สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การประยุกต์ใช้งานบัตรแม่เหล็กกับระบบAccess control

Application of magnetic card for Access control system

T042755

โดย

นายฐิติพงส์ เตชะพิชญะ

นายณฐพงศ์ เภาพิจิตร

นายประดิษฐ์ ธรรมนนทิกุล

เลขหมู่.....
เลขทะเบียน 42755
วัน, เดือน, ปี 7 มิ.ย. 2545

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

b11211508

การประยุกต์ใช้งานบัตรแม่เหล็กกับระบบAccess control

Application of magnetic card for Access control system

โดย

นาย ฐิติพงศ์ เตชะพิชญะ เลขประจำตัว 40010200

นาย ณฐพงศ์ เภาพิจิตร เลขประจำตัว 40010206

นาย ประดิษฐ์ ธรรมนนทิกุล เลขประจำตัว 40010432

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. กิตติ ไพฑูรย์วัฒนกิจ

ปริญญานิพนธ์ปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ ปีการศึกษา 2543

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งานบัตรแม่เหล็กกับระบบAccess control

ผู้จัดทำ

1. นายฐิติพงศ์ เตชะพิชญะ
2. นายณฐพงศ์ เภาพิจิตร
3. นายประดิษฐ์ ธรรมนนทิกุล

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. กิตติ ไพฑูรย์วัฒนกิจ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประยุกต์ใช้งานบัตรแม่เหล็กกับระบบAccess control

Application of magnetic card for Access control system

ผู้จัดทำ

1. นายฐิติพงศ์ เตชะพิชญะ
2. นายณฐพงศ์ เภาพิจิตร
3. นายประดิษฐ์ ธรรมนนทิกุล

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รองศาสตราจารย์ ดร. กิตติ ไพฑูรย์วัฒนกิจ)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การประยุกต์ใช้งานบัตรแม่เหล็กกับระบบAccess control

นาย ฐิติพงศ์ เตชะพิชญะ
นาย ณฐพงศ์ เภาพิจิตร
นายประดิษฐ์ ธรรมนนทิกุล
รองศาสตราจารย์ ดร. กิตติ ไพฑูรย์วัฒนกิจ
(อาจารย์ที่ปรึกษา)
ปีการศึกษา 2543

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เป็นการประยุกต์ใช้งานข้อมูลบนบัตรแม่เหล็ก โดยใช้ไมโครคอนโทรลเลอร์ตระกูลMCS – 51 ในการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก แล้วทำการส่งข้อมูลที่ได้เข้าไปยังคอมพิวเตอร์ผ่านทางพอร์ทอนุกรม เพื่อนำข้อมูลที่ได้ไปใช้งานในการควบคุมการผ่านเข้าออกประตู ตามอาคารต่างๆ และผู้ที่ได้รับอนุญาตให้ทำการผ่านเข้าออกได้นั้นจะสามารถควบคุมอุปกรณ์ไฟฟ้าต่างๆภายในอาคารนั้นผ่านทางหน้าจอคอมพิวเตอร์ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Application of magnetic card for Access control system

Titipong Techapichaya

Natapong Paopijit

Pradit Trammanontikul

Assoc. Prof. Dr. Kitti Paitoonwattanakij

2000

ABSTRACT

This project is an application of data in magnetic card reader. Microcontroller (MCS-51) is used to receive data from magnetic stripe reader and send it to personal computer by serial port. These data will be used in access-control system and a person who is authorized can control all electrical devices in a building.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในปัจจุบัน การอ่านและการบันทึกข้อมูลลงบนบัตรแม่เหล็ก ถูกนำไปใช้งานกันอย่างแพร่หลาย ไม่ว่าจะเป็นบัตรเครดิต บัตรเอทีเอ็ม บัตรประจำตัวนักศึกษา รวมถึงการนำเอาไปใช้ทางด้านรักษาความปลอดภัยเช่นบัตรรูดเปิด-ปิดประตู การใช้บัตรในการเช็คเวลาเข้าออกของพนักงาน หรือแม้แต่ในการใช้บัตรเช็คหรือเปิดดูข้อมูลที่เป็นความลับต่างๆ โดยผู้ถือบัตรจะมีสิทธิ์ในการที่จะสามารถใช้บริการต่างๆได้ แม้ว่าในปัจจุบันได้มีการนำเอาการบันทึก-อ่านข้อมูลแบบบาร์โค้ดหรือสมาร์ทการ์ดใช้แล้วก็ตาม แต่จะเห็นได้ว่าการใช้บัตรแม่เหล็กในการเก็บข้อมูลเพื่อนำไปใช้งานทางด้านต่างๆนั้นมีข้อดีมากกว่า ในด้านของการปลอมแปลงและเลียนแบบนั้นทำได้ยากกว่าแบบบาร์โค้ด ซึ่งแบบบาร์โค้ดนั้นสามารถทำการลอกเลียนได้ง่ายดายมากแค่เพียงนำแถบบาร์โค้ดนั้นไปถ่ายเอกสารก็สามารถใช้งานได้เหมือนตัวบัตรจริง ส่วนสมาร์ทการ์ดนั้นกระบวนการการผลิตนั้นจะทำได้ยากและใช้ต้นทุนสูงซึ่งเหมาะกับองค์กรหรือธุรกิจใหญ่ๆซึ่งไม่ต้องการให้มีการลอบปลอมแปลงบัตรเกิดขึ้นได้ อันจะก่อให้เกิดความเสียหายอย่างใหญ่หลวงต่อองค์กรหรือธุรกิจนั้น

ในโครงการนี้จะเป็นการประยุกต์ใช้งานเครื่องอ่านข้อมูลบนบัตรแม่เหล็กกับระบบรักษาความปลอดภัยในบ้าน หรือตามอาคารสำนักงานต่างๆ ซึ่งจำเป็นต้องมีการนำข้อมูลในบัตรแต่ละใบไปเก็บเป็นฐานข้อมูลของระบบ โดยตัวเครื่องอ่านข้อมูลบนบัตรแม่เหล็กนี้เองที่เป็นตัวกลางในการสื่อสารข้อมูลระหว่างตัวเครื่องกับผู้ใช้ เมื่อผู้ใช้มาทำการติดต่อผ่านบัตรของตนก็จะมีการเชื่อมต่อไปยังฐานข้อมูลเพื่อเรียกดูข้อมูลของเจ้าของบัตรทำให้ระบบสามารถตรวจสอบและตัดสินการให้สิทธิแก่เจ้าของบัตรนั้นได้ ในโครงงานนี้จะใช้เครื่องอ่านบัตรแม่เหล็กแบบรูด ทั้งนี้เพื่อความสะดวกและรวดเร็วในการติดต่อกับผู้ใช้ ซึ่งจะต้องใช้ไมโครคอนโทรเลอร์ MCS-51 มาเป็นตัวควบคุมการอ่านข้อมูลจากบัตรของเครื่องอ่านบัตรและ ควบคุมการติดต่อกับฐานข้อมูลบนคอมพิวเตอร์ส่วนบุคคล โดยผ่านทางพอร์ทอนุกรม มาตรฐาน RS-232 อีกทั้งยังเพิ่มอำนวยความสะดวกในการนำเอาระบบคอมพิวเตอร์ซึ่งใช้เก็บข้อมูล มาใช้ควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าในบ้านหรือภายในอาคาร ต่างๆ เช่นระบบไฟฟ้า ระบบปรับอากาศ ระบบแสงสว่างต่างๆ ซึ่งจะมีการตรวจสอบสถานะหรือสภาวะการทำงานของอุปกรณ์ต่างๆในระบบนั้น แล้วนำมาบนหน้าจอคอมพิวเตอร์เพื่อให้ผู้ใช้ทราบสถานะ และสามารถทำการสั่งเปิดปิดเครื่องใช้ไฟฟ้าเหล่านั้นได้ใน 2 ทิศทางคือจากทางคอมพิวเตอร์และทางสวิทช์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วัตถุประสงค์ของโครงงาน

1. โครงงานนี้มีวัตถุประสงค์เพื่อศึกษาการทำงานและการเขียนโปรแกรมใช้งานเครื่องอ่านข้อมูลบนบัตรแม่เหล็ก เพื่อติดต่อกับฐานข้อมูลผ่านทางพอร์ทอนุกรม มาตรฐานRS-232 ซึ่งจะเป็นการส่งข้อมูลติดต่อกันระหว่างคอมพิวเตอร์ส่วนบุคคล กับวงจรควบคุมระบบโดยไมโครคอนโทรลเลอร์ ซึ่งจะต้องให้มีการส่งข้อมูลตอบรับไปมาผ่านทางพอร์ทอนุกรม โดยมีการผิดพลาดเกิดขึ้นให้น้อยที่สุดหรือไม่มีเลย

2. เพื่อเพิ่มอำนวยความสะดวกในการควบคุมการเปิดปิดอุปกรณ์ไฟฟ้าซึ่งติดตั้งทั่วไปในอาคาร บ้านเรือน หรือหอพัก ให้สามารถดูและและควบคุมได้จากส่วนกลาง ซึ่งสามารถดูสถานะของอุปกรณ์เหล่านั้นทั้งหมดที่หน้าจอคอมพิวเตอร์ได้ ไม่ให้เกิดการเปิดทิ้งไว้ในขณะที่ไม่มีคนอยู่อันจะเป็นเหตุให้เกิดการสิ้นเปลืองพลังงานได้

3. เพื่อเพิ่มความปลอดภัยให้กับหน่วยงานต่างๆในปัจจุบันไม่ว่าจะเป็นโรงงาน ธุรกิจ ห้างร้าน หรือบ้านคนอยู่อาศัย เป็นการควบคุมไม่ให้บุคคลภายนอกผู้ไม่ได้รับสิทธิ์ เข้ามากระทำการมิดีมิร้าย อันจะส่งผลให้เกิดความเสียหายที่ตามมา อีกทั้งยังเป็นการจัดดูแลระบบในองค์กรให้มีความเป็นระเบียบเรียบร้อยในการควบคุมบุคลากรและให้เป็นไปตามที่ต้องการ โดยมีการเก็บสถานะบุคคลเข้า-ออก ตลอดเวลา

ขอบเขตของโครงงาน

จากลักษณะของโครงงานดังที่กล่าวมา จะประกอบไปด้วยอุปกรณ์แต่ละส่วนที่มีหน้าที่ต่างๆกัน ดังต่อไปนี้

1. บอร์ดไมโครคอนโทรลเลอร์ตระกูลMCS-51 ใช้สำหรับควบคุมอุปกรณ์ต่างๆเช่น การอ่านข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก การแสดงผลบนหน้าจอLCD การรับค่าคีย์บอร์ดที่ผู้ใช้ป้อนรหัส-ผ่าน ควบคุมการกระตุ้นสถานะของรีเลย์เพื่อควบคุมการเปิดปิดและตรวจสอบสถานะของอุปกรณ์ไฟฟ้าเพื่อส่งให้คอมพิวเตอร์ ส่งสัญญาณเตือนภัย เปิดปิดประตูเข้า-ออก และเป็นผู้ติดต่อสื่อสารรับส่งข้อมูลกับคอมพิวเตอร์โดยตรง

2. บอร์ดควบคุมเปิดปิด และตรวจสถานะอุปกรณ์ไฟฟ้า โดยจะใช้อุปกรณ์ประเภทรีเลย์ในวงจรทางด้านไฟฟ้าระดับแรงดัน220โวลท์ที่จ่ายให้กับอุปกรณ์ไฟฟ้า และจะมีส่วนตรวจสอบสถานะของอุปกรณ์ไฟฟ้าว่าเปิดหรือปิดอยู่ เพื่อส่งผลรายงานส่งไปให้คอมพิวเตอร์ทราบว่าสถานะของอุปกรณ์แต่ละตัวเป็นอย่างไร

3. บอร์ดติดต่อกับผู้ใช้ประกอบด้วยจอLCDเป็นตัวแสดงข้อความสื่อสารให้ผู้ใช้เข้าใจ และคีย์บอร์ดให้ผู้ใช้สามารถกดรหัสผ่านเพื่อให้ไมโครคอนโทรลเลอร์นำไปประมวลผลต่อไป

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้ สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. เครื่องอ่านข้อมูลบนบัตรแม่เหล็ก โดยการเขียนโปรแกรมสั่งงานไมโครคอนโทรลเลอร์ให้รับค่าที่อ่านได้จากบัตรแม่เหล็ก นำไปประมวลผลเทียบกับฐานข้อมูลที่คอมพิวเตอร์ได้ส่งมา

5. การสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์(MCS-51) จะเป็นการส่งข้อมูลแบบอนุกรม(Serial) ตามมาตรฐาน RS-232 ผ่านทางพอร์ทอนุกรมของคอมพิวเตอร์ซึ่งถือว่าเป็นสื่อกลางในการติดต่อกับอุปกรณ์ทั้งสอง

6. โปรแกรมที่ใช้แสดงผลบนจอคอมพิวเตอร์ เป็นโปรแกรมที่เขียนขึ้นโดยใช้โปรแกรมวิชวลเบสิก(Visual Basic6.0) ซึ่งใช้ในการเขียนโปรแกรมติดต่อส่งและรับข้อมูล กับไมโครคอนโทรลเลอร์ เขียนโปรแกรมควบคุมการจัดเก็บฐานข้อมูลในคอมพิวเตอร์ และส่วนของกราฟฟิกที่ใช้ในการควบคุมการเปิดปิดอุปกรณ์ไฟฟ้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2

ทฤษฎีหรือหลักการ

2.1 ความรู้เบื้องต้นเกี่ยวกับบัตรแม่เหล็ก

รูปแบบของข้อมูลที่บันทึกบนแถบแม่เหล็กและคุณลักษณะต่างๆของบัตรแม่เหล็กจะเป็นไปตามมาตรฐานสากล ISO 7810 – 7813 โดยมีรายละเอียดดังต่อไปนี้

2.1.1 *มาตรฐานของบัตรแม่เหล็ก*

รูปที่ 2.1 แสดงขนาดมาตรฐานของบัตรแม่เหล็ก

จากรูปที่ 2.1 ตามมาตรฐานของ ISO เราสามารถพิจารณาแทร็คต่างๆที่ใช้งานได้เป็น 3 แทร็ค โดยมีรายละเอียดในการใช้งานแต่ละแทร็คดังนี้

แทร็คที่ 1 เป็นแทร็คที่สามารถทำการอ่านข้อมูลได้อย่างเดียว เรียกว่า Read only โดยข้อมูลประกอบด้วยตัวอักษรและตัวเลขอยู่ในช่วงบริเวณเส้นขนาน a และ เส้นขนาน b รูปแบบของข้อมูลใน แทร็คที่หนึ่งจะเป็นดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SS	FC	PAN	FS	NAME	FS	Additional Data	Discretionary Data	ES	LRC

SS = Start Sentinel FS = Field Separator

ES = End Sentinel LRC = Longtitudinal Redundancy Check

FC = Format Code PAN = Primary Account Number

รูปที่ 2.2 รูปแบบของข้อมูลในแทร็คที่ 1 ของบัตรแม่เหล็ก

ข้อมูลในแทร็คที่ 1 จะมีทั้งหมด 76 ตัวอักขระ (ไม่รวมตัวอักขระเริ่มต้น , ตัวอักขระปิดท้ายและ LRC) โดยความหมายของตัวอักขระต่างๆจะกล่าวถึงในแทร็คที่ 2 เนื่องจากในโครงงานนี้จะใช้ข้อมูลของบัตรแม่เหล็กในแทร็คที่ 2 มาใช้งาน

แทร็คที่ 2 เป็นแทร็คที่สามารถอ่านข้อมูลได้อย่างเดียวเช่นกัน แต่ข้อมูลจะเป็นลักษณะตัวเลขเพียงอย่างเดียว อยู่ในช่วงเส้นขนาน b และเส้นขนาน c ซึ่งเป็นแทร็คที่ใช้งานโดยทั่วไป (รวมทั้งในโครงงานนี้ด้วย) รูปแบบการจัดเรียงข้อมูลในแทร็คที่ 2 เป็นดังนี้

SS	PAN	FS	Additional data	Discretionary Data	ES	LRC

SS = Start Sentinel FS = Field Separator

ES = End Sentinel LRC = Longtitudinal Redundancy Check

PAN = Primary Account Number

รูปที่ 2.3 รูปแบบของข้อมูลในแทร็คที่ 2 ของบัตรแม่เหล็ก

ข้อมูลในแทร็คที่ 2 ของบัตรแม่เหล็กจะมีรายละเอียดดังนี้

1. ก่อนถึงส่วนที่เป็นข้อมูลจริงของแทร็ค จะประกอบด้วยตัวอักขระที่เป็นศูนย์ทั้งหมด (Leading clock zeros) โดยจะมีประมาณ 22 ตัวอักษร

2. ตัวอักขระที่เป็นตัวบอกจุดเริ่มต้นของชุดข้อมูลในแทร็คที่ 2 ของบัตร (Start Sentinel-SS) คือ 01011 หรือ " B " ในเลขฐาน 16

3. จะมีส่วนที่เป็นข้อมูลของบัตร (ไม่รวมตัวอักขระเริ่มต้น,ตัวอักขระปิดท้าย และตัวอักขระตรวจสอบข้อผิดพลาด) จะมีทั้งหมด 37 ตัวอักษร โดยข้อมูลทั้งหมดจะเป็นตัวเลขเท่านั้น

4. ตัวอักขระ FS (Field Separator) เป็นตัวอักขระที่บอกถึงการจบของชุดข้อมูล ซึ่งจะใช้เลขฐานสอง "1101" หรือ " D " ในเลขฐาน 16 แทน FS

5. ตัวอักขระที่บอกจุดสิ้นสุดของชุดข้อมูล (ตัวอักขระปิดท้าย)ในแทร็คที่ 2 ของบัตร (End Sentinel – ES) คือ 11111 หรือ " F " ในเลขฐาน 16

6. หลังจากตัวอักขระปิดท้ายข้อมูลแล้ว จะมีตัวอักขระ LRC (Longtitudinal Redundancy Check) ซึ่งจะเป็นตัวอักขระที่ใช้ตรวจสอบความผิดพลาดในการอ่านข้อมูลจากบัตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7. ส่วนที่เหลือของบัตรหลังจาก LRC แล้ว จะเป็นตัวอักขระที่เป็นศูนย์ทั้งหมดจนสุดบัตร (Trailing clocking zeros) โดยจะมีประมาณ 22 ตัวอักษรเช่นกัน

8. ข้อมูลทุกตัว (ยกเว้นส่วนที่เป็น Leading clock zeros และ Trailing clocking zeros) จะมีส่วนที่เป็นข้อมูลจริงๆ 4 บิต และจะมีบิตพาริตี้อีก 1 บิต โดยจะใช้พาริตี้คู่ โดยจะมีรูปแบบดังนี้

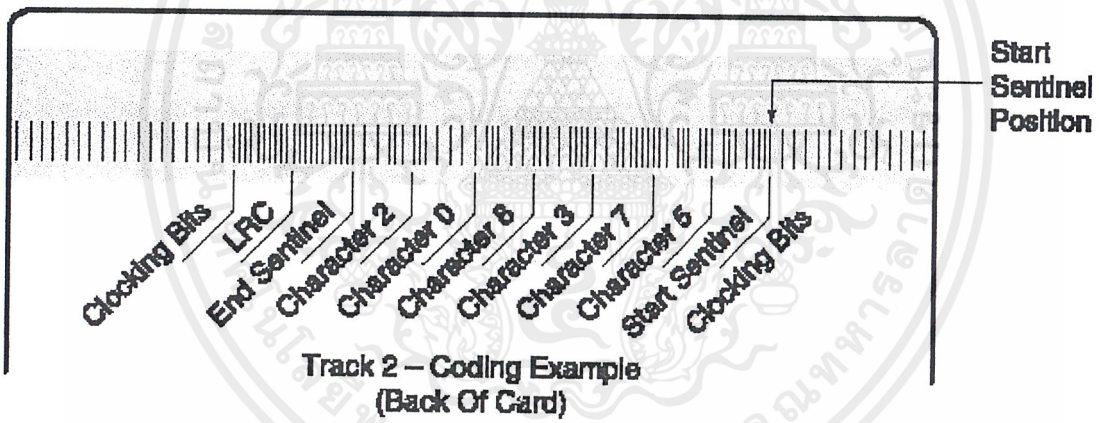
Parity	B3	B2	B1	B0

แต่ลำดับของข้อมูลที่อ่านได้จากเครื่องอ่านจะเป็นดังนี้

B0	B1	B2	B3	parity

นั่นคือจะอ่านบิตศูนย์เข้ามาก่อน และอ่านบิตพาริตี้เข้ามาเป็นลำดับสุดท้าย

โดยการเขียนข้อมูลลงในบัตรจริงๆ จะมีลักษณะดังรูป



รูปที่ 2.4 รูปแบบของการเขียนข้อมูลลงในบัตรแม่เหล็ก

ตัวอย่างเช่น ข้อมูล 123 จะถูกเขียนลงในบัตรเรียงตามลำดับ ดังนี้

“00...0000”	leading clocking zeros	ประมาณ 22 ตัวอักษร
“11010”	hex “ B ”	(01011) หรือ SS
“10000”	ข้อมูลเลข 1	(00001)
“01000”	ข้อมูลเลข 2	(00010)
“11001”	ข้อมูลเลข 3	(00011)
“11111”	hex “ F ”	(11111)
“00100”	ข้อมูลเลข 4	(00100) LRC check
“000..000”	trailing clocking zeros	ประมาณ 22 ตัวอักษร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. การคำนวณค่า LRC (Longtitudinal Redundancy Check)

การคำนวณค่า LRC ของข้อมูลในแทร็ค 2 ของบัตรแม่เหล็ก ทำได้ดังนี้

9.1 เขียนข้อมูลทุกตัวอักษรในรูปของเลขฐาน 2 เรียงกัน ดังตัวอย่างข้างล่าง

9.2 จะไม่ใช่บิตพาริตี้ในการคำนวณ LRC

9.3 นับจำนวนบิตที่เป็นเลข 1 ทั้งหมดในแนวตั้งของแต่ละแถว

9.4 ถ้าหากจำนวนของเลข 1 เป็นจำนวนคู่ ให้ใส่เลข 0 ลงไปเป็นค่า LRC แต่ถ้าหากจำนวนของเลขหนึ่งเป็นเลขคี่ ก็ให้ใส่เลข 1 ลงไปเป็นค่า LRC แทน ลักษณะนี้ก็คือการนำข้อมูลทุกๆ ตัวโดยที่ไม่นับบิตพาริตี้ มาทำการ exclusive or กันทีละบิตนั่นเอง

9.5 จากนั้นจึงทำการใส่ค่าบิตพาริตี้ให้กับ LRC โดยจะใช้พาริตี้แบบคี่

	P	B3	B2	B1	B0
B	0	1	0	1	1
0	1	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	1	0	0	1	1
F	1	1	1	1	1
LRC	0	0	1	0	0

โดยที่ "B" คือ SS (Start sentinel)

"F" คือ ES (End sentinel)

ในที่นี้ค่าของ LRC คือ 00100

รูปที่ 2.5 แสดงตัวอย่างการคำนวณค่า LRC

แทร็คที่ 3 เป็นแทร็คที่สามารถทำการอ่านและเขียนข้อมูลลงไปได้ เก็บข้อมูลที่เป็นตัวเลข โดยจะอยู่ในช่วงเส้นขนาน c และเส้นขนาน d มีรูปแบบของการจัดเรียงข้อมูลเป็นดังนี้

SS	PAN	FS	Use and security data	Additional data	ES	LRC

SS = Start Sentinel FS = Field Separator

ES = End Sentinel LRC = Longtitudinal Redundancy Check

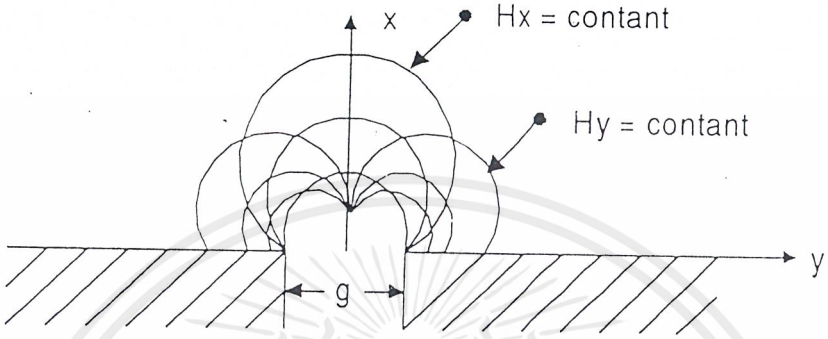
PAN = Primary Account Number

รูปที่ 2.6 รูปแบบของข้อมูลในแทร็คที่ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 การบันทึกเทปแม่เหล็กในระบบดิจิตอล (Magnetic tape recording)

ในการบันทึกสัญญาณลงบนแถบแม่เหล็กและการนำสัญญาณคืนกลับหรือเล่นกลับ (Playback) จะเกี่ยวข้องกับสนามแม่เหล็ก (Magnetic field) ดังแสดงในรูป 2.6

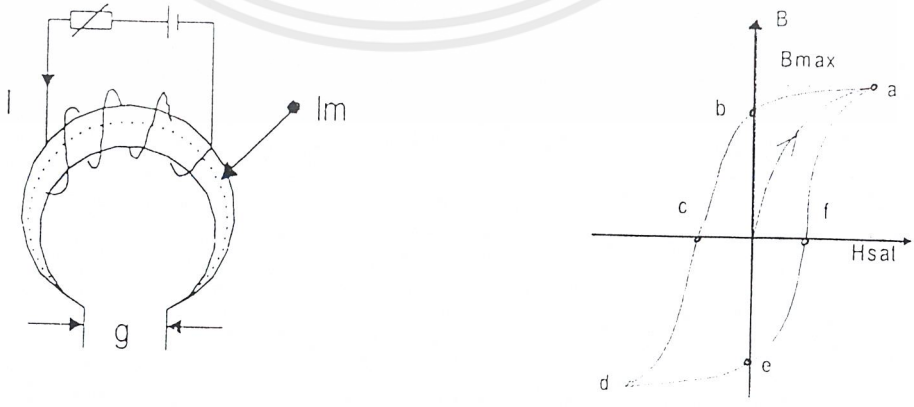


รูปที่ 2.7 แสดงความเข้มสนามแม่เหล็กที่เกิดขึ้นบริเวณหัวเทป

การอธิบายเบื้องต้นของการบันทึกลงบนพื้นผิวแม่เหล็ก จะเกี่ยวข้องกับกระบวนการทำให้เกิดความเป็นแม่เหล็ก (Magnetization) ในช่วงพื้นที่นั้นของส่วนที่จะถูกบันทึก (Recording medium) บริเวณช่องว่าง (Air gap) ของหัวบันทึก จะเกิดฟลักซ์แม่เหล็กรั่วในบริเวณความอิ่มตัวของแก็ป (Gap) ผ่านชั้นบางๆของวัสดุที่มีความเป็นแม่เหล็กเมื่อหัวบันทึกได้รับแรงเคลื่อนแม่เหล็กไฟฟ้า

$$\int H.dl = N.I = F = mmf$$

เมื่อสารแม่เหล็กถูกบันทึกถึงจุดอิ่มตัว (Saturation) ในความเป็นแม่เหล็กของวัสดุแต่ละชนิดจะเกิดความสัมพันธ์ของความหนาแน่นของเส้นแรงแม่เหล็ก (B) กับความเข้มแม่เหล็ก (H) ซึ่งความสัมพันธ์นี้จะเกิด ฮีสเทอรีซิสลูป (Hysteresis loop)



(a) magnetic head (b) hysteresis loop

รูปที่ 2.8 แสดงฮีสเทอรีซิสที่เกิดขึ้นในหัวเทป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูป 2.8 (a) แสดงโครงสร้างเบื้องต้นของหัวเทปในลักษณะวงจรแม่เหล็กเมื่อคิดเป็นวงจรแม่เหล็ก จากทฤษฎีของเคอร์ชอร์ฟกล่าวไว้ว่า " ผลรวมของกระแสทั้งหมดของวงจรเท่ากับศูนย์ "

$$\sum I = 0$$

$$\sum H.dl = I = 0$$

$$H_m * l_m + H_g * g = 0$$

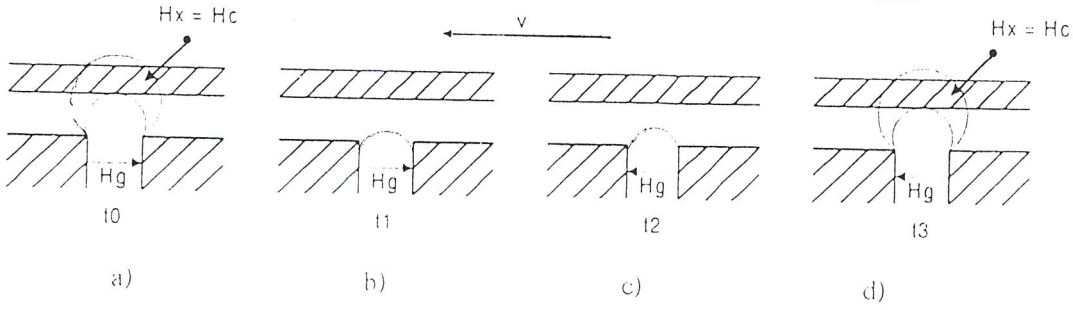
$$H \approx H_m = - H_g * (g/l_m)$$

เมื่อ H_g เป็นความเข้มสนามแม่เหล็กที่เกิดในช่องว่าง (Gap)

g เป็นความยาวของช่องว่าง (Gap) เมื่อ $g << l_a$ สามารถตัดทิ้งได้

l_a เป็นเส้นทางเดินของเส้นแรงแม่เหล็กในอากาศในระยะทางครึ่งวงกลม

จากรูป 2.8 (b) แสดงความสัมพันธ์ระหว่างความหนาแน่นของสนามแม่เหล็ก (B) กับความเข้มของสนามแม่เหล็ก (H) ในสารเฟอร์โรแมกเนติดที่เดิมไม่มีอำนาจแม่เหล็กอยู่เลยจะพบว่าเมื่อเพิ่มค่า H ขึ้นเรื่อยๆในช่วงแรกๆ B จะเพิ่มขึ้นอย่างรวดเร็ว แต่เมื่อ H มีค่าสูงขึ้น อัตราการเพิ่มของ B จะมีค่าน้อยและแทบจะไม่เพิ่มเลยคือ H เริ่มอิ่มตัว (Saturate) ณ จุด a ให้ค่า B มีค่าสูงสุดด (B_{max}) จากนั้นถ้าลดค่า H ลงจนกระทั่ง H = 0 ณ จุด b ช่วงนี้ค่า B จะยังไม่เท่ากับศูนย์เนื่องจากขนาดของความหนาแน่นสนามแม่เหล็กยังคงมีเหลืออยู่ (Retentivity or Remanance) ในสารแม่เหล็ก เรียกว่า B_r ต่อมาเปลี่ยนทิศทางในการป้อนกระแสไฟฟ้า คือ เพิ่มค่า H ในทางลบโดยค่อยๆเพิ่ม H ไปเรื่อยๆจน $H = - H_c$ ณ จุด c ทำให้ B มีค่าเท่ากับศูนย์ จากนั้นเพิ่มค่า H ในทางลบไปเรื่อยๆอีก ($H < -H_c$) จนกระทั่งเกิดการอิ่มตัวในทางลบอีกครั้งหนึ่ง ($-H_{sat}$) ณ จุด d ทำให้ B มีค่าสูงสุดทางลบอีกครั้ง ($-B_{max}$) ต่อจากนั้นเพิ่มค่า H จนกระทั่งถึงจุด e ทำให้ $B = -B_r$ และเพิ่ม H ขึ้นอีกเรื่อยๆจนกระทั่งถึงจุด f ถึงจะได้ $H = H_c$ โดย B = 0 จากนั้นถ้าต้องการให้ $B = B_{max}$ ก็เพิ่ม H ให้เท่ากับ H_{sat} อีกครั้งจะเป็นวงรอบของความสัมพันธ์เช่นนี้เรื่อยๆ วงของความสัมพันธ์ของ B-H นี้เรียกว่า ฮีสเทอรีซิสลูป (Hysteresis loop)



รูปที่ 2.9 แสดงการบันทึกในช่วงอิ่มตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูป 2.9 (a) ในช่วง t(0) เมื่อให้สนามที่หัวเทปอยู่ที่ระดับสูงสุด (Peak level) ในช่วงนี้เกิดความเข้มสูงสุด (High field) ของตัวกลางซึ่งได้แก่เส้นเทป ขณะที่เส้นเทปเคลื่อนไปด้วยความเร็ว (v) ในทิศทางที่แสดง ที่ t(1) H_g มีค่าเข้าใกล้ศูนย์จะทำให้เส้นเทปคงความเป็นแม่เหล็กในช่วง B_r ที่ t(2) กระแสและ H_g ที่ช่องว่างจะเริ่มเพิ่มขึ้นในทิศทางตรงกันข้ามและที่ t(3) จะเกิดความเข้มสูงสุดอีกครั้งในทิศทางตรงกันข้ามกับช่วง t(1) ดังแสดงในรูปที่ 2.9 (d)

2.3 หลักการเขียนข้อมูลลงบนแถบแม่เหล็ก

ในการเขียนข้อมูลลงบนแถบแม่เหล็กนั้น ข้อมูลจะมีลักษณะเป็นลอจิก 0 และ 1 ซึ่งจะต้องทำการแปลงให้อยู่ในรูปของกระแสเพื่อป้อนให้กับขดลวดในหัวอ่าน อันจะก่อให้เกิดการเปลี่ยนทิศทางของแมกเนไตเซชัน (Magnetization) ต่างๆกันตามทิศทางของกระแสที่ป้อน วิธีแปลงข้อมูลให้อยู่ในรูปของกระแสนี้เราเรียกว่า การเข้ารหัสทางแชนแนล (Channel coding) หรือ การเข้ารหัสมอดูเลชัน (Modulation coding) โดยกระบวนการในการเขียนสามารถอธิบายคร่าวๆได้ ดังนี้

1. ข้อมูลที่ต้องการจะเขียนลงแถบแม่เหล็กจะถูกนำมาผ่านวงจรเข้ารหัสเพื่อแปลงให้อยู่ในรูปของกระแส

2. วงจรเข้ารหัส จะทำการส่งกระแสที่ได้ไปยังหัวเขียน

3. การเปลี่ยนทิศของกระแสในขดลวด มีผลทำให้เกิดการกลับทิศการเรียงตัวของแมกเนไตเซชันตามลักษณะของข้อมูลที่เข้ารหัสเอาไว้

