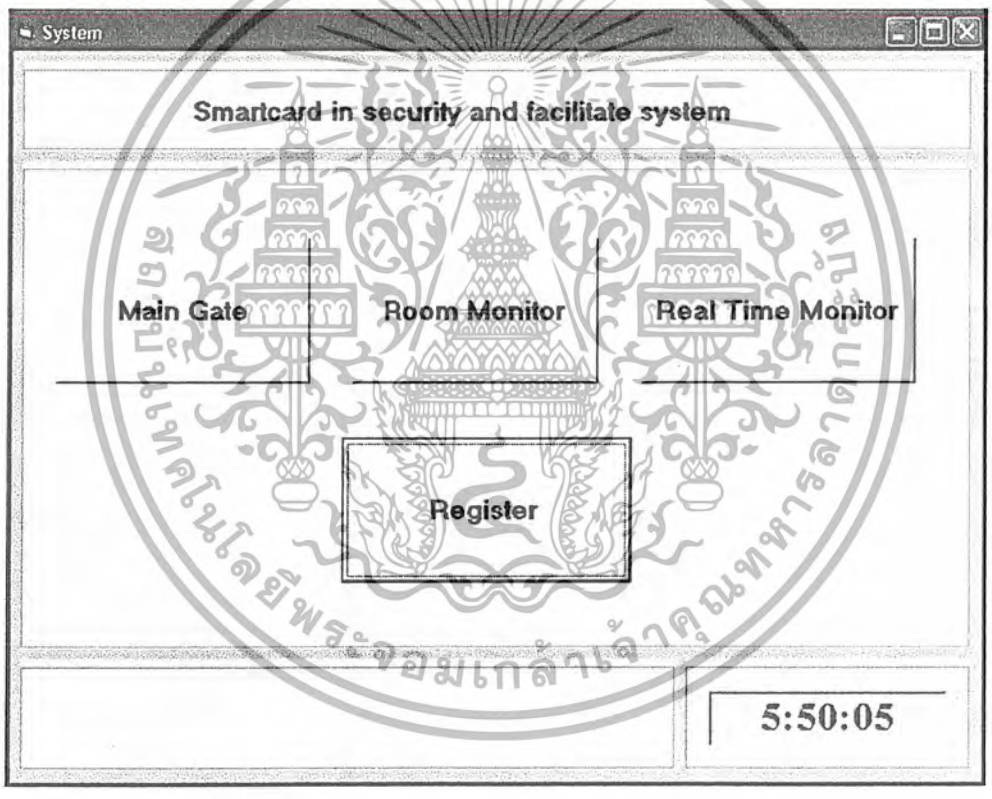
รูปที่ 5.6 โฟล์วชาร์ตการทำงานของวงจรจัดการการส่งรับส่งข้อมูลระหว่างเครื่องอ่านทุกเครื่องและคอมพิวเตอร์ส่วนกลาง

5.3 การทำงานของโปรแกรมรักษาความปลอดภัยและอำนวยความสะดวก



รูปที่ 5.5 โฟลชาร์ตการทำงานของโปรแกรมรักษาความปลอดภัยและอำนวยความสะดวก

การทำงานจะเริ่มต้นที่เมื่อเปิดโปรแกรมรักษาความปลอดภัยและอำนวยความสะดวกขึ้นมา ระบบ ก็จะทำการตรวจสอบว่ามีการส่งข้อมูลมาจากเครื่องอ่านเครื่องใดแล้วทำการดำเนินการตาม โปรแกรมย่อยนั้นๆ โดย ในระบบจะมี เครื่องลูกอยู่ 3 เครื่องคือ เครื่องที่ประตูหลัก เครื่องที่ร้านอาหาร และเครื่องที่ห้องพัก โดยทั้งหมดจะทำการติดต่อกับฐานข้อมูล Access 2000 และในโปรแกรมรักษา ความปลอดภัยและอำนวยความสะดวกนี้ จะมีส่วนสำหรับเรียก โปรแกรม บันทึกข้อมูลบัตรใหม่ และ Real Time Monitor รวมอยู่ด้วย ซึ่ง โปรแกรมย่อยทั้งหมดจะกล่าวถึงในลำดับต่อไป

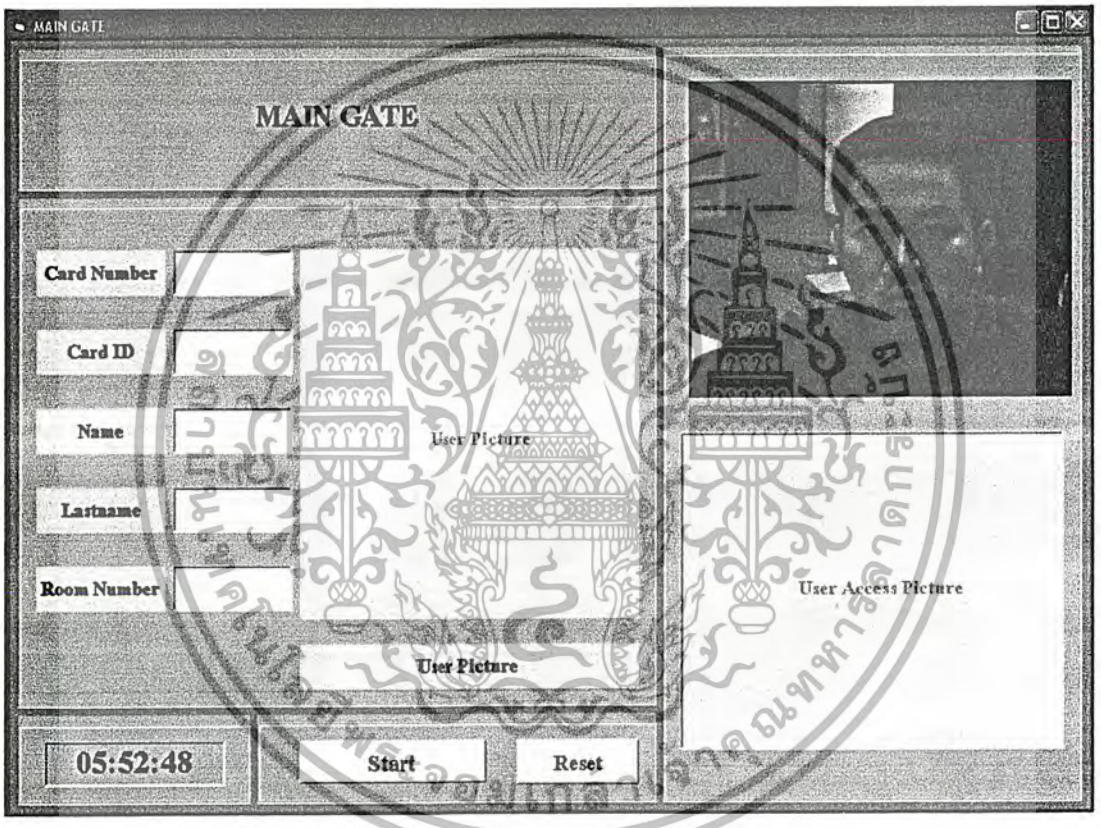


รูปที่ 5.6 แสดงรูปแบบของโปรแกรมรักษาความปลอดภัยและอำนวยความสะดวก

5.3.1 โปรแกรมการเข้าออกประตูหลัก

รูปที่ 5.7 โฟลชาร์ต การทำงานของโปรแกรมการเข้าออกประตูหลัก

เมื่อเครื่องอ่านได้อ่านรหัสบัตรผู้ใช้งานได้แล้ว จะทำการส่งข้อมูลที่อ่านได้มายังเครื่องคอมพิวเตอร์ซึ่งจะทำการค้นหาว่าบัตรที่เสียบเข้ามาเป็นบัตรที่ถูกต้องหรือไม่ ถ้าถูกต้องก็ทำการเปิดประแสดงฐานข้อมูลของผู้ใช้บัตรรายนั้นๆออกมา โดยจะมี โปรแกรม Visual Basic 6 เป็นตัวเชื่อมต่อระหว่างเครื่องอ่านบัตรและฐานข้อมูล ซึ่งเป็น Access 2000 โดยฐานข้อมูลจะประกอบไปด้วย หมายเลขบัตร, หมายเลขประจำบัตร, ชื่อ , นามสกุล ,หมายเลขห้อง จากนั้นจะทำการเก็บภาพการเข้าออกนั้นไว้ โดยทำการสั่งให้กล้องที่ติดตั้งไว้ถ่ายภาพ และบันทึกเวลาการเข้าประตูออกมาเป็นรายงาน

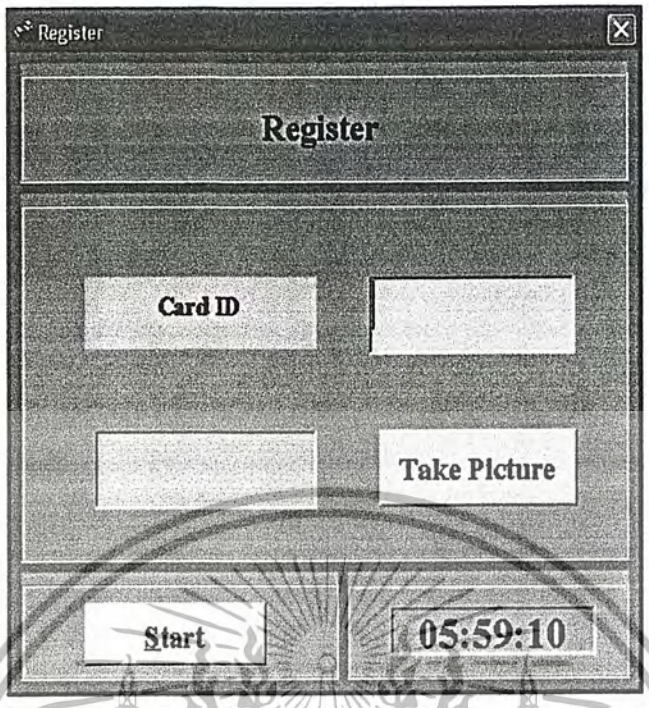


รูปที่ 5.8 แสดงฐานข้อมูลผู้ใช้งานที่ประตูหลัก

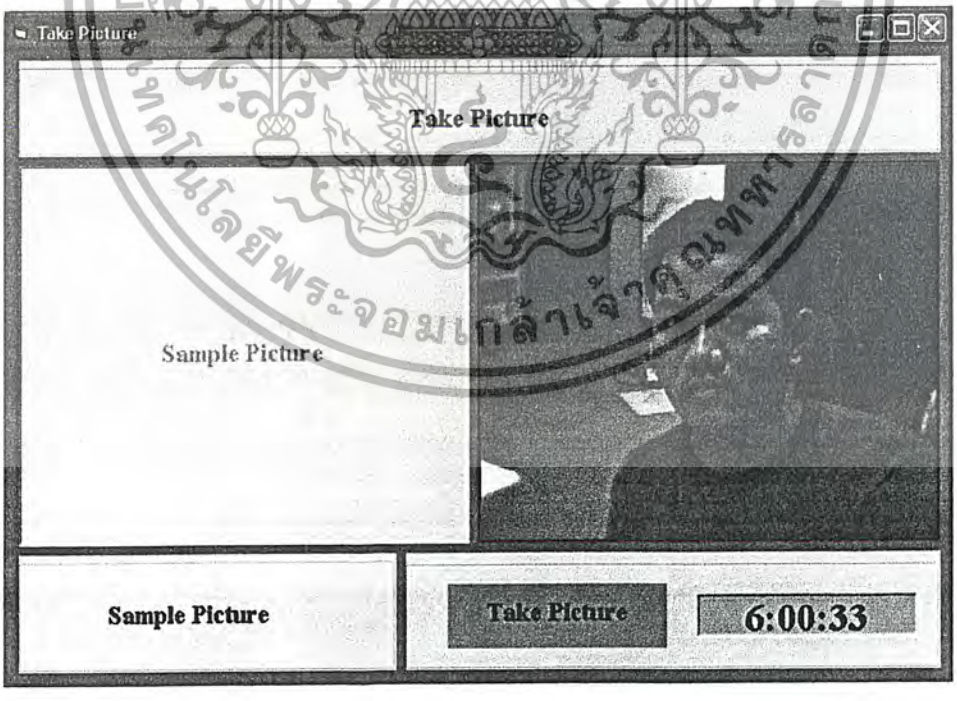
5.3.2 บันทึกข้อมูลบัตรใหม่

รูปที่ 5.9 โฟลชาร์ต การทำงานของโปรแกรมการบันทึกข้อมูลบัตรใหม่

หลักการทำงานจะเริ่มต้นเมื่อทำการเสียบบัตรแล้วโปรแกรมจะทำการแสดงเลขบัตรออกมา จากนั้นจะทำการเช็คว่ามีบัตรนี้อยู่ในฐานเข้ามูลหรือไม่ จากนั้นทำการถ่ายภาพโดยการสั่งให้กล้องถ่ายภาพแล้วนำสำเนาภาพไปเก็บไว้เพื่อใช้ในการอ้างอิงต่อไป



รูปที่ 5.10 แสดงรูปแบบของโปรแกรมบันทึกข้อมูลบัตรใหม่

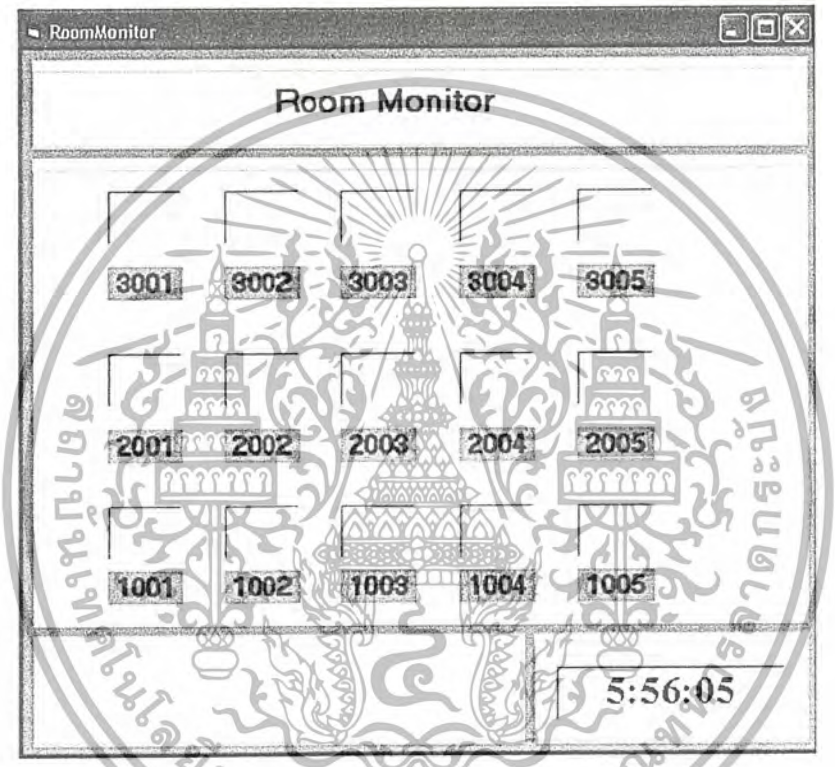


รูปที่ 5.11 แสดงรูปแบบของโปรแกรมบันทึกภาพ

5.3.3 ห้องพัก

รูปที่ 5.12 โฟลชาร์ต การทำงานของโปรแกรมห้องพัก

เมื่อทำการเปิดโปรแกรมประตูหลัก เมื่อต้องการเข้าห้องแต่ละห้องจะต้องทำการการกดรหัสประจำห้องที่ถูกต้องแล้วเครื่องอ่านจะทำการติดต่อกับโปรแกรมหลัก โปรแกรมจะทำการบันทึกเวลาการเข้าออกและแสดงให้เห็นว่าห้องใดมีผู้ใช้งานอยู่โดยจะมีสถานะเป็นสีเข้ม



รูปที่ 5.13 แสดงรูปแบบของโปรแกรมห้องพัก

5.3.4 ร้านอาหาร

รูปที่ 5.14 โฟลชาร์ต การทำงานของโปรแกรมร้านอาหาร

เมื่อเริ่มทำงานโปรแกรมจะทำการตรวจสอบว่าบัตรนี้เป็นของ User ใด โดยทำการติดต่อ กับฐานข้อมูล จากนั้นทำการบันทึกรายจ่าย รวมทั้งบันทึกวันเวลาที่ใช้งานไว้ด้วย

บทที่ 6 การทดลองและผลการทดลอง

6.1 การทดลองวัดความผิดพลาดการรับส่งข้อมูลจากเครื่องอ่านมายังโปรแกรมฐานข้อมูล

6.1.1 ใช้สายรับสัญญาณยาว 9 ฟุตในการรับสัญญาณ

6.1.2 ความเร็วในการรับข้อมูล 9600 บิต/วินาที ผ่านพอร์ตคอม 1

6.1.3 ทำการทดลองโดยทำการรับข้อมูลจากบัตร TOT ตัวอย่าง10บัตรจำนวนบัตรละ100 ครั้ง

ผลการทดลองรับค่าจากเครื่องอ่านบัตรมายังโปรแกรมฐานข้อมูล

หมายเลขบัตร	หมายเลขประจำบัตร	จำนวนครั้งที่ถูกต้อง	จำนวนครั้งที่ผิดพลาด	เปอร์เซ็นต์ความผิดพลาด
1	1234066493	100	0	0%
2*	4B003A971F	100	0	0%
3*	1281124001	99	1	1%
4	1282173972	100	0	0%
5	1279070253	100	0	0%
6	1282173989	100	0	0%
7	1289043389	100	0	0%
8	1089045450	100	0	0%
9*	1326016520	99	1	1%
10	1301000116	100	0	0%
	รวม	998	2	0.02%

ตารางที่ 6.1 ผลการทดลองความผิดพลาดของเครื่องอ่านบัตร

หมายเหตุ 2* บัตรโทรศัพท์ของ China Telecom

3* ความผิดพลาดที่วัดได้เป็น FFFFFFFFFF

9* ความผิดพลาดที่วัดได้เป็น FFEBF13260

6.2 การทดลองวัดความผิดพลาดการรับส่งข้อมูลที่ระยะสายส่งต่างๆ

- ทำการอ่านหมายเลขบัตรที่ระยะความยาวของสายส่งต่างๆกัน คือ 10 , 20 และ 30 ฟุต ตามลำดับ แล้วบันทึกจำนวนครั้งที่ถูกต้องและจำนวนครั้งที่ผิดพลาด พร้อมคำนวณเปอร์เซ็นต์ความผิดพลาด

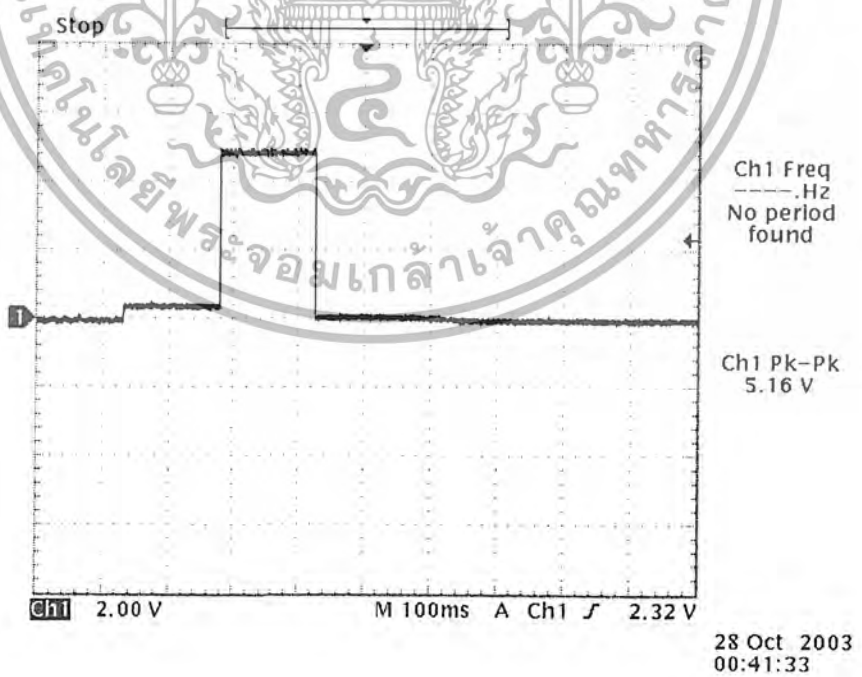
ความยาวสายส่ง (ฟุต)	จำนวนครั้งที่ถูกต้อง	จำนวนครั้งที่ผิดพลาด	เปอร์เซ็นต์ความผิดพลาด
10	100	0	0 %
20	99	1	1 %
30	99	1	1 %

ตารางที่ 6.2 ผลการทดลองความผิดพลาดของเครื่องอ่านบัตรในสายส่งระยะต่างๆ

6.3 การทดลองวัดรูปสัญญาณ ณ ตำแหน่งขาต่างๆในสมาร์ตการ์ด

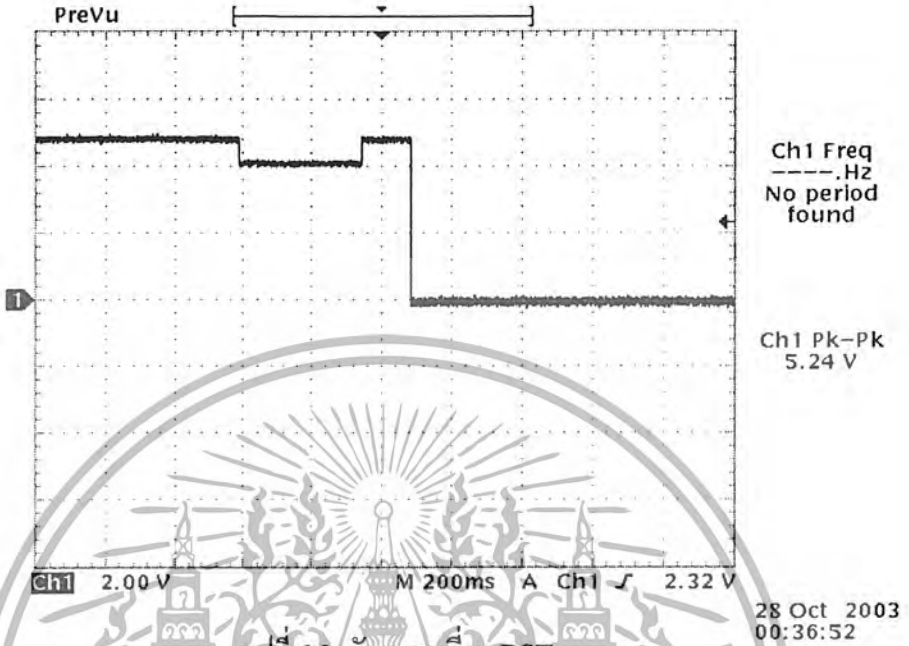
- ทำการวัดสัญญาณที่ขาต่างๆ ของสมาร์ตการ์ด ซึ่งได้แก่ Vcc , RST , CLK และ I/O

6.3.1 สัญญาณที่ขา Vcc



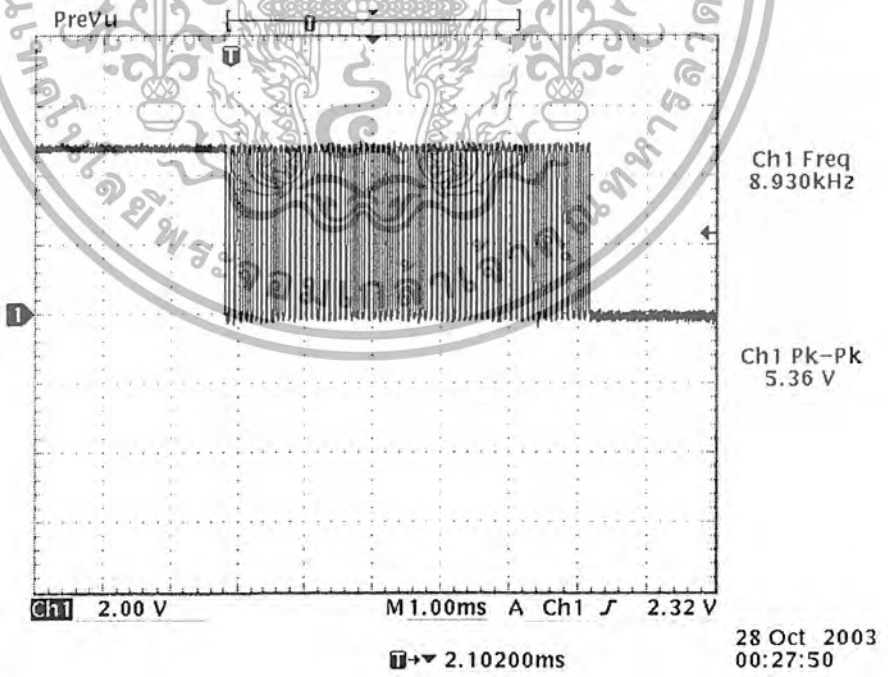
รูปที่ 6.1 รูปสัญญาณที่ขา Vcc

6.3.2 สัญญาณที่ขา RST

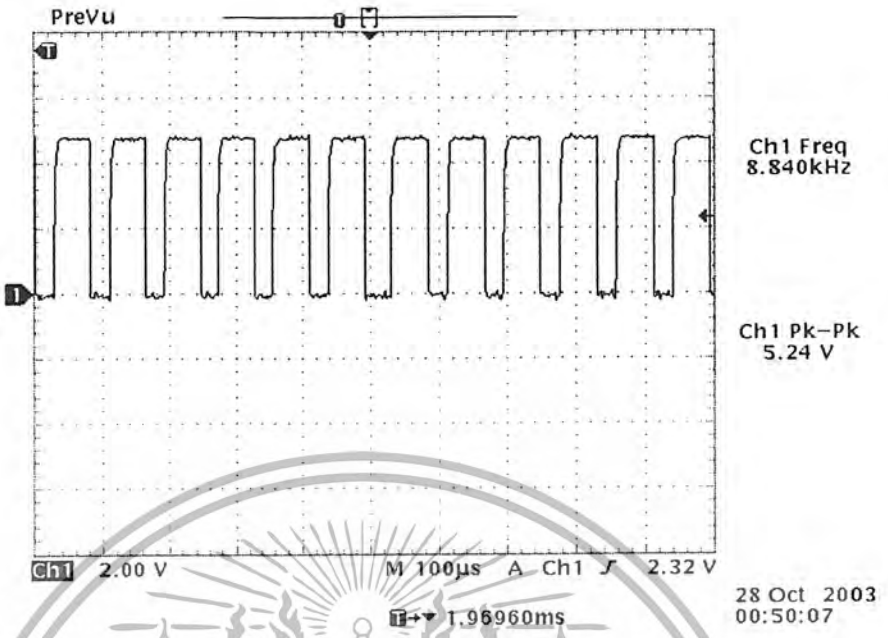


รูปที่ 6.2 สัญญาณที่ขา RST

6.3.3 สัญญาณที่ขา CLK

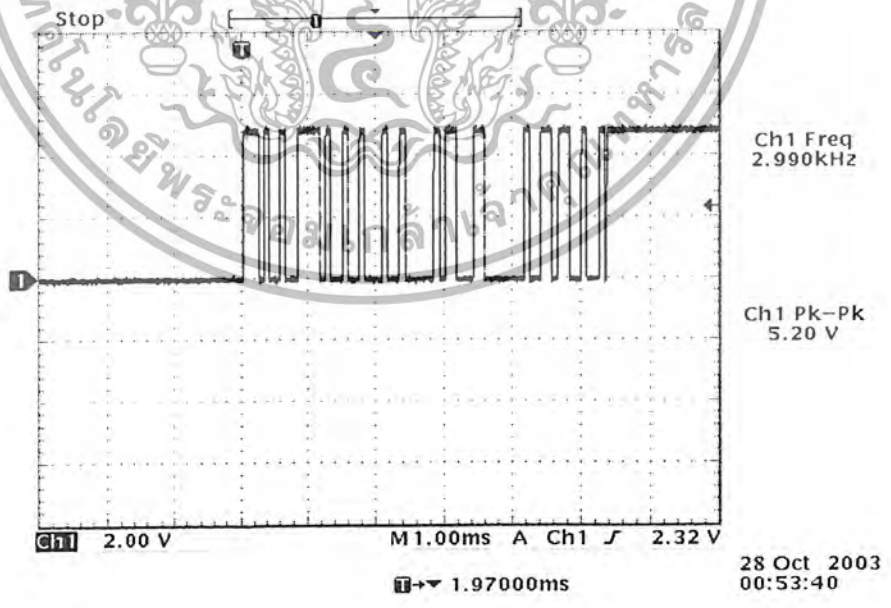


รูปที่ 6.3 สัญญาณที่ขา CLK

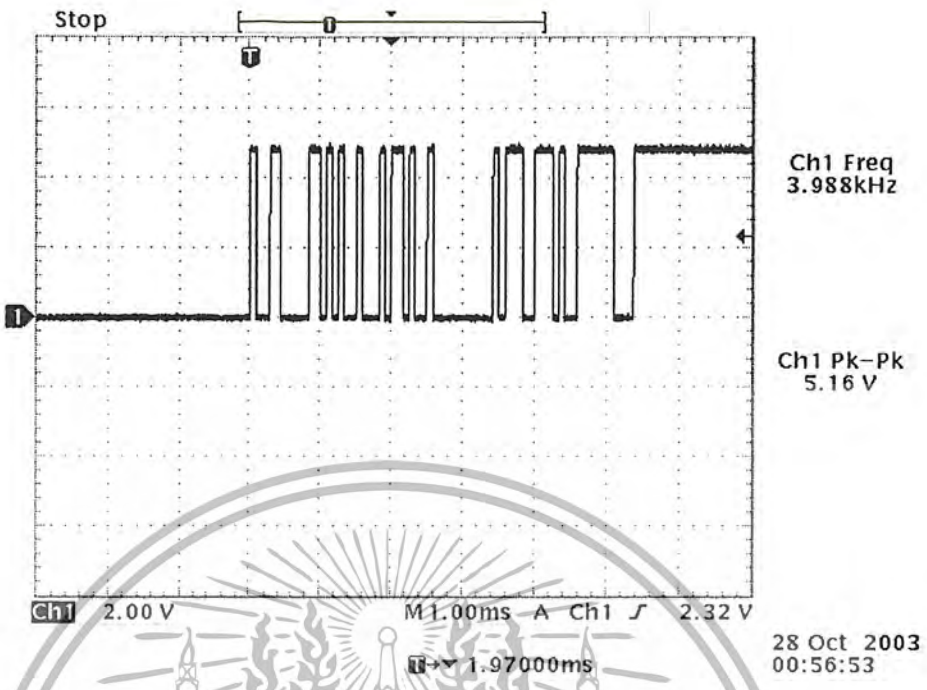


รูปที่ 6.4 สัญญาณที่ขา CLK (ขยายให้เห็นชัดมากขึ้น)

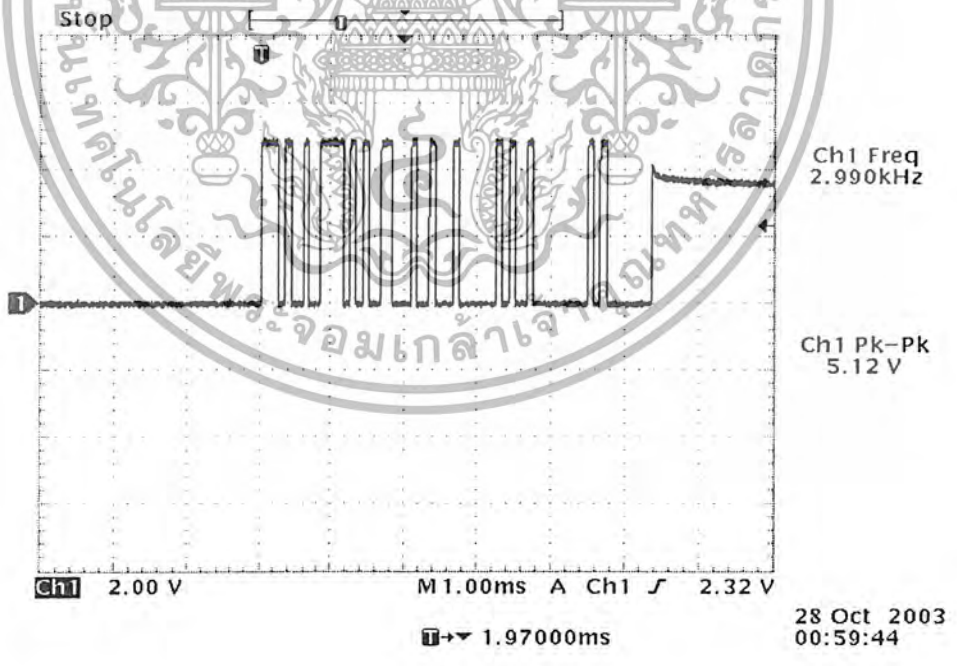
6.3.4 สัญญาณที่ขา I/O



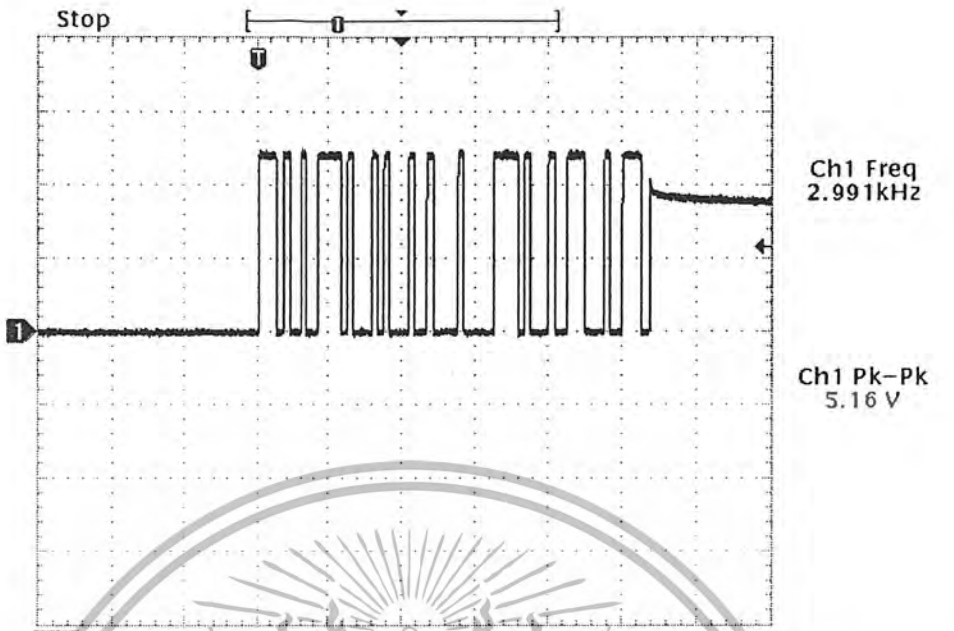
รูปที่ 6.5 สัญญาณที่ขา I/O ของการ์ด หมายเลข 1



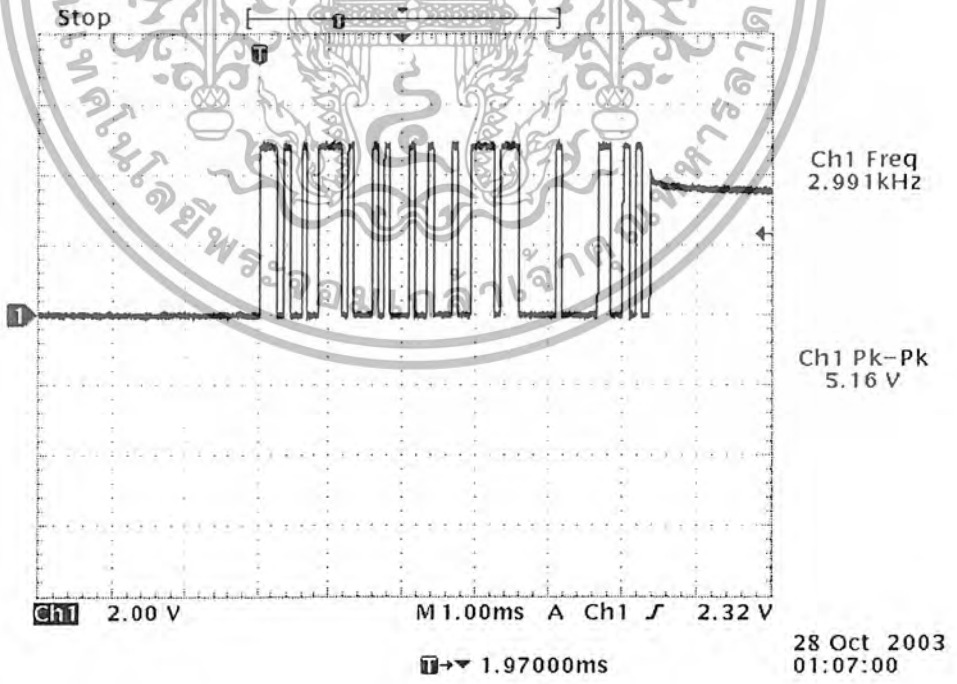
รูปที่ 6.6 สัญญาณที่ขา I/O ของการ์ด หมายเลข 2



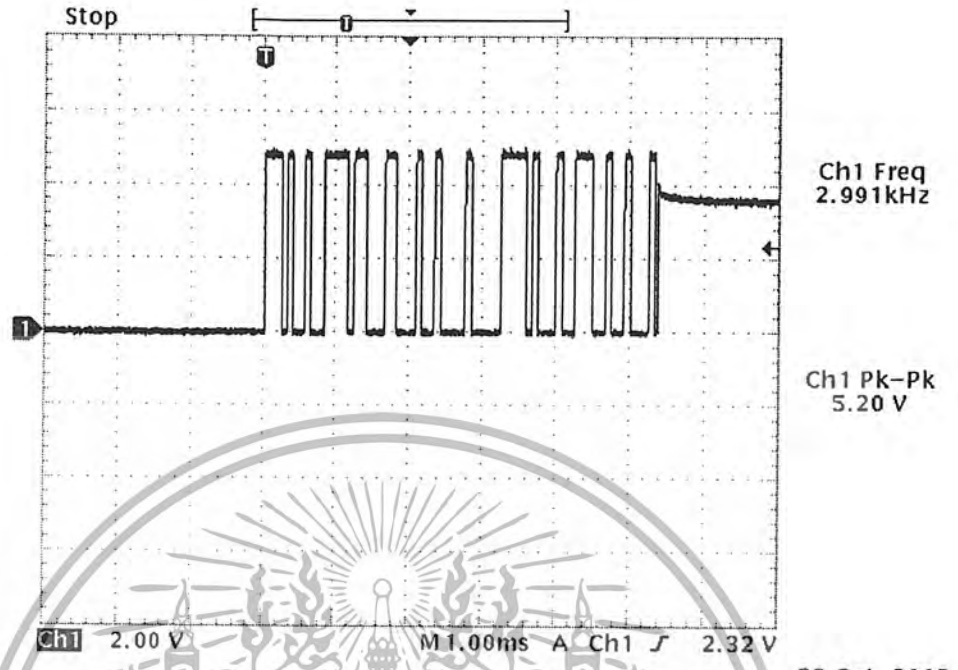
รูปที่ 6.7 สัญญาณที่ขา I/O ของการ์ด หมายเลข 3