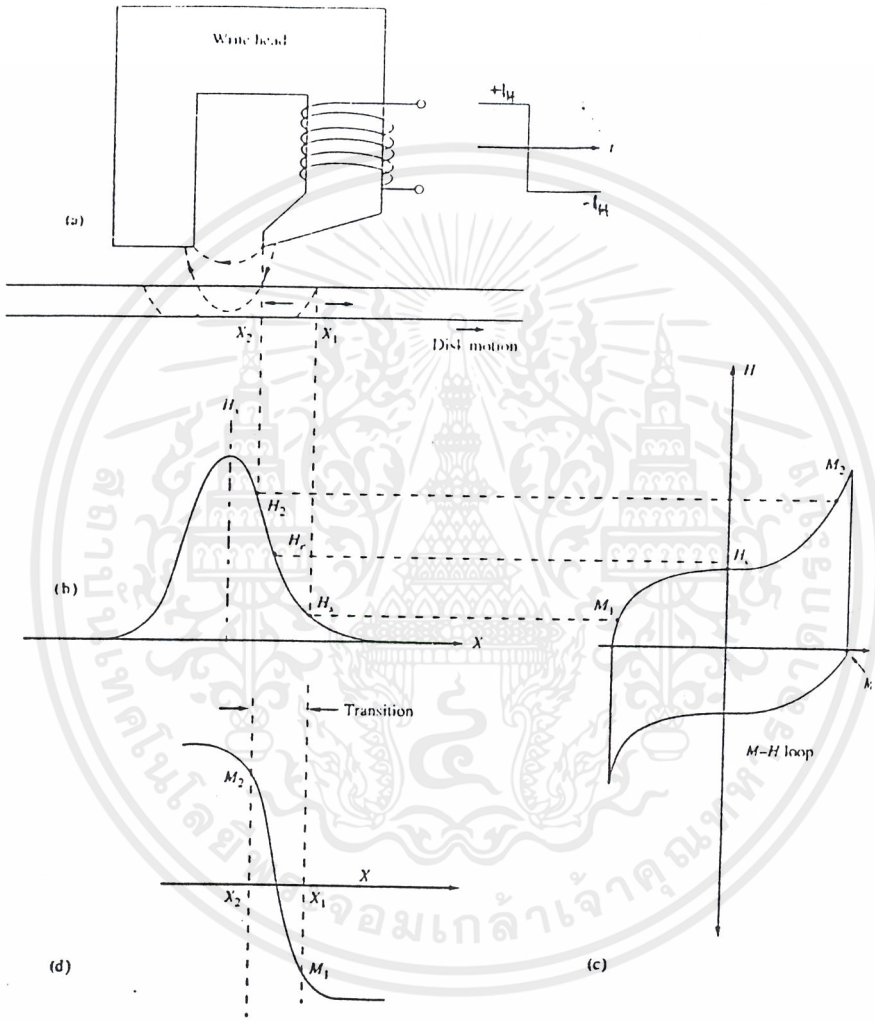
เมื่อทำการเขียนข้อมูลซึ่งอยู่ในรูปดิจิตอลลงบนแถบแม่เหล็ก สารแม่เหล็กจะถูกเหนี่ยวนำให้เกิดการอิ่มตัวในทิศทางใดทิศทางหนึ่ง ซึ่งถ้าสนามแม่เหล็กที่หัวอ่านสร้างขึ้นมานี้มีขนาดมากเพียงพอที่จะทำให้เกิดการเรียงตัวใหม่ของแมกเนไตเซชันทั้งหมด เราจะเรียกว่า การเขียนทับ (over writing)

พิจารณาจากรูป 2.10 จะเห็นว่าสารแม่เหล็กมีการเรียงตัวของแมกเนไตเซชันตามทิศทางของลูกศรในรูป 2.10 (a) จากนั้นกระแสจะถูกป้อนให้กับหัวอ่านเพื่อจะกลับทิศทางการเรียงตัวของแมกเนไตเซชัน จากรูป 2.10(b) จะเห็นว่า ขนาดของสนามแม่เหล็กมีค่าเปลี่ยนแปลงตามระยะทางจาก x_1 ถึง x_2 โดยที่ จุด x_1 สนามจะมีค่าอ่านที่สุด ในขณะที่จุด x_2 สนามจะมีความแรงมากที่สุด พิจารณาจากรูป 2.10(b) สนามที่จุด x_1 มีขนาด H_1 และที่จุด x_2 มีขนาด H_2 สนามนี้จะทำให้เกิดแมกเนไตเซชัน M_1 และ M_2 ลงบนแถบแม่เหล็ก ตามลำดับ ดังแสดงใน M-H Loop (รูป 2.10(c)) แมกเนไตเซชัน M_1 ที่จุด x_1 มีขนาดน้อยมากจึงไม่มีผลต่อการเปลี่ยนทิศทางของแมกเนไตเซชันมากเท่าใดนัก แต่ที่จุด x_2 ซึ่ง M_2 มีค่าเกือบถึงจุดอิ่มตัวทำให้แมกเนไตเซชันมีการกลับทิศทางการเรียงตัวใหม่เกือบทั้งหมด เมื่อทำการพลอตค่าแมกเนไตเซชันบนแถบแม่เหล็กเทียบกับระยะ x จะได้ดังรูป 2.10(d) จากกราฟเราจะได้ข้อสังเกตดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. จากรูป 2.10(b) ถ้ากราฟมีค่าความชันมากขึ้น จะทำให้ช่วงการเปลี่ยนสถานะของแมกเนไตเซชันน้อยลง

2. จากรูป 2.10(c) ความชันของกราฟ M-H มีผลต่อช่วงการเปลี่ยนสถานะของแมกเนไตเซชัน



รูปที่ 2.10 แสดงกราฟต่างๆในกระบวนการเขียนข้อมูลลงบนแถบแม่เหล็ก

2.4 หลักการอ่านข้อมูลจากแถบแม่เหล็ก

พิจารณาจากเมื่อเราทำการเขียนข้อมูลลงบนแถบแม่เหล็ก ข้อมูลเหล่านั้นก็จะถูกแปลงให้อยู่ในรูปของกระแสป้อนให้กับหัวเขียน ตามวิธีการเข้ารหัสแบบต่างๆ (ดังจะได้กล่าวต่อไป) การกลับทิศทางไปมาของกระแสนี้ จะมีผลต่อสนามที่แก็ปของหัวเขียนมีการกลับทิศด้วย ซึ่งจะทำให้การเรียงตัวของแมกเนไตเซชันในสารแม่เหล็กมีการเปลี่ยนแปลงไปด้วย เมื่อจะทำการอ่านข้อมูลออกมา

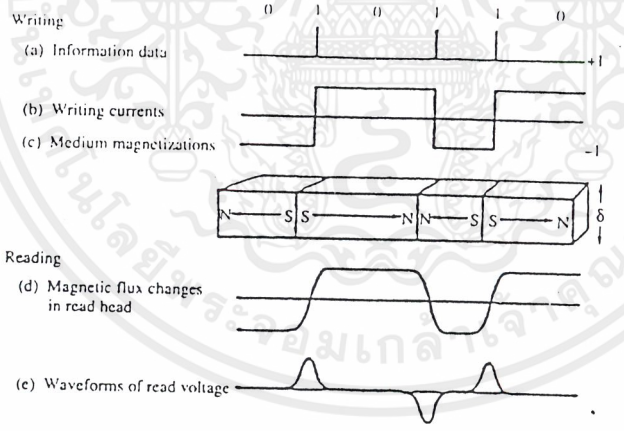
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากแถบแม่เหล็ก เราจะต้องเปลี่ยนวงจรเข้ารหัสข้อมูลซึ่งต่ออยู่กับหัวเขียน ไปเป็นวงจรพรีแอมพลิไฟร์เพื่อขยายสัญญาณที่ได้จากหัวอ่าน โดยหลักการทั่วไปในการอ่านข้อมูลจากแถบแม่เหล็กมีดังนี้

1. เมื่อแถบแม่เหล็กเคลื่อนที่ผ่านหัวอ่าน จะเกิดการเหนี่ยวนำให้เกิดฟลักซ์ขึ้นที่หัวอ่านนั้นตามทิศทางของแมกเนไตเซชันบนแถบแม่เหล็ก

2. การเปลี่ยนทิศทางไปมาของฟลักซ์แม่เหล็กที่หัวอ่านจะทำให้มีแรงดันเกิดขึ้น ซึ่งเราจะนำแรงดันนี้ไปใช้ประโยชน์ในการอ่านข้อมูลต่อไป

จะสังเกตได้ว่าขณะที่หัวอ่านเคลื่อนที่ไป แมกเนไตเซชันที่จุดนั้นจะทำให้เกิดฟลักซ์แม่เหล็กขึ้นที่หัวอ่าน จากกฎของฟาราเดย์กล่าวไว้ว่า $V = d\phi/dt$ นั่นคือเมื่อเกิดการเปลี่ยนแปลงฟลักซ์แม่เหล็กเทียบกับเวลา จะเหนี่ยวนำให้เกิดแรงดันขึ้น จากรูป 2.11(a), 2.11(b) แสดงถึงการเข้ารหัสข้อมูลเพื่อเขียนลงไปในแถบแม่เหล็ก วิธีที่แสดงดังในภาพคือ กระแสจะมีการกลับทิศทุกครั้งเมื่อข้อมูลเปลี่ยนจากศูนย์เป็นหนึ่ง ซึ่งผลของการเขียนจะทำให้เกิดการเรียงตัวของแมกเนไตเซชันดังแสดงในรูป 2.11(c) ส่วนในด้านการอ่าน พบว่าเมื่อหัวอ่านเคลื่อนที่ผ่านจุดที่เป็นแถบเปลี่ยนสถานะ(ดังรูป 2.11(d)) การเปลี่ยนแปลงของแมกเนไตเซชันนี้จะเหนี่ยวนำให้เกิดแรงดันค่าน้อยๆขึ้นที่หัวอ่าน แรงดันนี้จะถูกนำไปผ่านวงจรขยายและนำไปผ่านวงจรถอดรหัสข้อมูลต่อไป



รูปที่ 2.11 กราฟแสดงการอ่านและเขียนข้อมูลจากแถบแม่เหล็ก

2.5 การเข้ารหัสข้อมูลในบัตรแม่เหล็ก

ในระบบคอมพิวเตอร์ มีวิธีมากมายในการเข้ารหัสข้อมูล " 1" และ " 0 " แต่ในส่วนของบัตรแม่เหล็กแล้ว ISO ได้กำหนดมาตรฐานในการเข้ารหัสข้อมูลไว้โดยจะใช้หลักการ F2F

F2F ย่อมาจาก Frequency – double frequency ซึ่งวิธีนี้จะมีข้อดีคือ สามารถทำการ self clocking ได้ นั่นคือ เราสามารถอ่านค่าบิตข้อมูลต่างๆจากบัตรแม่เหล็กได้โดยไม่ขึ้นอยู่กับความเร็วในการเคลื่อนที่ของบัตรผ่านหัวเทป ด้วยเหตุนี้ ทำให้มนุษย์ทั่วไปสามารถทำการรูดบัตรผ่านเครื่องอ่านได้ไม่ว่าจะรูดช้าหรือเร็วก็ตาม ในการเข้ารหัสแบบ F2F นี้ มีหลักการคือ จะมีการกำหนดระยะห่างระหว่าง

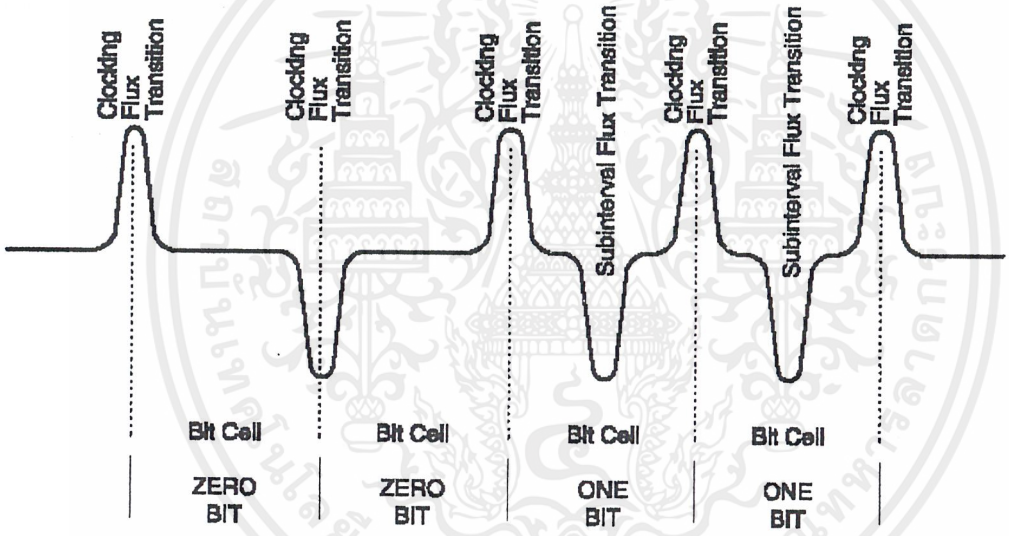
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จุดที่มีการเปลี่ยนทิศทางของฟลักซ์บนแถบแม่เหล็กให้มีค่าคงที่ค่าหนึ่ง ซึ่งระยะห่างนี้จะใช้ในการบันทึกข้อมูลที่เป็น 0 ส่วนข้อมูลที่เป็น 1 นั้น ระยะห่างระหว่างจุดที่มีการเปลี่ยนแปลงทิศทางของฟลักซ์นั้นจะมีความยาวเป็นครึ่งหนึ่งของที่กำหนดไว้สำหรับข้อมูลที่เป็น 0 (นั่นคือกำหนดให้ฟลักซ์ที่ออกจากหัวเทปเป็น 1 เท่าของความถี่และ 2 เท่าของความถี่นั่นเอง)

ในแทร็คที่ 1 และ 3 ค่าคงที่ที่กำหนดไว้สำหรับการบันทึกข้อมูล 0 คือ 0.0047619 นิ้ว หรืออาจกล่าวว่ามีความหนาแน่นของบิตเป็น 210 บิตต่อนิ้ว ก็ได้ สำหรับข้อมูลที่เป็น 1 นั้นจะมีระห่างระหว่างจุดเปลี่ยนทิศทางของฟลักซ์เป็น 0.0023809 นิ้ว (เป็นครึ่งหนึ่งของการบันทึกข้อมูล 0 นั่นเอง)

ในแทร็คที่ 2 จะมีค่าคงที่ที่กำหนดสำหรับการบันทึกข้อมูล 0 คือ 0.013333 นิ้ว หรือ 75 บิตต่อนิ้วนั่นเอง ในทำนองเดียวกันการบันทึกข้อมูล 1 จะมีค่าระยะทางเป็น 0.0066665 นิ้ว

หลักการ F2F สามารถแสดงให้เห็นได้ดังรูป



รูปที่ 2.12 แสดงการเปลี่ยนแปลงทิศทางของฟลักซ์ในการเข้ารหัสแบบ F2F

เครื่องที่ใช้ในการเข้ารหัสข้อมูลนั้นจะต้องสามารถกำหนดระยะของของจุดเปลี่ยนแปลงทิศทางของฟลักซ์ให้ได้อย่างแม่นยำเพื่อมิให้เกิดความผิดพลาดในการอ่านข้อมูลกลับภายหลัง

ก่อนที่จะทำการเข้ารหัสข้อมูลอักขระตัวแรก เราจะต้องทำการเข้ารหัสข้อมูลที่บอกจุดเริ่มต้นก่อน (Start sentinel) ในแทร็ค 1 SS จะมี 7 บิต ส่วนในแทร็ค 2 และ 3 จะมีเพียงแค่ 5 บิต (เหมือนเช่นตัวอักขระทั่วๆไปในแทร็ค 2) ตัวอย่างเช่นในแทร็ค 2 SS คือ 01011

หลังจากที่ตัวอักขระข้อมูลตัวสุดท้ายถูกเข้ารหัสบนบัตรแม่เหล็กแล้ว ก็จะมีการเข้ารหัสข้อมูลที่บอกจุดสิ้นสุดของข้อมูล (End sentinel) และหลังจาก ES จะทำการเข้ารหัสข้อมูลที่ใช้ในการตรวจสอบข้อผิดพลาด (LRC) ดังได้กล่าวรายละเอียดแล้วในตอนต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระยะก่อนถึง ES จะมีการใส่ข้อมูลที่เป็น 0 ทั้งหมดเอาไว้ (clocking bits) ทั้งนี้เพื่อมีประโยชน์ในการหา ES ได้ง่ายขึ้น เนื่องจาก ES มีรหัสคือ 01011 ซึ่งดังที่เคยกล่าวมาแล้วว่าการรับข้อมูลจากเครื่องอ่านเข้ามานั้นจะทำในทิศทางกลับกัน นั่นคือจะรับข้อมูลที่เป็นเลข 1 เข้ามาก่อน ดังนั้นเมื่อใดที่มีการรับข้อมูลที่เป็นเลข 1 เข้ามาเป็นครั้งแรก นั่นแสดงว่าเราตรวจพบ ES แล้วนั่นเอง ดังนี้

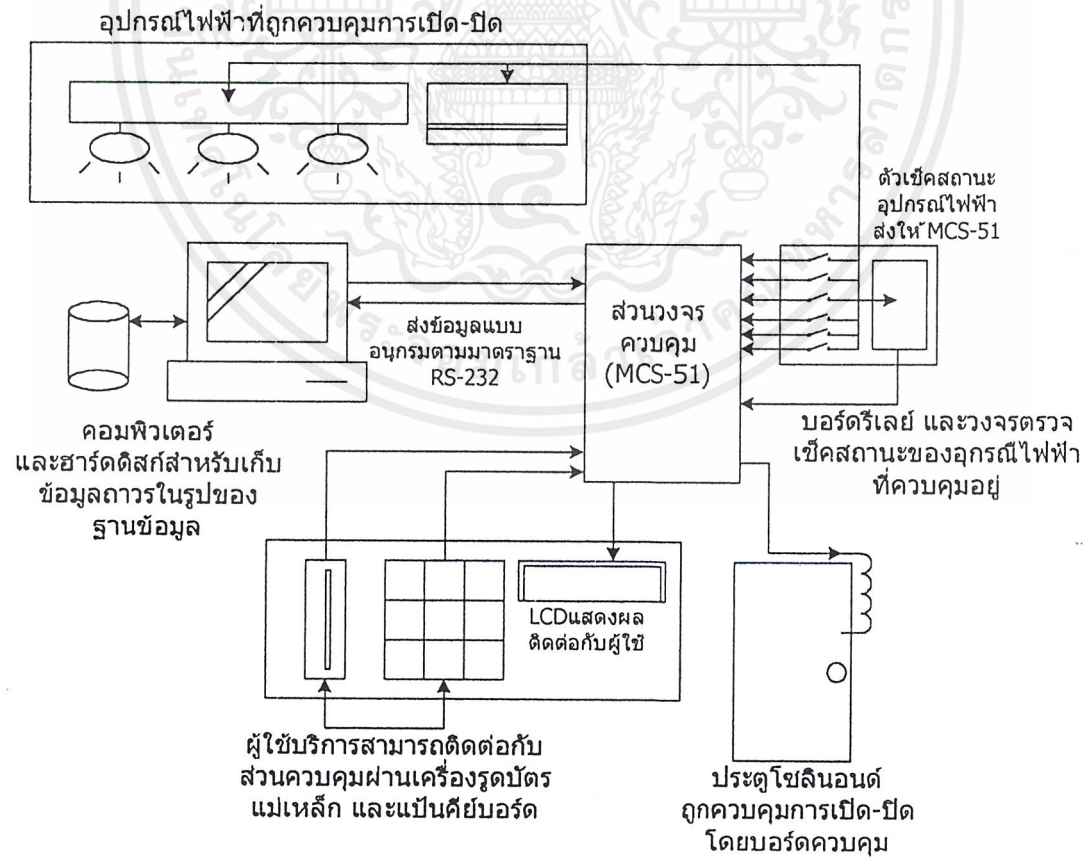
0000000000011010..... DATA……11111(LRC)000000000000

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

หลักการสร้างและการออกแบบ

หลักการออกแบบโครงงานจะเป็นการติดต่อกันระหว่างคอมพิวเตอร์กับไมโครคอนโทรเลอร์(MCS-51) โดยใช้การส่งข้อมูลสื่อสารกันระหว่างอุปกรณ์ทั้งสองตัวซึ่งจะทำงานสลับกันเป็นอุปกรณ์ต้นทางและอุปกรณ์ปลายทางโดยผู้ใช้สามารถป้อนข้อมูลได้ทั้งสองทาง คือ ควบคุมจากทางด้านคอมพิวเตอร์ภายในตัวอาคารได้โดยตรงเป็นในลักษณะผู้ควบคุมดูแลระบบในการจัดการฐานข้อมูล หรือผ่านทางแป้นคีย์บอร์ดและเครื่องอ่านข้อมูลจากบัตรแม่เหล็กที่ติดอยู่กับประตูนอกตัวอาคารซึ่งเป็นการสั่งงานให้กับไมโครคอนโทรลเลอร์ในรูปแบบของผู้ใช้บริการเพื่อให้ทำการติดต่อดึงข้อมูลจากคอมพิวเตอร์อีกต่อหนึ่งและทำการประมวลผลควบคุมอุปกรณ์อื่น ๆ เช่น ประตูเปิด-ปิด หรือ อุปกรณ์ไฟฟ้าภายในอาคาร ซึ่งจำเป็นจะต้องมีการเขียนโปรแกรมควบคุมอุปกรณ์ทั้งฝ่ายส่งและฝ่ายรับที่ดีพอเพื่อไม่ให้เกิดการผิดพลาดในการสื่อสารกันขึ้น



รูปที่ 3.1 แสดงแผนภาพการทำงานของอุปกรณ์ทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 ลักษณะโครงสร้างของระบบ

โครงสร้างของระบบที่ควบคุมโดยรวมจะประกอบไปด้วย การควบคุมในส่วนของฮาร์ดแวร์ ซึ่งประกอบไปด้วยวงจรในส่วนควบคุม(MCS-51) ส่วนของการส่งสัญญาณติดต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์โดยผ่านพอร์ทอนุกรม และบอร์ดควบคุมอุปกรณ์ไฟฟ้า และสำหรับในส่วนของซอฟต์แวร์ ซึ่งประกอบไปด้วยโปรแกรมภาษาแอสเซมบลี้ที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์ โปรแกรมควบคุมการทำงานของคอมพิวเตอร์และการส่งข้อมูลผ่านพอร์ทอนุกรม รวมทั้งส่วนที่เป็นฐานข้อมูลใช้สำหรับเก็บข้อมูล

3.2 การออกแบบทางด้านฮาร์ดแวร์

3.2.1 <u>การใช้งานเครื่องอ่านบัตรแม่เหล็กร่วมกับ Microcontroller</u>

เครื่องอ่านบัตรแม่เหล็กที่ใช้สำหรับอ่านข้อมูลจากบัตรแม่เหล็กในโครงงานชิ้นนี้สามารถใช้งานร่วมกับไมโครคอนโทรลเลอร์ได้ในหลายรูปแบบ ขึ้นอยู่กับความสามารถของ MPU(Microprocessor Unit) และลักษณะของระบบที่เราใช้งานด้วย โดยทั่วไปจะมีการใช้งานใน 2 ลักษณะดังต่อไปนี้ คือ

- การใช้งานในลักษณะรับข้อมูลทีละบิต (Single bit input programming)
- การใช้งานผ่านทาง USART (Universal Synchronous/Asynchronous Receiver – Transmitter)

3.2.1.1 <u>การใช้งานในลักษณะรับข้อมูลทีละบิต (Single bit input programming)</u>

การใช้งานในรูปแบบนี้จะไม่จำเป็นต้องมีวงจรภายนอกเพื่อช่วยในการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก การรับข้อมูลจะกระทำโดยใช้โปรแกรมที่เขียนขึ้น ซึ่งการเขียนโปรแกรมจะต้องมีความแน่นอนในเรื่องของเวลาในการรับข้อมูลด้วย ทำให้เกิดข้อเสียคือระหว่างมีการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก ไมโครคอนโทรลเลอร์จะไม่สามารถทำงานอย่างอื่นได้ ทั้งนี้เพื่อให้ข้อมูลที่รับได้มีความถูกต้องและแน่นอน ในโปรเจคนี้จะทำการรับข้อมูลจากเครื่องอ่านด้วยวิธีนี้

3.2.1.2 <u>ใช้งานผ่านUSART(Universal Synchronous/Asynchronous Receiver–Transmitter)</u>

จะเป็นการใช้ USART ช่วยในการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก การใช้งานในรูปแบบนี้จะมีข้อได้เปรียบคือ ช่วยลดปัญหาเรื่องความแน่นอนของเวลาในการรับข้อมูลได้ โดยการใช้งานจะ ต่อสัญญาณ CARD PRESENT เข้ากับช่องสัญญาณ DSR ของ USART ส่วนข้อมูลจะถูกรับผ่านทาง RXD ของ USART และจะมีสัญญาณ STROBE ซึ่งใช้ควบคุมการรับข้อมูลโดยต่อผ่านทาง RXC ของ USART จะเห็นว่า USART จะกำหนดการรับข้อมูลเอง ทำให้ข้อมูลที่ได้มีความถูกต้องมากกว่าการเขียนโปรแกรมเพื่อรับข้อมูลด้วยตนเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

3.2.2 หลักการและวิธีการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก

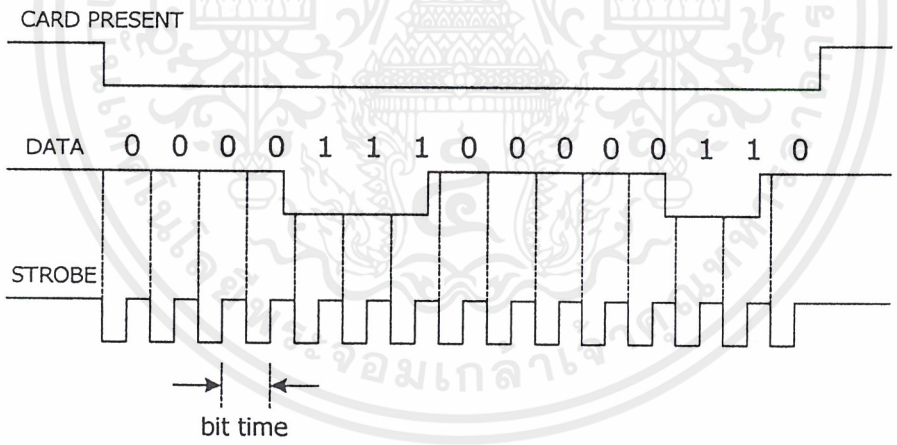
3.2.2.1 การอ่านข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก เครื่องอ่านบัตรแม่เหล็กจะมีสายสัญญาณที่สำคัญอยู่ 3 เส้น คือสายสัญญาณข้อมูล (DATA) , สายสัญญาณควบคุมการรับข้อมูล (STROBE) และ สายสัญญาณตรวจสอบว่ามีบัตรเข้ามาหรือไม่ (CARD PRESENT) โดยการรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็กสามารถทำได้ดังนี้

1) ต่อสายสัญญาณSTROBE จากเครื่องอ่านบัตรแม่เหล็กเข้ากับพอร์ทของไมโครคอนโทรลเลอร์ซึ่งสามาถทำการเขียนโปรแกรมเพื่อตรวจจับสัญญาณนาฬิกาขาลงได้ (Negative-edge)

2) ต่อสายสัญญาณ DATA จากเครื่องอ่านบัตร เข้ากับพอร์ทใดพอร์ทหนึ่งของไมโครคอนโทรลเลอร์

3) ต่อสายสัญญาณ CARD PRESENT ซึ่งทำหน้าที่ตรวจสอบว่ามีการผ่านบัตรเข้ามาที่หัวอ่านหรือไม่ เข้ากับอีกพอร์ทหนึ่งของไมโครคอนโทรลเลอร์

โดยลักษณะสัญญาณต่างๆเมื่อทำการอ่านข้อมูลจะเป็นดังนี้



รูปที่ 3.2 แสดงสัญญาณต่างๆที่อ่านได้จากเครื่องอ่านบัตรแม่เหล็ก

โดยสายสัญญาณแต่ละเส้นมีความหมายดังนี้

1) สัญญาณ DATA เป็นสายสัญญาณของข้อมูลจากบัตรแม่เหล็ก ข้อมูลที่อ่านได้จากสายสัญญาณนี้ก็คือข้อมูลจากบัตรแม่เหล็กนั่นเอง

2) สัญญาณ STROBE เป็นสายสัญญาณที่ควบคุมการอ่านข้อมูลจากสายสัญญาณ DATA โดยเราจะสามารถอ่านข้อมูลได้เมื่อสัญญาณ STROBE เปลี่ยนสภาวะจาก high เป็น low (Negative edge)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

42755

3) สัญญาณ CARD PRESENT เป็นสายสัญญาณที่บอกให้รู้ว่าขณะนี้มีบัตรผ่านเข้ามายังเครื่องอ่านหรือไม่ โดยสัญญาณนี้จะเปลี่ยนสภาวะเป็น low เมื่อตรวจพบว่ามีการเปลี่ยนแปลงฟลักซ์ที่หัวอ่าน จากนั้นจะเปลี่ยนสถานะกลับไปเป็น high อีกครั้งหลังจากที่บัตรเคลื่อนที่พ้นหัวอ่านไปแล้วประมาณ 150 ms

ในสภาวะปกติที่ยังไม่มีการรูดบัตร สัญญาณทั้งสามช่องจะมีสถานะเป็น high (+Vcc) แต่เมื่อมีการรูดบัตรแม่เหล็ก สัญญาณ CARD PRESENT จะเปลี่ยนสถานะเป็นlow หลังจากนั้นก็จะมีสัญญาณนาฬิกาปรากฏที่ช่องสัญญาณ STROBE ใช้ในการตรวจจับเพื่ออ่านข้อมูล โดยจะทำการอ่านข้อมูลทุกครั้งที่สัญญาณ STROBE มีสภาวะเป็นเปลี่ยนจาก high เป็น low ข้อมูลที่อ่านได้นั้นจะมีลักษณะกลับกับความเป็นจริง กล่าวคือ ถ้าอ่านค่าจากสัญญาณ DATA ได้สภาวะ low แสดงว่าข้อมูลจริงมีค่าเป็น 1 เมื่ออ่านข้อมูลได้หนึ่งตัวแล้วก็จะมีการนำข้อมูลนั้นไปเก็บไว้ในหน่วยความจำก่อนที่จะเริ่มรับข้อมูลตัวต่อไป และจะทำการรับข้อมูลไปเรื่อยๆจนกว่าสัญญาณ STROBE จะหยุดไปเนื่องจากไม่มีแถบแม่เหล็กเคลื่อนผ่านหัวอ่าน และสัญญาณ CARD PRESENT จะกลับสู่สภาวะ high เหมือนเดิม เมื่อตรวจพบดังนี้แล้วก็จะทำการนำข้อมูลที่อ่านได้ทั้งหมดมาตรวจว่ามีข้อผิดพลาดเกิดขึ้นหรือไม่โดยจะตรวจจากบิตพาริตี้ และLRC(Longitudinal Redundancy Check) ถ้าหากไม่ตรวจพบข้อผิดพลาดใดๆก็จะทำการเปลี่ยนข้อมูลให้อยู่ในรูปที่สามารถนำไปใช้ได้ เช่น เปลี่ยนข้อมูลให้อยู่ในรูปของ ASCII (American Standard Code for Information Interchange)

ในการเก็บข้อมูลที่อ่านได้ลงในหน่วยความจำนั้นมีวิธีการอยู่ 2 วิธี วิธีการแรกเราจะเก็บข้อมูลที่อ่านได้ในแต่ละครั้งไว้ในหน่วยความจำ 1 ไบต์ ซึ่งวิธีนี้จะมีข้อดีคือสามารถเขียนโปรแกรมเพื่อจัดการกับข้อมูลได้ง่าย แต่จะมีข้อเสียคือ อาจมีบางบิตในหน่วยความจำไม่ถูกใช้ อาจเนื่องมาจากข้อมูลที่อ่านได้ในแต่ละครั้งมีขนาดไม่ถึง 8 บิต (1 ไบต์) และด้วยการเก็บข้อมูลแบบนี้เราจะไม่สามารถอ่านข้อมูลในทิศทางย้อนกลับได้ อีกวิธีที่ใช้ในการเก็บข้อมูลคือจะทำการเก็บข้อมูลที่อ่านได้เรียงไปเรื่อยๆในหน่วยความจำ (วิธีนี้จะทำให้ไม่มีบิตที่ไม่ถูกใช้ในหน่วยความจำเลย) แต่ข้อเสียของวิธีนี้ก็คือการเขียนโปรแกรมเพื่อแยกข้อมูลแต่ละตัวนั้น จะทำได้ยากในกรณีที่ข้อมูลที่อ่านได้แต่ละครั้งมีขนาดไม่เท่ากับ 8 บิต (1 ไบต์) เช่นข้อมูลในแทร็ค 2 ของบัตรแม่เหล็กจะประกอบด้วย 5 บิต ดังนั้นข้อมูลที่อ่านได้ในแต่ละครั้งจะมี 5 บิต เมื่ออ่านข้อมูลครั้งแรก ข้อมูล 5 บิต นี้จะถูกเก็บไว้ในไบต์แรกของหน่วยความจำ ซึ่งจะมีพื้นที่เหลืออยู่ 3 บิต จากนั้นเมื่อเราอ่านข้อมูลชุดที่สองจากบัตรแม่เหล็ก ก็จะนำข้อมูล 3 บิตมาเก็บไว้ในหน่วยความจำไบต์แรกส่วนที่เหลือและนำข้อมูลอีก 2 บิตไปเก็บไว้ใน 2 บิตแรกของหน่วยความจำไบต์ที่สองต่อไป จะเห็นว่าเป็นการยากในการเขียนโปรแกรมเพื่อแยกข้อมูลแต่ละชุดออกจากกัน อย่างไรก็ตามไม่ว่าจะใช้วิธีใด ข้อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มูลที่เราอ่านได้นั้นจะมีลักษณะกลับกับในความเป็นจริง นั่นคือ ถ้าข้อมูลเป็น 1011 ในการอ่านข้อมูลจริงๆจะอ่านได้ในลักษณะนี้ คือ 1 – 1 – 0 – 1 ตามลำดับ

3.2.2.2 <u>หลักในการเขียนโปรแกรมเพื่อรับข้อมูลจากเครื่องอ่าน</u>

เริ่มแรกเราจะทำการเขียนโปรแกรมเพื่อตรวจจับสัญญาณ CARD PRESENT ก่อน เมื่อใดที่สัญญาณนี้เปลี่ยนสถานะเป็น low จะเริ่มตรวจสอบสัญญาณ STROBE ต่อไป โดยเมื่อใดที่สัญญาณ STROBE เปลี่ยนจากสถานะ high เป็น low เราจะทำการอ่านข้อมูลจากสายสัญญาณ DATA ช่วงตอนต้นของบัตร ข้อมูลที่อ่านได้จะมีค่าเป็น 0 ทั้งหมดเนื่องจากเป็น leading clocking zeros (ดังได้กล่าวไว้ในทฤษฎีตอนต้น) ในขณะที่ตัวอักขระแรกที่เราจะอ่านได้ก็คือ SS (Start sentinel) ซึ่งมีค่า 01011 จากที่เคยกล่าวไว้ในตอนต้นแล้วว่าการอ่านข้อมูลจากเครื่องอ่านนั้น เราจะได้ข้อมูลดังนี้คือ 1 – 1 – 0 – 1 – 0 ดังนั้นเมื่อเราอ่านข้อมูลเข้ามาแล้วจะต้องทำการตรวจสอบว่าเมื่อใดที่ข้อมูลเปลี่ยนจาก 0(ซึ่งเป็น leading clocking zeros) เป็น 1(บิตแรกที่ของ SS ที่เราอ่านได้) แสดงว่าเราตรวจพบ SS แล้ว ก็จะทำการเก็บค่าข้อมูล 1 นั้นไว้ในหน่วยความจำและทำการรับข้อมูลบิตถัดไปมา จนครบ 5 บิต(1 อักขระในแทร็คที่ 2 จะมีข้อมูล 5 บิต) แล้วก็จะทำการเลื่อนตำแหน่งของหน่วยความจำออกไปเพื่อเก็บข้อมูลอักขระถัดไป ตรวจสอบจนรับครบ 40 ตัวอักขระแล้วก็จะทำการหยุดรับค่าจากบัตร เนื่องจากในแทร็คที่ 2 จะมีข้อมูลแค่ 40 ตัวอักษรเท่านั้น เมื่อรับข้อมูลครบแล้วก็จะทำการตรวจสอบความผิดพลาดของข้อมูลที่รับมา โดยจะทำการตรวจสอบพาริตี้บิต และ LRC ในการตรวจสอบบิตพาริตี้ จะใช้แฟลกพาริตี้ในไมโครคอนโทรลเลอร์เป็นตัวตรวจสอบ ซึ่งข้อมูลในบัตรแม่เหล็กจะเป็นพาริตี้คี่ เมื่อตรวจสอบพาริตี้ผ่านแล้ว ก็จะทำการตรวจสอบ LRC ต่อไปโดยนำข้อมูลทั้งหมดมา Exclusive Or กัน แล้วนำผลลัพธ์ที่ได้มาเทียบกับอักขระ LRC ที่อ่านได้จากบัตรแม่เหล็กว่าตรงกันหรือไม่ (ในที่นี้จะไม่ทำการตรวจสอบบิตพาริตี้ของ LRC) หากตรงกันแสดงว่าข้อมูลที่รับมาครั้งนั้นถูกต้อง ก็จะนำข้อมูลที่ได้นั้นไปแปลงเป็นรหัส ASCII (ตารางในการแปลงแสดงไว้ในภาคผนวก) เพื่อที่เมื่อส่งผ่านทางพอร์ท อนุกรมของคอมพิวเตอร์แล้วจะสามารถตีความเป็นตัวอักษรที่ถูกต้องได้ (ในที่นี้ใช้รหัส ASCII ที่เป็นพาริตี้คู่)