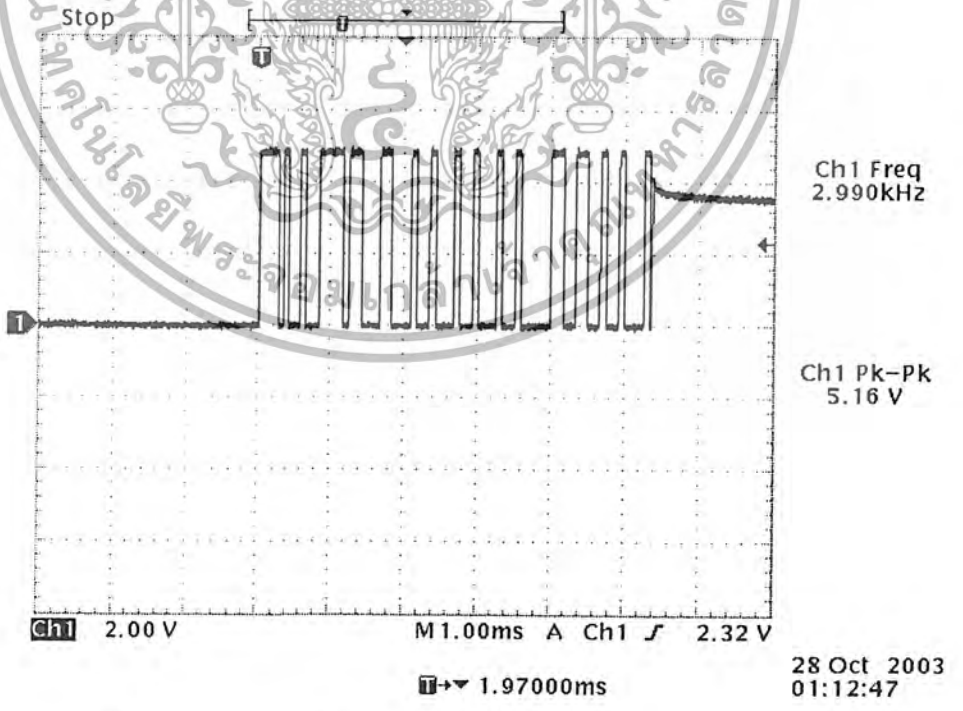
รูปที่ 6.8 สัญญาณที่ขา I/O ของการ์ด หมายเลข 4



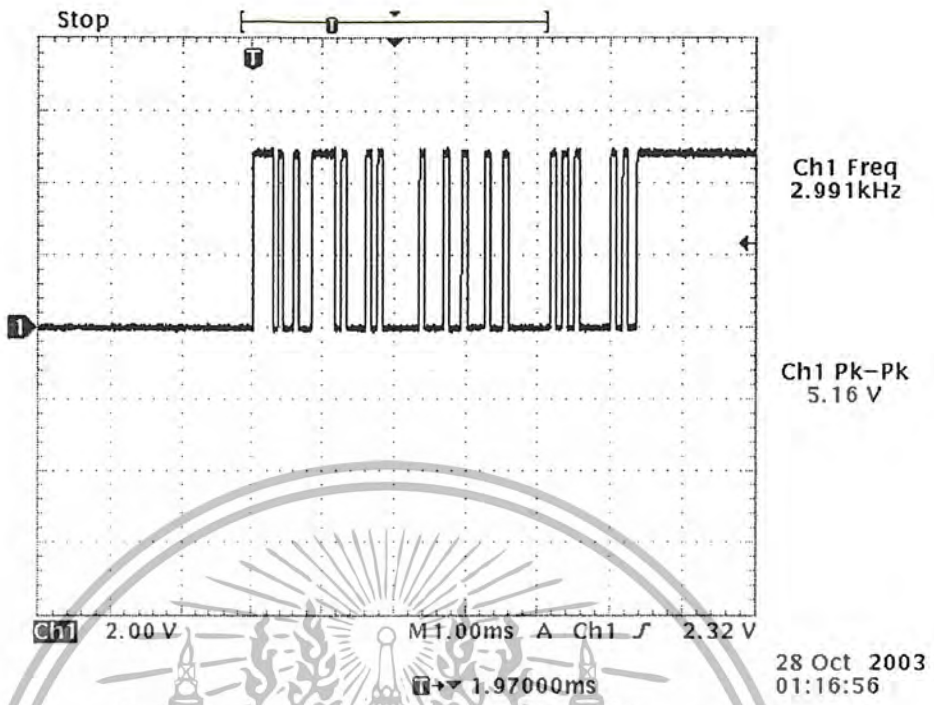
รูปที่ 6.9 สัญญาณที่ขา I/O ของการ์ด หมายเลข 5



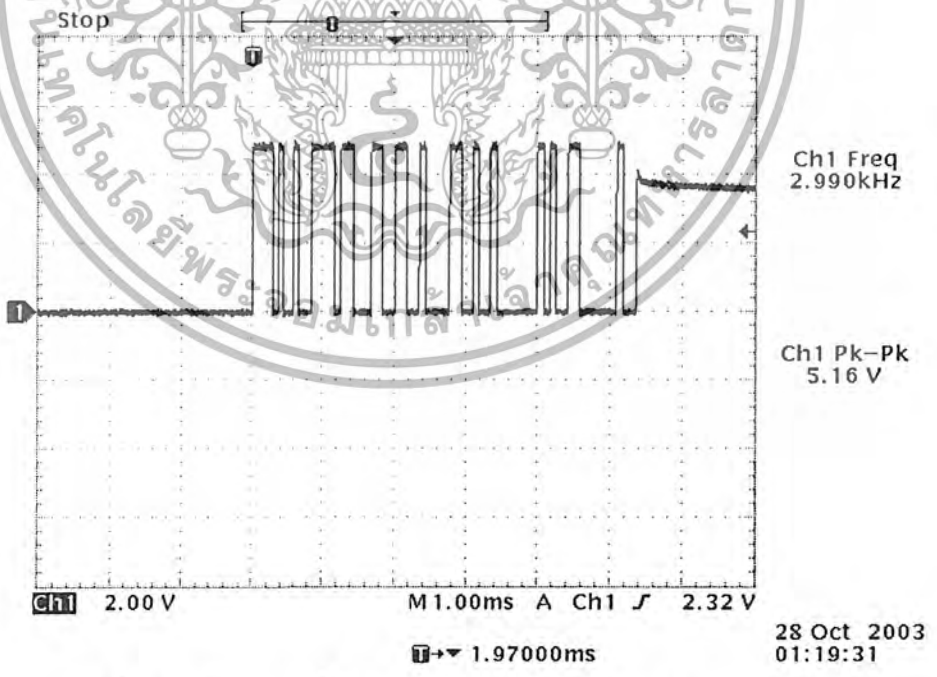
รูปที่ 6.10 สัญญาณที่ขา I/O ของการ์ด หมายเลข 6



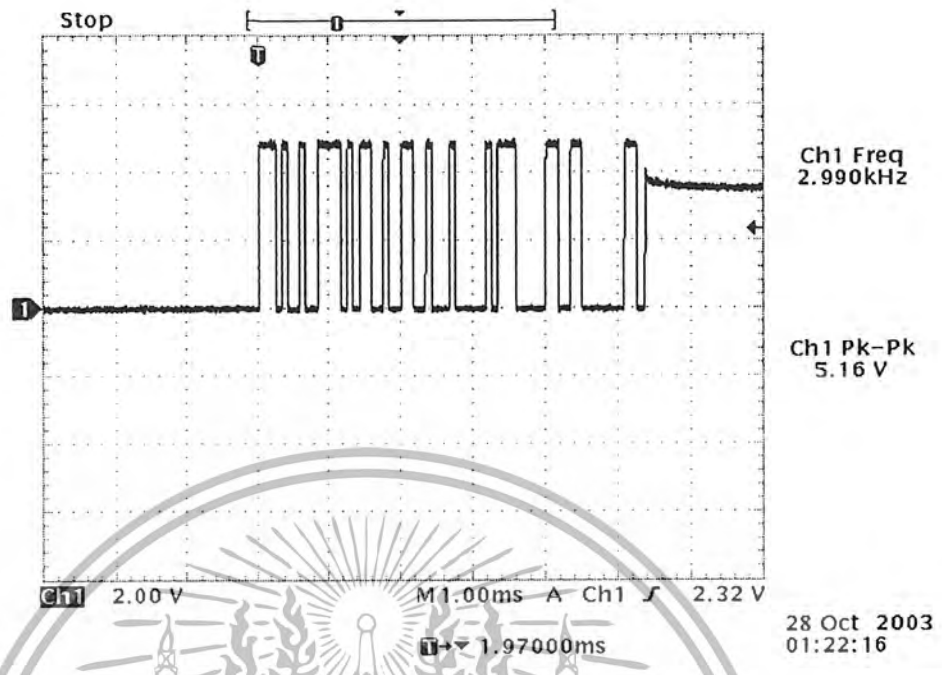
รูปที่ 6.11 สัญญาณที่ขา I/O ของการ์ด หมายเลข 7



รูปที่ 6.12 สัญญาณที่ขา I/O ของการ์ด หมายเลข 8



รูปที่ 6.13 สัญญาณที่ขา I/O ของการ์ด หมายเลข 9



รูปที่ 6.14 สัญญาณที่ขา I/O ของการ์ด หมายเลข 10

6.4 การทดลองโปรแกรมรักษาความปลอดภัยและอำนวยความสะดวก

6.4.1 ผลการทดลองการลงทะเบียนบัตร

6.4.1.1 เปิดโปรแกรมรักษาความปลอดภัยและอำนวนความสะดวก

6.4.1.2 เลือกที่ Register

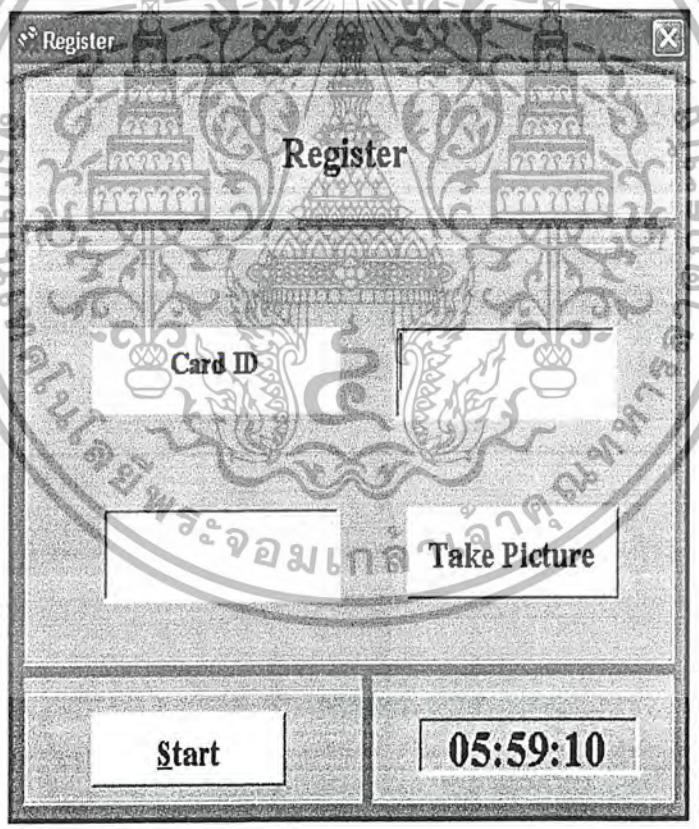
6.4.1.3 จะมีรูปโปรแกรมขึ้นมาดังรูป

6.4.1.4 เสียบบัตรใบใหม่เข้า SmartCard Reader Module

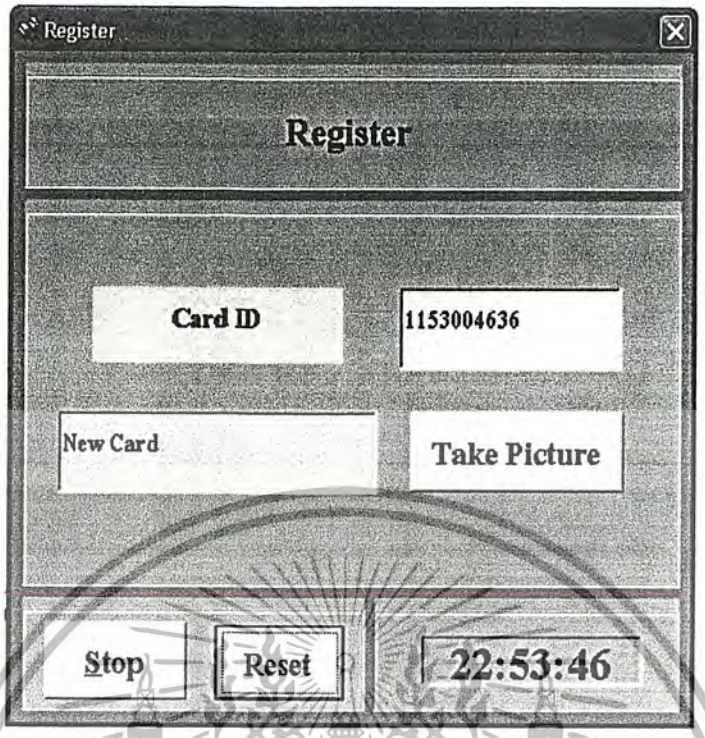
6.4.1.5 เมื่อทำการเสียบบัตร จะขึ้นว่า New Card พร้อมกับขึ้นเลขบัตร

6.4.1.6 ทำการเปิดฐานข้อมูล Project.mdb ทำการบันทึกเลขบัตรใบใหม่ลงไป

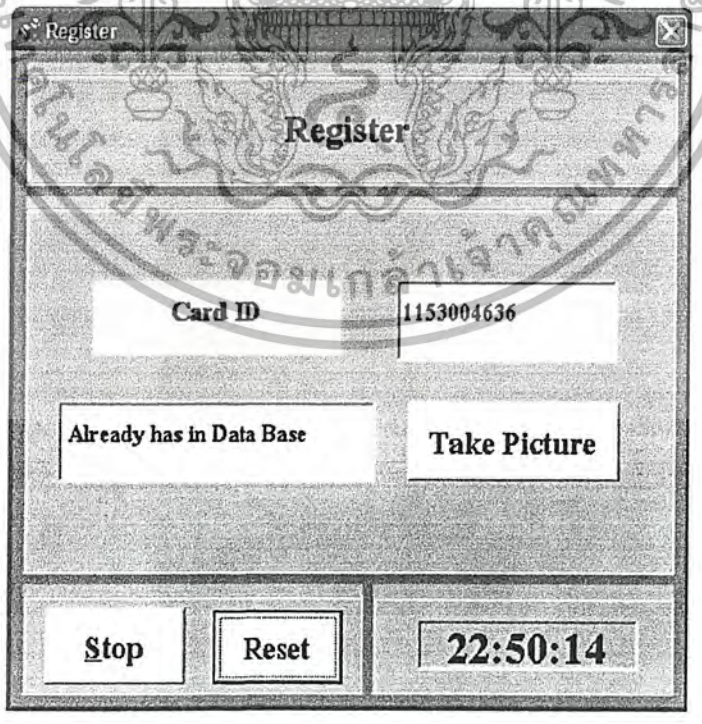
6.4.1.7 ทำการเสียบบัตรใบใหม่อีกครั้ง จะขึ้นคำว่า Already has in data base แสดงว่าการลงทะเบียนบัตรใหม่สำเร็จ



รูปที่ 6.15 รูปโปรแกรมส่วน Regigter ก่อนทำการเสียบบัตร



รูปที่ 6.16 รูปโปรแกรมส่วน Regigter หลังการเสียบบัตรใหม่



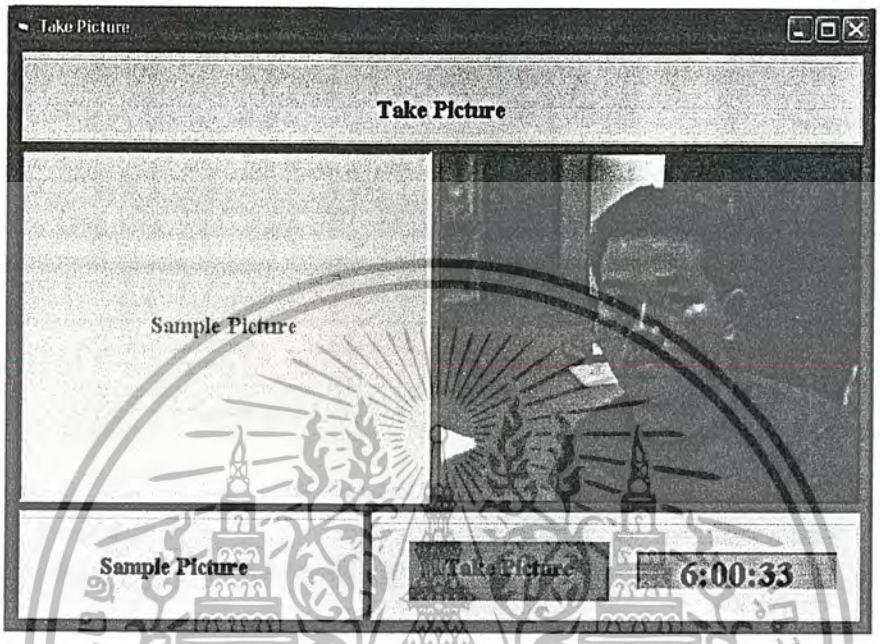
รูปที่ 6.17 รูปโปรแกรมส่วน Regigter เมื่อทำการบันทึกข้อมูลลงฐานข้อมูลแล้ว

6.4.2 ผลการทดลองการบันทึกภาพผู้ใช้รายใหม่

6.4.2.1 เมื่อทำการบันทึกข้อมูลบัตรใบใหม่เรียบร้อยแล้วจากนั้นเลือก Take Picture

6.4.2.2 กดที่ Take Picture ก็จะขึ้นภาพตัวอย่างที่บันทึกไว้ขึ้นมา

6.4.2.3 เข้าไปแก้ชื่อภาพให้ตรงกับชื่อหมายเลขบัตร



รูปที่ 6.17 รูปโปรแกรมส่วน Regigter เมื่อทำการบันทึกข้อมูลลงฐานข้อมูลแล้ว

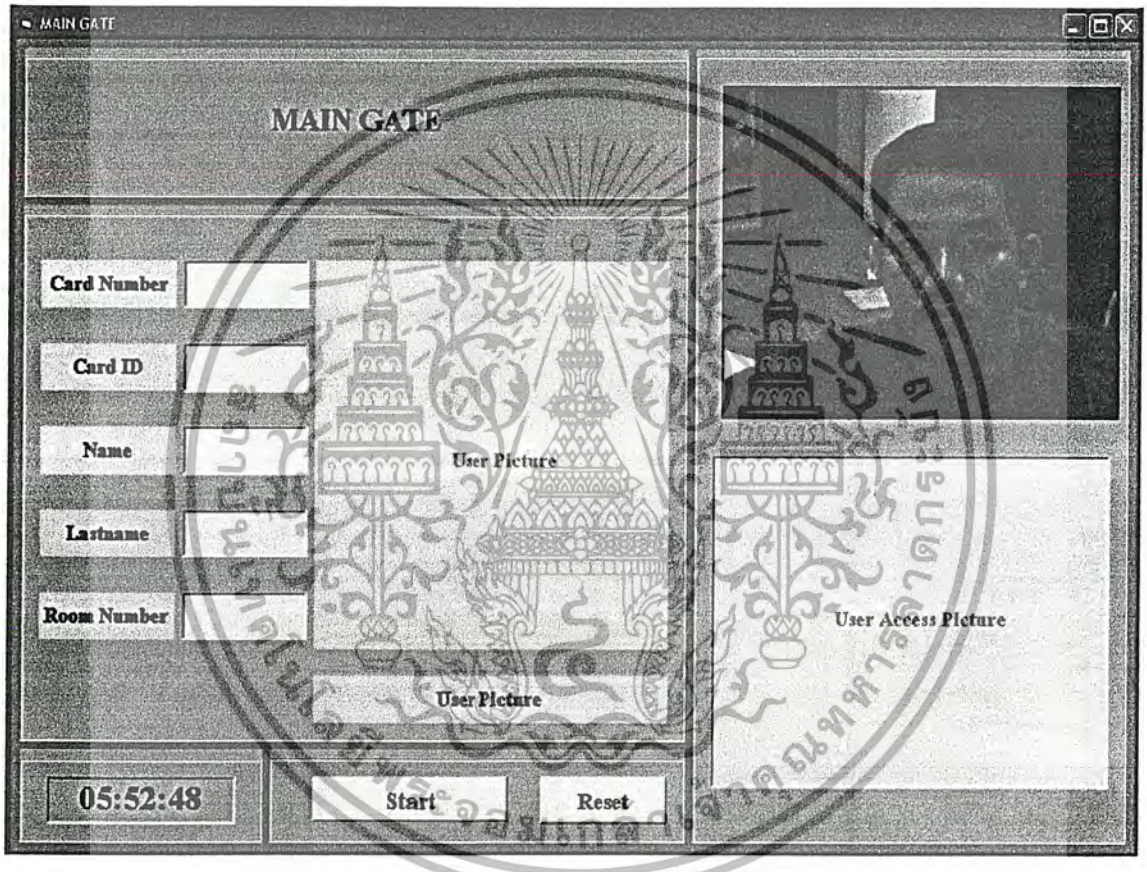
รูปที่ 6.17 รูปโปรแกรมส่วน Regigter เมื่อทำการบันทึกข้อมูลลงฐานข้อมูลแล้ว

6.4.3 ผลการทดลองการเข้าออกที่ประตูหลัก

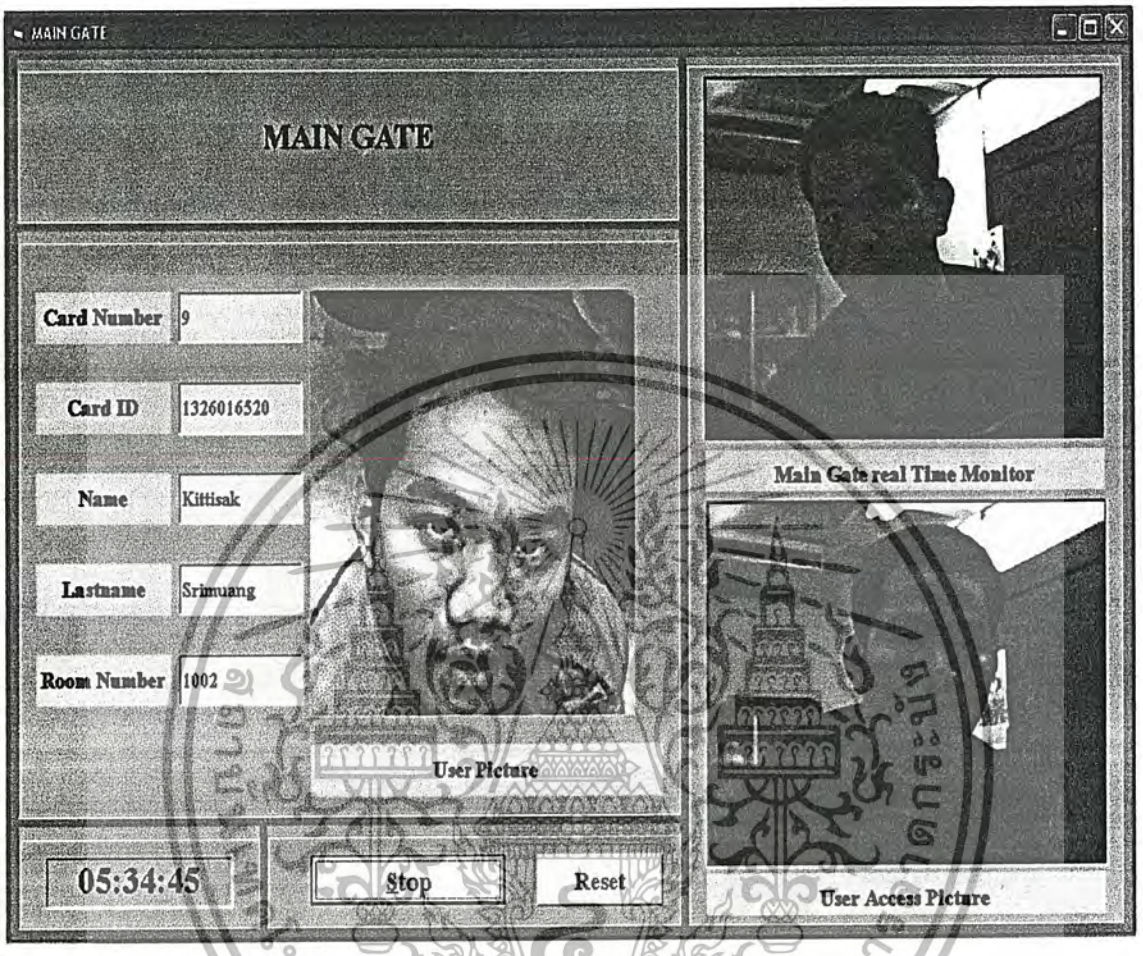
6.4.3.1 เปิดโปรแกรมในส่วน Main Gate

6.4.3.2 โปรแกรมจะขึ้นหน้าต่างที่แสดงภาพ Real Time ที่ประตูหลักออกมา

6.4.3.3 ทำการเสียบบัตรถ้าบัตรถูกต้อง โปรแกรมก็จะแสดง รูปภาพอ้างอิงที่เก็บไว้ พร้อมทั้งถ่ายภาพที่ประตู ขณะทำการเสียบบัตร พร้อมทั้งบันทึกเวลาไว้ด้วย



รูปที่ 6.18 รูปโปรแกรมส่วน ประตูหลัก



รูปที่ 6.19 รูปโปรแกรมส่วน ประตูหลักเมื่อทำการเสียบบัตร

6.4.3.4 ผลการทดลองการเข้าประตูหลัก

หมายเลขบัตร	หมายเลขห้อง	วันเวลา	ผู้ถือบัตร
CardID 1282173989	RoomNumber 3004	วันที่ 23 เวลา 15 นาฬิกา 51 นาที 22 วินาที	UserName Nattawut Boonpramook
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 15 นาฬิกา 52 นาที 49 วินาที	UserName Makutpong Krisnathevin
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 11 วินาที	UserName Kittisak Srimuang
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 28 วินาที	UserName Niruj Serivivatanawongse
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 40 วินาที	UserName Makutpong Krisnathevin
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 5 นาที 5 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 7 นาที 55 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 8 นาที 4 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 21 นาที 45 วินาที	UserName Niruj Serivivatanawongse
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 16 นาฬิกา 26 นาที 51 วินาที	UserName Kittisak Srimuang
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 16 นาฬิกา 26 นาที 56 วินาที	UserName Makutpong Krisnathevin
CardID 1282173989	RoomNumber 3004	วันที่ 23 เวลา 16 นาฬิกา 27 นาที 0 วินาที	UserName Nattawut Boonpramook
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 16 นาฬิกา 27 นาที 4 วินาที	UserName Niruj Serivivatanawongse
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 16 นาฬิกา 29 นาที 21 วินาที	UserName Kittisak Srimuang
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 16 นาฬิกา 29 นาที 34 วินาที	UserName Kittisak Srimuang

ตารางที่6.3 ผลการทดลองรายงานการเข้าออกประจำเดือน 3 ปี 2547

6.4.4 ผลการทดลองโปรแกรมห้องพัก

6.4.4.1 ทำการเลือกโปรแกรมในส่วน Main Gate

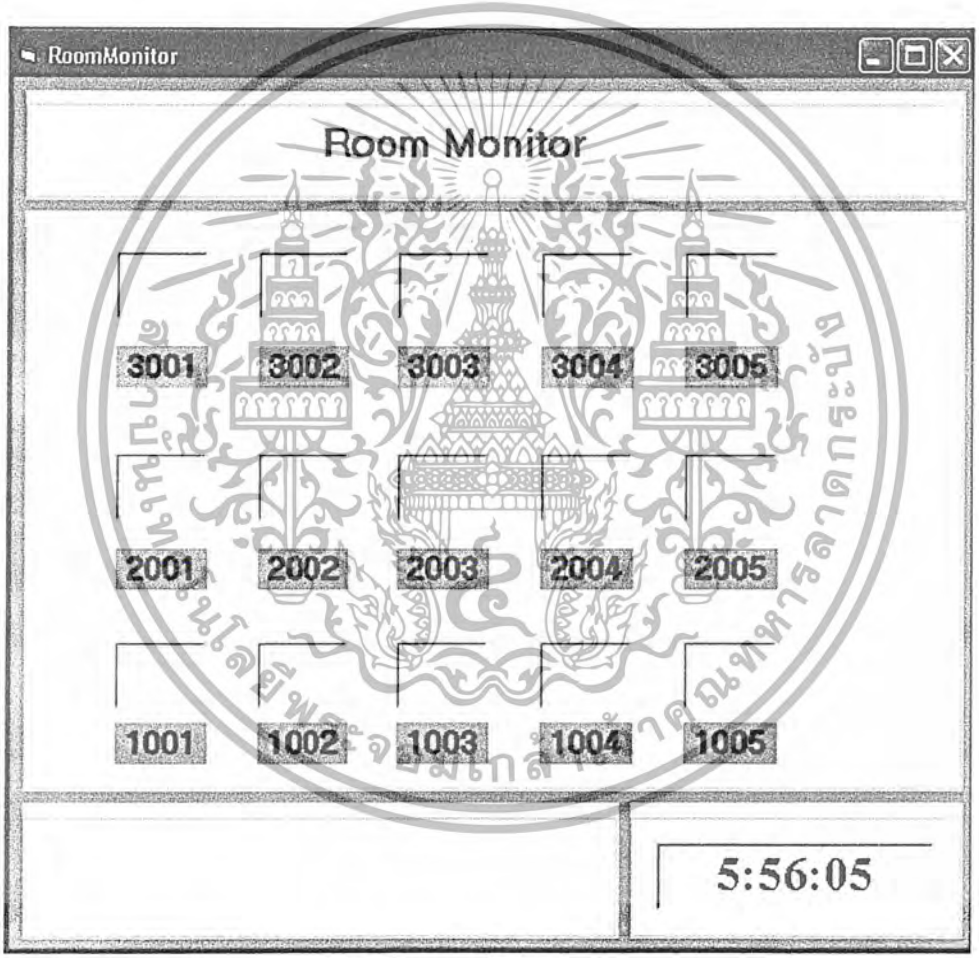
6.4.4.2 ทำการเสียบบัตรที่ประตูห้องพัก

6.4.4.3 ทำการกดรหัสประจำห้องพัก

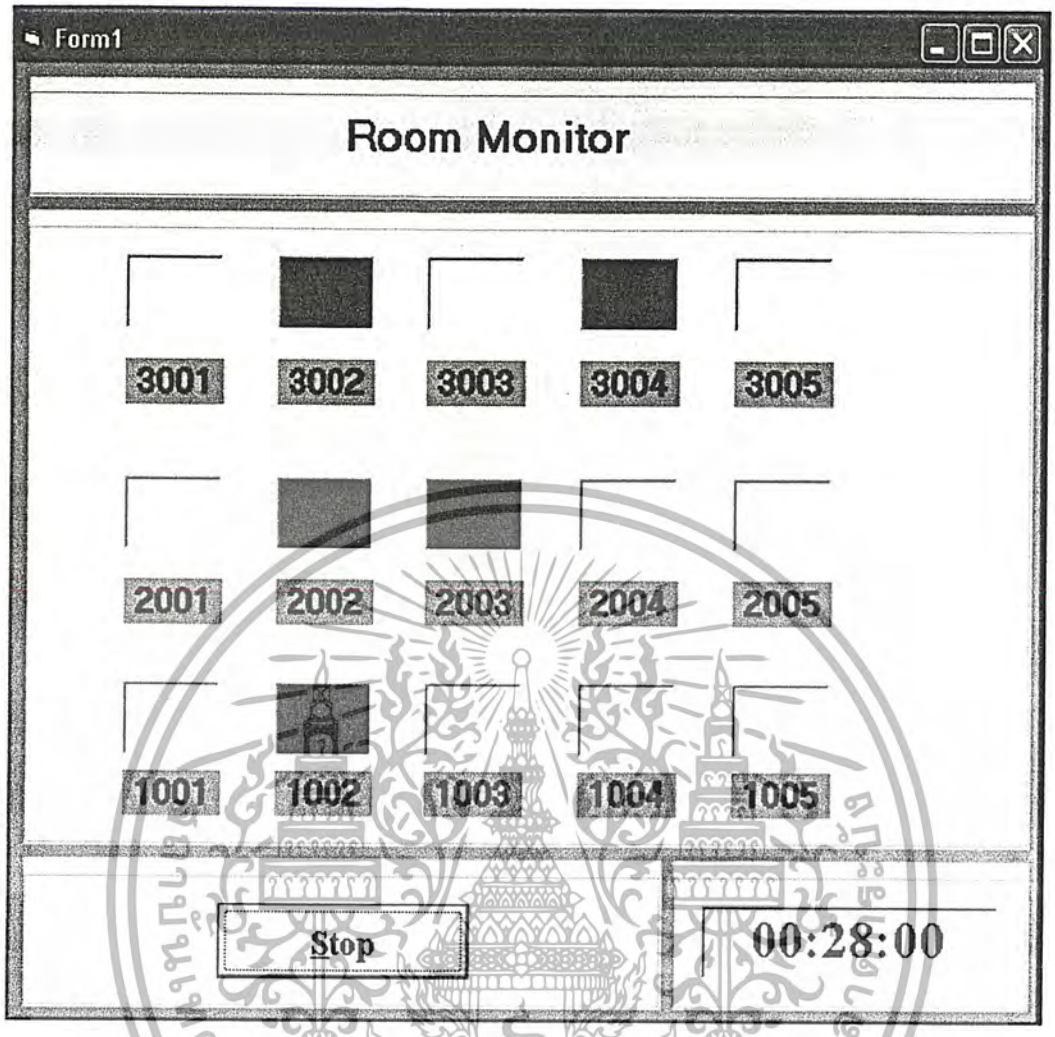
6.4.4.4 เมื่อทำการเข้าห้องพักแต่ละครั้งโปรแกรมในส่วนนี้จะทำการบันทึกเวลาการเข้าออกไว้

6.4.4.5 เลือกที่ Room Monitor จะพบว่าห้องที่มีคนเข้าจะแสดงสถานะเป็น สีเขียวเข้ม

6.4.4.6 เมื่อทำการเสียบบัตรที่ห้องเดิมอีกครั้งเมื่อเปิด Room Monitor อีกครั้งจะพบว่าสถานะของห้องห้องนั้นจะเปลี่ยนกับมาเป็นสีอ่อนอีกครั้ง



รูปที่ 6.20 รูปโปรแกรมส่วนห้องพักเมื่อยังไม่ได้ทำการเสียบบัตร



รูปที่ 6.21 รูปโปรแกรมส่วนห้องพักเมื่อทำการเสียบบัตร

6.4.4.7 ผลการทดลองการเข้าออกห้องพัก

หมายเลขบัตร	หมายเลขห้อง	วันเวลา	ผู้ถือบัตร
CardID 1279070253	RoomNumber 3002	วันที่ 23 เวลา 11 นาฬิกา 0 นาที 50 วินาที	UserName Songkhram Sundaranu
CardID 1282173989	RoomNumber 3004	วันที่ 23 เวลา 11 นาฬิกา 0 นาที 55 วินาที	UserName Nattawut Boonpramook
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 11 นาฬิกา 1 นาที 1 วินาที	UserName Makutpong Krisnathevin
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 11 นาฬิกา 1 นาที 21 วินาที	UserName Makutpong Krisnathevin
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 11 นาฬิกา 2 นาที 10 วินาที	UserName Kittisak Srimuang
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 2 นาที 15 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 2 นาที 23 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 2 นาที 41 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 3 นาที 50 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 12 นาที 6 วินาที	UserName Niruj Serivivatanawongse
CardID 1341073360	RoomNumber 2003	วันที่ 23 เวลา 11 นาฬิกา 18 นาที 30 วินาที	UserName Niruj Serivivatanawongse
CardID 1326016520	RoomNumber 1002	วันที่ 23 เวลา 11 นาฬิกา 18 นาที 37 วินาที	UserName Kittisak Srimuang
CardID 1234066493	RoomNumber 3001	วันที่ 23 เวลา 11 นาฬิกา 18 นาที 44 วินาที	UserName Makutpong Krisnathevin
CardID 1282173989	RoomNumber 3004	วันที่ 23 เวลา 11 นาฬิกา 18 นาที 49 วินาที	UserName Nattawut Boonpramook
CardID 1279070253	RoomNumber 3002	วันที่ 23 เวลา 11 นาฬิกา 18 นาที 54 วินาที	UserName Songkhram Sundaranu
CardID 1279070253	RoomNumber 3002	วันที่ 23 เวลา 13 นาฬิกา 6 นาที 16 วินาที	UserName Songkhram Sundaranu
CardID 1279070253	RoomNumber 3002	วันที่ 23 เวลา 13 นาฬิกา 8 นาที 7 วินาที	UserName Songkhram Sundaranu
CardID 1281124001	RoomNumber 2365	วันที่ 22 เวลา 16 นาฬิกา 53 นาที 11 วินาที	UserName Pichai Raungroj
CardID 1282173989	RoomNumber 1254	วันที่ 23 เวลา 9 นาฬิกา 53 นาที 45 วินาที	UserName Nattawut Boonpramook
CardID 1282173989	RoomNumber 1254	วันที่ 23 เวลา 9 นาฬิกา 54 นาที 3 วินาที	UserName Nattawut Boonpramook
CardID 1282173989	RoomNumber 1254	วันที่ 23 เวลา 9 นาฬิกา 54 นาที 12 วินาที	UserName Nattawut Boonpramook
CardID 1282173989	RoomNumber 1254	วันที่ 23 เวลา 9 นาฬิกา 54 นาที 30 วินาที	UserName Nattawut Boonpramook
CardID 1234066493	RoomNumber 2351	วันที่ 23 เวลา 9 นาฬิกา 54 นาที 37 วินาที	UserName Makutpong Krisnathevin
CardID 1282173989	RoomNumber 1254	วันที่ 23 เวลา 9 นาฬิกา 54 นาที 43 วินาที	UserName Nattawut Boonpramook

ตารางที่6.4 ผลการทดลองรายงานการเข้าออกห้องพักประจำเดือน 3 ปี 2547

6.4.5 ผลการทดลองโปรแกรมส่วนร้านอาหาร

หมายเลขบัตร	หมายเลขห้อง	วันเวลา	ชื่อผู้ถือบัตร	รายการอาหาร
1282173989	3004	วันที่ 23 เวลา 15 นาฬิกา 51 นาที 22 วินาที	UserName Nattawut Boonpramook	ข้าวมันไก่ ราคา 20 บาท
1234066493	3001	วันที่ 23 เวลา 15 นาฬิกา 52 นาที 49 วินาที	UserName Makutpong Krisnathevin	สุกี้ ราคา 30 บาท
1326016520	1002	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 11 วินาที	UserName Kittisak Srimuang	ออส่วน ราคา 35 บาท
1341073360	2003	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 28 วินาที	UserNameNiruj Serivivatanawongse	ข้าวมันไก่ ราคา 20 บาท
1234066493	3001	วันที่ 23 เวลา 16 นาฬิกา 2 นาที 40 วินาที	UserName Makutpong Krisnathevin	สุกี้ ราคา 30 บาท
1341073360	2003	วันที่ 23 เวลา 16 นาฬิกา 5 นาที 5 วินาที	UserName Niruj Serivivatanawongse	ข้าวมันไก่ ราคา 20 บาท
1341073360	2003	วันที่ 23 เวลา 16 นาฬิกา 7 นาที 55 วินาที	UserName Niruj Serivivatanawongse	ออส่วน ราคา 35 บาท
1282173989	3004	วันที่ 23 เวลา 16 นาฬิกา 27 นาที 0 วินาที	UserName Nattawut Boonpramook	สุกี้ ราคา 30 บาท
1341073360	2003	วันที่ 23 เวลา 16 นาฬิกา 27 นาที 4 วินาที	UserName Niruj Serivivatanawongse	ข้าวมันไก่ ราคา 20 บาท
1326016520	1002	วันที่ 23 เวลา 16 นาฬิกา 29 นาที 34 วินาที	UserName Kittisak Srimuang	สุกี้ ราคา 30 บาท