3.2.3 <u>การออกแบบส่วนของวงจรควบคุม</u>

ส่วนควบคุมหลักเป็นส่วนที่สำคัญเป็นการควบคุมการทำงานของอุปกรณ์ทั้งหมด ในโครงงานนี้ได้เลือกใช้ไมโครคอนโทรลเบอร์ 89C51 (MCS-51) ทำการกำหนดค่าสัญญาณต่าง ๆ เพื่อทำการควบคุมอุปกรณ์โดยใช้สัญญาณข้อมูล (Data Bus) และสัญญาณแอดเดรส (Address Bus) ทำการประมวลผลรวมทั้งเป็นตัวสื่อสารและติดต่อกับคอมพิวเตอร์ส่วนบุคคลด้วย โดยใช้การสื่อสารแบบอนุกรมส่งผ่านขา TX และ RX โดยสัญญาณที่ส่งจะเป็นสัญญาณในรูปแบบ TTL (TTL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คือ ที่ระดับสัญญาณลอจิก 1 จะมีระดับแรงดันเป็น 5 โวลท์ และที่ระดับลอจิก 0 จะมีระดับแรงดันเป็น 0 โวลท์)

สำหรับการต่ออุปกรณ์ในวงจรควบคุมจะแสดงให้เห็นดังในรูปที่ 3.23 ซึ่ง MCS-51 จะต่อวงจรในแบบที่ใช้74LS373มาเป็นแลทช์ทำการมัลติเพล็กซ์สลับส่งข้อมูลกับแอดเดรสโดยจะแลทช์แอดเดรสไบต์ต่ำให้กับอุปกรณ์ภายนอกซึ่งก็คือIC เบอร์8255 เพื่อจะทำการขยายจำนวนพอร์ท โดยจะใช้ขาALEจากMCS-51 มาเป็นตัวควบคุมจังหวะการแลทช์ให้กับIC ตัวนี้ เนื่องจากว่าพอร์ท0 นั้นจะใช้งาน2หน้าที่ในการติดต่อกับอุปกรณ์ภายนอกซึ่งไม่สามารถที่จะใช้งานในเป็นแอดเดรสบัสในการอ้างอิงตำแหน่ง และบัสข้อมูลในการส่งและรับข้อมูลกับอุปกรณ์ภายนอกได้ในเวลาเดียวกัน ดังนั้นจึงต้องมี 74LS373 มาทำหน้าที่คงแอดเดรสเอาไว้(Latch) จากนั้นก็จะมีการมีการรับส่งข้อมูลกับอุปกรณ์ภายนอกโดยผ่านทางบัสข้อมูลนั่นเอง ซึ่งจะทำให้เราสามารถใช้งานพอร์ทได้อย่างเต็มประสิทธิภาพ นั่นคือจะสามารถใช้งานพอร์ททั่วไปได้ถึง 5 พอร์ท คือพอร์ท1 พอร์ท2 พอร์ทA , B และ C

ในการออกแบบจำเป็นต้องใช้พอร์ทสำหรับติดต่อกับอุปกรณ์ภายนอกหลายชนิดจำนวนหลายพอร์ทซึ่งพอร์ททีมีอยู่นั้นมีไม่เพียงพอ ดังนั้น จึงต้องมีการต่อไอซีเบอร์8255 เพิ่มอีกตัวเพื่อขยายพอร์ทให้มีจำนวนมากขึ้นอีกกว่าเดิม โดยอาศัยไอซีถอดรหัสเบอร์74HC138 ซึ่งใช้การป้อนอินพุท3ขา แต่จะเลือกให้เอาท์พุทออกได้8ทาง ในที่นี้จะเลือกให้มีเอาท์พุท2ตัวคือ Y0 ต่อกับ8255 ตัวที่1 และY1ต่อกับ8255ตัวที่2 เพื่อเลือกใช้พอร์ทของ8255ทั้ง2ตัวสลับกันไป โดยที่พอร์ทAของ 8255 ทั้ง2ตัวจะมีไอซีเบอร์74HC244 ซึ่งเป็นไอซีจ่ายกระแสเพิ่ม(Driver)ให้กับพอร์ทของ8255ในกรณีที่อุปกรณ์ภายนอกที่ต่อมีการดึงกระแสที่มาก ซึ่งอาจจะทำให้พอร์ทเกิดการเสียหายได้ในกรณีที่ไม่สามารถจ่ายกระแสได้อย่างเพียงพอ ส่วนขา $\overline{WR}$ และ $\overline{RD}$ ก็ต่อเข้ากับ8255ในตำแหน่งชื่อขาเดียวกันทั้ง 2 ตัวเพื่อเป็นสัญญาณป้อนไปบอกว่าต้องการอ่านหรือเขียนข้อมูล

เนื่องจากในการอ่านโปรแกรมจะใช้หน่วยความจำข้อมูล(ROM) ซึ่งเป็นแฟลชเมมโมรีภายในตัวไมโครคอนโทรลเลอร์เองที่มีเพียงพอ โดยไม่มีความจำเป็นต้องอาศัยหน่วยความจำโปรแกรมภายนอก ดังนั้นขา $\overline{EA}$ (External Access) จึงต้องต่อเข้ากับไฟเลี้ยง(เป็นลอจิก1) เพื่อเลือกใช้โปรแกรมจากหน่วยความจำโปรแกรมภายใน และขา PSEN ก็จะปล่อยลอยไว้เนื่องจากไม่มีการเรียกใช้งานจากหน่วยความจำโปรแกรมภายนอก

	Address	P7	P6	P5	P4	P3	P2	P1	P0	
8255 ตัวที่1	พอร์ท A	X	X	X	0	0	0	0	0	=00H
	พอร์ท B	X	X	X	0	0	0	0	1	=01H
	พอร์ท C	X	X	X	0	0	0	1	0	=02H

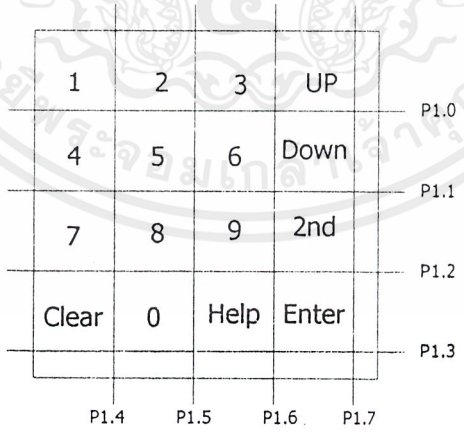
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	Address	P7	P6	P5	P4	P3	P2	P1	P0	
	คอนโทรลเวิร์ด	X	X	X	0	0	0	1	1	= 03H
8255 ตัวที่ 2	พอร์ท A	X	X	X	1	0	0	0	0	= 04H
	พอร์ท B	X	X	X	1	0	0	0	1	= 05H
	พอร์ท C	X	X	X	1	0	0	1	0	= 06H
	คอนโทรลเวิร์ด	X	X	X	1	0	0	1	1	= 07H

รูปที่ 3.3 แผนผังหน่วยความจำเพื่อเลือกอ่าน-เขียนค่าไปที่8255ทั้ง2ตัว ในวงจรรูปที่ 3.23

สำหรับอุปกรณ์ภายนอกที่มาเชื่อมต่อกับพอร์ทของวงจรควบคุมจะมีดังต่อไปนี้

1) คีย์บอร์ด จะต่ออยู่กับพอร์ทหนึ่งของไมโครคอนโทรเลอร์ เป็นตัวติดต่อกับผู้ใช้โดยให้ป้อนรหัสผ่านทุกครั้งที่มีการรูดบัตเวลาเข้า-ออก เพื่อให้ประตูเปิดออก หรือกรณีเลือกโหมดการรูดบัตรโดยกดปุ่ม2nd เพื่อเลือกที่จะแจ้งเวลาออกเป็นการเริ่มตั้งระบบรักษาความปลอดภัยให้ทำงาน และกดปุ่ม Enter เพื่อเลือกที่จะแจ้งเวลาขาเข้าซึ่งต้องมีการกดรหัสผ่าน โดยเครื่องอ่านบัตรแม่เหล็กจะอ่านโค้ดในบัตรของผู้ใช้แล้วส่งต่อให้ไมโครคอนโทรลเลอร์นำไปติดต่อกับคอมพิวเตอร์เพื่อดึงข้อมูลของโค้ดในบัตรนี้ฐานข้อมูลรวมทั้งรหัสผ่าน แล้วนำไปเทียบกับรหัสผ่านที่ผู้ใช้กด หากถูกต้องประตูก็จะเปิดออกเป็นต้น ซึ่งปุ่มใดจะทำหน้าที่อะไรขึ้นกับการเขียนโปรแกรมสั่งการไมโครคอนโทรลเลอร์ ดังรูป



รูปที่ 3.4 การต่อคีย์บอร์ดเข้ากับพอร์ท 1 ของไมโครคอนโทรลเลอร์

2) เครื่องอ่านข้อมูลบนบัตรแม่เหล็ก จะประกอบไปด้วยขาสัญญาณ 5 เส้น คือ

- **ขาCARD PRESENT** จะต่อกับเข้ากับพอร์ท P2.0 ของไมโครคอนโทรลเลอร์ ซึ่งเป็นสายสัญญาณที่บอกให้รู้ว่าขณะนี้มีบัตรผ่านเข้ามายังเครื่องอ่านหรือไม่ โดยสัญญาณนี้จะเปลี่ยนสภาวะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นLow เมื่อตรวจพบว่ามีการเปลี่ยนแปลง ฟลักซ์ที่หัวอ่าน จากนั้นจะเปลี่ยนสถานะกลับไปเป็น high อีกครั้งหลังจากที่บัตรเคลื่อนที่พ้นหัวอ่านไปแล้วประมาณ 150 ms

- **ขาDATA** จะต่อเข้ากับพอร์ท P2.1 ของไมโครคอนโทรลเลอร์ ซึ่งเป็นสายสัญญาณของข้อมูลจากบัตรแม่เหล็ก ซึ่งข้อมูลที่อ่านได้จากสายสัญญาณนี้ก็คือข้อมูลจากบัตรแม่เหล็กนั่นเอง

- **ขาSTROBE** จะต่อกับพอร์ท P2.2 ของไมโครคอนโทรลเลอร์เป็นสายสัญญาณที่ควบคุมการอ่านข้อมูลจากสายสัญญาณ DATA เปรียบเสมือนสัญญาณนาฬิกาของการรูดบัตรแต่ละครั้ง คือ เมื่อสัญญาณ STROBE เปลี่ยนสภาวะจาก High เป็น Low (Negative edge) เราจะสามารถอ่านข้อมูลบนบัตรแม่เหล็กได้ทีละบิตในแต่ละช่วงของสัญญาณขาลง

- **ขาไฟเลี้ยงและขากราวนด์ ของเครื่องอ่านบัตรแม่เหล็ก**

3) ด็อทเมตริกซ์ แอลซีดีโมดูล (LCD : Liquid Crystal Display) ใช้เป็นส่วนแสดงผลที่ไมโครคอนโทรลเลอร์ใช้ในการติดต่อกับผู้ใช้ ในโครงงานชิ้นนี้จะใช้ LCD 1 บรรทัด ขนาด16 ตัวอักษร ซึ่ง มีตัวคอนโทรลเลอร์ภายในคือเบอร์ HD44780 ทำหน้าที่รับข้อมูลจากอุปกรณ์ภายนอกมาและจัดการควบคุมตัวLCD Moduleให้ทำงานการแสดงผลต่างๆ เช่นการเคลียร์จอภาพ , การเกิดตัวอักษรปรากฏบนตำแหน่งต่างๆของหน่วยความจำภายในLCD

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
80h	81h	82h	83h	84h	85h	86h	87h	C0h	C1h	C2h	C3h	C4h	C5h	C6h	C7h

ตารางที่ 3.1 แสดงตำแหน่งของหน่วยความจำที่ใช้ในการแสดงผลข้อมูลเป็นตัวอักษรในLCD

<u>ส่วนขาสัญญาณที่ใช้ควบคุมคำสั่งสั่งการทำงานของ LCD</u> ประกอบไปด้วย

3.1) ขาที่3 ต่อกับตัวต้านทานปรับค่าได้และขา2(ขาไฟเลี้ยง 5 โวลท์) เพื่อใช้ปรับความสว่างของแสงที่ตัวหน้าจอLCD

3.2) ขาที่4 หรือขา RS(Register Select) ในโครงงานนี้จะต่อเข้ากับพอร์ทP2.5ของMCS-51 มีหน้าที่เลือกว่าจะให้ส่งข้อมูลจากภายนอก(ที่ขาข้อมูลทั้ง8เส้น) ในรูปแบบการเขียนข้อมูลให้แสดงบนหน้าจอLCD หรือเป็นการป้อนคำสั่ง(Command) ปรับรูปแบบการแสดงผลของLCD หากเป็นการเขียนข้อมูลจะให้ขานี้เป็นลอจิก 1 แต่ถ้าเป็นการป้อนคำสั่งจะให้ขานี้เป็นลอจิก 0

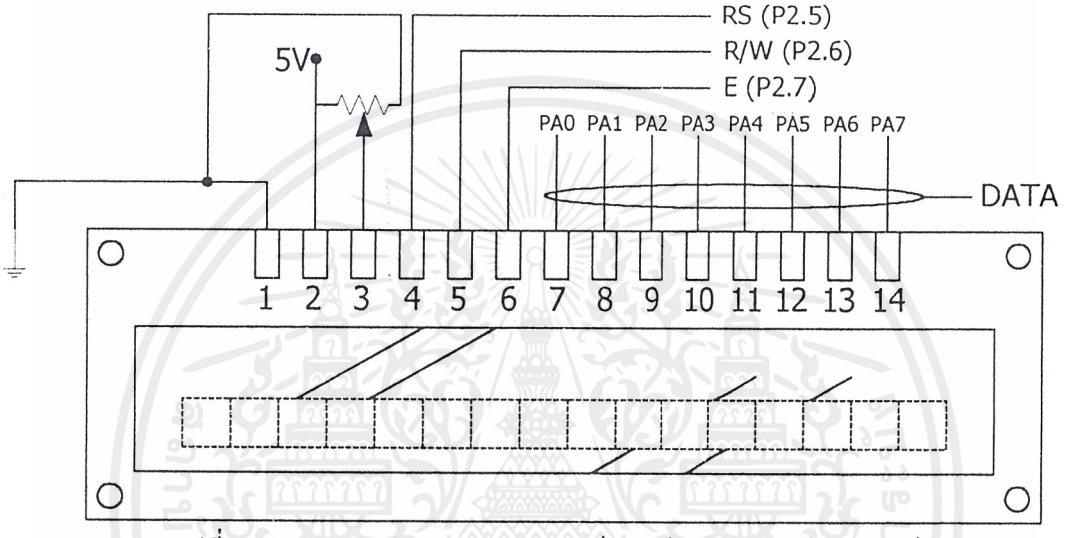
3.3) ขาที่5 หรือขา R/$\overline{W}$ (Read/Write) ซึ่งจะต่อเข้ากับพอร์ทP2.6 ของMCS-51 ซึ่งมีหน้าที่ในการเลือกการกระทำกับข้อมูลที่แสดงในหน่วยความจำแต่ละตำแหน่งตัวอักษรว่าจะให้ อ่านหรือเขียนข้อมูลจากตำ5แหน่งที่เคอร์เซอร์ชี้อยู่ หากเป็นการอ่านข้อมูลจะให้ขานี้เป็นลอจิก1 แต่ถ้าเป็นการเขียนข้อมูลเพื่อให้แสดงผลจะให้ขานี้เป็นลอจิก 0

3.4) ขาที่6 หรือขาE(Enable data) ซึ่งจะต่อเข้ากับพอร์ทP2.7 ของMCS-51 มีหน้าที่ Enable ให้ข้อมูลที่รออยู่ที่ขาDataทั้ง7เส้น ผ่านเข้าไปสู่คอนโทรลเลอร์ในLCDเพื่อนำไปประมวลผล ใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เวลาช่วง2ไซเคิล โดยการEnableแต่ละครั้งจะต้องใช้สัญญาณขาขึ้นเป็นเวลา2ไซเคิลและตามด้วยขาลง

3.5) ขาที่7 ถึง14 เป็นขาข้อมูล มีทั้งหมด 8บิต ซึ่งสามารถเป็นได้ทั้งข้อมูล(ตัวอักษร)หรือคำสั่งก็ได้ ทั้งนี้ขึ้นอยู่กับการเลือกขาควบคุมดังได้กล่าวมาแล้ว สำหรับโครงงานนี้จะต่อกับพอร์ท PA0 ถึง PA7 ของไอซี8255 ตัวที่1



รูปที่ 3.5 แสดงการต่อขา LCD กับพอร์ทของไมโครคอนโทรลเลอร์

4) ไฟแสดงสถานะ ในส่วนของโครงงานชิ้นนี้จะใช้ไปLEDจำนวน3ดวง คือสีเขียว สีเหลือง และสีแดง โดยมีการต่อกับทรานซิสเตอร์เพื่อทำหน้าที่เป็นสวิทช์เมื่อมีสัญญาณจากพอร์ทมากระตุ้นขาเบสของทรานซิสเตอร์

ไฟสีเหลือง จะต่ออยู่ที่พอร์ท PB4 แสดงถึงสภาวะปรกติ หรือระบบกำลังทำงานเป็นปกติ

ไฟสีเขียว จะต่ออยู่ที่พอร์ท PB5 แสดงถึงสภาวะผ่านหรือถูกต้องเมื่อมีการกระทำกับระบบ

ไฟสีแดง จะต่ออยู่ที่พอร์ท PB6 แสดงถึงสภาวะผิดพลาดหรือ ERROR

5) บอร์ดควบคุมอุปกรณ์ไฟฟ้า สำหรับในโครงงานนี้ได้กำหนดจำนวนอุปกรณ์ที่ควบคุมจำนวน8 อุปกรณ์ ซึ่งอาจจะเป็นหลอดไฟ เครื่องปรับอากาศ หรือเครื่องใช้ต่างๆซึ่งสามารถเพิ่มจำนวนอุปกรณ์ควบคุมได้ขึ้นอยู่กับจำนวนพอร์ทที่มีเหลือในส่วนวงจรไมโครคอนโทรลเลอร์ ในบอร์ดนี้จะประกอบด้วย ส่วนที่เป็นชุดรีเลย์ ทำหน้าที่ ตัด – ต่อไฟ220โวลท์กับอุปกรณ์ไฟฟ้าที่ควบคุมกับ สวิทช์ 2 ทาง ซึ่งจะต่อกับพอร์ทC ของไอซี8255ตัวที่2 และส่วนที่เป็นตัวตรวจเช็คสถานะของอุปกรณ์ไฟฟ้าว่าเปิดหรือปิดอยู่ ซึ่งจะทำการส่งค่าให้กับบอร์ดวงจรควบคุมนำไปส่งต่อให้กับคอมพิวเตอร์อีกที ซึ่งทำให้คอมพิวเตอร์สามารถนำข้อมูลที่ได้รับนี้มาแสดงบนหน้าจอเพื่อบอกสถานะแก่ผู้ใช้ได้ โดยในส่วนของตัวเช็คสถานะจะต่อเข้ากับพอร์ทB ของไอซี8255 ตัวที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.4 การสื่อสารกับคอมพิวเตอร์และการส่งข้อมูลผ่านพอร์ทอนุกรม

โดยปรกติเรามักจะใช้วงจรควบคุม(MCS-51) เป็นตัวควบคุมการทำงานของอุปกรณ์ต่างๆ เกือบทั้งหมด แต่ลำพังไมโครคอนโทรลเลอร์เพียงตัวเดียวไม่สามารถที่จะทำงานควบคุมอุปกรณ์หลายอย่าง หลายงานในเวลาเดียวกันได้ ถึงทำได้ก็จะต้องมีการรอคอยเกิดขึ้น ซึ่งอาจจะทำให้เกิดการล่าช้าได้ ดังนั้นจึงจำเป็นที่มักจะต้องมีอุปกรณ์ช่วยทำงานเสริมให้กับไมโครคอนโทรลเลอร์สำหรับงานควบคุมหลายๆอย่าง ซึ่งในส่วนของโครงงานนี้เห็นว่าเป็นโครงงานซึ่งเน้นระบบควบคุมเป็นหลักมีการทำงานหลายส่วนที่ค่อนข้างซับซ้อนจึงได้มีการนำคอมพิวเตอร์เข้ามาเป็นส่วนช่วยในการทำงานให้กับไมโครคอนโทรลเลอร์ ซึ่งจัดว่าคอมพิวเตอร์เป็นระบบที่ค่อนข้างแพร่หลาย ไม่เจาะจงลักษณะการใช้งาน ทำให้การประยุกต์ใช้งานใดๆจะขึ้นกับโปรแกรมที่ใช้ ไม่ต้องเปลี่ยนโครงสร้างทางฮาร์ดแวร์ใหม่ แต่จะไม่เหมาะกับงานระบบควบคุมเล็กๆเพราะดูจะเป็นการสิ้นเปลืองจนเกินไปที่จะต้องใช้คอมพิวเตอร์ ลำพังไมโครคอนโทรลเลอร์ก็สามารถทำได้ สำหรับหน้าที่ของการนำคอมพิวเตอร์มาใช้ในโครงงานนี้จะมีรายละเอียดดังต่อไปนี้

1) เป็นตัวให้เวลาปัจจุบันแก่ไมโครคอนโทรลเลอร์ แทนอุปกรณ์RTC(Real time Clock)

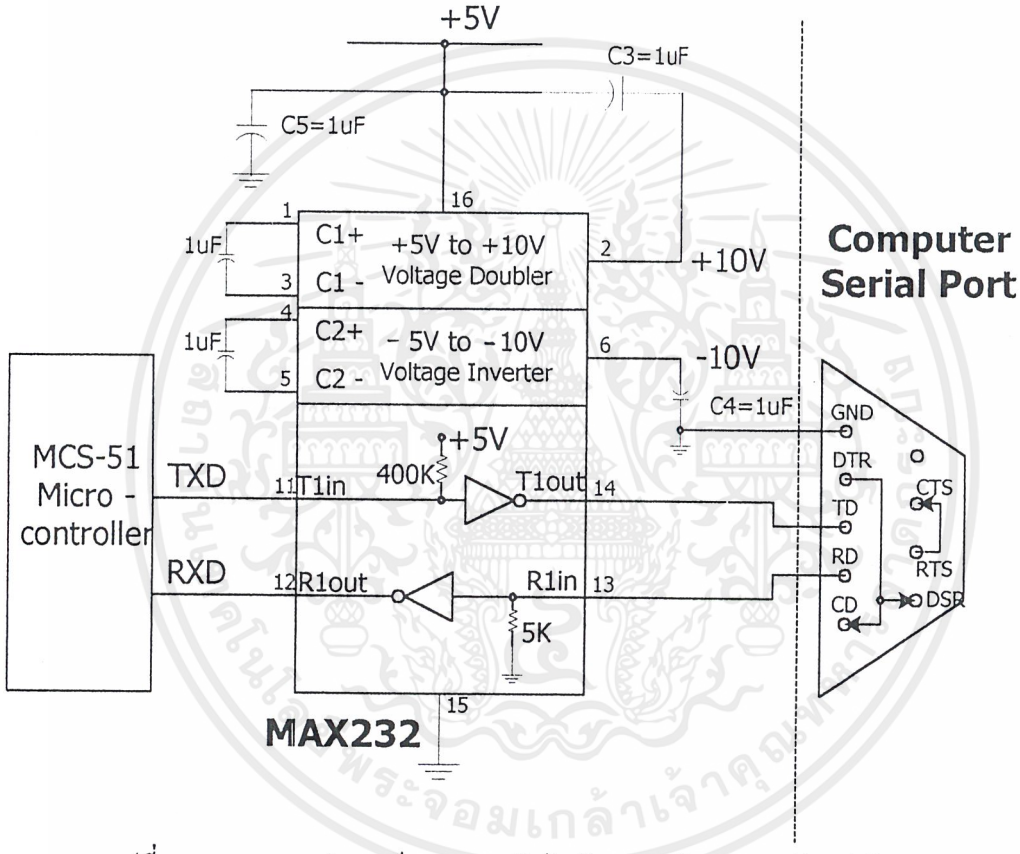
2) เป็นส่วนติดต่อกับผู้ใช้ภายในอาคาร โดยผู้ใช้สามารถติดต่อกับระบบโดยผ่านทางอินเตอร์เฟซกับโปรแกรมที่สร้างขึ้นด้วยโปรแกรมวิชวลเบสิก(Visual Basic 6.0) ซึ่งผู้ใช้จะสามารถควบคุมหรือตั้งเวลาการเปิด-ปิด อุปกรณ์ภายในอาคาร รวมถึงผู้ดูแลควมคุมระบบจะจัดการเกี่ยวกับฐานข้อมูลของระบบผ่านทางคอมพิวเตอร์ได้โดยตรง

3) เป็นตัวเก็บข้อมูลในรูปแบบของฐานข้อมูล(Database) ซึ่งจะเป็นข้อมูลแบบถาวร และสามารถเก็บเป็นสำนวนมากๆได้ สะดวกแก่การค้นหา เรียกใช้ และไม่เปลืองเนื้อที่ในการจัดเก็บ

รูปแบบของการสื่อสารระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์จะใช้การสื่อสารข้อมูลผ่านทางพอร์ทอนุกรม ตามมาตรฐาน RS-232C ซึ่งเป็นมาตรฐานที่กำหนดโดย EIA ซึ่งเป็นองค์กรอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ที่ใช้กันอย่างแพร่หลายโดยแบ่งการเชื่อมต่อออกเป็น 2 ลักษณะคือ DTE (Data Terminal Equipment) และ DCE (Data Communication Equipment) โดยปกติ DTE จะต้องต่อเข้ากับ DCE เสมอ เช่นการต่อเครื่องคอมพิวเตอร์ (อุปกรณ์ DTE) เข้ากับโมเด็ม (อุปกรณ์ DCE) เป็นต้น แต่ในที่นี้เป็นการติดต่อระหว่างพอร์ทอนุกรมบนคอมพิวเตอร์ ซึ่งในการสื่อสารจะถูกจัดการโดยชิพ UART (Universal Asynchoronous Receiver/Transmitter) กับรีจิสเตอร์วงจรควบคุมการส่งข้อมูลแบบอนุกรมในMCS-51 จึงเหมือนว่าเป็นการติดต่อสื่อสารระหว่างอุปกรณ์ DTE กับอุปกรณ์ DTE ด้วยกัน ซึ่งเรียกว่าสภาวะ Null Modem

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในสภาวะการติดต่อของพอร์ทอนุกรมบนคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ที่เรียกว่า Null Modem เป็นเหมือนการส่งข้อมูลไปมาระหว่าอุปกรณ์DTEทั้ง2 ซึ่งไม่ได้สลับซับซ้อนมากมายนัก ซึ่งจะพบเห็นโดยทั่วไปในการส่งไฟล์ระหว่างคอมพิวเตอร์ทั้ง 2 เครื่อง โดยขาส่งข้อมูล TD(Transmit data)ของทางคอมพิวเตอร์จะต่อกับขารับข้อมูล(RXD)ของMCS-51 และขารับข้อมูล RD(Receive data) ของคอมพิวเตอร์จะต่อกับขาส่งข้อมูล(TXD)ของMCS-51 เพื่อให้สามารถส่งรับข้อมูลได้แบบFull Duplex ส่วนการต่อขาอื่นๆที่พอร์ทอนุกรมบนคอมพิวเตอร์จะเป็นดังนี้



รูปที่ 3.6 แสดงการต่อพอร์ทอนุกรมกับไมโครคอนโทรลเลอร์ โดยใช้ MAX-232

DTR(Data terminal ready) ซึ่งมีหน้าที่แจ้งความพร้อมในการสื่อสารให้อุปกรณ์DCEได้ทราบ ในที่นี้จะต่อเข้ากับขา DSR(Data set ready)ของตัวมันเองซึ่งจะรับสัญญาณที่อุปกรณ์DCE ตอบกลับมาว่าพร้อมในการสื่อสารแล้วเช่นกัน และกับขาCD(Carrier detect)ซึ่งเป็นขาที่อุปกรณ์ DCEส่งมาบอกสถานะว่ากำลังติดต่อกับโมเด็มตัวอื่นอยู่หรือ โมเด็มกำลังได้รับสัญญาณที่พร้อมสำหรับการติดต่อสื่อสาร ซึ่งทันทีที่สัญญาณDTRแอกทีฟ สัญญาณ DSRและCDก็จะแอกทีฟในทันที ดังนั้นการต่อแบบนี้จึงเหมือนเป็นVirtual modem ที่มีความพร้อมสำหรับในการส่ง-รับ ข้อมูลได้ตลอดเวลาและติดต่อกับคอมพิวเตอร์ที่สื่อสารอยู่ตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RTS(Request to send) ส่งจาก DTE ไปยัง DCE มีหน้าที่บอกว่าขณะนี้อุปกรณ์DTEพร้อมที่จะรับข้อมูลแล้วให้อุปกรณ์DCEส่งข้อมูลมาได้ซึ่งจะต่อกับขาCTS(Clear to send) ของตัวมันเอง ซึ่งมีหน้าที่ส่งกลับไปบอกอุปกรณ์DTE ว่าข้อมูลกำลังจะส่งไป ดังนั้นในกรณีนี้จะมีการร้องขอการส่งข้อมูลและตอบรับความพร้อมในการส่งข้อมูล ด้วยตัวเองโดยทันที ซึ่งสามารถส่งหรือรับข้อมูลได้เลยโดยไม่สนใจว่าปลายทางพร้อมที่จะรับข้อมูลนั้นรึเปล่า

สำหรับพอร์ทอนุกรมของคอมพิวเตอร์ตามมาตรฐานRS-232 คุณลักษณะของสัญญาณทางไฟฟ้าที่ถูกใช้งานจะมี 2 ลักษณะคือ SPACE แสดงถึงไบนารี 0 หรือแรงดันไฟฟ้าบวก และMARKแสดงถึงไบนารี 1 หรือแรงดันไฟฟ้าลบ ซึ่งแรงดันไฟฟ้าบวก (สถานะ SPACE) อยู่ระหว่าง +5 ถึง +15 โวลต์ สำหรับเอาท์พุต และระหว่าง +3 ถึง +15 โวลต์ สำหรับอินพุต ซึ่งมีความแตกต่างนี้มีไว้เผื่อกรณีแรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณ ในทำนองเดียวกันแรงดันไฟฟ้าลบ (สถานะ MARK) ถูกกำหนดไว้ระหว่าง –5 ถึง –15 โวลต์ สำหรับเอาท์พุตและ –3 ถึง –15 โวลต์ สำหรับอินพุต ซึ่งถ้าให้สายสัญญาณที่ใช้ในการส่งมีความยาวเกินไป ระดับไฟฟ้าจะตกลงเกินขอบเขตที่ยอมรับได้ นอกจากนี้ ประจุไฟฟ้าที่เกิดขึ้นจะมีผลกับคุณภาพของสัญญาณ โดยการเปลี่ยนสถานะจากแรงดันไฟฟ้าบวกไปลบไม่ชัดเจน ซึ่ง RS-232-C ไม่ได้มุ่งหมายให้นำไปใช้กับระยะทางไกล และโดยทั่วไป 50 ฟุตเป็นระยะทางไกลที่สุดในการใช้สายปกติที่อัตราการส่งข้อมูลปกติ ถ้าหากอุปกรณ์อยู่ห่างกันมากอาจต้องใช้โมเด็ม หรือใช้มาตรฐานอื่นเข้ามาช่วย

ดังนั้นการที่เรานำพอร์ทอนุกรมของคอมพิวเตอร์ซึ่งมีระดับสัญญาณไฟฟ้าดังที่กล่าวมาแล้วทำให้เมื่อเรานำมาต่อกับขารับ-ส่งสัญญาณอนุกรมของไมโครคอนโทรลเลอร์ซึ่งเป็นระดับสัญญาณTTL(0 และ5 โวลท์) จำเป็นต้องมีตัวแปลงระดับสัญญาณให้สามารถใช้ร่วมกันได้ สำหรับในโครงงานนี้จะใช้ MAX-232 ซึ่งมีการต่อขาต่างๆดังรูปที่ 3.6 โดยภายในMAX-232 จะมีตัวเก็บประจุที่ทำหน้าผลิตสัญญาณไฟฟ้าขนาด+10โวลท์ และขนาด-10โวลท์จากแหล่งจ่ายไฟขนาด 5โวลท์ได้ ซึ่งขาT_{in} ทางฝั่งที่ของไมโครคอนโทรลเลอร์ส่งสัญญาณออกจะรับสัญญาณในช่วง -0.3 ถึงV_{cc}-0.3 โวลท์ แล้วจะแปลงเป็นสัญญาณที่ขาT_{out} เพื่อส่งให้พอร์ทคอมพิวเตอร์ในช่วง $\pm$15 โวลท์ ส่วนเมื่อพอร์ทคอมพิวเตอร์ส่งข้อมูลมายังไมโครคอนโทรลเลอร์ ก็จะแปลงสัญญาณที่เข้าขาR_{in}ซึ่งอยู่ในช่วง $\pm$ 30 โวลท์ให้กลายเป็นสัญญาณที่ขาR_{out}ในช่วง-0.3 ถึงV_{cc}+0.3 โวลท์ ซึ่งทำให้อุปกรณ์ทั้งสองสามารถติดต่อกันด้วยมาตรฐานแรงดันที่แตกต่างกันได้

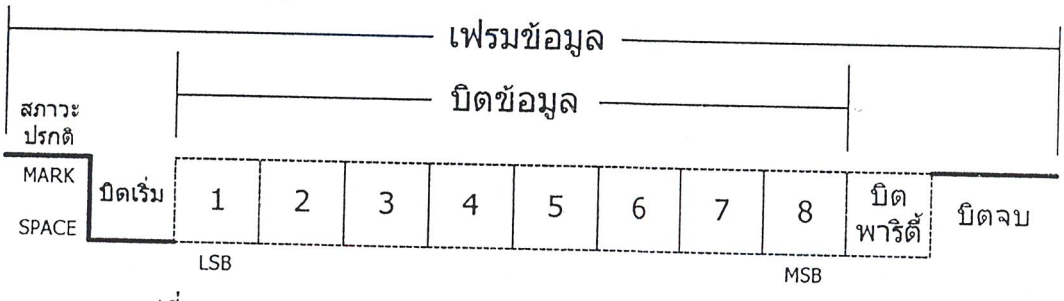
3.2.5 <u>รูปแบบของเฟรมข้อมูลในการส่งแบบอนุกรม</u>

การส่งข้อมูลติดต่อสื่อสารกันระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์จะ ส่งเป็นแบบอนุกรม(Serial Connection) ซึ่งจะช่วยประหยัดช่องสัญญาณไม่สิ้นเปลืองจำนวนสายส่งเท่ากับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบขนาน(Parallel) โดยจะเป็นการส่งที่ละบิตเรียงกันไปซึ่งทำให้ใช้สายสัญญาณในการส่งเพียง2 เส้นเท่านั้นทั้งยังสามารถส่งข้อมูลได้ไกลขึ้นกว่าแบบขนาน แต่ความเร็วในการส่งก็จะต้องลดลงน้อยกว่าแบบขนานมาก ปัญหาของการส่งแบบอนุกรมจะอยู่ที่ปลายทางจะแปลงสัญญาณข้อมูที่รับเข้ามาได้อย่างถูกต้องว่าเป็น0 หรือ1ได้อย่างไร สำหรับการส่งแบบอนุกรมสามารถแก้ได้โดยใช้สัญญาณในการรับส่งข้อมูล(Baud Rate)ของทั้ง 2 ฝั่งที่เท่ากัน ซึ่งต้องมีการส่งสัญญาณเวลาควบคู่ไปอีกช่องสัญญาณซึ่งจะเป็นการสิ้นเปลืองสำหรับการส่งระยะทางที่ไกลๆ อีกทั้งฝ่ายรับซึ่งรับข้อมูลที่เรียงมาอย่างต่อเนื่องนั้นจะไม่มีทางทราบได้เลยว่าข้อมูลแต่ละไบต์ตัวอักษรจะเริ่มที่บิตใดและจบที่บิตใด ซึ่งได้มีวิธีแก้ปัญหาเหล่านี้อยู่ 2 แบบคือ เทคนิคการส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) และเทคนิคการส่งข้อมูลแบบซิงโครนัส (Synchronous)

สำหรับในโครงงานนี้ได้เลือกใช้เทคนิคการส่งข้อมูลแบบ อะซิงโครนัส เนื่องจากการส่งข้อมูลแบบซิงโครนัสนั้นค่อนข้างยุ่งยากและเหมาะสำหรับการส่งข้อมูลจำนวนครั้งละมากๆซึ่งต้องการให้มีประสิทธิภาพมากที่สุดในการส่งและไม่มีข้อผิดพลาดเกิดขึ้น ซึ่งแบบอะซิงโครนัสนั้นในแต่ละตัวอักษรจะมีการส่งบิตเริ่มต้น(Start bit) ซึ่งมีค่า 0 ไป1บิต แล้วตามด้วยข้อมูลของตัวอักษรจำนวน8บิต อาจมีพาริตี้บิตด้วยในกรณีที่มีการเช็คความถูกต้องของเฟรมตัวอักษรด้วย แล้วตามด้วยบิตจบ(Stop bit) ซึ่งมีค่าเป็น1 โดยในขณะที่ไม่มีการส่งตัวอักษร สภาวะของสายจะมีค่าเป็น1 ทันทีที่มีการส่งข้อมูลชิพUARTของพอร์ทอนุกรมจะทำให้สภาวะของสายเปลี่ยนจาก1เป็น0 เป็นเวลา1 บิต ไทม์เป็นการส่งบิตเริ่มต้น จากนั้นจึงค่อยเริ่มส่งสัญญาณข้อมูลของตัวอักษร 8บิต เมื่อส่งเรียบร้อยแล้วUARTก็จะทำให้สายมีสภาวะเป็น1 เป็นการส่งบิตจบ ดังรูปที่ 3.7 การส่งแบบนี้จะเป็นการส่งข้อมูลทีละตัวอักษร โดยที่ระยะเวลาของแต่ละตัวอักษรนานเท่าไรก็ได้ ซึ่งทำให้ UART สามารถตรวจเช็คความผิดพลาดของแต่ละตัวอักษรได้จากการเช็คบิตเริ่มต้นและบิตสุดท้ายของแต่ละตัวอักษรเสมอ รวมทั้งการเช็คพาริตี้ด้วย แต่สำหรับวิธีนี้ฝั่งรับจะต้องถูกตั้งให้อ่านข้อมูลด้วยอัตราเดียวกับการส่งข้อมูลของฝั่งส่ง นั่นคือสัญญาณเวลาจะเท่ากัน นอกจากนี้จำนวนของบิตจบและค่าของพาริตี้บิตจะต้องถูกตั้งไว้ให้ตรงกับฝั่งส่งด้วย

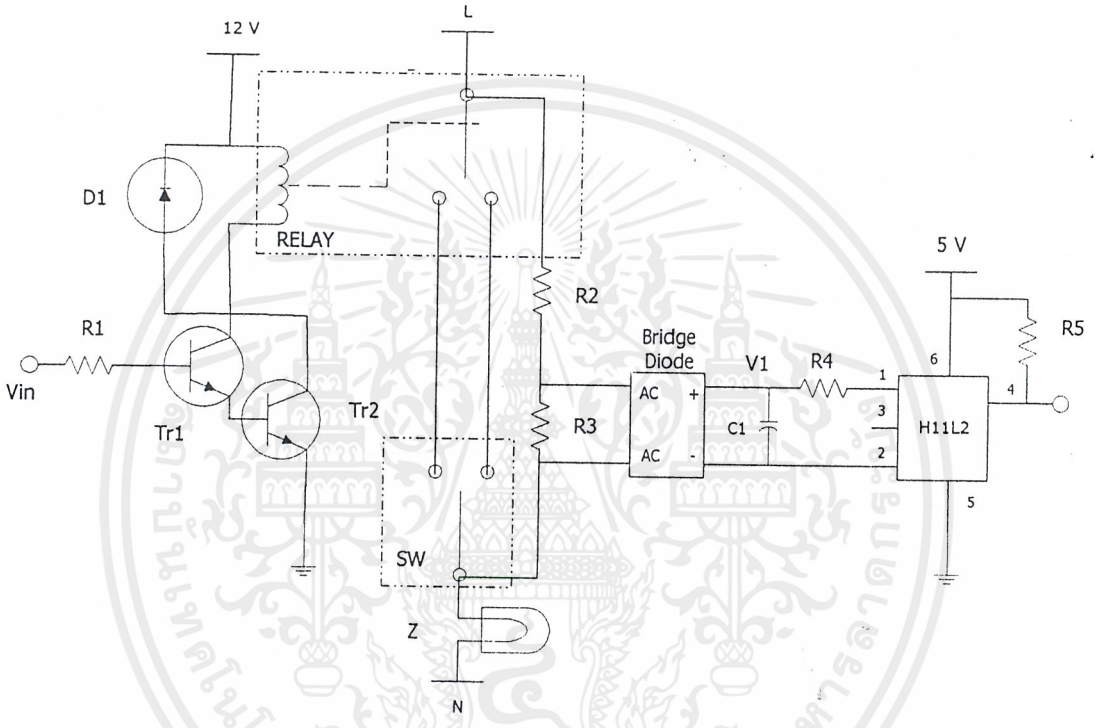


รูปที่ 3.7 แสดงเฟรมการส่งข้อมูลแบบอะซิงโครนัสของหนึ่งตัวอักษร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.6 ในส่วนของวงจรควบคุมอุปกรณ์ไฟฟ้า(Switching Board)

เป็นส่วนที่ทำหน้าที่ควบคุมการปิด-เปิดของอุปกรณ์ไฟฟ้าต่างๆแทนการใช้สวิทช์ ในที่นี้จะมีการเปิดปิดอุปกรณ์ไฟฟ้า 2 ทาง คือ ทางหน้าจอคอมพิวเตอร์ และ ทางสวิทช์โยกปิด-เปิด จะมีส่วนตรวจจับสัญญาณ หากมีการเปิดหรือปิดอุปกรณ์ไฟฟ้า จะส่งสัญญาณไปยังเครื่องคอมพิวเตอร์ดังรูป 3.8



รูปที่ 3.8 วงจรควบคุมอุปกรณ์ไฟฟ้า

ให้ Z เป็นหลอดไฟขนาด 7 วัตต์ แทนอุปกรณ์ไฟฟ้าต่างๆ ซึ่งมีความต้านทานเท่ากับ

$$R = V^2/P = 220x220 / 7 = 7K\Omega$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

<u>หลักการออกแบบ</u>

ส่วนควบคุมการปิด-เปิด แบ่งเป็น 2 ส่วน

1) เปิด – ปิดทางหน้าจอคอมพิวเตอร์ ดังรูป 3.8 Tr1 และ Tr2 ต่อแบบดาร์ลิงตัน ทำหน้าที่ ขับรีเลย์ โดยเป็นลักษณะของสวิทช์ เมื่อมีแรงดันจากไมโครคอนโทรลเลอร์ มาป้อนให้กับดาร์ลิงตันทำงานให้อยู่สภาวะอิ่มตัว โดยการป้อนกระแสซึ่งถูกควบคุมโดย R1 การหาค่า R1 จะต้องทราบค่า กระแสที่ไหลคอลเลคเตอร์สูงสุดที่ไหลผ่าน Tr2

$$R_1 = (V_{in} - 2V_{be})\,\beta / I_{c(max)} \qquad \text{----------(3.1)}$$

$$\beta = \beta_1\beta_2 \qquad \text{----------(3.2)}$$

ในที่นี้ใช้ BD139 และ 2N2222A ตามคุณสมบัติของทรานซิสเตอร์ทั้งสอง

BD139 : $I_{c(max)} = 1.5\ A \quad \beta_2 = 100$

2N2222A : $\beta_1 = 200$

เพราะฉะนั้น

$$R_1 = (5-1.2)*20000/1.5$$
$$= 50666$$
$$\cong 50000$$

ใช้ $R_1 = 50\ K\Omega$

รีเลย์ ที่ใช้ จะเป็นแบบ 12 V_{dc} 1 contact ทำหน้าที่เป็นสวิทช์ปิด-เปิดอุปกรณ์ไฟฟ้าต่างๆ โดยมีการต่อ ไดโอด(D_1) เพื่อไม่ให้ขดลวดในตัวรีเลย์คายพลังงานไปยังทรานซิสเตอร์เมื่อไม่ได้จ่ายแรงดัน ถ้าหากไม่ต่อไดโอด จะทำให้ทรานซิสเตอร์เสียได้

2) ส่วนปิด-เปิดทางสวิทช์ จะใช้สวิทช์โยก 3 ทาง ในการเปิดปิดไฟ ซึ่งการเปิด-ปิดในแต่ละครั้งจะมีการส่งสัญญาณไปยังเครื่องคอมพิวเตอร์ เพื่อบอกสถานะให้ทราบ ดังรูป 3.8 ได้มีการแบ่งแรงดันส่วนหนึ่งเพื่อเป็นตัวตรวจจับสัญญาณโดยใช้ H11L2 ได้มีการใส่บริดจ์เรียงกระแสให้เป็นกระแสตรงก่อนจากนั้นใส่ตัวเก็บประจุเพื่อทำให้เรียบ ก่อนเป็นอินพุทให้กับ H11L2 มีหลักการออกแบบดังนี้

กำหนดค่าของ H11L2 $I_f = 1mA \quad V_f = 1V \quad V_1 = 2\ V$

เพราะฉะนั้น $R_4 = (V_1 - V_f)/I_f = (2-1)/1m = 1K\Omega$

ให้ $R_2 = 3K\Omega$, $I = 2\ mA_{rms}$

$$V_{3k} = (V_m + 1.4)/\sqrt{2}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$= (2+1.4)/\sqrt{2}$

$= 2.4\ V_{rms}$

จะมีแรงดันตกคร่อมที่เหลือ $= 220 - 2.4 = 217.6\ V_{rms}$

ต้องการให้กระแสไหลในวงจรก่อนสับสวิทช์ $= 2\ mA_{rms}$

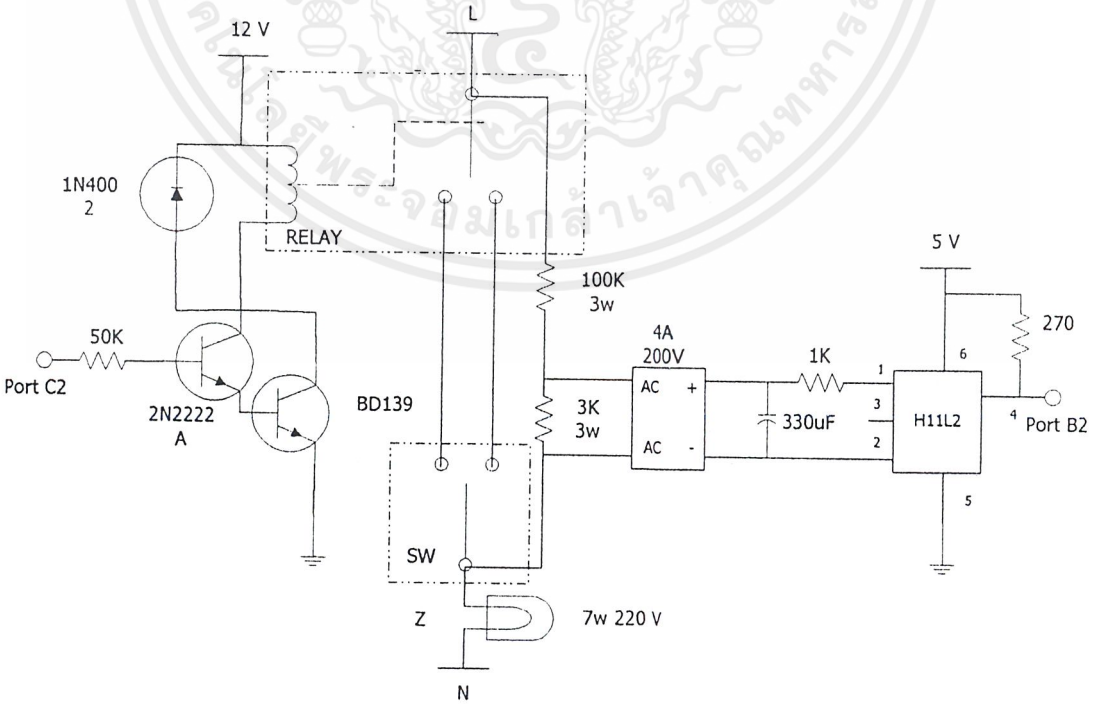
$R_1+Z = 217.6/2 = 108.8\ K\Omega$

$R_1 = 108.8 - 7 = 101.8 \cong 100K\Omega$

การทำงานของ Switching Board

สามารถเปิด- ปิดอุปกรณ์ไฟฟ้าได้ 2 ทางคือ ทาง หน้าจอคอมพิวเตอร์ และ สวิทช์จากรูปที่ 3.9 หากเปิดทางหน้าจอคอมพิวเตอร์ คอมพิวเตอร์จะส่งสัญญาณไปยังคอลโทรลเลอร์ และให้คอลโทรลเลอร์เป็นตัวแปลงรหัสข้อมูลว่าให้เปิดอุปกรณ์ตัวไหนก็จะส่ง แรงดัน 5 โวลท์ ในแต่ละบิทที่ต้องการ มากระตุ้นให้ทรานซิสเตอร์ทำงานและ รีเลย์ จะทำงานทำให้หลอดไฟติด

ก่อนเปิดอุปกรณ์ไฟฟ้า สัญญาณทางเอาท์พุทของ H11L2 เป็น 0 เพราะเป็นเกทแบบ open corrector เมื่อเปิดอุปกรณ์ไฟฟ้า จะทำให้กระแสไม่ไหลผ่าน R_1 และ R_2 ทำให้ H11L2 ไม่ทำงาน เอาท์พุท จะมีค่าเท่ากับ 5 โวลท์ ส่งไปยังคอลโทรลเลอร์และส่งไปยังคอมพิวเตอร์



รูปที่ 3.9 วงจรควบคุมอุปกรณ์ไฟฟ้า(เมื่อออกแบบแล้ว)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การออกแบบทางด้านซอร์ฟแวร์

ในส่วนของการเขียนโปรแกรมควบคุมการทำงานจะแบ่งได้2ส่วนคือ ในส่วนที่สั่งการไมโครคอนโทรลเลอร์ และส่วนที่สั่งการคอมพิวเตอร์ซึ่งใช้โปรแกรมวิชวลเบสิก6.0 ในการอินเตอร์เฟซติดต่อกับผู้ใช้ รวมถึงส่วนของการสื่อสารกันระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ด้วย

3.3.1 อัตราการส่ง-รับข้อมูล (Baud Rate) ในไมโครคอนโทรลเลอร์

ในโครงงานนี้จะใช้อัตราการส่งรับข้อมูลค่า 2400 BPS เนื่องจากในส่วนของไมโครคอนโทรลเลอร์เราจำเป็นต้องใช้ไทม์เมอร์เป็นตัวกำหนดBaud Rateเอง โดยหากใช้ออสซิเลเตอร์สร้างสัญญาณนาฬิกาขนาด11.059 MHz ป้อนให้กับMCS-51 จะสามารถให้มีอัตราบอดที่พอดีกับอัตราบอดมาตรฐานของคอมพิวเตอร์เช่น 1200,2400,4800,9600 BPS แต่สำหรับออสซิเลเตอร์ที่ให้สัญญาณนาฬิกาขนาด12 MHz ซึ่งง่ายต่อการคำนวณเวลาที่ใช้ในแต่ละไซเคิล แต่สำหรับอัตราบอดจะสร้างได้ไม่พอดี ดังตารางด้านล่างนี้

อัตราที่ต้องการ	อัตราที่ได้	ค่าของSMOD	ค่าของTH1	Error(%)
9600	8925	1	-7 (F9h)	7
2400	2404	0	-13 (F3h)	0.16
1200	1202	0	-26 (E6h)	0.16

ตารางที่ 3.2 แสดงการตั้งค่าไทม์เมอร์ที่ใช้สร้างอัตราบอดแต่ละค่า

ดังนั้น จึงเลือกใช้อัตราบอดที่2400BPS เนื่องจากมีข้อผิดพลาดเกิดขึ้นน้อยที่สุด ซึ่งไม่ส่งผลกระทบใดๆมากมายนักเนื่องจากในการส่งข้อมูลแต่ละจะส่งตัวอักษรจำนวนไม่มาก ไม่จำเป็นต้องใช้อัตราที่เร็วมากนัก

สำหรับMCS-51 ในการส่งข้อมูลผ่านพอร์ทอนุกรมจะทำการส่งและรับข้อมูลที่ขาTXD และขาRXDตามลำดับ ดังรูปที่3.6 โดยจะใช้Timer1 เป็นตัวสร้างอัตราบอด ในที่นี้จะเลือกSCON (Serial Port Control Register) ในโหมด1 เป็นโหมดUART 8 บิต คือข้อมูล8บิต บิตเริ่มต้นและบิตจบ ไม่มีบิตพาริตี้

$$\text{Baud Rate} = \frac{2^{SMOD}}{32} \times \left(\frac{f_{oscillator}}{12(256 - TH1)} \right)$$

ดังนั้นหากต้องการอัตราบอด 2400 BPS จะต้องตั้งค่าtimer1 ที่ TH1 = -13 หรือ F3h และตั้งค่า TCON (Timer Control Register) ให้อยู่ในโหมด 2 คือแบบ 8 bit Auto Reload และทำการตั้งค่า SMOD ให้เท่ากับ 0 เนื่องจากไม่ต้องการให้มีการเพิ่มอัตราเร็วในการส่งข้อมูลเป็น 2 เท่า โดยการปรับที่บิตที่ 7 ของ PCON

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับในการส่งข้อมูลในไมโครคอนโทรลเลอร์จะต้องเคลียร์ TI ให้เป็น 0 ก่อน และทำการเซ็ตค่า TR1 เพื่อให้มีอัตราบอดตามที่ได้ตั้งไว้ จากนั้นก็นำข้อมูลที่เป็นตัวอักษรแอสกี 8 บิตไปวางที่รีจิสเตอร์ SBUF ก็จะมีการส่งตัวอักษรนี้ออกไปเรียงกันทีละบิตจนครบซึ่งจะมีการใส่บิตเริ่มต้นและบิตจบเข้าไปในเฟรมข้อมูลด้วยเสมอเมื่อส่งครบแล้วTR1 จะเปลี่ยนเป็นลอจิก 1 โดยอัตโนมัติ ซึ่งเราจะต้องทำการเช็คบิต TI นี้เองว่าเป็น 1 แล้วจึงเคลียร์บิตนี้เพื่อทำการส่งตัวอักษรตัวต่อไปจนครบเรียกว่า การโพลลิ่ง (Polling)

และในการรับข้อมูลที่ส่งมาจากพอร์ตอนุกรมของคอมพิวเตอร์จะต้องทำการเซ็ทบิต REN ให้เป็น 1 ก่อนเพื่อยอมให้มีการรับข้อมูล และต้องเคลียร์บิต RI ให้เป็น 0 ก่อน และทำการเซ็ตค่า TR1 เพื่อให้มีอัตราบอดสำหรับข้อมูลที่จะส่งมา ข้อมูลจะถูกส่งมาวางไว้ในรีจิสเตอร์ SBUF ทีละตัวอักษร เมื่อครบแล้วเราต้องไปนำข้อมูลนั้นมาเก็บไว้เพื่อให้ SBUF ว่าง จากนั้นเราก็ทำการเคลียร์ RI เพื่อให้ตัวอักษรที่เรียงถัดมาได้มาอยู่ที่ SBUF เพื่อให้เรานำไปเก็บจนครบทุกตัวอักษร ดังนี้

โปรแกรมส่งข้อมูล 1 ตัวอักษร	**โปรแกรมรับตัวอักษร 1 ตัว**
CLR TI	CLR RI
SETB TR1	SETB TR1
MOV a,#ตัวอักษรแอสกี	SETB REN
MOV SBUF,A	JNB RI,$
JNB TI,$	MOV A,SBUF
CLR TI	CLR TI

ในการโพลลิ่ง ไมโครคอนโทรลเลอร์จะต้องคอยตรวจเช็คสถานะตรงนั้นอยู่ตลอดเวลาทำให้ไมโครคอนโทรเลอร์ไม่สามารถทำงานในส่วนอื่นได้เลยหรือหากจะแบ่งเวลาไปทำงานในส่วนอื่นด้วยก็จะอยู่ในรูปของระบบการแบ่งเวลาคือแบ่งเวลาทำโพลลิ่งช่วงหนึ่งแล้วโดดไปทำงานอื่นช่วงหนึ่งแล้วค่อยกลับมาทำโพลลิ่งใหม่ ดังจะทำให้การโพลลิ่งในแต่ละที่นั้นมีการผิดพลาดหรือหลุดการตรวจจับนั้นไปได้ ในการส่ง-รับข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์จะสามารถใช้การอินเตอร์รัพท์(INTERRUPT)ได้ นั่นคือทุกครั้งที่บิต TI หรือ RI กลายเป็น1 นั่นคือเมื่อมีการส่งข้อมูลออกไปหรือมีการรับข้อมูล 1 ตัวอักษรจะเกิดการอินเตอร์รัพท์ของการส่งแบบอนุกรมขึ้นซึ่งจะหยุดการทำงานในขณะนั้นแล้วโดดไปยังโปรแกรมย่อยตอบรับการอินเตอร์รัพท์(interrupt Service routine) ของการส่งข้อมูลแบบอนุกรมซึ่งจะอยู่ที่ตำแหน่ง 0023h ในที่นี้เราจำเป็นต้องมีการเขียนโปรแกรมตอบรับการอินเตอร์รัพท์และแยกการทำงานให้ถูกต้องว่าช่วงใดเป็นของการส่งข้อมูลออก ช่วงใดเป็นของการรับข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.2 การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์

ในการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ ในโครงงานนี้จะเป็นลักษณะการเขียนโปรแกรมเพื่อเรียกโปรแกรมย่อยซะเป็นส่วนใหญ่ ซึ่งพยายามให้สามารถตอบสนองความต้องการของผู้ใช้ที่ติดต่ออยู่หน้าคอมพิวเตอร์ได้อย่างถูกโหมดการทำงาน เนื่องจากในฟอร์มของโปรแกรมควบคุมคอมพิวเตอร์จะประกอบด้วยโหมดย่อยๆหลายๆ โหมดซึ่งเมื่อมีการคลิ้กใช้งานโหมดนั้น ไมโครคอนโทรลเลอร์จึงจำเป็นต้องโดดไปทำงานตามโหมดที่ควรจะทำ เพื่อให้มีการทำงานที่สอดคล้องกันระหว่างอุปกรณ์ทั้งสอง

ดังนั้นการอินเตอร์รัพท์จึงเป็นที่จำเป็นในโครงงานนี้ โดยเริ่มต้อนการทำงานไมโครคอนโทรลเลอร์จะทำการเซ็ตค่าและปรับแต่งโหมดการทำงานไม่ว่าจะเป็น8255 , LCD หรือกำหนดหน่วยความจำที่เข้าถึง จากนั้นก็จะอยู่ในสถานะรอรับคำสั่งจากคอมพิวเตอร์ ทันทีที่ผู้ใช้คลิ้กเปิดฟอร์มใดขึ้นมาก็จะมีการส่งโค้ดตัวอักษร 2 ตัวมาแจ้งยังไมโครคอนโทรเลอร์ ทำให้เกิดการอินเตอร์รัพท์ขึ้นทางพอร์ทอนุกรม โปรแกรมจะกระโดดไปทำงานที่โปรแกรมตอบสนองการเกิดอินเตอร์รัพท์ ของการส่งข้อมูลอนุกรมจะอยู่ที่ 0023h ที่ตำแหน่งนี้ก็จะมีการเขียนโปรแกรมเพื่อเช็คโค้ดที่รับมาว่าเป็นตัวอะไร หากตรงกับโหมดไหนก็ให้เคลียร์สแตก และกระโดดไปยังโปรแกรมช่วงที่เขียนสำหรับใช้งานเฉพาะโหมดในนี้ ด้วยการใช้คำสั่งCALLจะช่วยให้ง่ายต่อการเขียนโปรแกรม โดยที่บางโปรแกรมย่อยเราไม่จำเป็นที่จะต้องเขียนซ้ำหลายๆรอบ แต่จะเขียนเพียงรอบเดียวและใช้การCALLไปเรียกใช้ทุกครั้งที่ต้องการ แผนผังแสดงการทำงานของโปรแกรมหลักจะแสดงในรูปที่ 3.11

1) โปรแกรมย่อยสแกนคีย์ ในการเขียนโปรแกรมสแกนคีย์จะเริ่มไล่จากแถว1แล้วค่อยเป็นแถว2,แถว3และแถว4 ตามลำดับ เริ่มจากให้พอร์ททุกเส้นของพอร์ท1เป็น1หมด และป้อน 0 ในแถวที่กำลังพิจารณาอยู่ จากนั้นก็จะเริ่มทำการเช็คพอร์ที่เป็นคอลัมน์1,2,3,4 ตามลำดับ หากเจอว่าเป็น1 แสดงว่าไม่มีการกดในแถวนั้น แต่ถ้ามีบางคอลัมน์เป็น0 แสดงว่าได้มีการแตะกันของคอลัมน์ที่เรากำลังเช็คอยู่กับแถวที่เราเซ็ตให้เป็น0 เท่านี้ก็จะทราบว่าปุ่มอะไรถูกกด แต่ถ้าไม่พบก็เลื่อนไปที่แถวอื่นบ้างจากนั้นก็เคลียร์ทั้งพอร์ทให้เป็น1หมดแล้วค่อยป้อนค่า0 ในแถวใหม่ แล้วทำการเช็คแต่ละคอลัมน์ ดังที่ได้กล่าวไปแล้ว แต่ในขณะที่เช็คการกดของเลขใดแล้วควรจะหน่วงเวลาออกไปชั่วครู่เพื่อให้ปุ่มที่ถูกกด ตรวจพบการกดได้เพียงครั้งเดียวก่อนจะไม่ถูกกด ไม่เกิดการรับข้อมูลซ้ำหลายรอบในเวลาที่เร็วมากๆโดยไม่รู้ตัว

2) โปรแกรมแสดงผลในLCD เริ่มต้นด้วยการเขียนโปรแกรมย่อยโปรแกรมย่อยLCD Set ซึ่งจะเรียกใช้ทุกครั้งที่มีการสั่งตั้งค่าต่างๆให้กับLCD กับโปรแกรมย่อยLCD Write ซึ่งจะมีการเรียก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ทุกครั้งที่มีการเขียนข้อมูลให้LCDแสดงผล และโปรแกรมย่อยEnable LCDจะต้องเรียกใช้ทุกครั้งที่มีการเรียกใช้2โปรแกรมย่อยที่แล้วเพื่อให้LCD นำข้อมูลไปประมวลผล

โดยในการเขียนข้อมูลให้แสดงที่ตำแหน่งใดๆ เคอร์เซอร์จะเลื่อนไปตำแหน่งถัดไปให้อัตโนมัติ เริ่มจากป้อนคำสั่ง#01h แล้วเรียกโปรแกรมย่อยLCD_SETจะเป็นการเคลียร์หน่วยความจำที่แสดงผลทั้งหมด และเลื่อนเคอร์เซอร์อยู่ที่ 80h และถ้าหากเขียนข้อมูลมาอยู่ที่ 87h เคอร์เซอร์จะหยุดเลื่อนและจะไม่สามารถเขียนต่อไปได้อีกจนกว่าเราจะสั่งให้เคอร์เซอร์เลื่อนไปอยู่ที่ตำแหน่งถัดไปคือ c0h จึงจะเขียนข้อมูลให้แสดงต่อเป็นตัวอักษรที่ 9ได้ ดังโปรแกรมต่อไปนี้

```
หากต้องการตั้งค่าLCD
CALL delay
MOV a,#ค่าของคำสั่งที่จะตั้งค่าLCD
CALL lcd_set
CALL delay

หากต้องเขียนข้อมูลให้LCDแสดง
CALL delay
MOV a,#ตัวอักษรที่จะให้แสดง
CALL lcd_write
CALL delay

----------LCD Setting and writing ----------
 lcd_set: clr RS
          clr RW
          mov r1,#port_a
          movx @r1,a
          call enable
          call delay
          ret

lcd_write: setb RS
           clr RW
           mov r1,#port_a
           movx @r1,a
           call enable
           call delay
           ret
;.....................................................
enable:   setb E
          nop
          nop
          clr E
          ret
;.....................................................
delay:    mov r1,#0ffh
loop:     nop
          nop
          djnz r1,loop
          ret
```

รูปที่ 3.10 แสดงตัวอย่างของโปรแกรมควบคุมLCD

3) การทำงานในโหมด3 (โหมดฐานข้อมูล) เมื่อคอมพิวเตอร์ปลายทางมีการโหลดฟอร์มนี้ขึ้นมาจะทำให้มีการส่งโค้ด "m3"มาทางพอร์ทอนุกรม ทำให้เกิดการอินเตอร์รัพท์ขึ้น เมื่อโปรแกรมตอบรับการอินเตอร์รัพท์ทำการเช็คโค้ดที่ได้รับมาถูกต้องแล้วก็จะกระโดดไปยังโปรแกรมหลักของโหมดฐานข้อมูลซึ่งเป็นการรอรับค่าบัตรแม่เหล็กจากเครื่องอ่านบัตรแม่เหล็กเพื่อส่งให้คอมพิวเตอร์นำไปใช้งานต่อเช่นเพิ่มบัตรในฐานข้อมูลหรือดูข้อมูลของบัตรเป็นต้น ซึ่งในโหมดนี้จะใช้โปรแกรมย่อยเดียวกันกับโหมดออนไลน์เนื่องจากมีความคล้ายกันในแง่ของการรับข้อมูลจากบัตร จึงต้องมีการสร้างบิตเช็คสถานะว่าเป็นโหมดใดกันแน่(บิตerror_status3) โดยทุกครั้งที่เข้าโหมดนี้

เอกสารนี้ใช้สำหรับการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ในเชิงการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิตนี้จะถูกเซ็ตแต่สำหรับโหมดออนไลน์บิตนี้จะถูกเคลียร์ เมื่อรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็กแล้วจะต้องมีการตรวจเช็คความถูกต้องของข้อมูลในบัตรถึง 3 ขั้นตอนได้แก่การเช็คแบบพาริตี้ การเช็คแบบLRC และเช็คว่ามีบิตเริ่มต้นกับบิตจบ เมื่อตรวจสอบถูกต้องแล้วก็จะทำการแปลงเลขให้เป็นรหัสแอสกี้มีพาริตี้คู่ เนื่องจากเมื่อส่งข้อมูลทางพอร์ทอนุกรม UARTของพอร์ทจะมีการเช็คพาริตี้คู่ทุกตัวอักษรตามที่ได้กำหนดไว้ในคุณสมบัติของMSCommเพื่อกันการผิดพลาดที่เกิดจากการส่ง จากนั้นก็จะส่งข้อมูลที่อ่านได้ไปยังคอมพิวเตอร์ โดยการทำงานจะเป็นไปตามรูปที่ 3.12

4) การทำงานในโหมด1 (โหมดออนไลน์) สำหรับในโหมดนี้ก็จะคล้ายๆกันคือ เมื่อได้รับโค้ด"m1"จากคอมพิวเตอร์ก็จะ โดดไปยังโปรแกรมช่วงโหมดออนไลน์ซึ่งเริ่มต้นจะให้เลือกกดคีย์บอร์ดเพื่อเลือกการเข้าหรือการออก ไม่ว่าจะเป็นการกดเพื่อเข้าหรือออกก็ตามก็จะต้องรูดบัตร1ครั้งซึ่งจะมีการับข้อมูลจากเครื่องอ่านบัตรแม่เหล็กและตรวจเช็คความถูกต้องของข้อมูลเหมือนกับในโหมดฐานข้อมูลแต่จะมีการเพิ่มมาคือจะมีการส่งโค้ดปะไว้หน้าข้อมูลเพื่อให้โปรแกรมปลายทางสามารถแยกทำงานถูกโดยที่การเข้าจะใส่โค้ด"de" ส่วนการออกจะใส่โค้ด"en" และก็จะส่งข้อมูลพร้อมโค้ดปะหน้าไปยังคอมพิวเตอร์เพื่อเช็คว่าบัตรที่รูดมีอยู่ในฐานข้อมูลหรือไม่ ถ้ามีชื่ออยู่ก็จะส่งโค้ดกลับมาบอกว่าการ์ดผ่าน("cv")ระบบก็จะทำงานต่อได้ แต่ถ้าไม่ผ่านก็จะส่งโค้ด"ci" มาทำให้คอนโทรลเลอร์ปฏิเสธการการบริการผู้ใช้ต่อ ระบบจะเริ่มให้รูดบัตรใหม่จนกว่าจะถูก กลับมาที่บัตรผ่านเมื่อเป็นโหมดออกระบบก็จะเข้าสู่โหมดล็อก แต่สำหรับโหมดเข้าหลังจากที่คอมส่งรหัส"cv"มาแล้วก็จะหน่วงเวลาประมาณ1วินาทีเพื่อส่งชื่อเจ้าของบัตรไปแสดงที่LCDเป็นการทักทาย และอีก1วินาทีก็จะส่งรหัสผ่านของบัตรนั้นมาให้ไมโครคอนโทรลเลอร์เก็บไว้ จากนั้นไมโครคอนโทรลเลอร์ก็จะถามรหัสผ่านของผู้ใช้แล้วนำม่าเปรียบเทียบกันกับที่คอมพิวเตอร์ส่งมาหากถูกต้องระบบก็จะหยุดประตูจะเปิดให้เข้าได้ แต่ถ้าผิดก็จะวนกลับไปให้กดรหัสใหม่จำนวน3ครั้ง หากไม่ถูกเลยระบบก็จะล็อกเหมือนเดิม โดยแผนผังการทำงานจะเป็นไปตามรูปที่ 3.13