ตารางที่6.5 ผลการทดลองรายงานรายจ่ายประจำเดือน 3 ปี 2547

บทที่ 7 วิเคราะห์และสรุปผลการทดลอง

จากผลการทดลองในบทที่ 6 จะเห็นได้ว่า เปอร์เซ็นต์การผิดพลาดของข้อมูลที่เครื่องอ่านบัตรสมาร์ตการ์ดอ่านได้และส่งให้คอมพิวเตอร์นั้น มีความผิดพลาดน้อยมาก และสำหรับส่วนที่ผิดพลาดนี้อาจเนื่องมาจากการใส่บัตรอย่างไม่เหมาะสม เช่น ใส่บัตรไม่ลึกพอ หรือ การใส่ผิดด้าน เป็นต้น ซึ่งความผิดพลาดส่วนนี้สามารถแก้ไขได้โดยผู้ใช้ ในส่วนของบัตรหมายเลข 2 นั้น ซึ่งเป็นบัตรโทรศัพท์จาก China Telecom ซึ่งจะสังเกตว่ารหัสที่อ่านมาได้นั้นมีความแตกต่างจากบัตร TOT Card คือจะมีเลขฐานสิบหกรวมมาด้วย ซึ่งผลตรงนี้อาจเป็นเพราะบัตรสมาร์ตการ์ดใบนี้ เป็นบัตรคนละชนิดกับ TOT Card (SLE 4436)

สำหรับการทดลองในเรื่องของความผิดพลาดที่อาจเกิดจากระยะความยาวของสายส่งนั้น จากผลการทดลองทั้ง 3 ระยะคือ 10 , 20 และ 30 ฟุตนั้น จะเห็นได้ว่าไม่มีปัญหาในการส่งข้อมูล แต่ส่วนที่เกิดความผิดพลาดขึ้นมาเล็กน้อยนั้น มาจากความผิดพลาดของการใส่บัตรไม่เหมาะสมเช่นกัน

ที่น่าสังเกตอีกประเด็นหนึ่งคือ สัญญาณที่ได้รับจากขา I/O ของการ์ดที่เป็นบัตร TOT Card โดยจะมีข้อมูลที่เหมือนกันอยู่คือ ข้อมูลในช่วงไบต์ที่ 0-2 จะเห็นว่ามีรูปสัญญาณเหมือนกัน ซึ่งเนื่องมาจากข้อมูลส่วนนี้เป็นข้อมูล Customer Code ซึ่งได้กล่าวมาแล้วตามเนื้อหาในบทที่ 2

ภาคผนวก

โปรแกรม Microsoft Visual Basic Version 6.0 Enterprise

โปรแกรมส่วนการทำงานหลักของ โครงงาน

frmVBCom

```
Option Explicit
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim ChkStopLoopUnLess As Boolean
Dim EventMsg As String, ErrorMsg As String
Dim lStatus As Boolean
Dim lQuit As Boolean
Private Sub Command1_Click()  'Reset ค่าตัวแปรทุกตัวในคปรแกรม
    Text7.Text = ""
    Text3.Text = ""
    Text4.Text = ""
    Text6.Text = ""
    Text5.Text = ""
    Text1.Text = ""
    Text2.Text = ""
    Text8.Text = ""
    Text9.Text = ""
    Text10.Text = ""
    Text11.Text = ""
    Text12.Text = ""
    Text13.Text = ""
    Text14.Text = ""
    Text15.Text = ""
    Text16.Text = ""
```

```
    Text17.Text = ""
    Text18.Text = ""
    Picture1.Picture = LoadPicture()
    Picture2.Picture = LoadPicture()
End Sub

Private Sub Form_Load()
Dim CommPort As String, Handshaking As String, Settings As String
On Error Resume Next
  lStatus = False
  lQuit = False
  Text1.Text = ""
  App.Title = "Visual Basic : Serial Port"

  ComOnOff = False
  optModeRXTX.Item(0).Value = True
  SecRXTX = 0


  txtRXTX.SelLength = Len(txtRXTX)
  txtRXTX.SelText = ""
  txtRXTX.ForeColor = vbRed


  Settings = GetSetting(App.Title, "Properties", "Settings", "")
  If Settings <> "" Then
    MSComm1.Settings = Settings
    If Err Then
      MsgBox Error$, 48
      Exit Sub
    End If
```

```
    End If
    CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
    If CommPort <> "" Then MSComm1.CommPort = CommPort
      Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
    If Handshaking <> "" Then
      MSComm1.Handshaking = Handshaking
      If Err Then
        MsgBox Error$, 48
        Exit Sub
      End If
    End If
    Echo = GetSetting(App.Title, "Properties", "Echo", "")
    ChkStopLoopUnLess = False
On Error GoTo 0

End Sub

Private Sub UnLessLoop()
Dim Buffer
Dim Check As Boolean

MEVENT_DRIVEN:
  If lQuit Then
    Exit Sub
  End If
  Check = True
  Do
    If ChkStopLoopUnLess Then Exit Sub
    DoEvents
    If SecRXTX = 1 Then Check = False
  Loop Until Check = False
```

```
MPOLLING:
   Check = True
   Do
      If ChkStopLoopUnLess Then Exit Sub
      DoEvents
      If MSComm1.PortOpen = True Then
         Buffer = MSComm1.Input
      End If
      ShowData txtRXTX, (StrConv((Buffer), vbUnicode))
      If SecRXTX = 0 Then Check = False
   Loop Until Check = False

   If SecRXTX = 0 Then GoTo MEVENT_DRIVEN    'mode  Event-Driven
   If SecRXTX = 1 Then GoTo MPOLLING         'mode Polling

End Sub

Private Sub cmdOnOffPort_Click()
On Error GoTo ErrH

      If Not ComOnOff Then
         MSComm1.PortOpen = True
         ChkStopLoopUnLess = False
         cmdOnOffPort.Caption = "&Stop"
         ComOnOff = True
         lblOncom.Caption = "OnCom:" & MSComm1.Settings
         Call ShowMsgProgram(ModeRXTX)

      Else
         MSComm1.PortOpen = False
         ChkStopLoopUnLess = True
```

```
        cmdOnOffPort.Caption = "&Start"
        ComOnOff = False
        lblStatus.Caption = "Status Com : "
        lblOncom.Caption = "OffCom"

      End If

      'Set Interrupt or Polling
      Call optModeRXTX_Click(ModeRXTX)
      Call UnLessLoop

      Exit Sub
ErrH:
   MsgBox Err.Description

End Sub


Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   If ComOnOff Then
     cmdOnOffPort_Click
   End If
End Sub


Private Sub MSComm1_OnComm()
On Error GoTo ErrH
   Select Case MSComm1.CommEvent
      ' Event messages.
      Case comEvReceive
         Dim Buffer As Variant
```

```
    Buffer = MSComm1.Input
    ShowData Text1, (StrConv((Buffer), vbUnicode))
    EventMsg$ = " Receive "


        Room_Monitor

        Restaurant

        Main_Gate


Case comEvSend
    EventMsg$ = " Send "
Case comEvCTS
    EventMsg$ = "Change in CTS Detected"
Case comEvDSR
    EventMsg$ = "Change in DSR Detected"
Case comEvCD
    EventMsg$ = "Change in CD Detected"
Case comEvRing
    EventMsg$ = "The Phone is Ringing"
Case comEvEOF
    EventMsg$ = "End of File Detected"

' Error messages.
Case comBreak
    ErrorMsg$ = "Break Received"
Case comCDTO
    ErrorMsg$ = "Carrier Detect Timeout"
Case comCTSTO
```

```
        ErrorMsg$ = "CTS Timeout"
    Case comDCB
        ErrorMsg$ = "Error retrieving DCB"
    Case comDSRTO
        ErrorMsg$ = "DSR Timeout"
    Case comFrame
        ErrorMsg$ = "Framing Error"
    Case comOverrun
        ErrorMsg$ = "Overrun Error"
    Case comRxOver
        ErrorMsg$ = "Receive Buffer Overflow"
    Case comRxParity
        ErrorMsg$ = "Parity Error"
    Case comTxFull
        ErrorMsg$ = "Transmit Buffer Full"
    Case Else
        ErrorMsg$ = "Unknown error or event"
End Select

If Len(EventMsg$) Then

    lblStatus.Caption = "Status Com :" & EventMsg$



    Timer1.Enabled = True

ElseIf Len(ErrorMsg$) Then

    lblStatus.Caption = "Status Com : " & ErrorMsg$
```

```
        Beep

          MSComm1.PortOpen = False
        End If
        Timer1.Enabled = True

    Exit Sub
ErrH:
    MsgBox Err.Description
End Sub


Private Sub optModeRXTX_Click(Index As Integer)
    Select Case Index
    Case 0
        SecRXTX = 0
        If MSComm1.PortOpen = True Then MSComm1.RThreshold = 1
        ShowData txtRXTX, "Mode Event-driven" & Chr(13)    'Show msg
    Case 1
        SecRXTX = 1
        If MSComm1.PortOpen = True Then MSComm1.RThreshold = 0
        ShowData txtRXTX, "Mode Polling" & Chr(13)    'Show msg
    End Select

End Sub

Private Sub Timer1_Timer()

    lblTime.Caption = Format(Time(), "HH:MM:SS")

    If EventMsg$ = " Receive " Then
```

```
    lblStatus.Caption = "Status Com : "
  End If

End Sub
Private Sub txtRXTX_KeyPress(KeyAscii As Integer)
  '
  If MSComm1.PortOpen Then

    MSComm1.Output = Chr$(KeyAscii)

    If Not Echo Then

      txtRXTX.SelStart = Len(txtRXTX)
      KeyAscii = 0
    End If
  End If

End Sub


Public Static Sub ShowData(Term As Control, Data As String)
  On Error GoTo Handler
  Const MAXTERMSIZE = 16000
  Dim TermSize As Long, i
  TermSize = Len(Term.Text)
  If TermSize > MAXTERMSIZE Then
    Term.Text = Mid$(Term.Text, 4097)
    TermSize = Len(Term.Text)
  End If

  Term.SelStart = TermSize
```

```
Do
  i = InStr(Data, Chr$(8))
  If i Then
    If i = 1 Then
      Term.SelStart = TermSize - 1
      Term.SelLength = 1
      Data = Mid$(Data, i + 1)
    Else
      Data = Left$(Data, i - 2) & Mid$(Data, i + 1)
    End If
  End If
Loop While i

Do
  i = InStr(Data, Chr$(10))
  If i Then
    Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
  End If
Loop While i
If Term.Name <> Text1.Name Then
  i = 1
  Do
    i = InStr(i, Data, Chr$(13))
    If i Then
      Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
      i = i + 1
    End If
  Loop While i
End If
If Term.Name = Text1.Name Then
```

```
    If Not IsNumeric(Data) Then
      Exit Sub
    End If
  End If
  Term.SelText = Data

  Term.SelStart = Len(Term.Text)

Exit Sub

Handler:
  MsgBox Error$
  Resume Next
End Sub

Private Sub ShowMsgProgram(ByVal MODE As Integer)
Dim strShow As String

  If MODE = 0 Then
      strShow = "Visual Basic: Communication custom control ( MSComm ) " & Chr(13) & _
                "MODE : Event-driven communications" & Chr(13) & _
                "Com Port : " & MSComm1.CommPort & Chr(13) & _
                "Setting : " & MSComm1.Settings & Chr(13)
  Else
      strShow = "Visual Basic: Communication custom control ( MSComm ) " & Chr(13) & _
                "MODE : Polling communications" & Chr(13) & _
                "Com Port : " & MSComm1.CommPort & Chr(13) & _
                "Setting : " & MSComm1.Settings & Chr(13)
  End If
  txtRXTX.Text = ""                'Clear Text
  ShowData txtRXTX, strShow     'Show msg
```

```
End Sub

Private Sub DoQuery()

  Dim cn As New ADODB.Connection
  Dim rs As New ADODB.Recordset
  cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " & _
            "Data Source = Project.mdb"
  cn.Open

   rs.Open "Select * From staff Where SID = '" & Text2.Text & "'", cn

   If Not rs.EOF Then
     SID = rs("SID")
     RoomNumber = rs("RoomNumber")
     CardNumber = rs("CardNumber")
     Text15.Text = rs("Name")
     Text16.Text = rs("Lastname")
   End If
End Sub


Private Sub Data_Show()

  Dim ii As Integer
  Dim Name As Variant
  Dim iFileNum As Integer
    Text7.Text = SID
    Text3.Text = Text15.Text
    Text4.Text = Text16.Text
```

```
    Text6.Text = RoomNumber
    Text5.Text = CardNumber
    Picture1.Picture = LoadPicture(CardNumber + ".jpg")
        ii = Year(Date) + 543
      Text17.Text = "วันที่" & Day(Date) & " เวลา " & Hour(Time) & " นาฬิกา " & Minute(Time)
& " นาที " & Second(Time) & " วินาที "
        Text18.Text = Month(Date) & " ปี พ.ศ. " & ii

    Name = "รายงานการเข้าออกประจำเดือน" + Text18.Text + ".txt"
    iFileNum = FreeFile
    Open Name For Append As #iFileNum
    Print #iFileNum, "CardID " & (Text7.Text) & "      " & "RoomNumber " & (Text6.Text) & "
" & " " & (Text17.Text) & "      " & "UserName " & (Text3.Text) & "  " & (Text4.Text)
    Close #iFileNum

    Automatic


End Sub
Private Sub Automatic()
Dim filename As Variant
  filename = App.Path + "\" + Text17.Text + ".bmp"
    On Error Resume Next
  Call ezVidCap1.SaveDIB(filename)
    If Err Then
      MsgBox Err.Description, vbInformation, App.Title
    End If
    Picture2.AutoSize = True
    Picture2.Picture = LoadPicture(filename, vbLPLarge, vbLPColor)
```

```
    End Sub
Private Sub Main_Gate()
        If Len(Text1.Text) >= 2 And Text1.Text = "93" Then
                Text9.Text = Text1.Text
                Text1.Text = ""
                flage = 1
          End If
                If Len(Text1.Text) = 10 And flage = 1 Then
                    Text2.Text = Text1.Text
                    Text1.Text = ""
                     DoQuery
                     Data_Show
                     cmdOnOffPort_Click
                     cmdOnOffPort_Click
                     flage = 0
                   End If
End Sub
Private Sub Restaurant()
    Dim ii As Integer
    Dim Name As Variant
    Dim iFileNum As Integer
    Dim AA As Integer
    Dim BB As Integer
    Dim CC As Integer
    Name = "รายงานค่าใช้จ่ายประจำเดือน " + Text8.Text + ".txt"
    iFileNum = FreeFile

              If Len(Text1.Text) >= 2 And Text1.Text = "91" Then
                      Text1.Text = ""
                        flage = 2
              End If
```

```
If Len(Text1.Text) = 10 And flage = 2 Then
        Text2.Text = Text1.Text
        Text1.Text = ""
        DoQuery
        ii = Year(Date) + 543
        Text9.Text = "วันที่ " & Day(Date) & " เวลา " & Hour(Time) & " นาฬิกา " &
Minute(Time) & " นาที " & Second(Time) & " วินาที "
        Text8.Text = "เดือน " & Month(Date) & " ปี พ.ศ. " & ii
    End If
    If Len(Text1.Text) = 3 And flage = 2 Then
        If Text1.Text = "555" Or Text1.Text = "55" Then
        Open Name For Append As #iFileNum
        Print #iFileNum, "CardID " & (SID) & "      " & "RoomNumber " &
(RoomNumber) & "    " & " " & (Text9.Text) & "      " & "UserName " & (Text15.Text) & "  " &
(Text16.Text) & "  ข้าวมันไก่  ราคา 20 บาท"
        Close #iFileNum
        Text1.Text = ""
        AA = AA + "30"
    ElseIf Text1.Text = "666" Then
        Open Name For Append As #iFileNum
        Print #iFileNum, "CardID " & (SID) & "      " & "RoomNumber " &
(RoomNumber) & "    " & " " & (Text9.Text) & "      " & "UserName " & (Text15.Text) & "  " &
(Text16.Text) & "  สุกี้    ราคา 30 บาท"
        Close #iFileNum
        Text1.Text = ""
        AA = AA + "30"
    ElseIf Text1.Text = "777" Then
        Open Name For Append As #iFileNum
        Print #iFileNum, "CardID " & (SID) & "      " & "RoomNumber " &
(RoomNumber) & "    " & " " & (Text9.Text) & "      " & "UserName " & (Text15.Text) & "  " &
```

```
(Text16.Text) & "  ออส่วน   ราคา 35 บาท "

                    Close #iFileNum
                    Text1.Text = ""
                    AA = AA + "30"
                End If
            End If

End Sub

Private Sub Room_Monitor()
  Dim ii As Integer
   Dim Name As Variant
   Dim iFileNum As Integer

                If Len(Text1.Text) >= 2 And Text1.Text = "92" Then
                    Text9.Text = Text1.Text
                    Text1.Text = ""
                    flage = 3
                End If
                If Len(Text1.Text) = 10 And flage = 3 Then
                    Text2.Text = Text1.Text
                    Text1.Text = ""
                    flage = 0
                    DoQuery
                    ii = Year(Date) + 543
                    Text9.Text = "วันที่ " & Day(Date) & " เวลา " & Hour(Time) & " ¹
นาฬิกา " & Minute(Time) & " นาที " & Second(Time) & " วินาที "
                    Text8.Text = Month(Date) & " ปี พ.ศ.  " & ii
                    Name = "รายงานการเข้าออกประจำเดือน '" + Text8.Text + ".txt"
                    iFileNum = FreeFile
```

```
                    Open Name For Append As #iFileNum
                    Print #iFileNum, "CardID " & (SID) & "      " & "RoomNumber "
& (RoomNumber) & "    " & " " & (Text9.Text) & "      " & "UserName " & (Text15.Text) & "  " &
(Text16.Text)
                    Close #iFileNum
                    Room_Monitor_Transfer
                   cmdOnOffPort_Click
                   cmdOnOffPort_Click
               End If

End Sub


Private Sub Room_Monitor_Transfer()

   If RoomNumber = 3001 Then
     If Room3001 = 0 Then
         Room3001 = 3001
           Else
         Room3001 = 0
     End If
   End If
   If RoomNumber = 3002 Then
     If Room3002 = 0 Then
          Room3002 = 3002
            Else
          Room3002 = 0
        End If
     End If
  If RoomNumber = 3003 Then
     If Room3003 = 0 Then
```

```
            Room3003 = 3003
                Else
               Room3003 = 0
            End If
        End If
    If RoomNumber = 3004 Then
        If Room3004 = 0 Then
            Room3004 = 3004
                Else
               Room3004 = 0
            End If
        End If
    If RoomNumber = 3005 Then
        If Room3005 = 0 Then
            Room3005 = 3005
                Else
               Room3005 = 0
            End If
        End If
    If RoomNumber = 2001 Then
        If Room2001 = 0 Then
            Room2001 = 2001
                Else
               Room2001 = 0
            End If
        End If
    If RoomNumber = 2002 Then
        If Room2002 = 0 Then
            Room2002 = 2002
                Else
               Room2002 = 0
```

```
        End If
    End If
If RoomNumber = 2003 Then
    If Room2003 = 0 Then
        Room2003 = 2003
            Else
        Room2003 = 0
        End If
    End If
If RoomNumber = 2004 Then
    If Room2004 = 0 Then
        Room2004 = 2004
            Else
        Room2004 = 0
        End If
    End If
If RoomNumber = 2005 Then
    If Room2005 = 0 Then
        Room2005 = 3005
            Else
        Room2005 = 0
        End If
    End If
If RoomNumber = 1001 Then
    If Room1001 = 0 Then
        Room1001 = 1001
            Else
        Room1001 = 0
    End If
End If
If RoomNumber = 1002 Then
```

```
    If Room1002 = 0 Then
        Room1002 = 1002
          Else
        Room1002 = 0
      End If
  End If
If RoomNumber = 1003 Then
    If Room1003 = 0 Then
        Room1003 = 1003
          Else
        Room1003 = 0
      End If
  End If
If RoomNumber = 1004 Then
    If Room1004 = 0 Then
        Room1004 = 1004
          Else
        Room1004 = 0
      End If
  End If
If RoomNumber = 1005 Then
    If Room1005 = 0 Then
        Room1005 = 1005
          Else
        Room1005 = 0
      End If
  End If
End Sub
```

```
frmRoom_Mornitor
Option Explicit
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim ChkStopLoopUnLess As Boolean
Dim EventMsg As String, ErrorMsg As String
Dim lStatus As Boolean
Dim lQuit As Boolean

Private Sub Form_Load()

 If Room3001 = 3001 Then
  Picture3001.BackColor = &H404000
 End If

 If Room3002 = 3002 Then
  Picture3002.BackColor = &H404000
 End If

 If Room3003 = 3003 Then
  Picture3003.BackColor = &H404000
 End If
 If Room3004 = 3004 Then
  Picture3004.BackColor = &H404000
 End If
 If Room3005 = 3005 Then
  Picture3005.BackColor = &H404000
```

```
End If
 If Room2001 = 2001 Then
  Picture2001.BackColor = &H404000
 End If
If Room2002 = 2002 Then
  Picture2002.BackColor = &H404000
End If
If Room2003 = 2003 Then
  Picture2003.BackColor = &H404000
End If
If Room2004 = 2004 Then
  Picture2004.BackColor = &H404000
End If
If Room2005 = 2005 Then
  Picture2005.BackColor = &H404000
End If
 If Room1001 = 1001 Then
  Picture1001.BackColor = &H404000
 End If
If Room1002 = 1002 Then
  Picture1002.BackColor = &H404000
End If
If Room1003 = 1003 Then
  Picture1003.BackColor = &H404000
End If
If Room1004 = 1004 Then
  Picture1004.BackColor = &H404000
End If
If Room1005 = 1005 Then
  Picture1005.BackColor = &H404000
End If
```

```
End Sub


Private Sub Timer1_Timer()
lblTime = Time
End Sub

frmMainGateVedio
Option Explicit
 Private Sub cmdSaveDIB_Click()
   Dim iFileNum As Integer
   Dim filename As Variant
   Dim Name As Variant
   Dim ii As Integer

 text9.Text = "วันที่ " & Day(Date) & "เดือน " & Month(Date) & " ปี " & Year(Date) & " เวลา " &
Hour(Time) & " นาฬิกา " & Minute(Time) & " นาที " & Second(Time) & " วินาที "
   filename = App.Path + "\" + text9.Text + ".bmp"
     On Error Resume Next
   Call ezVidCap1.SaveDIB(filename)
     If Err Then
       MsgBox Err.Description, vbInformation, App.Title
     End If
       ii = Year(Date) + 543
         Text10.Text = "วันที่ " & Day(Date) & " เวลา " & Hour(Time) & " นาฬิกา " & Minute(Time)
& " นาที " & Second(Time) & " วินาที "
         Text8.Text = Month(Date) & " ปี พ.ศ.  " & ii


   End Sub
```

```
Private Sub Timer2_Timer()
 lblTime.Caption = Time()
End Sub


frmRegister
Option Explicit
Dim SecRXTX As Integer
Dim ModeRXTX As Integer
Dim ComOnOff As Boolean
Dim StatusComm As String
Dim ChkStopLoopUnLess As Boolean
Dim EventMsg As String, ErrorMsg As String
Dim lStatus As Boolean
Dim lQuit As Boolean

Private Sub Command1_Click()
TakePicture.Show
End Sub

Private Sub Command2_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text5.Text = ""

End Sub

Private Sub Form_Load()
Dim CommPort As String, Handshaking As String, Settings As String
```

```
On Error Resume Next
  lStatus = False
  lQuit = False
  Text1.Text = ""
  App.Title = "Visual Basic : Serial Port

  ComOnOff = False
  optModeRXTX.Item(0).Value =
  SecRXTX = 0


  txtRXTX.SelLength = Len(txtRXTX)
  txtRXTX.SelText = ""
  txtRXTX.ForeColor = vbRed


  Settings = GetSetting(App.Title, "Properties", "Settings", "")
  If Settings <> "" Then
    MSComm1.Settings = Settings
    If Err Then
      MsgBox Error$, 48
      Exit Sub
    End If
  End If
  CommPort = GetSetting(App.Title, "Properties", "CommPort", "")
  If CommPort <> "" Then MSComm1.CommPort = CommPort
    Handshaking = GetSetting(App.Title, "Properties", "Handshaking", "")
  If Handshaking <> "" Then
    MSComm1.Handshaking = Handshaking
    If Err Then
```

```
        MsgBox Error$, 48
        Exit Sub
      End If
    End If
    Echo = GetSetting(App.Title, "Properties", "Echo", "")
    ChkStopLoopUnLess = False

On Error GoTo 0 '

End Sub

Private Sub UnLessLoop()
Dim Buffer
Dim Check As Boolean

MEVENT_DRIVEN:
  If lQuit Then
    Exit Sub
  End If
  Check = True
  Do
    If ChkStopLoopUnLess Then Exit Sub
    DoEvents
    If SecRXTX = 1 Then Check = False
  Loop Until Check = False
MPOLLING:
  Check = True
  Do
    If ChkStopLoopUnLess Then Exit Sub
    DoEvents
    If MSComm1.PortOpen = True Then
```

```
        Buffer = MSComm1.Input
      End If
      ShowData txtRXTX, (StrConv((Buffer), vbUnicode))
      If SecRXTX = 0 Then Check = False
    Loop Until Check = False

    If SecRXTX = 0 Then GoTo MEVENT_DRIVEN    'mode  Event-Driven
    If SecRXTX = 1 Then GoTo MPOLLING         'mode Polling

End Sub

Private Sub cmdOnOffPort_Click()
On Error GoTo ErrH

      If Not ComOnOff Then
        MSComm1.PortOpen = True
        ChkStopLoopUnLess = False
        cmdOnOffPort.Caption = "&Stop"
        ComOnOff = True
        lblOncom.Caption = "OnCom:" & MSComm1.Settings
        Call ShowMsgProgram(ModeRXTX)
      Else
        MSComm1.PortOpen = False
        ChkStopLoopUnLess = True
        cmdOnOffPort.Caption = "&Start"
        ComOnOff = False
        lblStatus.Caption = "Status Com : "
        lblOncom.Caption = "OffCom"
      End If

      'Set Interrupt or Polling
```

```
    Call optModeRXTX_Click(ModeRXTX)
  Call UnLessLoop

    Exit Sub
ErrH:
  MsgBox Err.Description

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
  If ComOnOff Then
    cmdOnOffPort_Click
  End If
End Sub

Private Sub MSComm1_OnComm()
On Error GoTo ErrH
  Select Case MSComm1.CommEvent
    ' Event messages.
    Case comEvReceive
      Dim Buffer As Variant
      Buffer = MSComm1.Input
      ShowData Text1, (StrConv((Buffer), vbUnicode))
      EventMsg$ = " Receive "

      DoQuery
```

```
        cmdOnOffPort_Click
        cmdOnOffPort_Click
'    End If
Case comEvSend
    EventMsg$ = " Send "
Case comEvCTS
    EventMsg$ = "Change in CTS Detected"
Case comEvDSR
    EventMsg$ = "Change in DSR Detected"
Case comEvCD
    EventMsg$ = "Change in CD Detected"
Case comEvRing
    EventMsg$ = "The Phone is Ringing"
Case comEvEOF
    EventMsg$ = "End of File Detected"

' Error messages.
Case comBreak
    ErrorMsg$ = "Break Received"
Case comCDTO
    ErrorMsg$ = "Carrier Detect Timeout"
Case comCTSTO
    ErrorMsg$ = "CTS Timeout"
Case comDCB
    ErrorMsg$ = "Error retrieving DCB"
Case comDSRTO
    ErrorMsg$ = "DSR Timeout"
Case comFrame
    ErrorMsg$ = "Framing Error"
```

```
    Case comOverrun
        ErrorMsg$ = "Overrun Error"
    Case comRxOver
        ErrorMsg$ = "Receive Buffer Overflow"
    Case comRxParity
        ErrorMsg$ = "Parity Error"
    Case comTxFull
        ErrorMsg$ = "Transmit Buffer Full"
    Case Else
        ErrorMsg$ = "Unknown error or event"
End Select

If Len(EventMsg$) Then
    lblStatus.Caption = "Status Com : " & EventMsg$
    Timer1.Enabled = True
ElseIf Len(ErrorMsg$) Then
    lblStatus.Caption = "Status Com : " & ErrorMsg$
    Beep
'   Ret = MsgBox(ErrorMsg$, 1, "Click Cancel to quit, OK to ignore.")
' If Ret = 2 Then
      MSComm1.PortOpen = False
    End If
```

```
    Timer1.Enabled = True
'   End If

    Exit Sub
ErrH:
    MsgBox Err.Description
End Sub


Private Sub optModeRXTX_Click(Index As Integer)
    Select Case Index
    Case 0
        SecRXTX = 0
        If MSComm1.PortOpen = True Then MSComm1.RThreshold = 1
        ShowData txtRXTX, "Mode Event-driven" & Chr(13)     'Show msg
    Case 1
        SecRXTX = 1
        If MSComm1.PortOpen = True Then MSComm1.RThreshold = 0
        ShowData txtRXTX, "Mode Polling" & Chr(13)      'Show msg
    End Select

End Sub

Private Sub Timer1_Timer()

    lblTime.Caption = Format(Time(), "HH:MM:SS")

    If EventMsg$ = " Receive " Then
        lblStatus.Caption = "Status Com :"
    End If
```

```
End Sub
Private Sub txtRXTX_KeyPress(KeyAscii As Integer)
  '
  If MSComm1.PortOpen Then

    MSComm1.Output = Chr$(KeyAscii)

    If Not Echo Then

      txtRXTX.SelStart = Len(txtRXTX)
      KeyAscii = 0
    End If
  End If

End Sub


Public Static Sub ShowData(Term As Control, Data As String)
  On Error GoTo Handler
  Const MAXTERMSIZE = 16000
  Dim TermSize As Long, i
  TermSize = Len(Term.Text)
  If TermSize > MAXTERMSIZE Then
    Term.Text = Mid$(Term.Text, 4097)
    TermSize = Len(Term.Text)
  End If

  Term.SelStart = TermSize

  Do
    i = InStr(Data, Chr$(8))
```

```
    If i Then
      If i = 1 Then
        Term.SelStart = TermSize - 1
        Term.SelLength = 1
        Data = Mid$(Data, i + 1)
      Else
        Data = Left$(Data, i - 2) & Mid$(Data, i + 1)
      End If
    End If
  Loop While i

  Do
    i = InStr(Data, Chr$(10))
    If i Then
      Data = Left$(Data, i - 1) & Mid$(Data, i + 1)
    End If
  Loop While i
  If Term.Name <> Text1.Name Then
    i = 1
    Do
      i = InStr(i, Data, Chr$(13))
      If i Then
        Data = Left$(Data, i) & Chr$(10) & Mid$(Data, i + 1)
        i = i + 1
      End If
    Loop While i
  End If
  If Term.Name = Text1.Name Then
    If Not IsNumeric(Data) Then
      Exit Sub
    End If
```

```
    End If
    Term.SelText = Data

    Term.SelStart = Len(Term.Text)

Exit Sub

Handler:
    MsgBox Error$
    Resume Next
End Sub

Private Sub ShowMsgProgram(ByVal MODE As Integer)
Dim strShow As String

    If MODE = 0 Then
        strShow = "Visual Basic: Communication custom control ( MSComm ) " & Chr(13) & _
                  "MODE : Event-driven communications" & Chr(13) & _
                  "Com Port : " & MSComm1.CommPort & Chr(13) & _
                  "Setting : " & MSComm1.Settings & Chr(13)
    Else
        strShow = "Visual Basic: Communication custom control ( MSComm ) " & Chr(13) & _
                  "MODE : Polling communications" & Chr(13) & _
                  "Com Port : " & MSComm1.CommPort & Chr(13) & _
                  "Setting : " & MSComm1.Settings & Chr(13)
    End If
    txtRXTX.Text = ""                   'Clear Text
    ShowData txtRXTX, strShow     'Show msg

End Sub
```

```
Private Sub DoQuery()

  Dim iFileNum As Integer
  Dim Name As Variant
  Dim ii As Integer
  Dim cn As New ADODB.Connection
  Dim rs As New ADODB.Recordset
  cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; " & _
          "Data Source = Project.mdb"
  cn.Open
   rs.Open "Select * From staff Where SID = '" & Text2.Text & "'", cn

     If Not rs.EOF Then
        Text5.Text = rs("SID")

        Text3.Text = " Already has in Data Base "

     Else

        Text5.Text = Text2.Text
        Text3.Text = "New Card"

  End If

  End Sub
```

```
frmSmart_Card_in_security_and_facillitate_system
Option Explicit

Private Sub Command1_Click()
   frmNewVBCom.Show
End Sub

Private Sub Command3_Click()
frmNewRoomMonitor.Show
End Sub

Private Sub Command4_Click()
Register.Show
End Sub

Private Sub Command5_Click()
frmMainGateVedio.Show
End Sub

Private Sub Timer1_Timer()
lblTime = (Time)
End Sub
```

```
Frm_Take_Picture
Option Explicit
   Private Sub Form_Load()
Me.Show 'show form
Me.Refresh
ezVidCap1.Preview = True
ezVidCap1.AutoSize = True
ezVidCap1.CenterVideo = True
End Sub
Private Sub Label1_Click()
Dim filename As Variant
 text9.Text = "วันที่  " & Day(Date) & " เดือน" & Month(Date) & "ปี " & Year(Date) & " เวลา " &
Hour(Time) & " นาฬิกา " & Minute(Time) & " นาที " & Second(Time) & " วินาที "
   filename = App.Path + "\" + text9.Text + ".bmp"
      On Error Resume Next
    Call ezVidCap1.SaveDIB(filename)
      If Err Then
         MsgBox Err.Description, vbInformation, App.Title
      End If

   '  Picture1.AutoSize = True
      Picture1.Picture = LoadPicture(filename, vbLPLarge, vbLPColor)
   End Sub
'End Sub

Private Sub Timer2_Timer()
 lblTime.Caption = Time()
End Sub
```