5) การทำงานในโหมด2 (โหมดแสดงสถานะไฟ) โหมดนี้จะแตกต่างกับ2โหมดก่อนเพราะไม่มีบัตรแม่เหล็กเข้ามาเกี่ยวข้อง เริ่มต้นจากเมื่อมีการ โหลดฟอร์มแสดงสถานะไฟที่คอมพิวเตอร์ก็จะมีการส่ง โค้ด"m2" มายังคอนโทรลเลอร์ซึ่งแน่นอนว่าจะต้องเกิดการอินเตอร์รัพท์พอร์ทอนุกรมขึ้น และก็จะมีการกระโดดไปยังโปรแกรมหลักของการแสดงสถานะหลอดไฟ ซึ่งจะเป็นการส่งค่าของพอร์ทที่ต่ออยู่กับตัวเช็คสถานะ(Opto Isolate) ซึ่งจะบอกได้ว่าขณะนี้หลอดไฟเปิดหรือปิดอยู่พร้อมทั้งใส่โค้ด2ตัวหน้า "op" ไปยังคอมพิวเตอร์ จากนั้นคอมพิวเตอร์ปลายทางก็จะนำค่าพอร์ททั้งพอร์ท ไปทำการเช็คสภาวะ0หรือ1 กลายเป็นสถานะของหลอดไฟแต่ละดวงได้ จากนั้นก็จะขึ้นแสดงบนหน้าจอคอมว่าหลอดดวงไหน ติดอยู่หรือดับอยู่บ้าง โดยจะส่งสถานะนี้ไปหาคอมพิวเตอร์อยู่เรื่อยๆตราบเท่าที่ยังทำงานอยู่ในช่วงโปรแกรมหลักนี้อยู่ และก็จะเปิดช่วงให้มีการรอรับอินเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รัพท์ประมาณ2วินาทีกว่า เพื่อให้ผู้ใช้หน้าคอมพิวเตอร์สามารถที่จะคลิ้กเพื่อสั่งเปิดปิดไปได้ โดยในช่วงที่รออินเตอร์รัพท์อยู่หากผู้ใช้ต้องการคลิ้กเพื่อเปิดหรือปิดไฟดวงใดก็จะส่งโค้ดของหลอดดวงนั้นเพิ่มด้วยโค้ด "Zแล้วตามด้วยหมายเลขของหลอดไฟ" ซึ่งก็จะทำให้เกิดการอินเตอร์รัพท์และก็จะกระโดดเข้าโปรแกรมย่อยการสั่งการป้อนข้อมูลให้หลอดไฟซึ่งก็จะนำข้อมูลจากโค้ดที่รับมาไปทำการเช็คสถานะของหลอดว่าเป็นติดหรือดับอยู่ และสถานะขาเบสของทรานซิสเตอร์ที่ใช้ทริกว่าเป็น1 หรือ 0 อยู่ เนื่องจากสวิทช์เป็นแบบเปิดได้ 2 ทางจะกระตุ้น 0 หรือ1 เพื่อให้ติดหรือดับก็ย่อมได้ ซึ่งต้องดูเทียบกันด้วยว่า หากไฟติดอยู่(Opto=1) และขาเบสเป็น1 เราต้องป้อน 0เพื่อให้ไฟดับ แต่ถ้าขาเบสเป็น0 เราต้องป้อน1 เพื่อให้ไฟดับเป็นต้น และเมื่อไม่มีการอินเตอร์รัพท์หรือคอมพิวเตอร์ส่งข้อมูลมา ระบบก็จะกลับไปส่งค่าสถานะไปให้คอมพิวเตอร์แทน สลับกัน แสดงได้ดังรูปที่ 3.16

6) โปรแกรมย่อยการอ่านข้อมูลจากบัตรแม่เหล็ก เริ่มจากเมื่อมีบัตรผ่านเข้ามายังเครื่องอ่านบัตรแม่เหล็กจะเกิดฟลักซ์แม่เหล็กตัดกันส่งผลให้ขาCard Present ตกลงเป็น0 และจะเป็นศูนย์ตลอดจนกว่าจะรูดสุดบัตรแม่เหล็ก จากนั้นก็จะเป็นการเช็คหาตัวอักษรเริ่มต้น(1011b)ซึ่งบิตแรกที่จะพบคือ1 ดังนั้นเราจึงหาบิตแรกที่เป็น1ให้พบ โดยในการอ่านข้อมูลจากบัตรทีละบิตนั้นจะต้องอ่านค่าที่ขา $\overline{DATA}$ ทุกครั้งที่ขา Strobe=0 ซึ่งควรจะหน่วยเวลาสักนิดเพื่อกันโอกาสที่จะรับข้อมูลผิดพลาด และเนื่องจากขา $\overline{DATA}$ เป็นขอบขาลง เราจึงต้องอ่านข้อมูลแบบกลับ ในที่นี้เราควรต้องหาบิตเริ่มต้นที่ขา $\overline{DATA}$ =0 นั่นเอง ดังนั้นหากพบแล้วให้นำไปเก็บไว้ในCarry Flag(C) จากนั้นก็ทำการRotateไปทางขวาให้ครบทั้ง5บิต(ข้อมูลในบัตรมี5บิต บิตที่5เป็นพาริตีคี่) จากนั้นก็เลื่อนไปทางขวาอีก 3 ครั้งเพื่อให้บิตแรกที่รับเข้ามาเป็นLSB นำไปเก็บไว้ในหน่วยความจำตามที่ต้องการพร้อมทั้งคอมพลิเมนต์ จากนั้นแล้วก็เพิ่มตำแหน่งขึ้นเพื่อจะได้รับตัวอักษรถัดมา(5บิตถัดไป)ในขั้นตอนเดิมไปเรื่อยๆจนครบ เมื่อCard Presentกลับไปเป็น1อีกครั้งเป็นอันสิ้นสุดการอ่านข้อมูล แผนภาพแสดงการทำงานดังรูปที่ 3.18

7) โปรแกรมตรวจเช็คความผิดพลาดของข้อมูลที่รูดจากบัตร แบ่งเป็น 3 แบบได้แก่

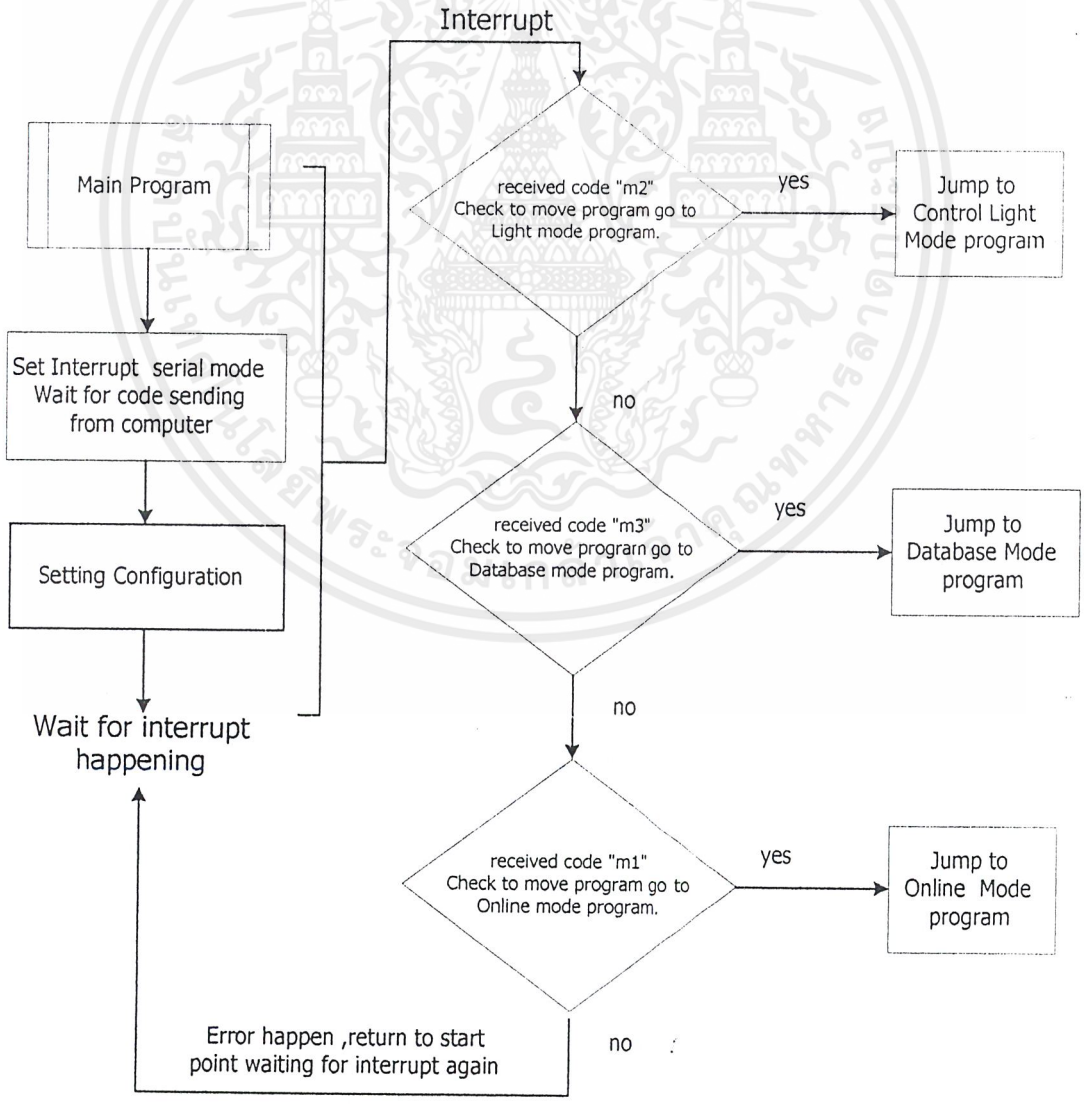
7.1) โปรแกรมเช็คพาริตี้ ในกรณีที่มีการเช็คพาริตี้ สำหรับบัตรแม่เหล็กจะมีการเก็บบิตพาริตีเป็นบิตที่5ของข้อมูลซึ่งเป็นพาริตีชนิดคี่ ดังนั้นเมื่อเราจะทำการเช็คพาริตีโดยไมโครคอนโทรลเลอรีเราต้องทำการเช็คทั้ง8บิต(ซึ่งก็คือรวมบิตพาริตีคี่ไปด้วย) บิตอื่นๆที่ไม่ใช่ข้อมูลจะทำให้เป็น0 ซึ่งสามารถทำได้ด้วยการดูที่บิต7ของPSW(Program Status Word) ซึ่งเป็นพาริตีคู่ ดังนั้นการคิดพาริตีคู่โดยรวมบิตที่5ที่เป็นพาริตีคี่เข้าไปด้วยถ้าถูกต้องจะได้พาริตีคู่เป็น0เสมอ หากได้1 แสดงว่าข้อมูลมีการผิดพลาดเกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.2) โปรแกรมเช็คLRC ในบัตรแม่เหล็กจะมีอัขระที่เป็นตัวเช็คความถูกต้องของข้อมูลเก็บไว้อยู่หลังอักขระจบ ซึ่งเป็นการนำข้อมูลทุกตัวมาทำการExclusive ORกันทั้งหมด(ข้อมูลเพียง4บิต ไม่นับบิตพาริตี) ดังนั้นเราจะนำข้อมูลทุกข้อมูลที่เก็บไว้มา mark พาริตีออกโดยการANDกับ 00001111b แล้วนำมาExclusiveกันให้หมด ค่าที่ได้จะนำมาเทียบกับข้อมูลหลังอักขระจบ(LRC) โดยนำมาลบกัน หากได้เป็นศูนย์ก็แสดงว่าไม่ผิดพลาด แผนผังแสดงได้ดังรูปที่ 3.20

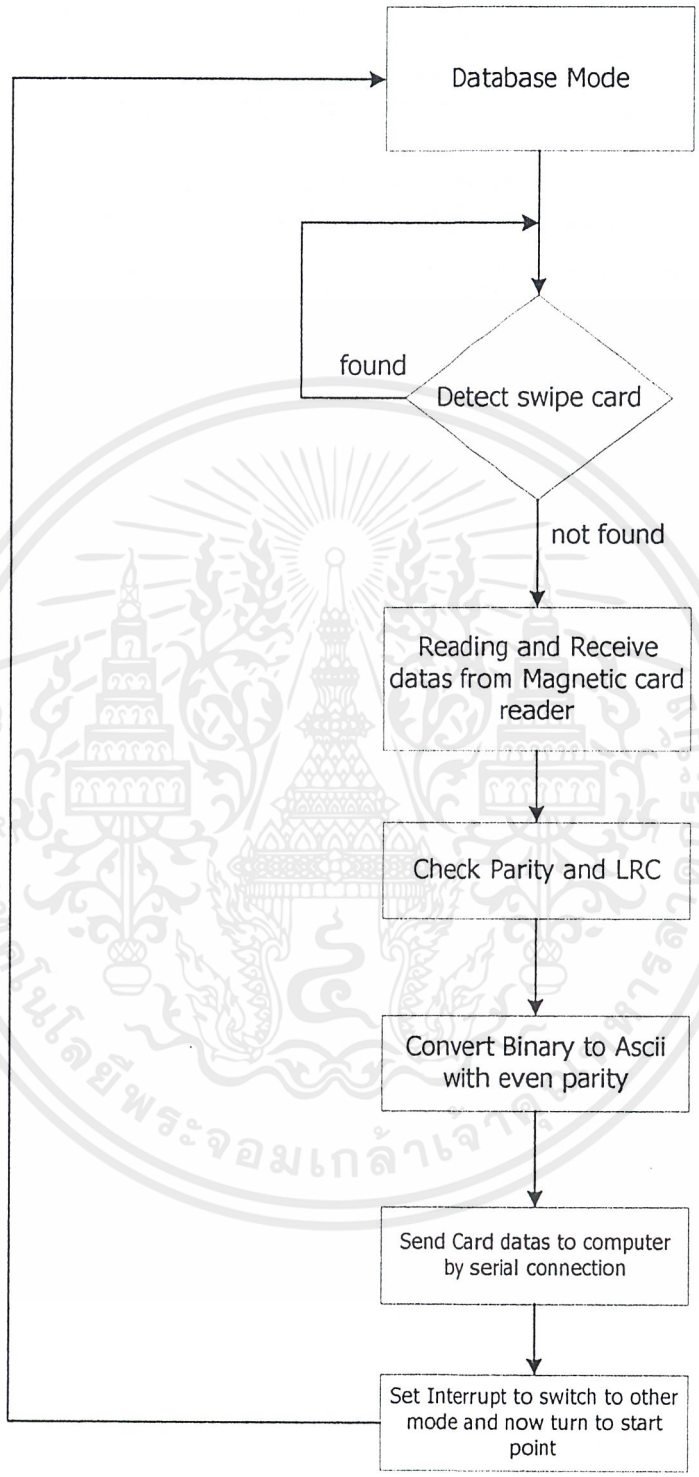
7.3) โปรแกรมเช็คอักขระเริ่มและอักขระจบ ในส่วนนี้ไม่มีอะไรมากแค่เพียงหาอักขระเริ่มต้น(00001011b) และหาอักขระจบ (00001111) ให้พบ หากไม่พบก็จะถือว่าการอ่านข้อมูลเกิดความผิดพลาด

แผนผังแสดงการทำงานของโปรแกรมทั้งหมด



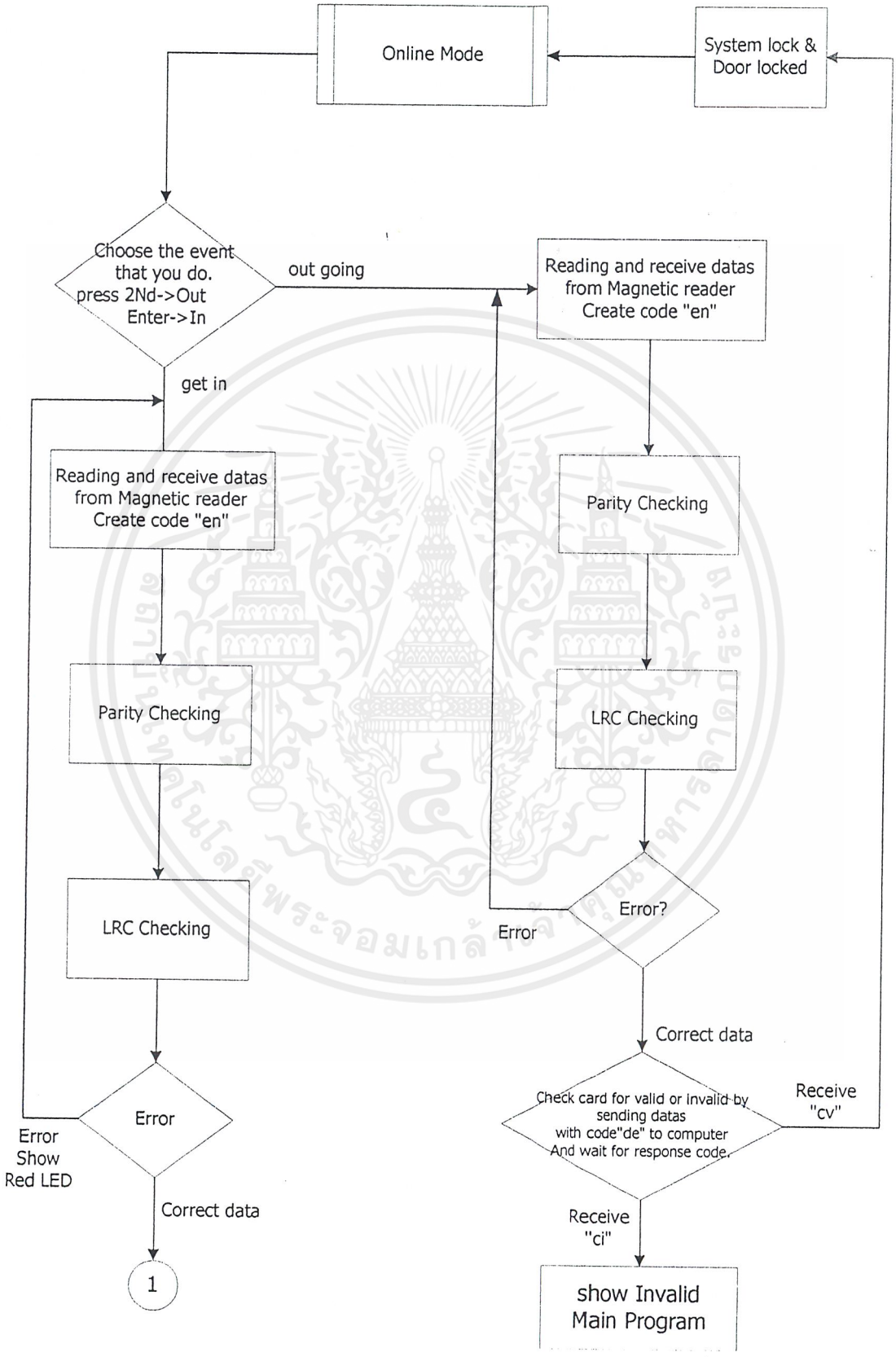
รูปที่ 3.11 แผนผังการทำงานของโปรแกรมหลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



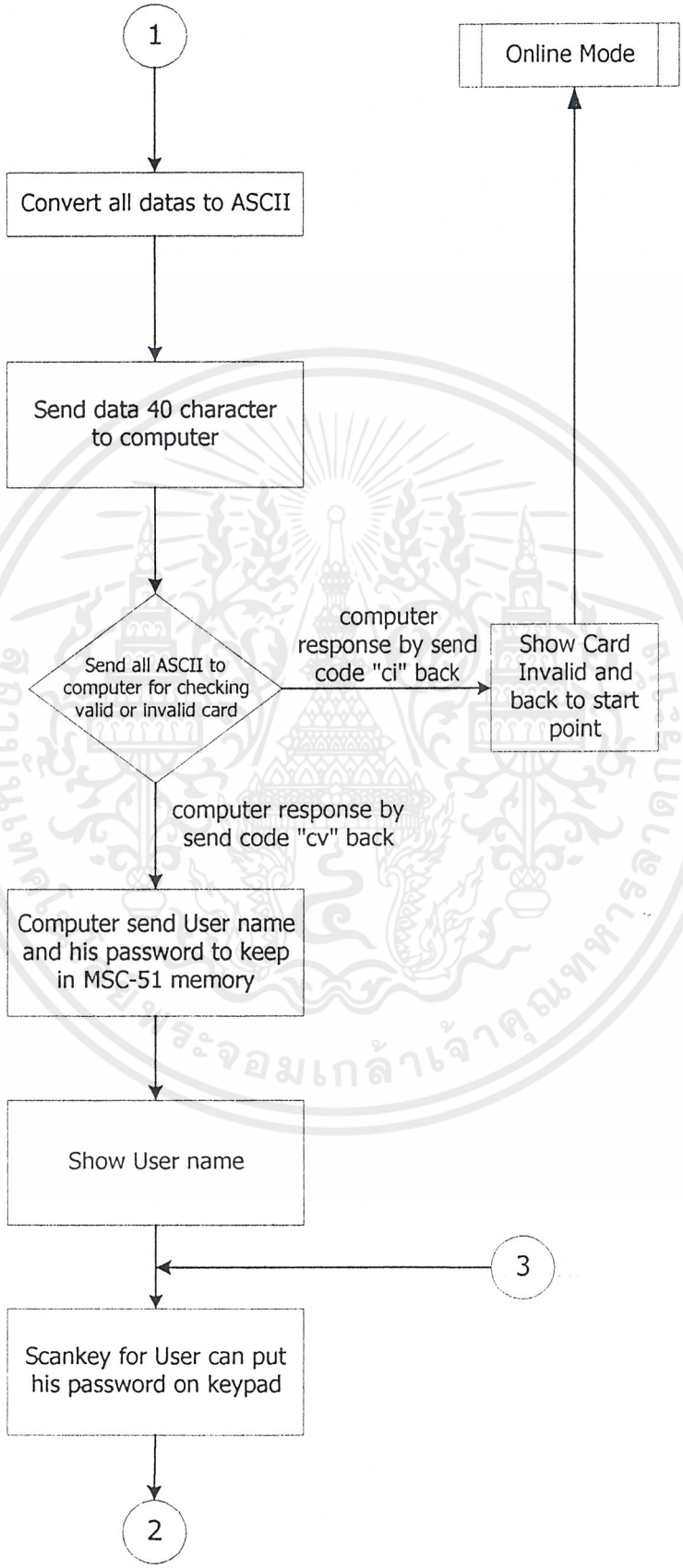
รูปที่ 3.12 แผนผังการทำงานของโปรแกรมหลักในช่วงโหมดของฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



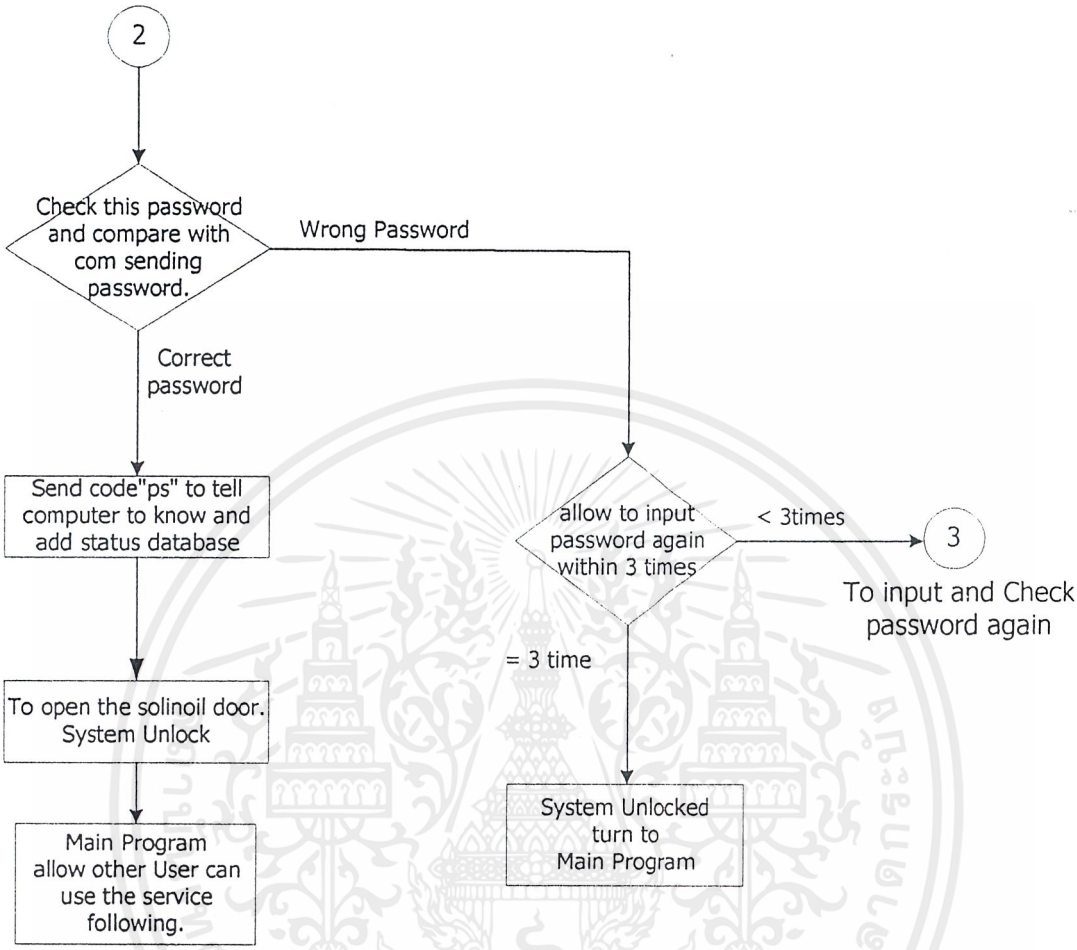
รูปที่ 3.13 แผนผังการทำงานของโปรแกรมหลักในช่วงโหมดการออนไลน์(1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

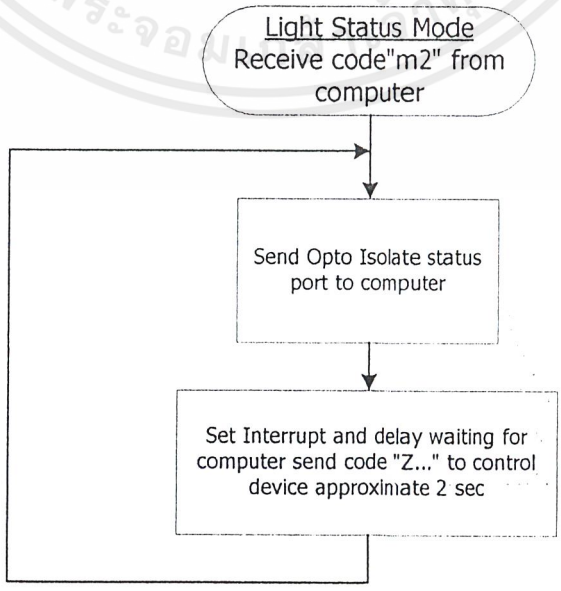


รูปที่ 3.14 แผนผังการทำงานของโปรแกรมหลักในช่วงโหมดการออนไลน์(2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

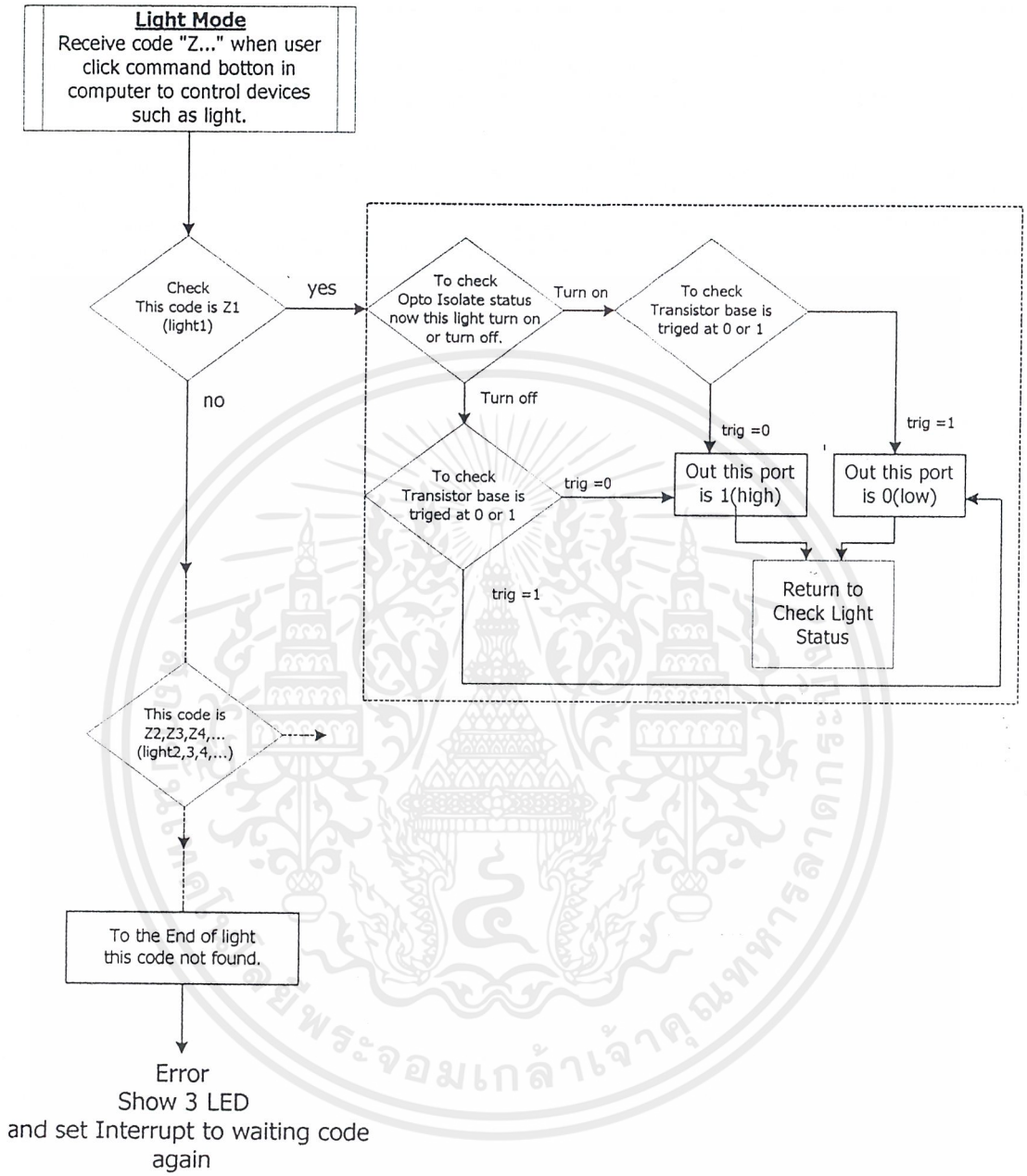


รูปที่ 3.15 แผนผังการทำงานของโปรแกรมหลักในโหมดการออนไลน์(3)



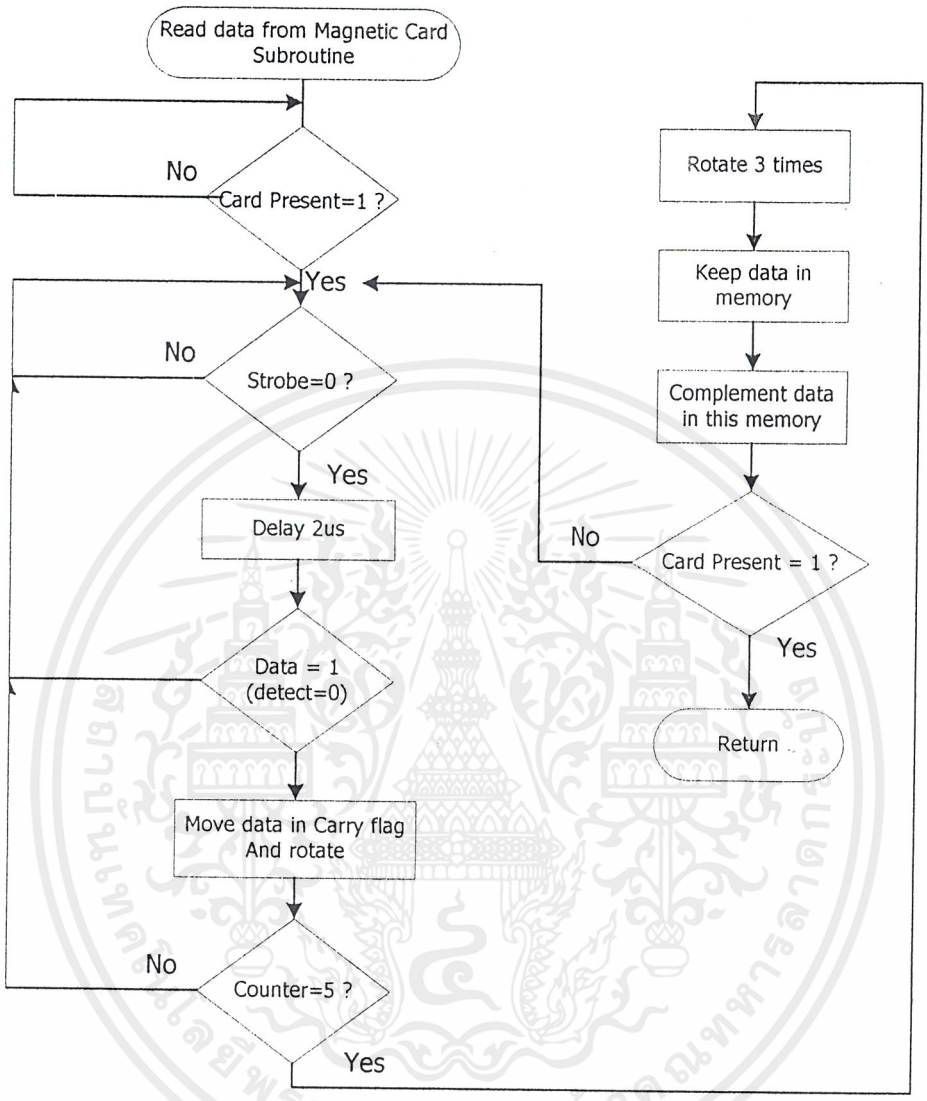
รูปที่ 3.16 แผนผังแสดงการทำงานในโหมดแสดงสถานะไฟปรากฏหน้าคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



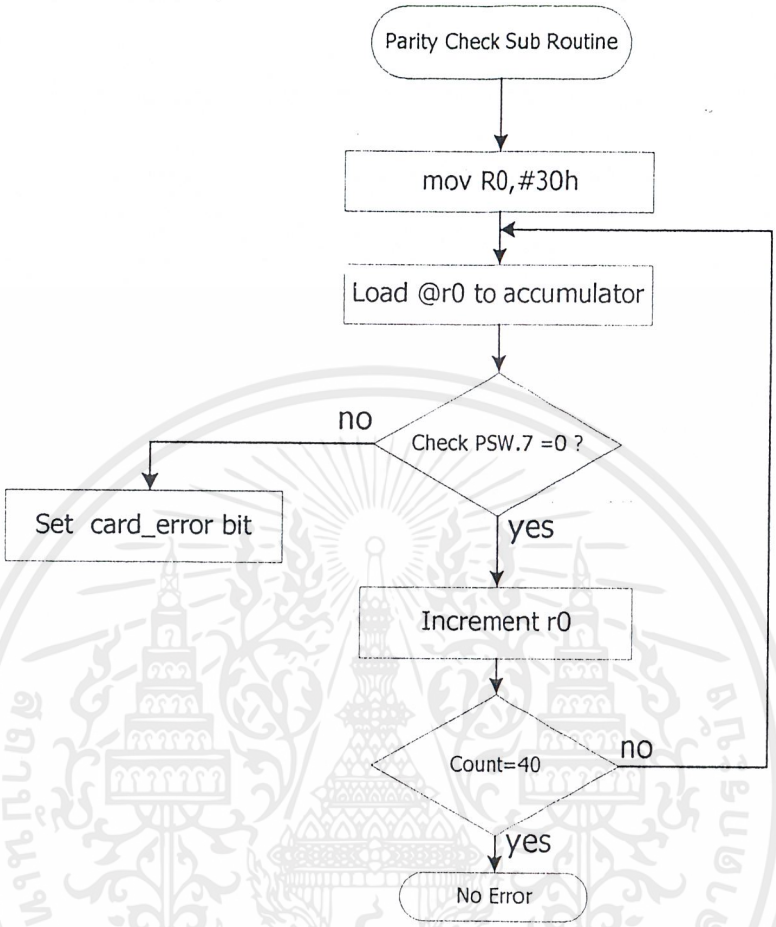
รูปที่ 3.17 แผนผังการทำงานของการควบคุมอุปกรณ์ไฟฟ้าเมื่อมีผู้ส่งโค้ดมาทำการควบคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