โปรแกรม mul_tx31.asm สำหรับตัวจัดการรับข้อมูลจากเครื่องอ่านแต่ละเครื่อง

```
        org     00h
        addr    equ     70h
        mem     equ     71h
        setb    p3.7
        mov     addr,#41h
        clr     p1.7
        mov     pcon,#00h
        mov     scon,#0f0h
        mov     tmod,#20h
        mov     th1,#0fdh
        setb    tr1
start:  mov     r1,#40h
        mov     a,addr
        setb    tb8
        mov     sbuf,a
        jnb     ti,$
        clr     ti
        clr     tb8

        clr     sm2
keep_rx: jnb    ri,$
        clr     ri
        mov     a,sbuf
        cjne    a,#0ffh,keep_rx1
        sjmp    next_addr

keep_rx1: mov   @r1,a
        cjne    a,#'*',d
```

```
                inc     r1
                mov     mem,r1
                sjmp    dd
d:              inc     r1
                sjmp    keep_rx


dd:             mov     r1,#40h
ddd:            mov     a,@r1
                mov     p1,a
                acall   clk
                inc     r1
                mov     a,r1
                cjne    a,mem,ddd
                sjmp    next_addr

                setb    sm2

next_addr:      inc     addr
                mov     a,addr
                cjne    a,#44h,start
                mov     addr,#41h
                ajmp    start


delay_100ms:    mov     r7,#50
delay_100ms1:   mov     r6,#0e0h
delay_100ms2:   nop
                nop
                djnz    r6,delay_100ms2
                djnz    r7,delay_100ms1
                ret
```

```
clk:    clr   p3.7
        acall delay_100ms
        setb  p3.7
        ret

delay:  mov   r5,#0ffh
        djnz  r5,$
        ret

        end
```

โปรแกรม mul_tx32.asm สำหรับจัดส่งข้อมูลที่ได้จากโปรแกรม mul_tx31.asm ไปยังคอมพิวเตอร์กลาง

```
addr    equ     70h
mem     equ     71h
        org     00h
        sjmp    main0
        org     03h
        sjmp    int_rt

int_rt:         mov     a,p1
                acall   tx_procedure
                reti

tx_procedure:   mov     sbuf,a
                jnb     ti,$
                clr     ti
                acall   delay_100ms
                ret

main0:          mov     ie,#10000001b
                mov     p1,#0ffh
                setb    it0
                mov     pcon,#00h
                mov     scon,#040h
                mov     tmod,#20h
                mov     th1,#0fdh
                setb    tr1
                sjmp    $

delay_100ms:    mov     r7,#50
```

```
delay_100ms1:  mov    r6,#0e0h
delay_100ms2:  nop
               nop
               djnz   r6,delay_100ms2
               djnz   r7,delay_100ms1
               ret

               end
```

โปรแกรม main03.asm สำหรับเครื่องอ่านหน้าประตูหลัก

```
lcd_en          bit     p3.4
r_w             bit     p3.5
con_data        bit     p3.6
crd_in          bit     p0.4
crd_pwr         bit     p0.3
crd_rst         bit     p0.6
crd_clk         bit     p0.5
crd_io          bit     p0.2
buzzer          bit     p3.2

KPAD_ROW0   BIT     P2.0
KPAD_ROW1   BIT     P2.1
KPAD_ROW2   BIT     P2.2
KPAD_ROW3   BIT     P2.3
KPAD_COL0   BIT     P2.4
KPAD_COL1   BIT     P2.5
KPAD_COL2   BIT     P2.6
flag            equ     2fh
last_add        equ     30h
serial_add      equ     40h
card_buff       equ     60h
serial_num      equ     70h
chk_id          equ     7bh
error_counter   equ     7ch
KPAD_DATA   equ     50h
how_char        equ     51h
key_in          equ     52h
key_chk         equ     58h
```

```
ready_ser           bit     f0
data_complete_flag  bit     flag.0
send_data_flag      bit     flag.1

        org     00h
        ljmp    main

        org     23h
        ljmp    ser_int

initial: clr    con_data
        mov     P1,#38h
        acall   lcd_clk
        acall   lcd_delay
        mov     P1,#0ch
        acall   lcd_clk
        acall   lcd_delay
        mov     p1,#06h
        acall   lcd_clk
        acall   lcd_delay
        mov     p1,#01h
        acall   lcd_clk
        acall   lcd_delay
        ret

lcd_clk: setb   lcd_en
        acall   lcd_delay
        clr     lcd_en
        acall   lcd_delay
        ret
```

```
lcd_clr:      clr     con_data
              mov     p1,#01h
              acall   lcd_clk
              acall   lcd_delay
              ret

lcd_delay:    mov     r6,#2
lcd_delay1:   mov     r7,#0f9h
lcd_delay2:   nop
              nop
              djnz    r7,lcd_delay2
              djnz    r6,lcd_delay1
              ret

delay:        mov     r3,#0ffh
delay1:       mov     r4,#0ffh
              djnz    r4,$
              djnz    r3,delay1
              ret

setline:      clr     con_data
              mov     p1,r2
              acall   lcd_clk
              acall   lcd_delay
              ret

show:         setb    con_data
              mov     p1,a
              acall   lcd_clk
              acall   lcd_delay
              ret
```

```
show_message: mov   r3,#2
              mov   r4,#8
              mov   r2,#80h
              acall setline
line1:        mov   a,#00h
              movc  a,@a+dptr
              inc   dptr
              acall show
              djnz  r4,line1
              mov   r2,#0c0h
              acall setline
              mov   r4,#8
              djnz  r3,line1
              mov   r2,#80h
              acall setline

              ret

delay_show:   mov   r5,#10
delay_show1:  acall delay
              djnz  r5,delay_show1
              ret

delay_100ms:  mov   r7,#10
delay_100ms1: mov   r6,#0e6h
delay_100ms2: nop
              djnz  r6,delay_100ms2
```

```
                djnz    r7,delay_100ms1
                ret

delay_next:             mov     r5,02h
delay_next_step:        mov     r3,0ffh
delay_next_step1:       mov     r4,0ffh
                        djnz    r4,$
                        djnz    r3,delay_next_step1
                        djnz    r5,delay_next_step
                        ret

clear:          clr     con_data
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

rdclk:          mov     b,#08h
                djnz    b,$
                ret

card_reset:     setb    crd_rst
                setb    crd_clk
                acall   rdclk
                clr     crd_clk
                acall   rdclk
                clr     crd_rst
                acall   rdclk
                ret

crd_readbit:    MOV     R7,#08          ;8 bit Read = 1 byte
```

```
                MOV   A,#0
BitCount:       SETB  crd_clk          ;set 1 clock
                SETB  crd_io           ;set bit for read
                LCALL RDCLK            ;delay time
                CLR   C                ;clear flag "C"
                JNB   Crd_IO,isZERO    ;check = "0" or = "1"
                SETB  C
isZERO:         RRC   A                ;Shift data
                LCALL RDCLK            ;delay time
                CLR   Crd_CLK          ;rise clock
                LCALL RDCLK            ;delay time
                DJNZ  R7,BitCount
                RET

sound:          cpl   buzzer
                acall delay_100ms
                setb  buzzer
                ret

tx_procedure:   mov   sbuf,a
                acall delay_100ms
                ret
show_wrong_crd:     mov   dptr,#wrong_crd
                    acall show_message
                    mov   r0,#10
loop_delay:         acall delay
                    djnz  r0,loop_delay
                    mov   dptr,#remove_crd
                    acall show_message
                    ljmp  chk_crd_out
```

```
ser_int:        push    acc
                jbc     ri,ser_rx
                clr     ti
                sjmp    exit_ser_int1
ser_rx:         mov     a,sbuf
                mov     last_add,a
                cjne    a,#43h,exit_ser_int1
                jb      ready_ser,exit_ser_int
                clr     ti
                jb      data_complete_flag,send_data
                mov     sbuf,#0ffh
send_data:      setb    send_data_flag
exit_ser_int:   clr     ready_ser
exit_ser_int1:  pop     acc
                reti

main:           mov     p0,#11111100b
                mov     p3,#10001111b
                mov     ie,#10010000b
                mov     p1,#0ffh
                clr     send_data_flag
                clr     data_complete_flag
                clr     r_w
                mov     scon,#0f0h
                mov     tmod,#20h
                mov     th1,#0fdh
                setb    tr1
                mov     a,#00h
                ;mov    error_counter,#00
                acall   delay
                acall   initial
```

```
                mov     dptr,#text
                acall   show_message
main1:          setb    ready_ser
                jb      ready_ser,$

check_crd_in:   jnb     crd_in,no_crd
                acall   delay
                jnb     crd_in,no_crd

                sjmp    read_crd
no_crd:         mov     sbuf,#0ffh
                sjmp    main1

read_crd:       clr     crd_pwr
                mov     r0,#card_buff
                acall   delay
                acall   card_reset
READC:          MOV     R6,#8           ;nCounter for read 8 BIT = 1 BYTE
READI:          acall   crd_readBit     ;step = 1 Byte return REG 'A'
                MOV     @R0,A
                INC     R0
                DJNZ    R6,READI
                setb    crd_pwr


                mov     r1,#63h
                mov     r0,#serial_num
continue:       mov     a,@r1
                push    acc
                swap    a
```

```
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                pop     acc
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                inc     r1
                cjne    r1,#68h,continue

                mov     sbuf,#0ffh

                mov     r1,#serial_num
                mov     dptr,#id_card1
chk_id_crd:     mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd1
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd
                ljmp    pass

chk_id_crd1:    mov     r1,#serial_num
                mov     dptr,#id_card2
chk_id_crd11:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd2
                inc     dptr
                inc     r1
```

```
                cjne    r1,#7Ah,chk_id_crd11
                ljmp    pass

chk_id_crd2:    mov     r1,#serial_num
                mov     dptr,#id_card3
chk_id_crd22:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd3
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd22
                ljmp    pass

chk_id_crd3:    mov     r1,#serial_num
                mov     dptr,#id_card4
chk_id_crd33:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd4
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd33
                ljmp    pass

chk_id_crd4:    mov     r1,#serial_num
                mov     dptr,#id_card5
chk_id_crd44:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd5
```

```
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd44
                ljmp    pass

chk_id_crd5:    mov     r1,#serial_num
                mov     dptr,#id_card6
chk_id_crd55:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd6
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd55
                ljmp    pass

chk_id_crd6:    mov     r1,#serial_num
                mov     dptr,#id_card7
chk_id_crd66:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd7
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd66
                ljmp    pass

chk_id_crd7:    mov     r1,#serial_num
                mov     dptr,#id_card8
chk_id_crd77:   mov     a,#00h
                mov     chk_id,@r1
```

```
                movc    a,@a+dptr
                cjne    a,chk_id,show_wrong_crd1
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd77
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

pass:                   setb    data_complete_flag
                        jnb     send_data_flag,$

                        mov     r2,#2
loop_chk_busy:          mov     a,last_add
                        cjne    a,#41,send_head0
                        acall   delay
                        djnz    r2,loop_chk_busy
                        mov     dptr,#busy_try_again
                        acall   show_message
                        acall   delay
                        acall   delay
                        ljmp    main
show_wrong_crd1:                ljmp    show_wrong_crd

send_head0:     mov     tmod,#21h
                mov     th1,#0fdh
                mov     tl1,#0fdh
                setb    tr1
                mov     scon,#50h
                acall   delay_100ms
                mov     r2,#2

send_head:      mov     dptr,#header
```

```
send_head1:   mov     a,#0
              movc    a,@a+dptr
              acall   tx_procedure
              inc     dptr
              djnz    r2,send_head1

              mov     r0,#serial_num
              mov     dptr,#number
tx_loop:      mov     a,@r0
              movc    a,@a+dptr
              acall   tx_procedure
              inc     r0
              cjne    r0,#7Ah,tx_loop
              mov     a,#'*'
              acall   tx_procedure
              mov     a,#0ah
              acall   tx_procedure
              mov     a,#0dh
              acall   tx_procedure
              acall   sound
              clr     p3.7
              clr     data_complete_flag
              clr     send_data_flag
              mov     dptr,#welcome
              acall   show_message
              acall   delay_show
              mov     dptr,#remove_crd
              acall   show_message
              acall   delay_show

chk_crd_out:  jb      crd_in,chk_crd_out1
```

```
                acall   delay
                jb      crd_in,chk_crd_out
                setb    p3.7
                ljmp    main
chk_crd_out1:   acall   sound
                sjmp    chk_crd_out

text:           db      "-Insert Ur Card-"
number:         db      "0123456789ABCDEF"
id_card1:       db      1,2,3,4,0,6,6,4,9,3          ;**
id_card2:       db      1,2,7,9,0,7,0,2,5,3          ;**
id_card3:       db      1,2,8,1,1,2,4,0,0,1          ;**
id_card4:       db      1,2,8,2,1,7,3,9,7,2
id_card5:       db      1,2,8,2,1,7,3,9,8,9          ;**
id_card6:       db      1,2,8,9,0,4,3,3,8,9
id_card7:       db      1,3,2,6,0,1,6,5,2,0          ;**
id_card8:       db      1,3,4,1,0,7,3,3,6,0          ;**
wrong_crd:      db      " Wrong ID-Card! "
remove_crd:     db      " Remove Ur Card "
header:         db      "93"
enter_pwd:      db      " Enter Password "
text1:          db      "...Thank You...."
number_pwd:     db      "s123456789#0*"
password:       db      "212278"
error:          db      " Wrong Password "
contact_officer: db     "Contact Officer!"
busy_try_again: db      "Busy.. Try Again"
welcome:        db      "Welcome To KMITL"
end
```

โปรแกรม room02.asm สำหรับเครื่องอ่านหน้าห้องพัก

```
lcd_en          bit     p3.4
r_w             bit     p3.5
con_data        bit     p3.6
crd_in          bit     p0.4
crd_pwr         bit     p0.3
crd_rst         bit     p0.6
crd_clk         bit     p0.5
crd_io          bit     p0.2
buzzer          bit     p3.2

KPAD_ROW0   BIT     P2.0
KPAD_ROW1   BIT     P2.1
KPAD_ROW2   BIT     P2.2
KPAD_ROW3   BIT     P2.3
KPAD_COL0   BIT     P2.4
KPAD_COL1   BIT     P2.5
KPAD_COL2   BIT     P2.6
flag                equ     2fh
last_add            equ     30h
serial_add          equ     40h
card_buff           equ     60h
serial_num          equ     70h
chk_id              equ     7bh
error_counter       equ     7ch
KPAD_DATA           equ     50h
how_char            equ     51h
key_in              equ     52h
key_chk             equ     58h
```

```
ready_ser           bit     f0
data_complete_flag  bit     flag.0
send_data_flag      bit     flag.1

        org     00h
        ljmp    main

        org     23h
        ljmp    ser_int

GET_KPAD:       MOV     P2,#0FFH
                MOV     KPAD_DATA,#0

CHK_COL0:       CLR     KPAD_COL0
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL0_DETECT
                AJMP    CHK_COL1
COL0_DETECT:    MOV     KPAD_DATA,#01
                AJMP    GET_ROW
CHK_COL1:       SETB    KPAD_COL0
                CLR     KPAD_COL1
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL1_DETECT
                AJMP    CHK_COL2

COL1_DETECT:    MOV     KPAD_DATA,#02
                AJMP    GET_ROW

CHK_COL2:       SETB    KPAD_COL1
```

```
                CLR   KPAD_COL2
                MOV   A,P2
                ANL   A,#00FH
                CJNE  A,#00FH,COL2_DETECT
                RET

COL2_DETECT:    MOV   KPAD_DATA,#03
GET_ROW:        CLR   KPAD_COL0
                CLR   KPAD_COL1
                CLR   KPAD_COL2

                JB    KPAD_ROW0,CHK_ROW1
                RET

CHK_ROW1:       JB    KPAD_ROW1,CHK_ROW2
                MOV   A,KPAD_DATA
                ADD   A,#3
                MOV   KPAD_DATA,A
                RET

CHK_ROW2:       JB    KPAD_ROW2,CHK_ROW3
                MOV   A,KPAD_DATA
                ADD   A,#6
                MOV   KPAD_DATA,A
                RET

CHK_ROW3:       MOV   A,KPAD_DATA
                ADD   A,#9
                MOV   KPAD_DATA,A
                RET
```

```
initial:        clr     con_data
                mov     P1,#38h
                acall   lcd_clk
                acall   lcd_delay
                mov     P1,#0ch
                acall   lcd_clk
                acall   lcd_delay
                mov     p1,#06h
                acall   lcd_clk
                acall   lcd_delay
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

lcd_clk:        setb    lcd_en
                acall   lcd_delay
                clr     lcd_en
                acall   lcd_delay
                ret

lcd_clr:        clr     con_data
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

lcd_delay:      mov     r6,#2
lcd_delay1:     mov     r7,#0f9h
lcd_delay2:     nop
                nop
```

```
                djnz    r7,lcd_delay2
                djnz    r6,lcd_delay1
                ret

delay:          mov     r3,#0ffh
delay1:         mov     r4,#0ffh
                djnz    r4,$
                djnz    r3,delay1
                ret

setline:        clr     con_data
                mov     p1,r2
                acall   lcd_clk
                acall   lcd_delay
                ret

show:           setb    con_data
                mov     p1,a
                acall   lcd_clk
                acall   lcd_delay
                ret

show_star:      mov     a,#'*'
                acall   show
                ret

show_message:   mov     r3,#2
                mov     r4,#8
                mov     r2,#80h
                acall   setline
line1:          mov     a,#00h
```

```
                movc    a,@a+dptr
                inc     dptr
                acall   show
                djnz    r4,line1
                mov     r2,#0c0h
                acall   setline
                mov     r4,#8
                djnz    r3,line1

                mov     r2,#80h
                acall   setline
                ret

delay_show:     mov     r5,#10
delay_show1:    acall   delay
                djnz    r5,delay_show1
                ret

delay_100ms:    mov     r7,#10
delay_100ms1:   mov     r6,#0c6h
delay_100ms2:   nop
                djnz    r6,delay_100ms2
                djnz    r7,delay_100ms1
                ret

delay_next:     mov     r5,02h
delay_next_step:mov     r3,0ffh
delay_next_step1:       mov     r4,0ffh
                djnz    r4,$
                djnz    r3,delay_next_step1
                djnz    r5,delay_next_step
```

```
                ret

clear:          clr     con_data
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

rdclk:          mov     b,#08h
                djnz    b,$
                ret

card_reset:     setb    crd_rst
                setb    crd_clk
                acall   rdclk
                clr     crd_clk
                acall   rdclk
                clr     crd_rst
                acall   rdclk
                ret

crd_readbit:    MOV     R7,#08          ;8 bit Read = 1 byte
                MOV     A,#0
BitCount:       SETB    crd_clk         ;set 1 clock
                SETB    crd_io          ;set bit for read
                LCALL   RDCLK           ;delay time
                CLR     C               ;clear flag "C"
                JNB     Crd_IO,isZERO   ;check = "0" or = "1"
                SETB    C
isZERO:         RRC     A                       ;Shift data
                LCALL   RDCLK           ;delay time
```

```
                CLR   Crd_CLK        ;rise clock
                LCALL  RDCLK         ;delay time
                DJNZ   R7,BitCount
                RET

sound:          cpl    buzzer
                acall  delay_100ms
                setb   buzzer
                ret

tx_procedure:   mov    sbuf,a
                acall  delay_100ms
                ret

show_wrong_crd:     mov    dptr,#wrong_crd
                    acall  show_message
                    mov    r0,#10
loop_delay:         acall  delay
                    djnz   r0,loop_delay
                    mov    dptr,#remove_crd
                    acall  show_message
                    ljmp   chk_crd_out

ser_int:        push   acc
                jbc    ri,ser_rx
                clr    ti
                sjmp   exit_ser_int1
ser_rx:         mov    a,sbuf
                mov    last_add,a
                cjne   a,#42h,exit_ser_int1
                jb     ready_ser,exit_ser_int
```

```
                clr     ti
                jb      data_complete_flag,send_data
                mov     sbuf,#0ffh
send_data:      setb    send_data_flag
exit_ser_int:   clr     ready_ser
exit_ser_int1:  pop     acc
                reti

main:           mov     p0,#11111100b
                mov     p3,#10001111b
                mov     ie,#10010000b
                mov     p1,#0ffh
                clr     send_data_flag
                clr     data_complete_flag
                clr     r_w
                mov     scon,#0f0h
                mov     tmod,#20h
                mov     th1,#0fdh
                setb    tr1
                mov     a,#00h
                mov     error_counter,#00
                acall   delay
                acall   initial
                mov     dptr,#text
                acall   show_message
main1:          setb    ready_ser
                jb      ready_ser,$

check_crd_in:   jnb     crd_in,no_crd
                acall   delay
                jnb     crd_in,no_crd
```

```
                sjmp    read_crd
no_crd:         mov     sbuf,#0ffh
                sjmp    main1

read_crd:       clr     crd_pwr
                mov     r0,#card_buff
                acall   delay
                acall   card_reset
READC:          MOV     R6,#8                   ;nCounter for read 8 BIT = 1 BYTE
READI:          acall   crd_readBit             ;step = 1 Byte return REG 'A'
                MOV     @R0,A
                INC     R0
                DJNZ    R6,READI
                setb    crd_pwr

                mov     r1,#63h
                mov     r0,#serial_num
continue:       mov     a,@r1
                push    acc
                swap    a
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                pop     acc
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                inc     r1
                cjne    r1,#68h,continue
```

```
                mov     sbuf,#0ffh

                mov     r1,#serial_num
                mov     dptr,#id_card
chk_id_crd:     mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,show_wrong_crd1
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
start_get_pwd:          mov     a,#00h
                        mov     how_char,#00h
                        mov     kpad_data,#00h
                        mov     r0,#key_in
                        acall   delay
                        acall   initial
                        mov     dptr,#enter_pwd
                        acall   show_message
loop:                   acall   get_kpad
                        mov     a,kpad_data
                        cjne    a,#00h,display
                        sjmp    loop
show_wrong_crd1:        ljmp    show_wrong_crd

display:                mov     r1,how_char
                        cjne    r1,#00h,display2
                        acall   lcd_clr
display2:               cjne    a,#10,display4
                        acall   lcd_clr
```

```
                mov     how_char,#00h
                mov     kpad_data,#00h
                mov     r0,#key_in
                sjmp    loop
display4:       cjne    a,#12,display3
                mov     dptr,#error
                acall   show_message
                acall   delay_show
                inc     error_counter
                mov     a,error_counter
                cjne    a,#3,start_get_pwd
                mov     dptr,#contact_officer
                acall   show_message
                acall   sound
                sjmp    $
display3:       mov     dptr,#number_pwd
                movc    a,@a+dptr
                mov     @r0,a
                inc     r0
                acall   show
                inc     how_char
                mov     r1,how_char
                acall   delay
                acall   delay
                cjne    r1,#6,loop
chk_ent:        acall   get_kpad
                mov     a,kpad_data
                cjne    a,#00h,chk_ent2
                sjmp    chk_ent
chk_ent2:       cjne    a,#10,chk_ent3
```

```
                acall   lcd_clr
                mov     how_char,#00h
                mov     kpad_data,#00h
                sjmp    loop

sound_warn:     acall   sound
                sjmp    chk_ent
incorrect:      mov     dptr,#error
                acall   show_message
                mov     r0,#10
loop_delay1:    acall   delay
                djnz    r0,loop_delay1
                inc     error_counter
                mov     a,error_counter
                cjne    a,#3,start_get_pwd1
                acall   sound
                mov     dptr,#contact_officer
                acall   show_message
                sjmp    $
start_get_pwd1: ljmp    start_get_pwd

chk_ent3:       cjne    a,#12,sound_warn
                mov     a,#00h
                mov     r1,#key_in
                mov     dptr,#password
chk_pwd:        movc    a,@a+dptr
                mov     key_chk,@r1
                cjne    a,key_chk,incorrect
                mov     a,#00h
                inc     dptr
                inc     r1
```

```
                cjne    r1,#58h,chk_pwd
                setb    data_complete_flag
                jnb     send_data_flag,$
                mov     dptr,#text1
                acall   show_message
                mov     r2,#2
loop_chk_busy:  mov     a,last_add
                cjne    a,#41,send_head0
                acall   delay
                djnz    r2,loop_chk_busy
                mov     dptr,#busy_try_again
                acall   show_message
                acall   delay
                acall   delay
                ljmp    start_get_pwd

send_head0:     mov     tmod,#21h
                mov     th1,#0fdh
                mov     tl1,#0fdh
                setb    tr1
                mov     scon,#50h
                acall   delay_100ms
                mov     r2,#2

send_head:      mov     dptr,#header
send_head1:     mov     a,#0
                movc    a,@a+dptr
                acall   tx_procedure
                inc     dptr
                djnz    r2,send_head1
```

```
                mov     r0,#serial_num
                mov     dptr,#number
tx_loop:        mov     a,@r0
                movc    a,@a+dptr
                acall   tx_procedure
                inc     r0
                cjne    r0,#7Ah,tx_loop
                mov     a,#'*'
                acall   tx_procedure
                mov     a,#0ah
                acall   tx_procedure
                mov     a,#0dh
                acall   tx_procedure
                acall   sound
                clr     p3.7
                clr     data_complete_flag
                clr     send_data_flag
                mov     dptr,#remove_crd
                acall   show_message
                acall   delay_show

chk_crd_out:    jb      crd_in,chk_crd_out1
                acall   delay
                jb      crd_in,chk_crd_out
                setb    p3.7
                ljmp    main
chk_crd_out1:   acall   sound
                sjmp    chk_crd_out

text:           db      "-Insert Ur Card-"
number:         db      "0123456789ABCDEF"
```

```
id_card:        db      1,2,7,9,0,7,0,2,5,3
wrong_crd:      db      " Wrong ID-Card! "
remove_crd:     db      " Remove Ur Card "
header:         db      "92"
enter_pwd:      db      " Enter Password "
text1:          db      "...Thank You...."
number_pwd:     db      "s123456789#0*"
password:       db      "212278"
error:          db      " Wrong Password "
contact_officer: db     "Contact Officer!"
busy_try_again: db      "Busy.. Try Again"

end
```