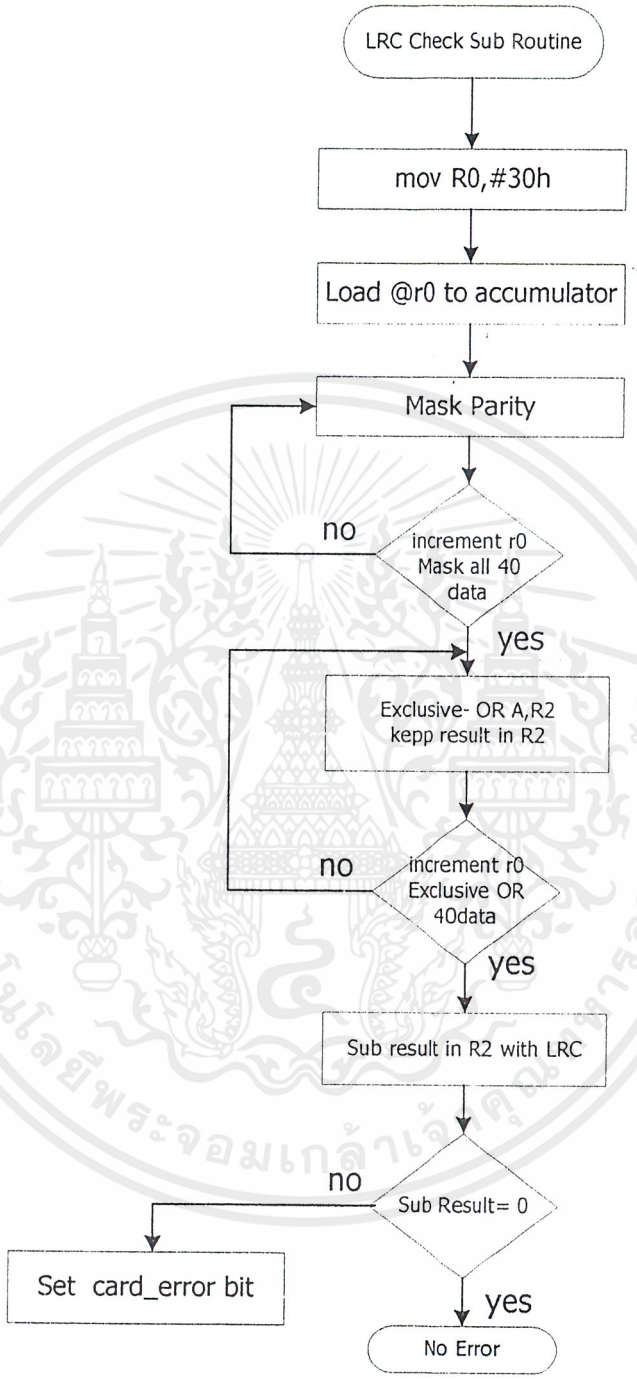
รูปที่ 3.18 แผนผังการทำงานของโปรแกรมย่อยอ่านและรับค่าจากเครื่องอ่านบัตรแม่เหล็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.19 แผนผังการทำงานของโปรแกรมย่อยการเช็คพาริตีของบัตรแม่เหล็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.20 แผนผังการทำงานของโปรแกรมย่อยการเช็คLRC ของอักขระท้ายบนบัตรแม่เหล็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 ในส่วนของโปรแกรมควบคุมการส่งข้อมูลของพอร์ทอนุกรมบนคอมพิวเตอร์

สำหรับการเขียนโปรแกรมควบคุมการส่งข้อมูลบนคอมพิวเตอร์จะใช้คอนโทรล MSComm (Communication) ซึ่งเป็นคอลโทรลตัวหนึ่งที่ช่วยในการติดต่อกับพอร์ตอนุกรม (Serial Port) ซึ่งสามารถทำการรับ-ส่งข้อมูลผ่านทางพอร์ตอนุกรมได้ด้วยคอนโทรลนี้ โดยจะใช้การตอบสนองต่อเหตุการณ์แบบ Event-Driven นั่นก็คือคอนโทรลจะทำหน้าที่ตรวจสอบการเกิดขึ้นหรือการร้องขอให้เกิดเหตุการณ์ต่าง ๆ กับพอร์ตอนุกรมโดยอัตโนมัติ เช่นมีการส่งข้อมูลออกหรือรับข้อมูลเข้ามา และเมื่อพบจะเกิดเหตุการณ์(Event) CommEvent กับ OnCommEvent ขึ้น ซึ่งจะมีการกระโดดเข้าไปทำงานในโพรซีเยอร์เหตุการณ์OnComm เหมือนกับการเกิดอินเตอร์รัพท์ในMCS-51 คุณสมบัติที่ใช้ติดต่อกับพอร์ทอนุกรมที่จำเป็นในโครงงานนี้มีรายละเอียดดังต่อไปนี้

การตั้งค่าคุณสมบัติที่เกี่ยวข้องกับการส่ง-รับข้อมูลของพอร์ทอนุกรมที่ใช้ในโครงงานนี้

- **MSComm1.Comport =1** (เป็นการกำหนดพอร์ทอนุกรมที่จะติดต่อด้วยเช่นCom1,Com2)

- **MSComm1.Setting = "2400,e,7,1"** (เป็นการกำหนดคุณสมบัติของการส่งข้อมูลว่า ส่งด้วยอัตราบอด2400,ให้มีการตรวจพาริตีคู่,ข้อมูลเป็นแบบแอสกี 7 บิต โดยบิตที่8เป็นบิตพาริตี,และจำนวนบิตจบ)

- **MSComm1.Inputlen = 0** (กำหนดจำนวนอักษรที่ดึงมาจากบัฟเฟอร์รับข้อมูลในแต่ละครั้งที่มีการเรียกคุณสมบัติInput โดยหากเซ็ตเป็น0 จะหมายถึงให้ดึงข้อมูลมาทั้งหมดที่มีอยู่ในบัฟเฟอร์)

- **MSComm1.PortOpen = True** (เป็นการเปิดพอร์ทให้ใช้งานได้)

- **MSComm1.Rthreshold =จำนวนตัวอักษร** (เป็นการกำหนดจำนวนตัวอักษรที่รับเข้ามาที่ทำให้เกิดเหตุการณ์ CommEvent กับ OnCommEvent ขึ้นซึ่งก็จะมีการเรียกโปรแกรมย่อยOnComm ซึ่งจะมีการเขียนโปรแกรมตรวจสอบตัวเลขของเหตุการณ์ที่เกิดขึ้นจากคุณสมบัติ CommEvent ว่าอยู่ในสภาวะใดหากเป็นสภาวะรับข้อมูล(ComEvReceive) ก็จะมีการเรียกคุณสมบัติInput เพื่อนำข้อมูลที่รับเข้ามาไปประมวลผลต่อ หากคุณสมบัติRthresholdนี้ถูกตั้งให้เป็น0 จะหมายความว่าจะไม่มีการเกิดเหตุการณ์ CommEvent กับ OnCommEvent ขึ้น ซึ่งเราจะต้องทำการตรวจเช็คสภาวะเหตุการณ์เอง

- **MSComm1.Output ="ข้อมูลเป็นตัวอักษร"** (เป็นการส่งข้อมูลออกนอกพอร์ทอนุกรม)

- **Variable = MSComm1.Input** (เป็นการดึงข้อมูลที่รับเข้ามาเก็บไว้ในบัฟเฟอร์มาเก็บไว้ในตัวแปรVariableที่เป็นStringหรือVariant ซึ่งขึ้นกับสถานะของInputlen ว่าจะให้ดึงข้อมูลมาจากบัฟเฟอร์ครั้งละกี่ตัวอักษร)

- **MSComm1.InBufferCount = 0** (เป็นคุณสมบัติให้นับจำนวนตัวอักษรที่ถูกเก็บค้างอยู่ในบัฟเฟอร์ แต่ถ้าหากเรากำหนดให้เท่ากับ 0 จะเป็นการเคลียร์ข้อมูลในบัฟเฟอร์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เหตุการณ์ **OnComm**

เกิดขึ้นเมื่อมีการเปลี่ยนแปลงค่าของคุณสมบัติ CommEvent ซึ่งเป็นการบอกถึงการเกิดข้อผิดพลาดหรือมีการสื่อสารเกิดขึ้นก็ได้ ซึ่งจะมีรูปแบบโพรซีเยอร์เหตุการณ์ ดังนี้

Private Sub object.OnComm ()

โดยปกติเมื่อคอนโทรล MSComm มีการเรียกโพรซีเยอร์เหตุการณ์ OnComm เรามักจะมีการเขียนโค้ดภายในโพรซีเยอร์เหตุการณ์นี้ เพื่อทำการตรวจสอบค่าของคุณสมบัติ CommEvent ทั้งนี้เพื่อตรวจสอบสถานะของการสื่อสารหรือข้อผิดพลาดที่เกิดขึ้นนั่นเอง ดังนี้

1. ในการรับข้อมูลเข้า เหตุการณ์ที่ใช้มากที่สุดก็คือ Event **ComEvReceive** ซึ่งเราใช้ในการเขียนโปรแกรมเพื่อตอบสนองข้อมูลที่ส่งเข้ามายังพอร์ทอนุกรมของคอมพิวเตอร์(รับข้อมูล) ซึ่งก็คือเมื่อคอมพิวเตอร์มีการรับข้อมูลที่พอร์ทอนุกรมขึ้นจะเกิดเหตุการณ์นี้ขึ้นทุกครั้งเปรียบเสมือนมีการอินเตอร์รัพท์ของไมโครคอนโทรลอร์ซึ่งได้ตั้งค่าไว้ที่ **MSComm1.Rthreshold = จำนวนตัวอักษรที่ต้องการให้เกิดเหตุการณ์OnCommขึ้น** ซึ่งในโครงงานนี้จะใช้ในโหมดนี้ตลอด เนื่องจากในการรับข้อมูลแต่ละครั้งเราจะทราบจำนวนข้อมูลที่รับมาตายตัวทำให้เราสามารถตั้งค่าคุณสมบัติRthreshold ตามจำนวนข้อมูลที่จะถูกส่งมาได้ หมายความว่าข้อมูลทั้งหมดหากส่งครบเมื่อไรก็จะมีจำนวนเท่ากับ Rthreshold พอดีก็จะเกิดเหตุการณ์OnCommขึ้นเมื่อข้อมูลถูกส่งมาครบนั้นเอง จากนั้นเราก็ไปเขียนโปรแกรมรับค่าข้อมูลภายใต้เงื่อนไขของเหตุการณ์นั้นโดยใช้ ตัวแปร **=MSComm1.input** จะเป็นการรับข้อมูลทั้งหมดในบัฟเฟอร์ที่ได้ส่งมา ไปเก็บในตัวแปรที่มารับค่านั่นเอง ทั้งนี้เนื่องจากเราได้ตั้งคุณสมบัติ **MSComm1.Inputlen = 0** ไว้แล้วเป็นการกำหนดสภาวะการรับข้อมูลว่าให้ดึงมาครั้งละกี่ตัว(การเซตให้เท่ากับ 0 เป็นการระบุว่าต้องการรับข้อมูลทั้งหมดในทีเดียว) เมื่อรับข้อมูลมาเก็บไว้ในจตัวแปรได้แล้วเราก็สามารถนำข้อมูลนั้นมาใช้งานได้

2. ในการส่งข้อมูลออก ในที่นี้การส่งข้อมูลเราสามารถเป็นผู้ระบุเองได้ว่าต้องการส่งอะไรเป็นจำนวนเท่าใด ดังนั้นเราจึงไม่จำเป็นต้องให้มีอินเตอร์รัพท์เวลาส่งข้อมูลออก ซึ่งต่างกับเวลารับเนื่องจากเราจะไม่มีทางทราบได้เลยว่าข้อมูลจะมาถึงเมื่อไร นอกจากเราจะทำการเช็คเหตุการณ์หรือจำนวนข้อมูลในบัฟเฟอร์ตลอดเวลา แต่สำหรับในการส่งเรารู้ว่าเมื่อไรจะส่งเพราะเราเป็นผู้ส่งเอง ดังนั้นเราจึงให้คุณสมบัติ **MSComm1.Sthreshold = 0** เพื่อไม่ให้เกิดเหตุการณ์ OnComm ขึ้นเมื่อเราส่งข้อมูล เราจะสามารถส่งข้อมูลออกได้โดย **MSComm1.Output = ข้อมูลที่จะส่ง** เทานี้ข้อมูลก็จะถูกส่งออกโดยทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.4 <u>โปรแกรมคอมพิวเตอร์</u> ในส่วนนี้เป็นโปรแกรมที่ใช้ติดต่อ(Interface) กับผู้ใช้ที่อยู่หน้าคอมพิวเตอร์ ซึ่งอาจจะเป็นผู้คุมระบบ ดูแลฐานข้อมูลหรือผู้ใช้บริการที่ต้องการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า ซึ่งได้เลือกใช้โปรแกรมวิชวลเบสิก เวอร์ชัน6.0

ในส่วนของฟอร์ม2,ฟอร์ม4 และฟอร์ม7 เป็นฟอร์มที่มีหน้าที่เกี่ยวกับฐานข้อมูล สภาวะออนไลน์ และฟอร์มควบคุมอุปกรณ์ไฟฟ้า ตามลำดับ ซึ่งจะมีการใช้คอนโทรลMSComm ในการติดต่อรับ-ส่งข้อมูลผ่านพอร์ทอนุกรม ส่วนฟอร์มอื่นๆจะใช้สำหรับทำงานภายในคอมพิวเตอร์ เช่น เกี่ยวกับฐานข้อมูล ซึ่งไม่มีการติดต่อกับอุปกรณ์ฮาร์ดแวร์ภายนอก ดังนั้นจะไม่กล่าวถึง

1) ฟอร์ม2 เป็นฟอร์มที่เป็นตัวจัดการกับฐานข้อมูลในระบบซึ่งภายในโครงงานนี้จะใช้ตัวแปรที่เป็นอ็อปเจ็คข้อมูลในการติดต่อกับฐานข้อมูลซึ่งจะช่วยให้การจัดการข้อมูลสามารถทำได้อย่างมีประสิทธิภาพและมีความยืดหยุ่นในการใช้งานลักษณะต่าง ๆ มากกกว่า Data Control ซึ่งได้เลือกใช้อ็อปเจ็คข้อมูลเป็นชนิดไดนาเซ็ต (Dynaset Type Recordset Object) ซึ่งเป็นอ็อปเจ็คข้อมูลที่สามารถเลือกนำข้อมูลในบางฟิลด์หรือทุก ๆ ฟิลด์ของตารางหรือแม้แต่จะนำฟิลด์ข้อมูลจากหลาย ๆ ตารางมารวมกันก็ได้โดยจะใช้คำสั่ง Select ของ SQL ใน การกำหนดโครงสร้างและเลือกดึงข้อมูลตามความต้องการได้

ในการทำงานกับฐานข้อมูลทุก ๆ อย่างในวิชวลเบสิก จะถูกจัดการฐานข้อมูลหรือ JET ดังนั้นโปรแกรมที่ต้องการทำงานกับฐานข้อมูลจะต้องกำหนดรายละเอียดและวิธีการติดต่อกับระบบจัดการฐานข้อมูลเสียก่อนซึ่งทำได้โดยการสร้างอ็อปเจ็คพื้นที่ทำงานกับระบบจัดการฐานข้อมูล (Workspace Object) ดังนี้

Dim ชื่อตัวแปร(ws_object) AS Workspace

Dim ชื่อตัวแปร(db_object) AS Database

Dim ชื่อตัวแปร(dyn_object) AS Recordset (ตัวแปรอ็อปเจ็คข้อมูลที่ใช้ในการติดต่อกับฐานข้อมูล)

หลังจากที่ได้อ็อปเจ็คพื้นที่ทำงานกับระบบจัดการข้อมูลแล้ว ก่อนจะสามารถอ่านโครงสร้างหรือจัดการใด ๆ กับข้อมูลที่อยู่ภายในฐานข้อมูลนั้นได้ ฐานข้อมูลที่ต้องการนั้นจะต้องถูกเปิด (Open) ก่อน ในลักษณะเช่นเดียวกับการเปิดแฟ้มข้อมูลปกติทั่วไปก่อนที่จะทำการอ่านหรือเขียนแฟ้มข้อมูลนั้น

Set ws_object = DBEngine.workspace(0)

Set db_object = ws_object.OpenDatabase (" Path ที่เก็บ Database ",False(เป็นการใช้ฐานข้อมูลร่วมกับผู้อื่น),False(ใช้ฐานข้อมูลแบบมีการแก้ไขได้))

จากนั้นเป็นขั้นตอนการสร้างฐานข้อมูลจำลองขึ้นโดยเลือกดึงโครงสร้างจากฐานข้อมูลจริงมาเก็บไว้ในตัวแปร dyn_object โดยใช้คำสั่ง SQL ในการเลือกดึงในแบบที่ต้องการได้ ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Set dyn_object = ProjectDB.OpenRecordset("คำสั่ง SQL")

สำหรับคำสั่ง SQL ที่ใช้มีโครงสร้างดังนี้

SELECT ฟิลด์ข้อมูลที่ต้องการดึง FROM ตารางที่ต้องการติดต่อ WHERE 'โดยมีเงื่อนไข' ORDER BY ฟิลด์ที่ต้องการเรียงลำดับ

ดังนั้นในส่วนของ From ซึ่งใช้เป็นตัวจัดการฐานข้อมูลโดยตรงจึงประกอบไปด้วยคำสั่งดังต่อไปนี้

dyn_object.AddNew เป็นการเพิ่มเร็คคอร์ดว่างเพื่อทำการเพิ่มข้อมูล ซึ่งตัวชี้เลื่อนมาชี้ที่นี่

dyn_object.edit เป็นการแก้ไขเร็คคอร์ดที่กำลังชี้อยู่

dyn_object.delete เป็นการลบเร็คคอร์ดที่กำลังชี้อยู่

ซึ่ง คุณสมบัติAddNew และ Edit จะทำได้โดยสมบูรณ์ก็ต่อเมื่อมีการกดปุ่ม Save ด้วยทุกครั้ง โดยที่สามารถกดปุ่มCancel เพื่อยกเลิกการกระทำคำสั่งนั้นได้

เมื่อมีการSaveจะต้องมีการกำหนดค่าให้กับแต่ละฟิลด์ในฐานข้อมูล dyn_object("ชื่อฟิลด์") = ข้อมูลที่เป็นชนิดเดียวกับข้อมูลในฟิลด์นั้น แล้วตามด้วย dyn_opject.Update dyn และ object.Requery เป็นการยืนยันในการเปลี่ยนแปลงและแก้ไขฐานข้อมูลนั้น

หากเป็นการ Cancel จะมีการใช้คำสั่ง dyn_object.CancelUpdate ซึ่งจะสามารถทำได้หลังจากการใช้ฟังก์ชันเพิ่มหรือแก้ไขข้อมูลเท่านั้น

ส่วนหลักอีกส่วนหนึ่งของฟอร์มนี้คือการรอรับข้อมูลจากบัตรแม่เหล็กซึ่งเมื่อโหลดฟอร์มนี้ขึ้นมาจะมีการส่งโค้ด "M3" ออกไปให้ไมโครคอนโทรลเลอร์กระโดดไปทำงานในช่วงโปรแกรมรอรับค่าจากบัตรแม่เหล็กซึ่งเมื่อไรที่มีการรูดบัตรแม่เหล็กข้อมูลที่ถูกปะด้วยโค้ด de จะถูกส่งกลับมาที่คอมพิวเตอร์(จำนวน 42 ตัว เท่ากับที่กำหนดไว้ใน Rthreshold) ซึ่งเราก็สามารถรับข้อมูลทั้ง 42 ตัวในโปรแกรมย่อย OnComm ในเหตุการณ์ ComEvReceive ด้วยคำสั่ง str = MSComm1.input จากนั้นก็จะนำข้อมูลนั้นไปตัดออกเลือกเฉพาะข้อมูลที่อยู่ระหว่างตัวอักขระเริ่มต้นกับอักขระสุดท้ายซึ่งเป็น login name ของผู้ใช้แต่ละคน ซึ่งก็จะนำค่านี้ไปหารายชื่อและรายละเอียดอื่น ๆ ของผู้ใช้นำมาแสดงบนหน้าจอ โดยใช้คำสั่งของ SQL คือ

SELECT * FROM member(ดึงทุกฟิลด์มาจากตารางชื่อmember) WHERE member.id ='login.text'

แต่ถ้าหากไม่พบ login นี้ก็จะทำการถามผู้ใช้ว่าต้องการเพิ่มข้อมูลนี้เข้าไปสู่ฐานข้อมูลหรือเปล่าเป็นต้น

2) ฟอร์ม 4 เป็นฟอร์มใช้ที่ใช้สำหรับให้ผู้ใช้บริการล็อกอินเข้าสู่ระบบโดยในฟอร์มนี้จะมีทั้งการรับข้อมูลจากบัตรแม่เหล็กและการดึงข้อมูลจากฐานข้อมูลซึ่งคล้ายกับฟอร์ม1ที่ได้กล่าวมาแล้วซึ่งโหมดนี้ค่อนข้างมีความซับซ้อนมากกว่าโหมดอื่น เนื่องจากในเหตุการณ์ ComEvReceive

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะมีการเลือกเป็น 3 สถานะตามโค้ด 2 ตัวอักษรที่ส่งมาพร้อมกับข้อมูลเป็นการบอกให้เราทราบว่าข้อมูลที่ส่งมานั้นส่งมาเพื่อใช้ทำอะไร ซึ่งการที่จะตัดเอาโค้ด 2 ตัวหน้าออกจากข้อมูลทั้ง 42 ตัวทำได้โดยการใช้คำสั่ง Check = Mid$(Str(ชื่อตัวแปรที่ใช้รับอินพุท),1(ตำแหน่งที่เริ่มตัดคำ),2(ตัดไปเป็นจำนวน 2 ตัวอักษร)) เท่านี้ก็จะได้โค้ด 2 ตัวหน้ามาเก็บไว้ในตัวแปรชื่อ Check

หากโค้ดเป็น de หมายถึงเป็นกรณีที่คนรูดบัตรเพื่อเข้าบ้าน ส่วน en เป็นการรูดบัตรเพื่อออกจากบ้านทั้ง 2 กรณีจำเป็นที่จะต้องเขียนสถานะลงไปในตารางฐานข้อมูลชื่อ Status เพื่อเก็บวันที่เวลาและสถานะว่า IN หรือ OUT โดยในการเขียนข้อมูลลงในฐานข้อมูลก็จะต้องสร้างตัวแปรที่เป็น Recordset และทำการเปิดฐานข้อมูลทุกครั้งดังที่ได้กล่าวมาแล้ว จากนั้นก็ทำการเรียกใช้คุณสมบัติ AddNew และทำการกำหนดค่าใช้ฟิลด์ข้อมูลดังนี้ dyn_object("ชื่อฟิลด์") = ข้อมูลที่เป็นชนิดเดียวกับข้อมูลในฟิลด์นั้น แล้วตามด้วย dyn_opject.Update dyn และ object.Requery

เริ่มต้นเมื่อได้รับข้อมูลจากบัตรแม่เหล็กพร้อมกับโค้ด 2 ตัวหน้าก็จะต้องเลือกเฉพาะโค้ดไปเก็บไว้และเลือกส่วนที่เป็นข้อมูล 40 ตัวไปดึงข้อมูลจากฐานข้อมูลโดยใช้คำสั่ง SQL ดังที่ได้กล่าวมาแล้วในฟอร์ม2 ซึ่งหากพบ login นั้นอยู่จริงก็จะนำเอาข้อมูลของผู้รูดบัตรมาโชว์ในTextbox ซึ่งการโชว์ข้อมูลจะทำได้โดย Textbox.text = dyn_object.Fields ("ชื่อฟิลด์ที่จะดึงมา") ซึ่งจะต้องเช็คและทำการส่งโค้ดกลับไปบอกไมโครคอนโทรลเลอร์ว่าบัตรที่รูดนั้นมีข้อมูลอยู่ในฐานข้อมูลจริงหรือไม่หากมีก็จะส่งโค้ด "cv" แต่ถ้าไม่มีการจะส่งโค้ด "ci" ซึ่งถ้าหากเป็น "cv" ก็จะดำเนินการต่อโดยถ้าเป็นการรูดเพื่อออกจากบ้านก็จะทำการเขียนสถานะการOUTเลยและระบบก็จะทำการล็อก แต่ถ้าเป็นการรูดเพื่อเข้า แต่ละครั้งในการส่งออกจะใช้ timer ในการAction ให้ทำการส่งโค้ดออกเพื่อจะได้กำหนดจังหวะในการส่งรับข้อมูลให้พอดีกันไม่ให้เกิดการชนกันของการส่งข้อมูลแต่ละครั้งโดยหลังจากส่ง "cv" ไปแล้วก็จะหน่วงเวลาไป1วินาทีแล้วก็จะทำการส่งชื่อUsernameไปให้ไมโครคอนโทรลเลอร์ และก็จะหน่วงเวลาประมาณ3วินาที เพื่อส่งรหัสผ่านต่อมา จากนั้นก็เป็นส่วนของไมโครคอนคทรลเลอร์ที่จะติดต่อกับผู้ใช้โดยหากผู้ใช้กดรหัสผ่านถูกต้องไมโครคอนโทรลเลอร์ก็จะส่งโค้ด "ps" กลับมาอีกครั้ง เป็นการยืนยันว่ารหัสผ่านถูกต้อให้ทำการบันทึกเวลาINได้ เป็นอันสิ้นสุดการทำงานของโหมดนี้

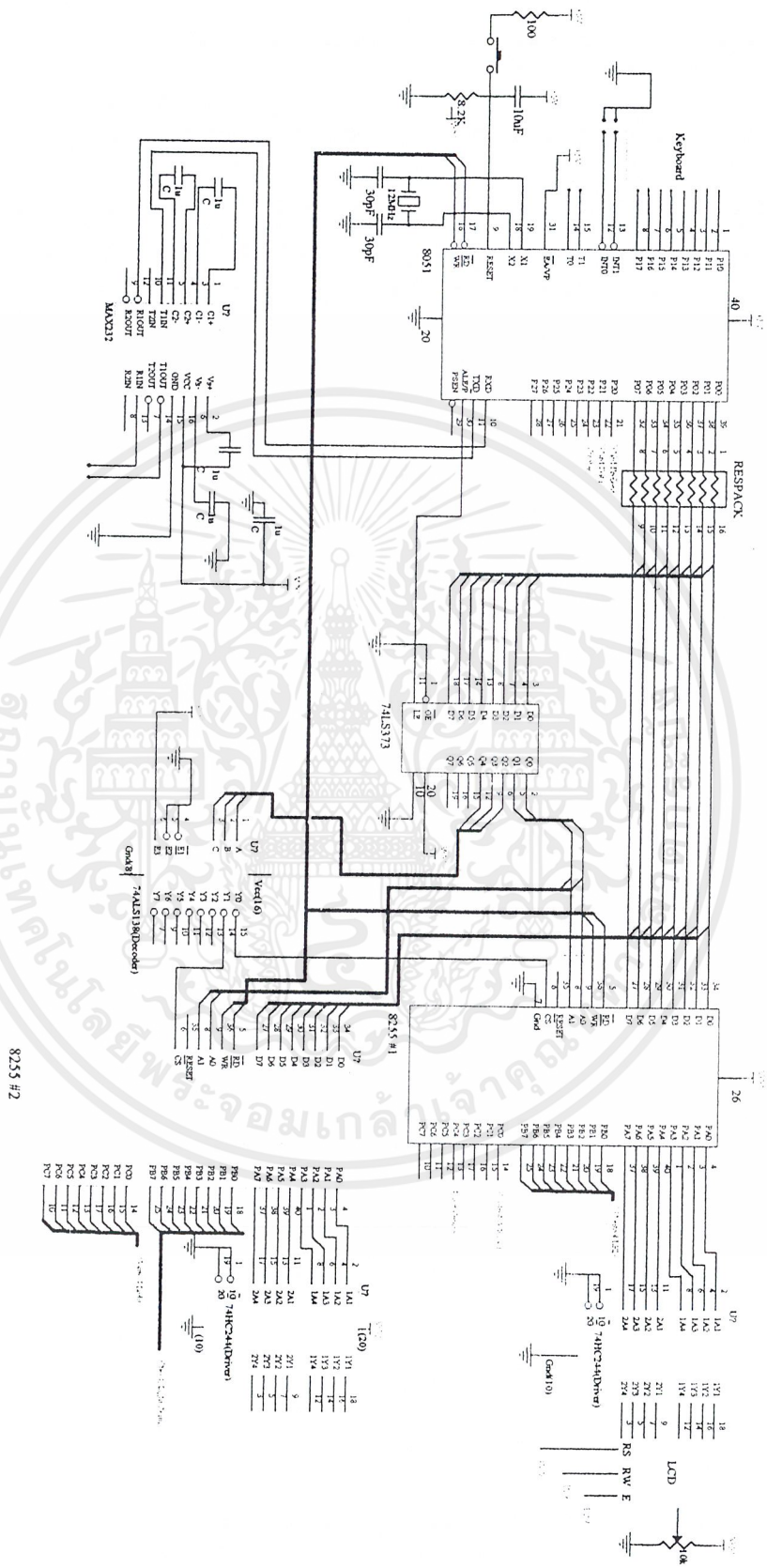
3) ฟอร์ม 7 เมื่อโหลดฟอร์มนี้ขึ้นมาจะมีการส่งโค้ด"m2" ออกไปเพื่อให้ไมโครคอนโทรลเลอร์กระโดดมาทำโปรแกรมส่วนนี้ คือจะส่งข้อมูลที่เป็นค่าพอร์ทของไมโครคอนโทรลเลอร์ที่ต่ออยู่กับตัวเช็คสถานะ ซึ่งได้แปลงจากเลขฐาน2 เป็นรหัสแอสกีเรียบร้อยแล้ว ส่งมาเป็นเลข 0,1 ทั้งหมด8ตัวอักษร มาพร้อมโค้ด"op" ซึ่งในฟอร์มนี้จะต้องปรับคุณสมบัติRthreshold ให้รับข้อมูลทั้งหมด10ตัวคือข้อมูล8ตัวกับโค้ดอีก2ตัว เมื่อรับมาก็จะเลือกเฉพาะข้อมูลสถานะไฟ8ตัวอักษรมาทำการอ่านที่ละเลขเรียงลำดับไป โดยแต่ละดวง(เราจำลองเป็นรูปภาพไฟให้กด)จะมีตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แปรเก็บสถานะon กับoff อยู่ ซึ่งในการนำค่าพอร์ทตัวเช็คสถานะมาใส่ที่ละตัวแล้วก็ทำการเปลี่ยนแลงค่าตัวแปรประจำหลอดให้มีค่าตรงกับค่าจริงซึ่งหลังจากมีการเปลี่ยนแปลงค่าตัวแปรทุกครั้งรูปไอคอนรูปหลอดไปสว่างและมืด จะต้องปรากฏให้ตรงกับค่าตัวแปรของหลอดนั้นด้วย

และเมื่อมีการคลิ้กรูปหลอดไฟใกดสถานะของหลอดตัวแปรนั้นก็จะกลับกันจากเปิดเป็นปิด และจาก ปิด เป็นเปิด รวมถึงรูปภาพไอคอนที่ปรากฏจะต้องสัมพันธ์กับค่าตัวแปรด้วย พร้อมทั้งจะส่งโค้ด"Z..." เป็นZแล้วตามด้วยหมายเลขหลอดไฟที่จะควบคุม ส่งไปให้ไมโครคอนโทรลเลอร์ ซึ่งจะรู้สถานะที่สั่งพร้อมกับตรวจสภาวะเอง แล้วก็จะปรับสถานะไฟปัจจุบันให้เป็นตรงกันข้ามซึ่งจะทำให้สภาวะการติด-ดับของหลอดไฟจริง สอดคล้องกับรูปที่ปรากฏบนหน้าจอโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.23 แสดงรูปวงจรไมโครคอนโทรลเลอร์

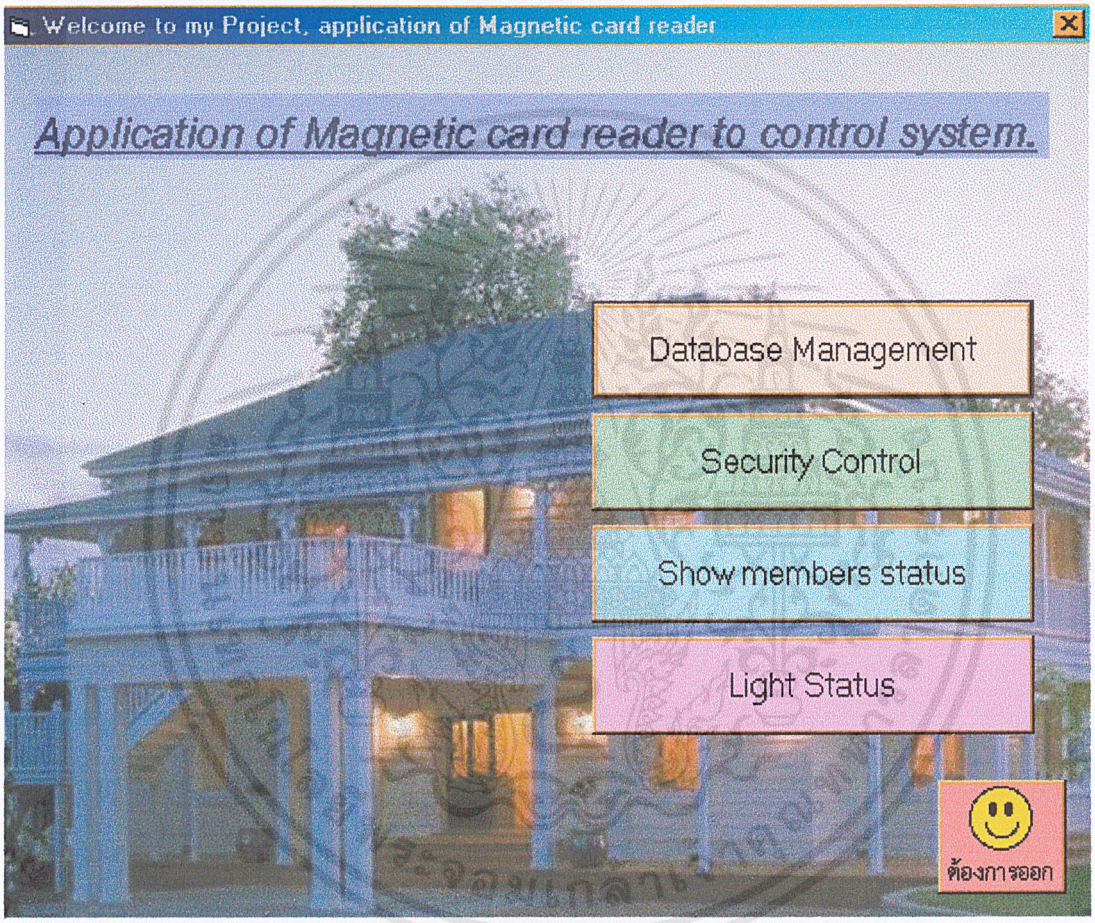
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

ผลการทดลอง

4.1 ส่วนโปรแกรมที่ติดต่อกับผู้ใช้

เมื่อทำการทดลองรันโปรแกรม จะปรากฏหน้าต่างส่วนที่เป็นโปรแกรมหลัก ดังรูป



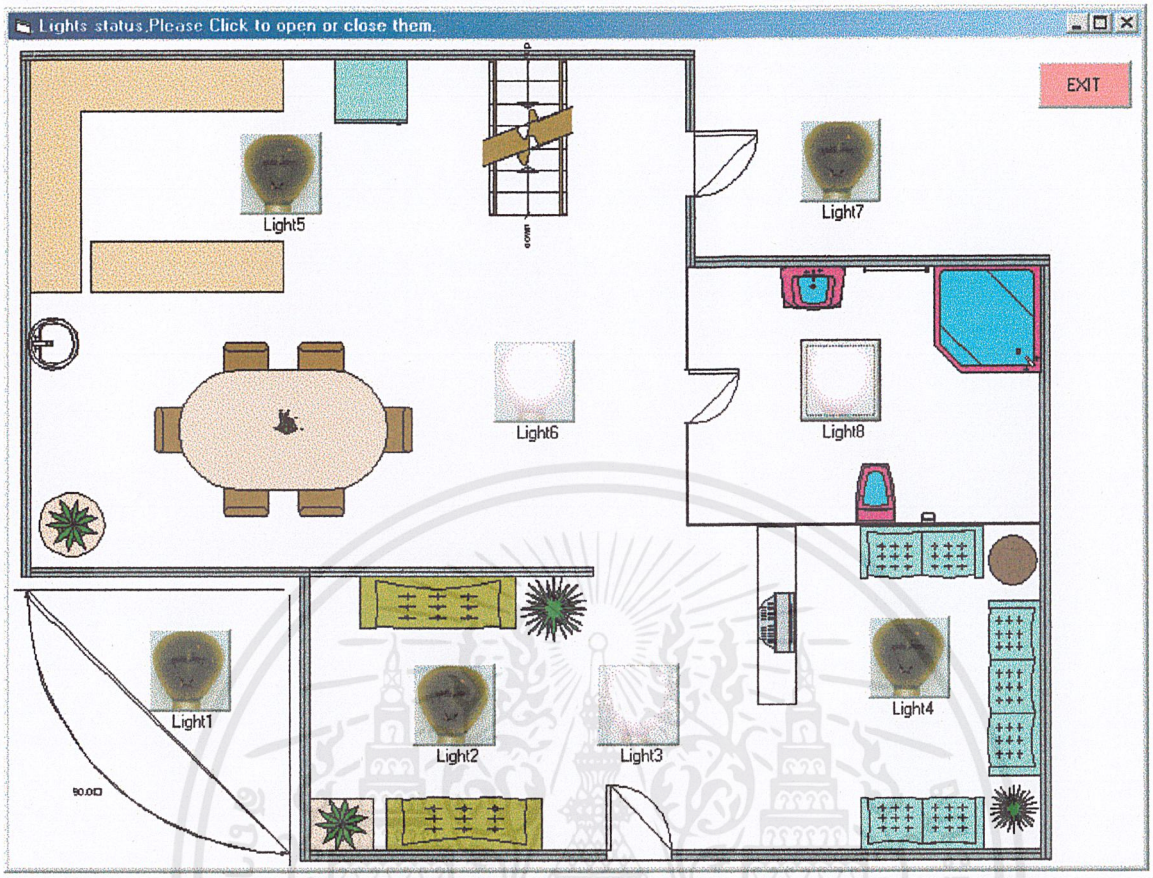
รูปที่ 4.1 แสดงส่วนที่เป็นโปรแกรมหลัก

จากหน้าต่างนี้ เราสามารถเลือกที่จะทำงานได้ 4 รูปแบบ ดังนี้

4.1.1 เลือกทำงานในโหมดควบคุมอุปกรณ์ไฟฟ้า

จากหน้าต่างที่เป็นโปรแกรมหลัก ถ้าเราเลือกที่ปุ่ม Light Status (แสดงในรูปที่ 4.1) จะปรากฏหน้าต่างที่เป็นแผนที่ภายในอาคาร และอุปกรณ์ต่างๆที่สามารถควบคุมได้ผ่านทางคอมพิวเตอร์ (ในที่นี้คือหลอดไฟ) ดังรูปที่ 4.2 เราสามารถเลือกคลิกที่รูปเครื่องใช้ไฟฟ้าต่างๆเพื่อเปิดหรือปิดอุปกรณ์เหล่านั้นผ่านทางหน้าจอคอมพิวเตอร์ได้ เมื่อหลอดไฟกำลังถูกเปิดอยู่ในขณะนั้น ภาพของหลอดไฟบนจอคอมพิวเตอร์จะเปลี่ยนเป็นสีขาวเพื่อแสดงให้เราทราบ ดังรูปที่ 4.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

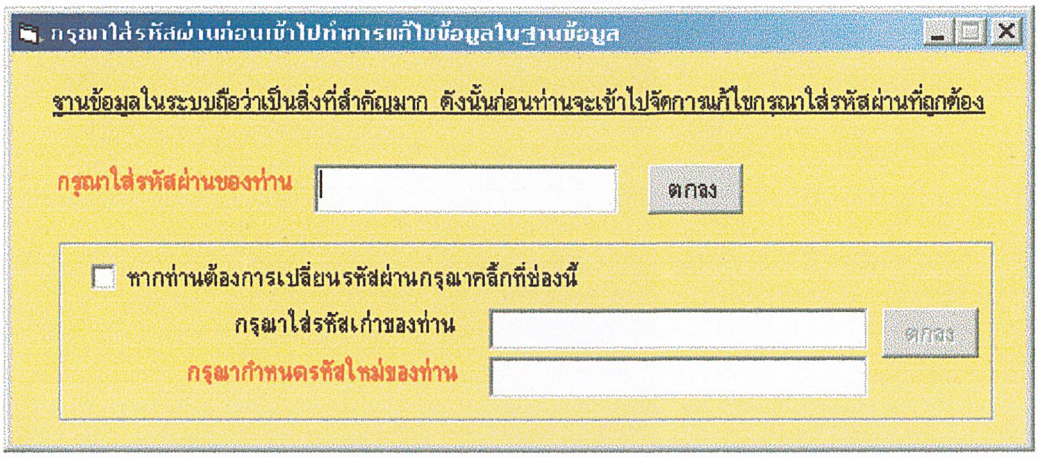


รูปที่ 4.2 แสดงหน้าต่างโปรแกรมส่วนควบคุมอุปกรณ์ไฟฟ้า และแสดงสถานะหลอดไฟที่ถูกเปิดอยู่

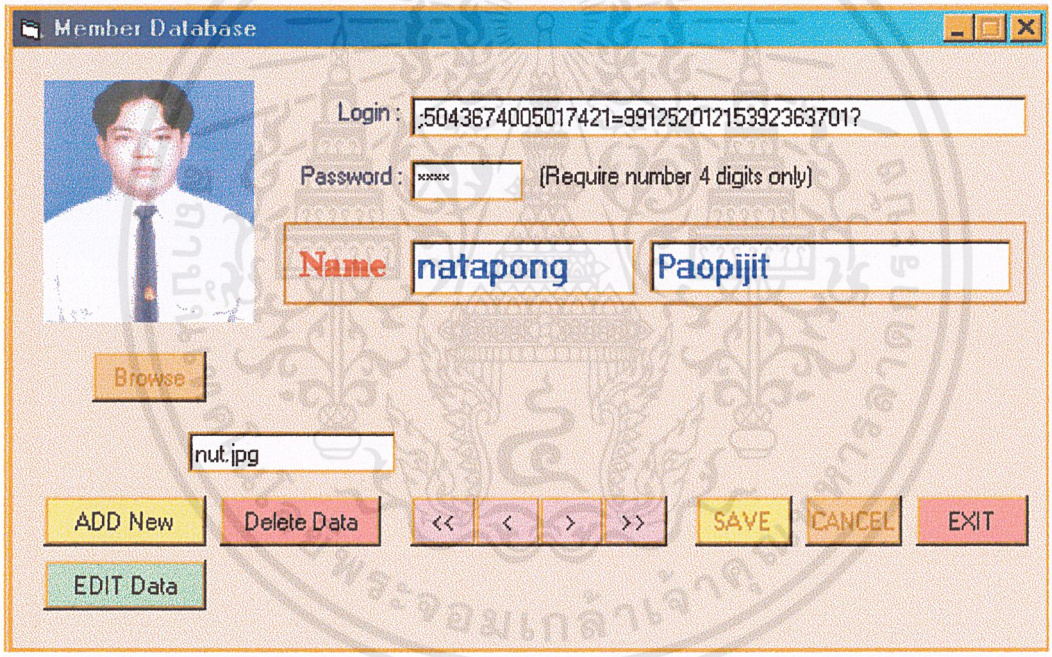
4.1.2 <u>เลือกทำงานในโหมดจัดการกับฐานข้อมูล</u>

มื่อมีบุคคลที่ต้องการจะเข้ามาภายในอาคารนอกเหนือจากที่ได้รับอนุญาตไว้ในตอนต้น บุคคลผู้นั้นจะต้องได้รับอนุญาตก่อนโดยทำการเพิ่มข้อมูลของบัตรใหม่เข้าไปในฐานข้อมูล เจ้าของระบบต้องเลือกทำงานในโหมดจัดการกับฐานข้อมูลจากหน้าโปรแกรมหลัก โดยเมื่อทำการเลือกคลิกที่ปุ่ม จัดการกับฐานข้อมูล จะปรากฏหน้าต่างให้ใส่รหัสผ่าน (เพื่อรักษาความปลอดภัย) ดังแสดงในรูป 4.3 เมื่อใส่รหัสผ่านถูกต้องแล้วก็จะเข้าสู่หน้าจอที่แสดงข้อมูลในบัตร , ชื่อ , รูปภาพ และรหัสที่ใช้ในการผ่านเข้าประตู ของบุคคลนั้นๆ ดังแสดงในรูปที่ 4.4 ขณะที่ผู้ใช้เข้าสู่การทำงานในโหมดนี้หน้าจอ LCD จะปรากฏคำว่า " Swipe Card -> " และเมื่อทำการรูดบัตรใหม่ที่เครื่องอ่าน จะมีหน้าต่างถามว่าจะทำการเพิ่มข้อมูลนี้ในฐานข้อมูลหรือไม่ หรือหากเรากดที่ปุ่ม add new ในรูป 4.4 แล้วทำการรูดบัตรใหม่ที่เครื่องอ่าน ข้อมูลนั้นก็จะถูกเพิ่มเข้าไปในฐานข้อมูลโดยอัตโนมัติ (เมื่อเรากด save) นอกจากนี้เราสามารถแก้ไขข้อมูลเก่าที่มีอยู่แล้วหรือทำการลบข้อมูลได้ผ่านทางหน้าจอส่วนนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 แสดงหน้าต่างถามรหัสผ่านก่อนเข้าสู่โหมดจัดการกับฐานข้อมูล



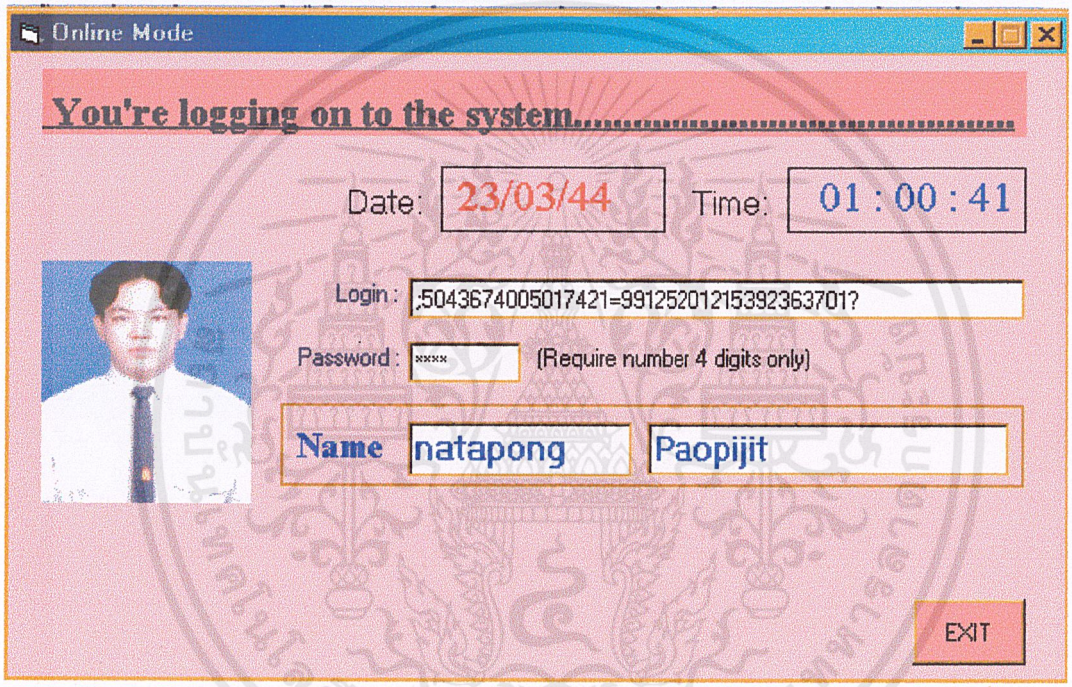
รูปที่ 4.4 แสดงหน้าต่างของโปรแกรมในโหมดจัดการกับฐานข้อมูล

4.1.3 <u>เลือกทำงานในโหมดรักษาความปลอดภัย</u> (security control)

เมื่อผู้ใช้เลือกที่ปุ่ม security control บนโปรแกรมหลัก จะเข้าสู่โหมดรักษาความปลอดภัย เมื่อมีผู้ทำการรูดบัตรเพื่อเข้าหรือออกจากอาคารข้อมูลนั้นจะปรากฏบนหน้าจอนี้ทันที หน้าจอนี้จะควบคุมการเข้าออกจากอาคารทั้งหมด ผู้ใช้ต้องทำการรันโปรแกรมในโหมดนี้ก่อนจึงจะสามารถออกจากอาคารได้ โดยเมื่อเข้าสู่การทำงานในโหมดนี้แล้ว ที่หน้าจอ LCD จะขึ้นอักษรคำว่า “online” หากผู้ใช้ต้องการออกจากอาคาร ก็จะต้องทำการกดที่ปุ่ม 2nd บนคีย์บอร์ด หลังจากนั้น LCD จะขึ้นข้อความ “Swipe Card” เพื่อให้ทำการรูดบัตร ข้อมูลจะถูกส่งและแสดงบนหน้าจอคอมพิวเตอร์ดังรูปที่ 4.5 หลัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้น LCD จะแสดงคำว่า “System Lock” เพื่อเป็นการรับทราบ และเมื่อผู้ใช้กลับเข้ามาในอาคารอีกครั้ง ก็จะต้องทำการกดที่ปุ่ม enter บนคีย์บอร์ด จอ LCD จะแสดงคำว่า “Swipe Card” อีกครั้ง เมื่อรูดบัตรแล้วข้อมูลจะแสดงบนคอมพิวเตอร์อีกครั้งหนึ่ง และ LCD จะปรากฏข้อความให้ใส่รหัสผ่าน เมื่อใส่รหัสผ่านถูกต้องแล้ว ประตูก็จะเปิดออก พร้อมกับแสดงคำว่า “ System Unlock” บนหน้าจอ LCD เมื่อทดลองทำการรูดบัตรที่ไม่มีอยู่ในฐานข้อมูล LCD จะแสดงคำว่า “Card invalid !” เพื่อเป็นการแจ้งให้ทราบ หากเราทำการใส่รหัสผ่านผิดครบ 3 ครั้ง ระบบก็จะกลับเข้าสู่สภาวะ “System Lock” เช่นเดิม ต้องทำการรูดบัตรใหม่อีกครั้ง

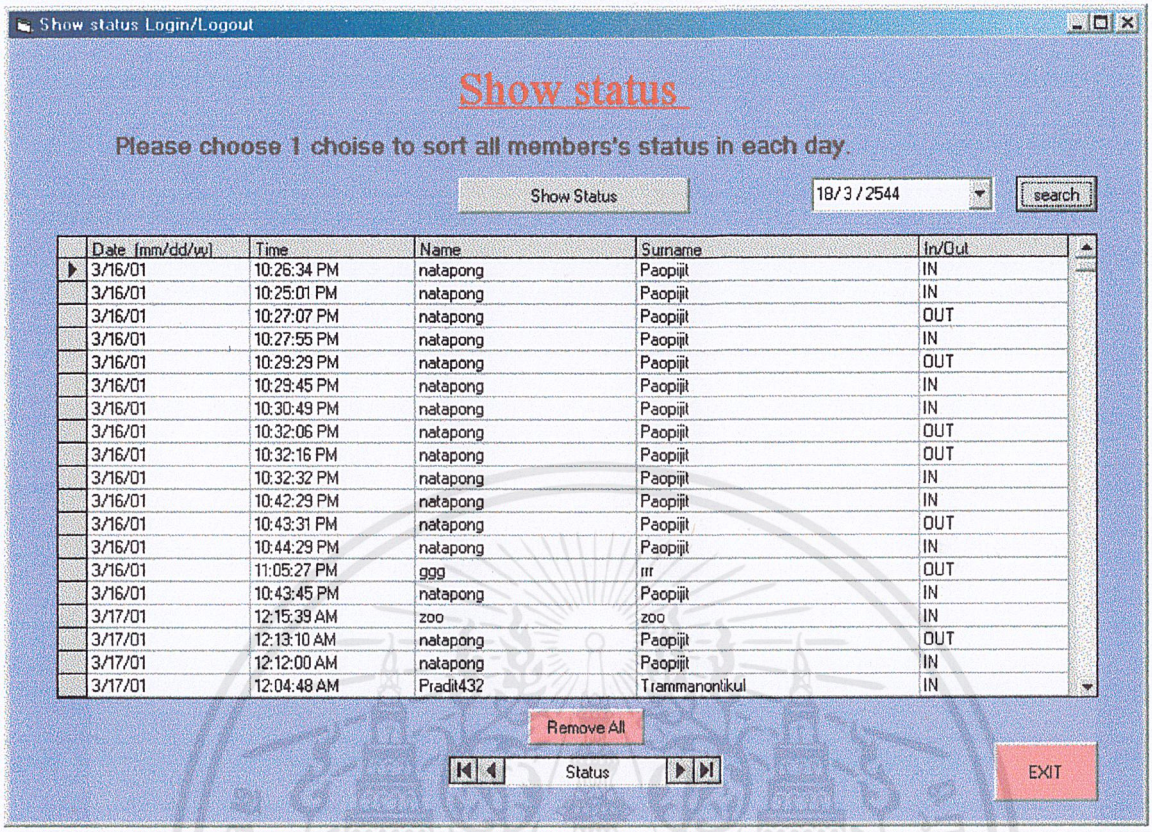


รูปที่ 4.5 แสดงหน้าต่างโปรแกรมในโหมดรักษาความปลอดภัย

4.1.4 <u>การทำงานในโหมดแสดงสถานะการผ่านเข้าออก</u>

ในโหมดนี้จะเป็นการแสดงรายชื่อบุคคลที่ทำการผ่านเข้าออกอาคารพร้อมทั้งวันและเวลาที่บุคคลนั้นเข้าหรือออกจากอาคาร เราสามารถเลือกตรวจสอบรายชื่อบุคคลภายในวันนั้นๆ หรือจะกำหนดให้แสดงรายชื่อบุคคลที่ผ่านเข้าออกทั้งหมดที่มีในฐานข้อมูลเลยก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Date (mm/dd/yy)	Time	Name	Surname	In/Out
3/16/01	10:26:34 PM	natapong	Paopijit	IN
3/16/01	10:25:01 PM	natapong	Paopijit	IN
3/16/01	10:27:07 PM	natapong	Paopijit	OUT
3/16/01	10:27:55 PM	natapong	Paopijit	IN
3/16/01	10:29:29 PM	natapong	Paopijit	OUT
3/16/01	10:29:45 PM	natapong	Paopijit	IN
3/16/01	10:30:49 PM	natapong	Paopijit	IN
3/16/01	10:32:06 PM	natapong	Paopijit	OUT
3/16/01	10:32:16 PM	natapong	Paopijit	OUT
3/16/01	10:32:32 PM	natapong	Paopijit	IN
3/16/01	10:42:29 PM	natapong	Paopijit	IN
3/16/01	10:43:31 PM	natapong	Paopijit	OUT
3/16/01	10:44:29 PM	natapong	Paopijit	IN
3/16/01	11:05:27 PM	ggg	rrr	OUT
3/16/01	10:43:45 PM	natapong	Paopijit	IN
3/17/01	12:15:39 AM	zoo	zoo	IN
3/17/01	12:13:10 AM	natapong	Paopijit	OUT
3/17/01	12:12:00 AM	natapong	Paopijit	IN
3/17/01	12:04:48 AM	Pradit432	Trammanontikul	IN

รูปที่ 4.6 แสดงหน้าต่างโปรแกรมในโหมดตรวจสอบสถานะการผ่านเข้าออก

4.2 การทดลองอ่านข้อมูลของบัตรต่างๆ

;	5043674005017421	=	99125201215392363701	?

ลักษณะข้อมูลของบัตรธนาคารกรุงไทย จะเริ่มต้นด้วย " ; " ซึ่งเป็นตัวอักขระเริ่มต้นของบัตรแม่เหล็กตามมาตรฐาน ISO และจะตามด้วยข้อมูลที่เป็นรหัสของบัตรเอทีเอ็ม ซึ่งในที่นี้คือ " 5043674005017421 " โดยจะมีเครื่องหมายเท่ากับเป็นตัวแบ่งข้อมูลระหว่าง รหัสของบัตร กับ Additional data และ Discretionary data (เป็น FS หรือ Field Separator) ข้อมูล Additional data อาจจะเป็นวันหมดอายุของบัตร หรือ รหัสบริการ (service code) ด้วยก็ได้ หลังจากข้อมูลที่เป็น Additional data และ Discretionary data แล้วก็จะเป็นตัวอักขระบอกจุดจบของข้อมูล (End Sentinel) และจะมี LRC ซึ่งใช้ตรวจสอบความผิดพลาดในการอ่านข้อมูลอยู่ในส่วนท้ายสุด(ในที่นี้ไม่ได้แสดงไว้)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

;	00000	21711754349717	=	12990004584	?
;	00000	21681178215317	=	12990005840	?

ลักษณะข้อมูลของบัตรธนาคารกรุงเทพ จะมีลักษณะต่างจากบัตรทั่วๆไปคือ จะมีเลขศูนย์นำหน้า และส่วนที่อยู่หน้า ตัวแบ่งข้อมูลจะไม่ใช่รหัสของบัตรเอทีเอ็ม หรือรหัสบัญชีแต่อย่างใด ซึ่งเราไม่สามารถทราบรายละเอียดของความหมายในส่วนนี้ได้เนื่องจากเป็นลักษณะเฉพาะของธนาคารนั้นๆ

;	3100902202239	033306841600071060617774	?
;	3101202667787	101230464501003106042073	?
;	3100901503995	103330627000003606010202	?

ในส่วนของบัตรประชาชน ข้อมูลส่วนแรกจะเป็นเลขประจำตัวประชาชน และต่อด้วยข้อมูลอื่นๆตามหลัง แต่จะไม่มีตัวแบ่งแยกข้อมูล (Field Separator) เหมือนดังเช่นในบัตรธนาคารต่างๆ

4.3 <u>การทดลองส่วนวงจรควบคุม</u>

H11L2 จะมีกระแสไม่เกิน10มิลลิแอมป์ จาก datasheet จึงได้มีการทดลองเปลี่ยนค่า R4 ดังตาราง

R4 (KΩ)	Vdc (v)	V3k (Vp-p)	I_L (mA)
1//2	1.8	6.4	1.2
1	2	6.1	1
2	2.7	8	0.85
∝	7.4	17	0

ซึ่งจะเห็นได้ว่า หากมีการเปลี่ยนค่า R_4 เป็นผลให้ค่าต่างๆเปลี่ยนไปหมดเพราะว่าตรงที่ต่อคร่อม R_2 เป็นการต่อแบบขนานทำให้มีการแบ่งกระแส และ แรงดันที่ R_2 จะเปลี่ยนไปตามค่าที่เปลี่ยนในตาราง เป็นการยากในการเปลี่ยนค่าตามที่ต้องการ วงจรนี้ไม่เหมาะที่จะใช้ควบคุมอุปกรณ์ที่มีจำนวนมากๆ ควรใช้ PLC ในการควบคุมแทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปและวิจารณ์

สำหรับโครงงานเครื่องบันทึก-อ่านบัตรแม่เหล็กนี้ ในส่วนของครึ่งเทอมแรกได้ทำเพียงส่วนของการบันทึกข้อมูลลงในบัตรแม่แหล็ก ซึ่งผลปรากฏพบว่าค่อนข้างยากและมีสัญญาณรบกวนที่มากเนื่องจากสัญญาณข้อมูลที่ใช้ต่ำมากๆ และต้องขยายด้วยอัตราการขยายที่สูงเป็น1000เท่า รวมกับความจำเป็นที่ต้องใช้อุปกรณ์ตรวจวัดสัญญาณที่ค่อนข้างดี และละเอียดมากๆ ทำให้ไม่สามารถทำการทดลองได้สะดวกนักได้ เนื่องจากไม่มีเครื่องมือที่ดี ดังที่ได้กล่าวมาแล้ว จึงไม่ประสบความสำเร็จในการทดลองเท่าไรนัก ดังนั้นในส่วนของเทอมที่ 2 จึงได้เปลี่ยนมาใช้เครื่องอ่านบัตรแม่เหล็ก (จากการปรึกษาอาจารย์ที่ปรึกษา) ซึ่งมีการกำหนดมาตรฐานที่ตายตัว มีการพัฒนามานานหลายรุ่นด้วยอุปกรณ์การสร้างที่ดีมากๆ ดังนั้นในเทอมนี้จึงสามารถเขียนโปรแกรมสำหรับอ่านข้อมูลจากบัตรบัตรได้อย่างไม่ยากลำบากเท่าไรนัก และค่าที่อ่านได้ในแต่ละครั้งก็ไม่ค่อยจะเกิดข้อผิดพลาดเท่าไร รวมกับมีรูปแบบการเขียนโปรแกรมเพื่อตรวจสอบความผิดพลาดของการรับข้อมูลในบัตรถึง 3 แบบ ทำให้เมื่อบางครั้งการรูดบัตรไม่เรียบสนิทก็ทำให้รับข้อมูลผิดก็จะทราบว่าเกิดERROR ขึ้น ซึ่งสามารถที่จะรูดใหม่จนกว่าจะถูกได้

ในส่วนของเทอมที่ 2 สามารถที่จะทำการควบคุมการทำงานของระบบด้วยไมโครคอนโทรเลอร์ซึ่งจะควบคุมตั้งแต่การรับข้อมูลจากเครื่องอ่านบัตรแม่เหล็ก การแสดงผลของLCD การสั่งปิดเปิดไฟที่บอร์ดควบคุมอุปกรณ์ไฟฟ้า การควบคุมการเปิด-ปิดประตู การติดต่อสื่อสารกับคอมพิวเตอร์ทางพอร์ทอนุกรม การส่งข้อมูลติดต่อกันไปมาไม่มีปัญหาอะไร เนื่องจากใช้อัตราบอดในการส่งที่ค่อนข้างต่ำ ซึ่งโอกาสที่จะส่งข้อมูลทางพอร์ทอนุกรมผิดพลาดนั้นมีน้อยมาก รวมถึงการเขียนโปรแกรมเกี่ยวกับการจัดเก็บข้อมูลลงในฐานข้อมูล ซึ่งโดยรวมแล้วขั้นตอนทั้งหมดนี้ก็สามารถทำงานได้โดยไม่มีปัญหาแต่อย่างใด สามารถนำไปใช้เป็นระบบอำนวยความสะดวกและระบบควบคุมคนเข้า-ออกได้ดังวัตถุประสงค์ที่ตั้งใจไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

<u>ปัญหาที่พบในการทดลอง</u>

1) เนื่องจากอุปกรณ์ดีๆที่ใช้ในการทดลองหาไม่ค่อยง่ายนักเนื่องจากมีจำนวนจำกัดเช่น ออสซิโลสโคปที่สามารถวัดสัญญาณได้ละเอียด หรือมัลติมิเตอร์ รวมถึงแหล่งจ่ายไฟกระสตรง ทั้งแบบ 5 โวลท์และ12โวลท์ ซึ่งมีความจำเป็นต่อการทดลองในโครงงานนี้ ทำให้บางครั้งขาดแคลนอุปกรณ์ตรวจวัดเหล่านี้ ทำให้งานต้องหยุดชะงัก ล่าช้า และไม่สามารถวัดบางจุดที่สงสัยได้

2) โครงงานชุดนี้ประกอบด้วยส่วนที่เป็นฮาร์ดแวร์และซอฟแวร์ ซึ่งนอกจากซอฟแวร์ที่ใช้ควบคุมไมโครคอนโทรเลอร์แล้ว ยังต้องมีการเขียนโปรแกรมคอมพิวเตอร์อีกด้วยเช่นในโครงงานนี้ใช้ภาษาวิชวลเบสิกเวอร์ชัน6.0 เขียนในการติดต่อฐานข้อมูลและส่งข้อมูลติดต่อกับไมโครคอนโทรลเลอร์ แม้ว่าหนังสือในท้องตลาดจะมีขายอยู่มากเกี่ยวกับโปรแกรมนี้แต่ทว่าส่วนที่เกี่ยวกับการส่งข้อมูลแบบอนุกรมโดยใช้MSCommและมาตรฐานการส่งข้อมูลแบบRS-232 นั้นแทบจะไม่มีเลย ทำให้ต้องทำการทดลองหาข้อสรุปด้วยตนเอง จึงเสียเวลาในการค้นคว้าหาข้อมูลอยู่นานกว่าจะได้เริ่มทำการทดลองในขั้นตอนต่อไป

3) เนื่องจากเป็นโครงงานแรกที่ได้มีการเขียนโปรแกรมคอมพิวเตอร์ซึ่ง โดยส่วนใหญ่มักจะเป็นการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์เป็นส่วนใหญ่ ดังนั้นจึงขาดความชำนาญในการเขียนโปรแกรมคอมพิวเตอร์ ที่เป็น Object Oriented ในระดับสูงๆ ทำให้สามารถเขียนโปรแกรมได้ในระดับธรรมดาเบื้องต้น ไม่สามารถเขียนเป็นโปรแกรมระดับมืออาชีพได้

<u>จุดบกพร่องที่ควรจะแก้ไขเพื่อเป็นแนวทางในการพัฒนา</u>

สำหรับโครงงานนี้พบว่า ไม่ค่อยเกิดปัญหาข้อผิดพลาดเกิดขึ้นเท่าไรนัก ดังนั้นจุดที่ควรแก้ไขก็ควรจะเป็นเทคนิคในการเขียนโปรแกรมที่น่าจะมีความกระชับขึ้น มีการแก้ข้อผิดพลาด(Bug)ทุกกรณี สามารถป้องกันกรณีที่เกิดข้อผิดพลาดได้หมด ไม่เกิดRun time Errorในภายหลัง สิ่งแรกที่ควรพิจารณาคือเมื่อเปิดฟอร์มที่มีการใช้MSComm เหมือนกันขึ้นมาทั้งคู่จะเกิดเหตุการณ์การเปิดพอร์ทชนกันเกิดErrorได้ ซึ่งควรที่จะมีการเช็คว่าขณะเปิดฟอร์มนี้ฟอร์มอื่นก็ควรปิดการใช้งานพอร์ทไปก่อน อีกทั้งในการส่งโค้ดบางครั้งระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์อาจเกิดข้อผิดพลาดได้เป็นบางกรณีเนื่องจากการเกิดการส่ง-รับที่ไม่ตรงกันซึ่งเกิดจากการไม่มีการส่งสัญญาณตอบรับกันก่อนที่ดีพอ แต่ในกรณีจะเกิดขึ้นน้อยมาก นอกจากนี้ยังควรเพิ่มระบบจ่ายไฟสำรองให้กับโครงงานนี้ด้วยไม่ว่าจะเป็นแบตเตอรี่แบ็คอัพ หรือUPS เนื่องจากหากไฟดับ ระบบจะไม่สามารถทำงานได้เลย ซึ่งคนภายในจะไม่สามารถออกจากบริเวณได้ และคนนอกบริเวณก็จะไม่สามารถเข้าได้ และควรเพิ่มส่วนที่เป็นระบบป้องกันภัยด้วย เช่นเมื่อมีผู้บุกรุก สัญญาณเตือนภัยควรที่จะดังขึ้นมา และถ้าจะให้ดียิ่งขึ้นระบบควรมีการโทรออกไปยังเบอร์โทรศัพท์ของบุคคลที่เกี่ยวข้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่สำคัญ นอกจากนี้ระบบ ควบคุมอุปกรณ์ไฟฟ้าควรที่จะเขียนโปรแกรมเพื่อให้สามารถตั้งเวลาเปิด-ปิดอุปกรณ์ไฟฟ้าได้เพิ่มขึ้นมา จะช่วยเพิ่มความสะดวกได้มากขึ้น ในส่วนของวงจรควบคุมหลอไฟนั้น จะมีข้อเสียเมื่อต้องทำการควบคุมหลอดไฟเป็นจำนวนมาก ดังนั้นหากเปลี่ยนเป็นการควบคุมโดยใช้อุปกรณ์จำพวก PLC จะสะดวกและมีประสิทธิภาพในการทำงานมากกว่าเดิม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาพผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
          org 0000h
          ljmp main
          org 0003h
;^^^^^^^^^^^^^^^^^^^^^^ Serial Port Interupt ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
          org 0023h
 interupt_rq:clr tr1
          clr es
          clr ea
          mov a,sbuf
          mov 6ah,a
          clr ri
     int_w:jnb ri,int_w
          mov a,sbuf
          mov 6bh,a
          clr ri
          mov sp,#09h
          mov 09h,#00h
          mov 08h,#50h
          clr err_status3
          reti
          org 0050h
          jmp mode_on_ck1