โปรแกรม restau03.asm สำหรับเครื่องอ่านในร้านอาหาร

```
lcd_en          bit     p3.4
r_w             bit     p3.5
con_data        bit     p3.6
crd_in          bit     p0.4
crd_pwr         bit     p0.3
crd_rst         bit     p0.6
crd_clk         bit     p0.5
crd_io          bit     p0.2
buzzer          bit     p3.2

KPAD_ROW0   BIT     P2.0
KPAD_ROW1   BIT     P2.1
KPAD_ROW2   BIT     P2.2
KPAD_ROW3   BIT     P2.3
KPAD_COL0   BIT     P2.4
KPAD_COL1   BIT     P2.5
KPAD_COL2   BIT     P2.6
last_add        equ     30h
flag            equ     2fh
key_in          equ     40h     ;40-5d
card_buff       equ     60h     ;60-67
serial_num      equ     70h
chk_id          equ     7bh
KPAD_DATA   equ     5eh
how_char        equ     5fh
key_chk         equ     6ah
key_in_temp     equ     7ch
ready_ser               bit     f0
```

```
data_complete_flag      bit     flag.0
send_data_flag          bit     flag.1

        org     00h
        ljmp    main

        org     23h
        ljmp    ser_int

GET_KPAD:       MOV     P2,#0FFH
                MOV     KPAD_DATA,#0
CHK_COL0:       CLR     KPAD_COL0
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL0_DETECT
                AJMP    CHK_COL1
COL0_DETECT:    MOV     KPAD_DATA,#01
                AJMP    GET_ROW
CHK_COL1:       SETB    KPAD_COL0
                CLR     KPAD_COL1
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL1_DETECT
                AJMP    CHK_COL2
COL1_DETECT:    MOV     KPAD_DATA,#02
                AJMP    GET_ROW
CHK_COL2:       SETB    KPAD_COL1
                CLR     KPAD_COL2
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL2_DETECT
```

```
                RET

COL2_DETECT:    MOV   KPAD_DATA,#03
GET_ROW:        CLR   KPAD_COL0
                CLR   KPAD_COL1
                CLR   KPAD_COL2
                JB    KPAD_ROW0,CHK_ROW1
                RET

CHK_ROW1:       JB    KPAD_ROW1,CHK_ROW2
                MOV   A,KPAD_DATA
                ADD   A,#3
                MOV   KPAD_DATA,A
                RET

CHK_ROW2:       JB    KPAD_ROW2,CHK_ROW3
                MOV   A,KPAD_DATA
                ADD   A,#6
                MOV   KPAD_DATA,A
                RET

CHK_ROW3:       MOV   A,KPAD_DATA
                ADD   A,#9
                MOV   KPAD_DATA,A
                RET

initial:        clr   con_data
                mov   P1,#38h
                acall lcd_clk
                acall lcd_delay
                mov   P1,#0ch
```

```
                acall   lcd_clk
                acall   lcd_delay
                mov     p1,#06h
                acall   lcd_clk
                acall   lcd_delay
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

lcd_clk:        setb    lcd_en
                acall   lcd_delay
                clr     lcd_en
                acall   lcd_delay
                ret

lcd_clr:        clr     con_data
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

lcd_delay:      mov     r6,#2
lcd_delay1:     mov     r7,#0f9h
lcd_delay2:     nop
                nop
                djnz    r7,lcd_delay2
                djnz    r6,lcd_delay1
                ret

delay:          mov     r3,#0ffh
```

```
delay1:         mov     r4,#0ffh
                djnz    r4,$
                djnz    r3,delay1
                ret

setline:        clr     con_data
                mov     p1,r2
                acall   lcd_clk
                acall   lcd_delay
                ret

show:           setb    con_data
                mov     p1,a
                acall   lcd_clk
                acall   lcd_delay
                ret

show_message:   mov     r3,#2
                mov     r4,#8
                mov     r2,#80h
                acall   setline
line1:          mov     a,#00h
                movc    a,@a+dptr
                cjne    a,#0ffh,line11
                sjmp    exit_show_mess
line11:         inc     dptr
                acall   show
                djnz    r4,line1
                mov     r2,#0c0h
                acall   setline
                mov     r4,#8
```

```
                djnz    r3,line1

exit_show_mess:         ret

delay_100ms:    mov     r7,#100
delay_100ms1:   mov     r6,#0e6h
delay_100ms2:   nop
                nop
                djnz    r6,delay_100ms2
                djnz    r7,delay_100ms1
                ret

clear:          clr     con_data
                mov     p1,#01h
                acall   lcd_clk
                acall   lcd_delay
                ret

rdclk:          mov     b,#08h
                djnz    b,$
                ret

card_reset:     setb    crd_rst
                setb    crd_clk
                acall   rdclk
                clr     crd_clk
                acall   rdclk
                clr     crd_rst
                acall   rdclk
                ret
```

```
crd_readbit:    MOV   R7,#08          ;8 bit Read = 1 byte
                MOV   A,#0
BitCount:       SETB  crd_clk         ;set 1 clock
                SETB  crd_io          ;set bit for read
                LCALL RDCLK           ;delay time
                CLR   C                       ;clear flag "C"
                JNB   Crd_IO,isZERO   ;check = "0" or = "1"
                SETB  C
isZERO:         RRC   A                       ;Shift data
                LCALL RDCLK           ;delay time
                CLR   Crd_CLK         ;rise clock
                LCALL RDCLK           ;delay time
                DJNZ  R7,BitCount
                RET

sound:          cpl   buzzer
                acall delay_100ms
                setb  buzzer
                ret

show_wrong_crd:     mov   dptr,#wrong_crd
                    acall show_message
                    mov   r0,#10
loop_delay:         acall delay
                    djnz  r0,loop_delay
                    mov   dptr,#remove_crd
                    acall show_message
                    ljmp  chk_crd_out

ser_int:        push  acc
                jbc   ri,ser_rx
```

```
                clr     ti
                sjmp    exit_ser_int1
ser_rx:         mov     a,sbuf
                mov     last_add,a
                cjne    a,#41h,exit_ser_int1
                jb      ready_ser,exit_ser_int
                clr     ti
                jb      data_complete_flag,send_data
                mov     sbuf,#0ffh
send_data:      setb    send_data_flag
exit_ser_int:   clr     ready_ser
exit_ser_int1:  pop     acc
                reti

main:           mov     p0,#11111100b
                mov     p3,#10001111b
                mov     ie,#10010000b
                clr     send_data_flag
                clr     data_complete_flag
                mov     r0,#key_in
clr_key_in:             mov   @r0,#00
                inc     r0
                cjne    r0,#5eh,clr_key_in
                mov     p1,#0ffh
                clr     r_w
                mov     scon,#0f0h
                mov     tmod,#20h
                mov     th1,#0fdh
                setb    tr1
                mov     a,#00h
                acall   delay
```

```
                acall   initial
                mov     dptr,#text
                acall   show_message
main1:          setb    ready_ser
                jb      ready_ser,$

check_crd_in:   jnb     crd_in,no_crd
                acall   delay
                jnb     crd_in,no_crd
                sjmp    read_crd
no_crd:         mov     sbuf,#0ffh
                sjmp    main1

read_crd:       clr     crd_pwr
                mov     r0,#card_buff
                acall   delay
                acall   card_reset

READC:          MOV             R6,#8                   ;nCounter for read 8 BIT = 1 BYTE
READI:          acall   crd_readBit             ;step = 1 Byte return REG 'A'
                MOV             @R0,A
                INC     R0
                DJNZ    R6,READI
                setb    crd_pwr
                mov     sbuf,#0ffh
                mov     r1,#63h
                mov     r0,#serial_num
continue:       mov     a,@r1
                cjne    a,#0ffh,continue1
                acall   sound
                mov     dptr,#wrong_crd
```

```
                acall   show_message
                acall   delay
                acall   delay
                acall   delay
                mov     dptr,#remove_crd
                acall   show_message
                acall   delay
                ljmp    chk_crd_out

continue1:      push    acc
                swap    a
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                pop     acc
                anl     a,#0fh
                mov     @r0,a
                inc     r0
                inc     r1
                cjne    r1,#68h,continue
;-------------------------------
                mov     r1,#serial_num
                mov     dptr,#id_card1
chk_id_crd:     mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd1
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd
```

```
                ljmp    pass

chk_id_crd1:    mov     r1,#serial_num
                mov     dptr,#id_card2
chk_id_crd11:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd2
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd11
                ljmp    pass

chk_id_crd2:    mov     r1,#serial_num
                mov     dptr,#id_card3
chk_id_crd22:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd3
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd22
                ljmp    pass

chk_id_crd3:    mov     r1,#serial_num
                mov     dptr,#id_card4
chk_id_crd33:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd4
                inc     dptr
```

```
                inc     r1
                cjne    r1,#7Ah,chk_id_crd33
                ljmp    pass

chk_id_crd4:    mov     r1,#serial_num
                mov     dptr,#id_card5
chk_id_crd44:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd5
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd44
                ljmp    pass

chk_id_crd5:    mov     r1,#serial_num
                mov     dptr,#id_card6
chk_id_crd55:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,chk_id_crd6
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd55
                ljmp    pass

chk_id_crd6:    mov     r1,#serial_num
                mov     dptr,#id_card7
chk_id_crd66:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
```

```
                cjne    a,chk_id,chk_id_crd7
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd66
                ljmp    pass

chk_id_crd7:    mov     r1,#serial_num
                mov     dptr,#id_card8
chk_id_crd77:   mov     a,#00h
                mov     chk_id,@r1
                movc    a,@a+dptr
                cjne    a,chk_id,show_wrong_crd1
                inc     dptr
                inc     r1
                cjne    r1,#7Ah,chk_id_crd77

;-------------------------------.
pass:
start_get_list:         mov     a,#00h
                        mov     how_char,#00h
                        mov     kpad_data,#00h
                        mov     r0,#key_in_temp
                        mov     r5,#key_in
                        acall   delay
show_enter_code:                acall   lcd_clr
                        mov     P1,#0fh         ; แสดงผลพร้อมเคอร์เซอร์กระพริบ
                        acall   lcd_clk
                        acall   lcd_delay
                        mov     dptr,#enter_code
                        acall   show_message
```

```
loop:           acall   get_kpad
                mov     a,kpad_data
                cjne    a,#00h,display2
                sjmp    loop
show_wrong_crd1:ljmp    show_wrong_crd

display2:       cjne    a,#10,display3          ;เป็นclr
                mov     a,how_char
                cjne    a,#00h,clr_dis
                sjmp    ask_finish
clr_dis:        mov     how_char,#00h
                mov     kpad_data,#00h
                mov     r0,#key_in_temp
                acall   delay
                sjmp    show_enter_code

display3:       cjne    a,#12,display4
                mov     dptr,#not_list
                clr     con_data
                mov     P1,#0ch
                acall   lcd_clk
                acall   lcd_delay
                acall   show_message
                acall   delay
                acall   delay
                mov     how_char,#00h
                mov     r0,#key_in_temp
                sjmp    show_enter_code

display4:       mov     dptr,#number_kb
```

```
                movc    a,@a+dptr
                mov     @r0,a
                inc     r0
                acall   show
                inc     how_char
                mov     r1,how_char
                acall   delay
                acall   delay
                cjne    r1,#3,loop
chk_ent:        acall   get_kpad
                mov     a,kpad_data
                cjne    a,#00h,chk_ent2
                sjmp    chk_ent
chk_ent2:       cjne    a,#10,chk_ent3
                mov     how_char,#00h
                mov     kpad_data,#00h
                mov     r0,#key_in_temp
                sjmp    show_enter_code

sound_warn:     acall   sound
                sjmp    chk_ent

chk_ent3:       cjne    a,#12,sound_warn
                mov     r0,#key_in_temp
temp2keyin:     mov     a,@r0
                mov     b,r5
                mov     r1,b
                mov     @r1,a
                inc     r0
                inc     r5
                cjne    r0,#7fh,temp2keyin
```

```
                mov     r0,#key_in_temp
ask_finish:     clr     con_data
                mov     P1,#0ch
                acall   lcd_clk
                acall   lcd_delay
                acall   lcd_clr
                mov     dptr,#finish            ; finish?
                acall   show_message
                acall   delay
                acall   delay
loopfinish:     acall   get_kpad
                mov     a,kpad_data
                cjne    a,#00h,loopfinish1
                sjmp    loopfinish
loopfinish1:    cjne    a,#12,next_code
                mov     r0,#2
loopfinish3:    mov     a,last_add
                cjne    a,#02,loopfinish2
                acall   delay
                djnz    r0,loopfinish3
                mov     a,last_add
                cjne    a,#01,line_busy
loopfinish2:    mov     b,r5
                mov     r1,b
                mov     @r1,0ffh
                setb    data_complete_flag
                jnb     send_data_flag,$
                mov     dptr,#text1
                acall   show_message
```

```
                sjmp    send_data1
line_busy:      mov     dptr,#busy_try_again
                acall   show_message
                acall   delay
                ljmp    start_get_list

next_code:      acall   lcd_clr
                mov     P1,#0fh         ; แสดงผลพร้อมเคอร์เซอร์กระพริบ
                acall   lcd_clk
                acall   lcd_delay
                mov     b,a
                mov     dptr,#enter_code
                acall   show_message
                mov     how_char,#00
                mov     a,b
                cjne    a,#10,display4_1
                ljmp    loop
display4_1:     ljmp    display4
;-------------------------
tx_procedure:   mov     sbuf,a
                acall   delay_100ms
                ret
;--------------------------------
send_data1:     mov     tmod,#21h
        mov     th1,#0fdh
        mov     tl1,#0fdh
        setb    tr1
        mov     scon,#50h
        ;acall  delay_100ms
        mov     r2,#2
```

```
send_head:          mov    dptr,#header
send_head1:   mov    a,#0
              movc   a,@a+dptr
              acall  tx_procedure
              inc    dptr
              djnz   r2,send_head1

              mov    r0,#serial_num
              mov    dptr,#number
tx_loop:      mov    a,@r0
              movc   a,@a+dptr
              acall  tx_procedure
              inc    r0
              cjne   r0,#7Ah,tx_loop

send_code:    mov    r0,#key_in
send_code1:   mov    a,@r0
              cjne   a,#0ffh,send_code2
              sjmp   end_of_code
send_code2:   acall  tx_procedure
              inc    r0
              sjmp   send_code1

end_of_code:  mov    a,#'*'
              acall  tx_procedure
              mov    a,#0ah
              acall  tx_procedure
              mov    a,#0dh
              acall  tx_procedure
              acall  sound
              clr    data_complete_flag
```

```
                clr     send_data_flag
                ;acall  delay_100ms
                mov     dptr,#remove_crd
                acall   show_message
;               mov     r5,#10
;delay_show:    acall   delay
;               djnz    r5,delay_show

chk_crd_out:    jb      crd_in,chk_crd_out1
                acall   delay
                jb      crd_in,chk_crd_out
                setb    p3.7
                ljmp    main
chk_crd_out1:   acall   sound
                sjmp    chk_crd_out

id_card1:       db      1,2,3,4,0,6,6,4,9,3         ;**
id_card2:       db      1,2,7,9,0,7,0,2,5,3         ;**
id_card3:       db      1,2,8,1,1,2,4,0,0,1         ;**
id_card4:       db      1,2,8,2,1,7,3,9,7,2
id_card5:       db      1,2,8,2,1,7,3,9,8,9         ;**
id_card6:       db      1,2,8,9,0,4,3,3,8,9
id_card7:       db      1,3,2,6,0,1,6,5,2,0         ;**
id_card8:       db      1,3,4,1,0,7,3,3,6,0         ;**
text:           db      "-Insert Ur Card-"
enter_code:     db      " Food Code :",0ffh
number:         db      "0123456789ABCDEF"
id_card:        db      1,2,7,9,0,7,0,2,5,3
header:         db      "91"
wrong_crd:      db      " Wrong ID-Card! "
```

```
remove_crd:     db      " Remove Ur Card "
not_list:       db      "Wrong Food Code "
finish:         db      " Finish? ",0ffh
text1:          db      "...Thank You...."
number_kb:      db      "s123456789#0*"
busy_try_again: db      "Busy...Try Again"

end
```

บรรณานุกรม

1) ฐานข้อมูล Access 2002 XP บัณฑิต จามรภูมิ สำนักพิมพ์ สวัสดีไอที

2) คู่มือเรียน Visual Basic 6 ฉันทวุฒิ พืชผล พิชัย สันติกุลานนท์ พิมพ์ครั้งที่ 5 บริษัทโปรวิชั่น จำกัด

3) Visual Basic 6 ฉบับโปรแกรมเมอร์ กิตติ ภักดีวัฒนะกุล จำลอง ครูอุตสาหะ พิมพ์ครั้งที่ 11 สำนักพิมพ์ เคพีที คอมพ์ แอนด์ คอนซัลท์

4) Visual Basic 6 ฉบับฐานข้อมูล กิตติ ภักดีวัฒนะกุล จำลอง ครูอุตสาหะ พิมพ์ครั้งที่ 11 สำนักพิมพ์ เคพีที คอมพ์ แอนด์ คอนซัลท์

5) การเขียนโปรแกรม สำหรับ Client/Server ด้วย Visual Basic 6 และ ASP.VBScript.Access.SQL Server

6) พ.อ. เจนวิทย์ เหลืองอร่าม ปิยวิทย์ เหลืองอร่าม สำนักพิมพ์ เจ แอนด์ พี คอท เนท

7) เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม โดย อรรถพล บุญยะโภคา วรพจน์ กรแก้ววัฒนกุล ชัยวัฒน์ ลิ้มพรจิตร วิไล Innovation Experiment Co.,Ltd.

8) เทคโนโลยีสมาร์ตการ์ด เลศแซ่ตั้ง บริษัท ซีเอ็ดยูเคชั่น จำกัด(หมาชน)

9) http://www.bangkokbiznews.com/2003/kit/2003kit/0130/14.html

10) http://www.bangkokbiznews.com/2003/kit/2003kit/0130/14.html (wire less smart card)

11) http://library.kmitnb.ac.th/projects/edu/TCT/tct0013t.html (เครื่องอ่านสมาร์ตการ์ด)

12) http://tmec.nectec.or.th/smartcard/smartcard_main.htm (smartcard news)

13) http://www.cipitc.or.th/court_order_decision/IP_judgement/jm2545/ip_121_2545.html

14) http://www.thaiinternetwork.com/article/view.php?No=123 (wire less smart card)

15)http://web.ku.ac.th/schoolnet/snet1/network/smart_c.htm (บัตรอเนกประสงค์)

16) http://www.atshop.com/MRWINDOW/(tm110201)smartcard.html (Smart Card)

17) http://www.nectec.or.th/bid/hotissue_smartcard.htm (Business smart card)

18) http://securetech-corp.com/smart_ca.html #1ACS Smart Card 8K EEPROM

19) www.slb.com

20) http://www.smartcards.net/infosec/index.html (detail)

20) http://www.scmegastore.com (market)

21) http://www.schlumbergersema.com/caseStudies/infrastructure/deXaBadgeReprint.pdf (manual)