;^^^^^^^^^^^^^^^^^^^^^^^^ End Of Serial Port Interupt ^^^^^^^^^^^^^^^^^^^^^^^^

;#############################################################################

      main:card_present equ p2.0
          card_data    equ p2.1
          strobe       equ p2.2
          RS           equ p2.5
          RW           equ p2.6
          E            equ p2.7
          first_bit    equ 00h          ; verify for first bit-- use first bit
                                        ; of 20h
          card_err     equ 01h          ; verify for card error -- use second
                                        ; bit of 20h
          err_status   equ 02h          ; verify for account error
          err_status2  equ 03h
          err_status3  equ 04h
          port_a       equ 00h
          port_b       equ 01h
          port_c       equ 02h
          control_w    equ 03h
          port_a2      equ 04h
          port_b2      equ 05h
          port_c2      equ 06h
          control_w2   equ 07h
          mov 2ch,#00h
          mov dptr,#control_w
          mov a,#88h
          movx @dptr,a                  ; set control word for 8255
          mov a,#82h                    ; set port b1 as opto isolate
          mov dptr,#control_w2
          movx @dptr,a
          mov tmod,#21h                 ; T1 send 8bit auto reload,T0 send 16 bit
          mov th1,#-13
          mov tl1,#-13                  ; Setting baud Rate = 2400BPS
          mov pcon,#00h                 ; Smod = 0
          mov scon,#40h                 ; Mode1(10 bit data),TI=0,RI=0
                                        ; REN=0(not receive data)
          clr err_status3               ; use in database mode
          acall delay                   ; set LCD
          clr RS
          clr RW
          clr E
          lcall delay
          mov a,#00111100b
          acall lcd_set
          acall delay
          mov a,#00001100b
          acall lcd_set
          acall delay
          mov a,#01h
          acall lcd_set
          acall delay
          clr ri
          setb es                       ; enable interupt from serial port
          setb ea
          setb ren
          setb tr1
          jmp $                         ; wait to recieve mode from computer
;#############################################################################

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Check mode ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

;----- check character "M1" which mean we're going to execute in online mode ------
 mode_on_ck1:mov a,6ah
          cjne a,#01001101b,check_edit  ; check for character "M"
 mode_on_ck2:mov a,6bh
          cjne a,#10110001b,check_edit  ; check for character "1"
          mov a,#10h                    ; check mode pass -- Display orange LED
          mov dptr,#port_b
          movx @dptr,a
          mov a,#01h
          acall lcd_set
          acall delay
          mov a,#80h
          acall lcd_set
          acall delay
          mov a,#'O'                    ; show "Online" on LCD Module
          acall lcd_write
          acall delay
          mov a,#'n'
          acall lcd_write
          acall delay
          mov a,#'l'
          acall lcd_write
          acall delay
          mov a,#'i'
          acall lcd_write
          acall delay
          mov a,#'n'
          acall lcd_write
          acall delay
          mov a,#'e'
          acall lcd_write
          acall delay
          jmp first_scan                ; jump to scan key for
                                        ; button "encode" or "decode"
;-----------------------------------------------------------------------------
;------------------------ Check for add database -----------------------------------
   check_edit:mov a,6ah
          cjne a,#01001101b,check_modeli  ; check for character "M"
  check_edit2:mov a,6bh
```

```
            cjne a,#00110011b,check_modeli      ; check for character "3"
            ljmp database
;-------------------------------------------------------------------------------
;------------------------ Check for mode 2 -------------------------------------
check_modeli:mov a,6ah
            cjne a,#01001101b,check_light       ; check for character "M"
check_modeli2:mov a,6bh
            cjne a,#10110010b,check_light       ; check for character "2"
            ljmp Light_display
;-------------------------------------------------------------------------------
;---------------------- Check for control light --------------------------------
check_light:mov a,6ah                           ; check for character "
            cjne a,#5ah,jp_recv                 ; check for "z"
            mov r7,6bh
    light1:cjne r7,#0b1h,light2
            ljmp display1
    light2:cjne r7,#0b2h,light3
            ljmp display2
    light3:cjne r7,#33h,light4
            ljmp display3
    light4:cjne r7,#0b4h,light5
            ljmp display4
    light5:cjne r7,#35h,light6
            ljmp display5
    light6:cjne r7,#36h,light7
            ljmp display6
    light7:cjne r7,#0b7h,light8
            ljmp display7
    light8:cjne r7,#0b8h,jp_recv
            ljmp display8
     jp_recv:ajmp recieve_err
;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
;------------------------ display light 1 --------------------------------------
    display1:jb 60h,out_01                  ; opto = 0
      out_11:mov a,2ch
            orl a,#00000001b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_01:mov a,2ch
            anl a,#11111110b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 2 --------------------------------------
    display2:jb 61h,out_02                  ; opto = 0
      out_12:mov a,2ch
            orl a,#00000010b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_02:mov a,2ch
            anl a,#11111101b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 3 --------------------------------------
```



```
    display3:jb 62h,out_03
      out_13:mov a,2ch
            orl a,#00000100b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_03:mov a,2ch
            anl a,#11111011b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 4 --------------------------------------
    display4:jb 63h,out_04                  ; opto = 0
      out_14:mov a,2ch
            orl a,#00001000b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_04:mov a,2ch
            anl a,#11110111b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 5 --------------------------------------
    display5:jb 64h,out_05                  ; opto = 0
      out_15:mov a,2ch
            orl a,#00010000b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_05:mov a,2ch
            anl a,#11101111b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 6 --------------------------------------
    display6:jb 65h,out_06                  ; opto = 0
      out_16:mov a,2ch
            orl a,#00100000b                ; force 1
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
      out_06:mov a,2ch
            anl a,#11011111b                ; force 0
            mov dptr,#port_c2
            movx @dptr,a
            mov 2ch,a
            ljmp light_display
;-------------------------------------------------------------------------------
;------------------------ display light 7 --------------------------------------
    display7:jb 66h,out_07                  ; opto = 0
      out_17:mov a,2ch
            orl a,#01000000b                ; force 1
```

```
          mov dptr,#port_c2
          movx @dptr,a
          mov 2ch,a
          ljmp light_display
   out_07:mov a,2ch
          anl a,#10111111b              ; force 0
          mov dptr,#port_c2
          movx @dptr,a
          mov 2ch,a
          ljmp light_display
;-------------------------------------------------------------------------------
;------------------------------display light 8----------------------------------
  display8:jb 67h,out_08                ; opto = 0
   out_18:mov a,2ch
          orl a,#10000000b              ; force 1
          mov dptr,#port_c2
          movx @dptr,a
          mov 2ch,a
          ljmp light_display
   out_08:mov a,2ch
          anl a,#01111111b              ; force 0
          mov dptr,#port_c2
          movx @dptr,a
          mov 2ch,a
          ljmp light_display
;-------------------------------------------------------------------------------



;------------------ if recieved character is not match our definition ----------
 recieve_err:mov a,#0ffh
          mov dptr,#port_b
          movx @dptr,a
          mov 6ah,#00h
          mov 6bh,#00h
          setb ea
          setb es
          setb ren
          setb tr1
          mov a,#0ffh
          mov dptr,#port_b
          movx @dptr,a
          jmp $
;------------------ end of display error recieving mode character ------------------

;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ end of check mode ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>   DATABASE MODE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    database:clr ri
          clr ti
          setb ea
          setb es
          lcall delayx
          acall LCD_card
          mov 21h,#0e4h                 ;'d'   check for card decoding
          mov 22h,#65h                  ;'e'
recieve_data2:mov r0,#30h
          mov r1,#40
     clear2:mov @r0,#00h
          inc r0
          djnz r1,clear
   check_cp2:jb card_present,check_cp2
          clr ea
          clr es
```



```
          call card_read
          call check_data
          jb card_err,recieve_data2
          setb err_status3
          call convert
          jmp database
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>ONLINEMODE>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

  first_scan:mov 90h,#0ffh
          clr p1.3
          jb p1.7,second_scan
          jmp decode_card
second_scan:mov 90h,#0ffh
          clr p1.2
          jb p1.7,first_scan
encode_card:clr ea
          clr es
          mov a,#10h
          mov dptr,#port_b
          movx @dptr,a
          lcall delayx
          acall lcd_card
          mov 21h,#65h                  ; code 'e' ,add as a prefix of card
                                        ; data and send to computer
          mov 22h,#0eeh                 ; code 'n'
          acall recieve_data
          acall invalid_ck
          ljmp first_scan               ; jump to wait for swipe card again
                                        ; ( decode or encode )

decode_card:clr ea
          clr es
          mov a,#10h
          mov dptr,#port_b
          movx @dptr,a
          lcall delayx
          acall LCD_card
          mov 21h,#0e4h                 ;'d'   check for card decoding
          mov 22h,#65h                  ;'e'
          acall recieve_data
          acall invalid_ck
          jb err_status,first_scan
          clr ea
          clr es
   continue:acall hello
          lcall scan_pass
          jb err_status,sys_lock
          mov a,#20h
          mov dptr,#port_b
          movx @dptr,a
          clr ri
          clr ti
          setb ea
          setb es
          jmp first_scan

  sys_lock:lcall lcd_lock
          lcall delayx
          clr ri
          clr ti
          setb ea
          setb es
          ljmp first_scan
```

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> end of ONLINE MODE >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

;------------------- subroutine -- check for invalid card data -------------------
   invalid_ck:setberr_status
            mov a,6ah
            cjne a,#63h,valid_ck          ; check " ci " which comes from card
invalid
            mov a,6bh
            cjne a,#69h,valid_ck          ; if it's not " ci ",go to check whether
                                          ; it's " cv " or not
            jmp not_found                 ; it's " ci " -- card invalid -- database
                                          ; do not contain these account
    valid_ck:mov a,6ah
            cjne a,#63h,tempp             ; check " cv " which comes from card valid
            mov a,6bh
            cjne a,#0f6h,tempp            ; it's neither "ci" nor "cv" ,
                                          ; data sending error
            acall lcd_lock
            clr ti
            clr ri
            setb es                       ; enable interupt from serial port
                                          ; in case of form reopening
            setb ea
            clr err_status
            ret
      tempp:jmp status_err

   not_found:mov a,#01h                   ; card data not found in database --
                                          ; display " card invalid " in LCD
            acall lcd_set
            acall delay
            mov a,#80h
            acall lcd_set
            acall delay
            mov a,#'C'
            acall lcd_write
            acall delay
            mov a,#'a'
            acall lcd_write
            acall delay
            mov a,#'r'
            acall lcd_write
            acall delay
            mov a,#'d'
            acall lcd_write
            acall delay
            mov a,#14h
            acall lcd_set
            acall delay
            mov a,#'I'
            acall lcd_write
            acall delay
            mov a,#'n'
            acall lcd_write
            acall delay
            mov a,#'v'
            acall lcd_write
            acall delay
            mov a,#0c0h
            acall lcd_set
            acall delay
            mov a,#'a'
```



```
            acall lcd_write
            acall delay
            mov a,#'l'
            acall lcd_write
            acall delay
            mov a,#'i'
            acall lcd_write
            acall delay
            mov a,#'d'
            acall lcd_write
            acall delay
            mov a,#14h
            acall lcd_set
            acall delay
            mov a,#00100001b
            acall lcd_write
            acall delay
            mov a,#40h
            mov dptr,#port_b
            movx @dptr,a
            clr ti
            clr ri
            setb es                       ; enable interupt from serial port
                                          ; in case of form reopening
            setb ea
            ret
  status_err:mov a,#0ffh                  ; data recieving from computer is error
            mov dptr,#port_b              ; display orange,green and red LED
            movx @dptr,a
            clr ti
            clr ri
            setb es                       ; enable interupt from serial port
                                          ; in case of form reopening
            setb ea
            ret
;---------------------------- end of database checking ----------------------------

;--------------------- subroutine -- Display " System Lock " ----------------------
    LCD_lock:mov a,#01h
            lcall lcd_set
            lcall delay
            mov a,#80h
            acall lcd_set
            acall delay
            mov a,#'S'
            acall lcd_write
            acall delay
            mov a,#'y'
            acall lcd_write
            acall delay
            mov a,#'s'
            acall lcd_write
            acall delay
            mov a,#'t'
            acall lcd_write
            acall delay
            mov a,#'e'
            acall lcd_write
            acall delay
            mov a,#'m'
            acall lcd_write
            acall delay
            mov a,#14h
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            acall lcd_set
            acall delay
            mov a,#'L'
            acall lcd_write
            acall delay
            mov a,#0c0h
            acall lcd_set
            acall delay
            mov a,#'o'
            acall lcd_write
            acall delay
            mov a,#'c'
            acall lcd_write
            acall delay
            mov a,#'k'
            acall lcd_write
            acall delay
            mov a,#00100010b
            acall lcd_write
            acall delay
            mov a,#01011111b
            acall lcd_write
            acall delay
            mov a,#00100010b
            acall lcd_write
            acall delay
            mov a,#00100001b
            acall lcd_write
            acall delay
            ret
;------------------------ end of subroutine -- System Lock ------------------------

;---------------------- subroutine -- LCD Setting and writing ----------------------
   lcd_set:clr RS
            clr RW
            mov r1,#port_a
            movx @r1,a
            lcall enable
            lcall delay
            ;lcall delay
            ret
;.....
   lcd_write:setb RS
            clr RW
            mov r1,#port_a
            movx @r1,a
            lcall enable
            ;lcall delay
            lcall delay
            ret
;.....
    enable:setb E
            nop
            nop
            clr E
            ret
;.....
     delay:mov r1,#0ffh
      loop:nop
            nop
            djnz r1,loop
            ret
;------------------ End of subroutine -- LCD setting and writing -------------------
```



```
;------------------ Subroutine -- LCD display " Swipe card" ------------------------
  LCD_card:mov a,#01h
            lcall lcd_set
            lcall delay

            mov a,#80h
            acall lcd_set
            acall delay
            mov a,#'S'
            acall lcd_write
            acall delay
            mov a,#'w'
            acall lcd_write
            acall delay
            mov a,#'i'
            acall lcd_write
            acall delay
            mov a,#'p'
            acall lcd_write
            acall delay
            mov a,#'e'
            acall lcd_write
            acall delay
            mov a,#14h
            acall lcd_set
            acall delay
            mov a,#'C'
            acall lcd_write
            acall delay
            mov a,#'a'
            acall lcd_write
            acall delay

            mov a,#0c0h
            acall lcd_set
            acall delay
            mov a,#'r'
            acall lcd_write
            acall delay
            mov a,#'d'
            acall lcd_write
            acall delay
            mov a,#14h
            acall lcd_set
            acall delay
            mov a,#01111110b
            acall lcd_write
            acall delay
            ret
;------------------ End of subroutine -- LCD display " Swipe card " ---------------

;------------------------ Subroutine -- Show user's name --------------------------
    hello:mov a,#01h
            lcall lcd_set
            lcall delay

            mov a,#80h
            lcall lcd_set
            lcall delay
            mov a,#'H'
            lcall lcd_write
            lcall delay
```

```
            mov a,#'e'
            lcall lcd_write
            lcall delay
            mov a,#'l'
            lcall lcd_write
            lcall delay
            mov a,#'l'
            lcall lcd_write
            lcall delay
            mov a,#'o'
            lcall lcd_write
            lcall delay
;..................... recieve user's name from computer .....................
            mov r6,#10
            mov r1,#60h
            clr ri              ;don't want to recieve data
            setb tr1            ;start baudrate 2400 BPS
            setb ren
waitreceive:jnbri,waitreceive
            mov a,sbuf
            mov @r1,a
            inc r1
            clr ri
            djnz r6,waitreceive
;..................... Display user's name to LCD ............................
            mov a,#14h
            lcall lcd_set
            lcall delay
            mov a,60h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,61h
            anl a,#01111111b
            lcall lcd_write
            lcall delay

            mov a,#0c0h
            lcall lcd_set
            lcall delay
            mov a,62h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,63h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,64h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,65h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,66h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,67h
            anl a,#01111111b
            lcall lcd_write
```



```
            lcall delay
            mov a,68h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
            mov a,69h
            anl a,#01111111b
            lcall lcd_write
            lcall delay
;............................. recieve password from computer ..................
            mov r6,#4
            mov r1,#23h
            clr ri              ;don't want to recieve data
            setb tr1            ;start baudrate 2400 BPS
            setb ren
waitreceive_p:jnbri,waitreceive_p
            mov a,sbuf
            mov @r1,a
            inc r1
            clr ri
            djnz r6,waitreceive_p
            ;clr tr1
            mov a,#10h
            mov dptr,#port_b
            movx @dptr,a
;............................. show message " Password ? " .....................
      show_p:mov a,#01h
            lcall lcd_set
            lcall delay
            mov a,#'P'
            lcall lcd_write
            lcall delay
            mov a,#'a'
            lcall lcd_write
            lcall delay
            mov a,#'s'
            lcall lcd_write
            lcall delay
            mov a,#'s'
            lcall lcd_write
            lcall delay
            mov a,#'w'
            lcall lcd_write
            lcall delay
            mov a,#'o'
            lcall lcd_write
            lcall delay
            mov a,#'r'
            lcall lcd_write
            lcall delay
            mov a,#'d'
            lcall lcd_write
            lcall delay
            mov a,#0c0h
            lcall lcd_set
            lcall delay
            mov a,#00111111b            ; ? sign
            lcall lcd_write
            lcall delay
            ret
;-------------- End of subroutine -- recieve user's name from computer ------------

;------------- Subroutine -- recieve data from card and send to computer ----------
```

```
            ; r0 : point to first address
            ; r1 : count for 40 character
            ; r2 : check for 5 bit data
            ; r6 : use for delay 5 uS
recieve_data:mov r0,#30h
            mov r1,#40
      clear:mov @r0,#00h
            inc r0
            djnz r1,clear
   check_cp:jb card_present,check_cp
            call card_read
            call check_data
            jb card_err,recieve_data
            call convert
            ret

;..................... subroutine -- card_read ..................................
   card_read:mov r0,#30h                    ; point to start of ram
            setb first_bit
  card_read2:mov r2,#5                      ; check for 5 bit data
            clr a
  card_read3:jb card_present,return         ; wait for end of card
            jb strobe,card_read3            ; wait for clock = 0
            mov r6,#2                       ; delay
            djnz r6,$                       ; delay
            mov c,card_data                 ; read data
            jnb first_bit,card_read4
            jc card_read3                   ; if c = 1, it mean data = 0
                                            ; and that is clocking zero
            clr first_bit
  card_read4:rrc a
            jnb strobe,$                    ; wait for clock = 1
            djnz r2,card_read3
            rrc a
            rrc a
            rrc a
            cpl a                           ; complement data
            anl a,#00011111b
            mov @r0,a
            inc r0
            cjne r0,#58h,card_read2         ; check for 40 byte of data
     return:ret
;................... End of subroutine -- card_read ..............................
;..................... subroutine -- check_data ..................................
  check_data:call parity_ck                 ; count for 40 bytecall parity_ck
            jb card_err,err_found
            call lrc_ck
            jb card_err,err_found
            call final_ck
            jb card_err,err_found
            ret
  err_found:mov a,#40h
            mov dptr,#port_b
            movx @dptr,a
            ret
;.........................................
   parity_ck:clr card_err                   ; set to no errer
            mov r0,#30h                     ; point to first byte
            mov r1,#40                      ; count for 40 data
  parity_ck2:mov a,@r0
            cjne a,#0,parity_ck3
            ret                             ; it's possible that there're less than
                                            ; 40 character or there's no data
  parity_ck3:mov a,@r0
            jnb psw.0,parity_err            ; if psw.0 is set,it means that data
                                            ; is odd parity
            inc r0
            djnz r1,parity_ck2
            ret                             ; return with card_err = 0 --> no errer
  parity_err:setb card_err                  ; set error bit
            ret
;.........................................
      lrc_ck:clr card_err
            mov r0,#30h                     ; point to first data
            mov r2,#00h                     ; use to store result of xor
     ex_or:mov a,@r0
            anl a,#0fh                      ; mask off parity bit
            xch a,r2
            xrl a,r2
            xch a,r2                        ; now acc contain data and r2 contain
result
            inc r0
            cjne a,#0fh,ex_or               ; find end sentinel
            mov a,@r0                       ; move LRC data from card to accumulator
            anl a,#0fh                      ; mask off parity bit of LRC
            subb a,r2
            cjne a,#00h,lrc_err
            ret
   lrc_err:setb card_err
            ret
;.........................................
    final_ck:clr card_err
            mov r0,#30h                     ; point to first data of card
            mov a,@r0
            cjne a,#00001011b,final_err
            mov r0,#57h                     ; point to last data of card
     try_ck:mov a,@r0
            cjne a,#00h,final_ck2
            dec r0
            cjne r0,#30h,try_ck
            jmp final_err
  final_ck2:dec r0
            mov a,@r0
            cjne a,#00011111b,final_err     ; check whether it's es or not
            ret
   final_err:setb card_err
            ret
;---------------------- End of subroutine -- Check_data ----------------------------
;---------- subroutine -- convert data to ascii and send to computer --------------
            ; r0 : point to first data
            ; r2 : check for 40 data
     convert:mov r0,#30h
            mov r2,#40

      and1:mov a,@r0
            anl a,#00001111b
            mov @r0,a

     start:cjne@r0,#0001011b,zero
            mov @r0,#0bbh
            inc r0
            djnz r2,and1
            mov r0,#30h                     ; use to point to 20h while send data
            mov r2,#40                      ; count for sending 40 byte of data
            ljmp senddata
```

```
 zero:cjne@r0,#00000000b,one
      mov @r0,#30h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

  one:cjne@r0,#00000001b,two
      mov @r0,#0b1h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

  two:cjne@r0,#00000010b,three
      mov @r0,#0b2h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

three:cjne@r0,#00000011b,four
      mov @r0,#33h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

 four:cjne@r0,#00000100b,five
      mov @r0,#0b4h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

 five:cjne@r0,#00000101b,six
      mov @r0,#35h
      inc r0
      djnz r2,and1
      mov r0,#30h
      mov r2,#40
      ljmp senddata

  six:cjne@r0,#00000110b,seven
      mov @r0,#36h
      inc r0
      djnz r2,tem1
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem1:ljmp and1

seven:cjne@r0,#00000111b,eight
      mov @r0,#0b7h
      inc r0
      djnz r2,tem2
      mov r0,#30h
      mov r2,#40
```



```
      ljmp senddata
 tem2:ljmp and1

eight:cjne@r0,#00001000b,nine
      mov @r0,#0b8h
      inc r0
      djnz r2,tem3
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem3:ljmp and1

 nine:cjne@r0,#00001001b,as
      mov @r0,#39h
      inc r0
      djnz r2,tem4
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem4:ljmp and1

   as:cjne@r0,#00001010b,nd
      mov @r0,#3ah
      inc r0
      djnz r2,tem5
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem5:ljmp and1

   nd:cjne@r0,#00001100b,fs
      mov @r0,#3ch
      inc r0
      djnz r2,tem6
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem6:ljmp and1

   fs:cjne@r0,#00001101b,nd_2
      mov @r0,#0bdh
      inc r0
      djnz r2,tem7
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem7:ljmp and1

 nd_2:cjne@r0,#00001110b,es1
      mov @r0,#0beh
      inc r0
      djnz r2,tem8
      mov r0,#30h
      mov r2,#40
      ljmp senddata
 tem8:ljmp and1

  es1:cjne@r0,#00001111b,error
      mov @r0,#3fh
      inc r0
      djnz r2,tem9
      mov r0,#30h
      mov r2,#40
```

```
            ljmp senddata
     tem9:ljmp and1
;.....................................
    error:mov a,#0ffh
          mov dptr,#port_b
          movx @dptr,a
          mov @r0,#5ah                    ; ascii for "Z"
          inc r0
          djnz r2,tem0
          mov r0,#30h
          mov r2,#40
          ljmp senddata
     tem0:ljmp and1
;.....................................

senddata:setb tr1                         ;start timer1 with start to send data
          call send2bits                  ;send 2bits check status
          mov a,@r0
          clr ti
          mov sbuf,a
    wait1:jnb ti,wait1                    ;wait for send 1 byte(ti=0 replace)
          inc r0
          djnz r2,again
          mov a,#20h
          mov dptr,#port_b
          movx @dptr,a
          jnberr_status3,invalid_chk
          ret
    again:mov a,@r0
          clr ti
          mov sbuf,a
          jmp wait1
;.....................................
 invalid_chk:mov r6,#2
          mov r1,#6ah
          clr ri                          ;don't want to recieve data
          setb tr1                        ;start baudrate 2400 BPS
          setb ren
wait_invalid:jnbri,wait_invalid
          mov a,sbuf
          mov @r1,a
          inc r1
          clr ri
          djnz r6,wait_invalid
          ;clr tr1
          ret
;------ End of subroutine -- convert data to ascii and send back to computer ------
;-------------------------------------------------------------------------------
   delayx:mov r6,#06h    ;number of round
 againshowx:mov 8ch,#00h     ;set th0=00h
          mov 8ah,#00h     ;set tl0=00h
          setb tr0         ;start timmer
 waitshowx:jnbtf0,waitshowx
          clr tf0
          clr tr0
          djnz r6,againshowx
          ret
;-------------------------------------------------------------------------------
  delayshow:mov 8ch,#00h     ;set th0=00h
          mov 8ah,#00h     ;set tl0=00h
          setb tr0         ;start timmer
  waitshow:jnbtf0,waitshow
          clr tf0
          clr tr0
          djnz r6,delayshow
          ret
;-------------------------------------------------------------------------------
  send2bits:mov a,21h
          clr ti
          mov sbuf,a
    wait0:jnb ti,wait0
          mov a,22h
          clr ti
          mov sbuf,a
   wait00:jnb ti,wait00
          clr ti
          ret
;------------------ subroutine -- scan key -----------------------------------
  scan_pass:mov r5,#3
 scan_again:mov r0,#27h
          mov r7,#4
          mov a,#14h
          lcall lcd_set
          lcall delay
    row1:mov 90h,#0ffh
          clr p1.0
   col11:jb p1.4,col12
          lcall delayx
          mov @r0,#01h
          mov a,#00101010b
          lcall lcd_write
          lcall delay
          inc r0
          djnz r7,row1
          jmp go_on
   col12:jb p1.5,col13
          lcall delayx
          mov @r0,#02h
          mov a,#00101010b
          lcall lcd_write
          lcall delay
          inc r0
          djnz r7,row1
          jmp go_on
   col13:jb p1.6,row2
          lcall delayx
          mov @r0,#03h
          mov a,#00101010b
          lcall lcd_write
          lcall delay
          inc r0
          djnz r7,row1
          jmp go_on
    row2:mov 90h,#0ffh
          clr p1.1
   col21:jb p1.4,col22
          lcall delayx
          mov @r0,#04h
          mov a,#00101010b
          lcall lcd_write
          lcall delay
          inc r0
          djnz r7,row1
          jmp go_on
   col22:jb p1.5,col23
          lcall delayx
```

```
        mov @r0,#05h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,row1
        jmp go_on
 col23:jb p1.6,row3
        lcall delayx
        mov @r0,#06h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,temp
        jmp go_on
  row3:mov 90h,#0ffh
        clr p1.2
 col31:jb p1.4,col32
        lcall delayx
        mov @r0,#07h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,temp
        jmp go_on
 col32:jb p1.5,col33
        lcall delayx
        mov @r0,#08h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,temp
        jmp go_on
 col33:jb p1.6,row4
        lcall delayx
        mov @r0,#09h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,temp
        jmp go_on
  row4:mov 90h,#0ffh
        clr p1.3
 col42:jb p1.5,clear_b
        lcall delayx
        mov @r0,#00h
        mov a,#00101010b
        lcall lcd_write
        lcall delay
        inc r0
        djnz r7,temp
        jmp go_on
clear_b:jb p1.4,temp
        lcall delayx
        lcall show_p
        jmp scan_again
   temp:ljmp row1
;.....................................................
  go_on:mov 90h,#0ffh
```



```
        clr p1.3
 enter:jb p1.7,clear_b2
        lcall delayx
        mov a,23h
        anl a,#0fh
        mov 23h,a
        mov a,24h
        anl a,#0fh
        mov 24h,a
        mov a,25h
        anl a,#0fh
        mov 25h,a
        mov a,26h
        anl a,#0fh
        mov 26h,a
        jmp compare

clear_b2:jb p1.4,enter
        lcall delayx
        lcall show_p
        ljmp scan_again
 temp2:ljmp p_error
compare:mov a,23h
        cjne a,27h,temp2
        mov a,24h
        cjne a,28h,temp2
        mov a,25h
        cjne a,29h,temp2
        mov a,26h
        cjne a,2ah,temp2
        mov a,#20h
        mov dptr,#port_b
        movx @dptr,a
;.............................. send status .....................................
sendstatus:mov 21h,#11110000b
        mov 22h,#11110011b
        lcall send_40
;........................ display " System Unlock ^_^ ...........................
        mov a,#01h
        lcall lcd_set
        lcall delay
        mov a,#80h
        lcall lcd_set
        lcall delay
        mov a,#'S'
        lcall lcd_write
        lcall delay
        mov a,#'y'
        lcall lcd_write
        lcall delay
        mov a,#'s'
        lcall lcd_write
        lcall delay
        mov a,#'t'
        lcall lcd_write
        lcall delay
        mov a,#'e'
        lcall lcd_write
        lcall delay
        mov a,#'m'
        lcall lcd_write
        lcall delay
        mov a,#14h
```

```
        lcall lcd_set
        lcall delay
        mov a,#'U'
        lcall lcd_write
        lcall delay
        mov a,#0c0h
        lcall lcd_set
        lcall delay
        mov a,#'n'
        lcall lcd_write
        lcall delay
        mov a,#'l'
        lcall lcd_write
        lcall delay
        mov a,#'o'
        lcall lcd_write
        lcall delay
        mov a,#'c'
        lcall lcd_write
        lcall delay
        mov a,#'k'
        lcall lcd_write
        lcall delay
        mov a,#01011110b
        lcall lcd_write
        lcall delay
        mov a,#01011111b
        lcall lcd_write
        lcall delay
        mov a,#01011110b
        lcall lcd_write
        lcall delay
        mov a,#00000001b
        mov dptr,#port_c
        movx @dptr,a
door_ck:movx a,@dptr
        anl a,#11101111b
        cjne a,#11111111b,door_ck
        clr err_status
        ret
p_error:mov a,#01h
        lcall lcd_set
        lcall delay
        mov a,#80h
        lcall lcd_set
        lcall delay
        mov a,#'I'
        lcall lcd_write
        lcall delay
        mov a,#'n'
        lcall lcd_write
        lcall delay
        mov a,#'c'
        lcall lcd_write
        lcall delay
        mov a,#'o'
        lcall lcd_write
        lcall delay
        mov a,#'r'
        lcall lcd_write
        lcall delay
        mov a,#'r'
        lcall lcd_write
```



```
        lcall delay
        mov a,#'e'
        lcall lcd_write
        lcall delay
        mov a,#'c'
        lcall lcd_write
        lcall delay
        mov a,#0c0h
        lcall lcd_set
        lcall delay
        mov a,#'t'
        lcall lcd_write
        lcall delay
        mov a,#14h
        lcall lcd_write
        lcall delay
        mov a,#00100001b          ; ! sign
        lcall lcd_write
        lcall delay
        mov a,#40h
        mov dptr,#port_b
        movx @dptr,a
        mov r6,#50                ; set for delay
        lcall delayshow
        djnz r5,temp1
        setb err_status
        ret
  temp1:lcall show_p
        jmp scan_again
;---------------------- end of subroutine -- scan kay -------------------------------
;---------------------- subroutine -- send 42 character -----------------------------
send_40:mov r2,#40
        setb tr1                          ;start timer1 with start to send data
        call send2bits                    ;send 2bits check status
again_1:mov a,#00h
        clr ti
        mov sbuf,a
 wait_1:jnb ti,wait_1                     ;wait for send 1 byte(ti=0 replace)
        djnz r2,again_1
        clr ti
        ret
;------------------- End of subroutine -- send 42 character ------------------------

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Subroutine Light display >>>>>>>>>>>>>>>>>>>>>>

light_display:clr ea
        clr es
        mov r7,#8
        mov dptr,#port_b2
        movx a,@dptr
        mov r2,a
        mov 21h,#6fh          ;o
        mov 22h,#0f0h         ;p
        call send2bits
        mov a,#0bbh           ; ;
        setb tr1                          ;start timer1 with start to send data
        clr ti
        mov sbuf,a
 wait_b:jnb ti,wait_b                     ;wait for send 1 byte(ti=0 replace)
        clr ti
light_a:mov a,r2
        rlc a
        mov r2,a
```

Form1 - 1

```
Private Sub Command1_Click()
Form4.Show
End Sub

Private Sub Command2_Click()
Form3.Show
End Sub

Private Sub command3_Click()
Unload Me
Unload Form2
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form7
End Sub

Private Sub command4_Click()
Form6.Show
End Sub

Private Sub command5_Click()
Form7.Show
End Sub
```

Form2 - 1

```
Dim str As Variant
Dim check As Variant
Dim code1 As String
Dim checkstatus As Boolean
Public projectWs As Workspace
Public projectDb As Database
Public memberdyn As Recordset
Public finddyn As Recordset
Private Sub Command1_Click()
Form5.Show
End Sub
'เริ่มต้น โหลดฟอร์ม
Private Sub Form_Load()
 On Error GoTo handleError
 Unload Form4
 Unload Form7
 '**********************************************
 MSComm1.CommPort = 1
 MSComm1.Settings = "2400,e,7,1"
 MSComm1.InputLen = 0
 MSComm1.PortOpen = True
 MSComm1.RThreshold = 42
 MSComm1.InBufferCount = 0
 '**********************************************
 Set projectWs = DBEngine.Workspaces(0)
 Set projectDb = projectWs.OpenDatabase("database.mdb", False, False)
 Set memberdyn = projectDb.OpenRecordset("SELECT * FROM member ORDER BY name", dbOpenDynaset)
 checkstatus = False
 Command1.Enabled = False
 memberdyn.MoveFirst
 Call displayfield
 Call savecancel(False)
 addstatus = False
 code1 = "M3"
 Timer1.Enabled = True
 Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
Private Sub displayfield()
'show data
login.Text = memberdyn.Fields("ID")
pass.Text = memberdyn.Fields("password")
name1.Text = memberdyn.Fields("name")
surname.Text = memberdyn.Fields("surname")
Text2.Text = memberdyn.Fields("pic")
Image1.Picture = LoadPicture("picture\" & Text2.Text)
End Sub
```

Form2 - 2

```
Private Sub l1_Click()
memberdyn.MovePrevious
If memberdyn.BOF = False Then
  Call displayfield
  enableoperation (True)
  l1.Enabled = True
  ll1.Enabled = True
  r1.Enabled = True
  rr1.Enabled = True
Else
  enableoperation (False)
  l1.Enabled = False
  ll1.Enabled = False
  r1.Enabled = True
  rr1.Enabled = True
End If
End Sub
Private Sub ll1_Click()
memberdyn.MoveFirst
  Call displayfield
enableoperation (True)
l1.Enabled = False
ll1.Enabled = False
r1.Enabled = True
rr1.Enabled = True
End Sub
Private Sub r1_Click()
memberdyn.MoveNext
If memberdyn.EOF = False Then
  Call displayfield
  enableoperation (True)
  l1.Enabled = True
  ll1.Enabled = True
  r1.Enabled = True
  rr1.Enabled = True
Else
   enableoperation (False)
  r1.Enabled = False
  rr1.Enabled = False
   l1.Enabled = True
   ll1.Enabled = True
End If
End Sub
Private Sub rr1_Click()
memberdyn.MoveLast
Call displayfield
enableoperation (True)
l1.Enabled = True
```

Form2 - 3

```
ll1.Enabled = True
r1.Enabled = False
rr1.Enabled = False
End Sub
Private Sub cmdexit1_Click()
Unload Me
Form1.Show
End Sub
'add edit cancel
Private Sub enableoperation(Enableflag As Boolean)
cmdadd1.Enabled = Enableflag
cmdedit1.Enabled = Enableflag
cmddel1.Enabled = Enableflag
End Sub
'save  cancel
Private Sub savecancel(Enableflag As Boolean)
cmdsave1.Enabled = Enableflag
cmdcancel1.Enabled = Enableflag
End Sub
'Clear Textbox
Private Sub clear()
login.Text = ""
pass.Text = ""
name1.Text = ""
surname.Text = ""
End Sub
'add1
Private Sub cmdadd1_Click()
On Error GoTo handleError
addstatus = True
r1.Enabled = False
rr1.Enabled = False
l1.Enabled = False
ll1.Enabled = False
login.SetFocus
Call enablewrite
Call enableoperation(False)
Call savecancel(True)
login.Text = ""
Command1.Enabled = True
Call clear
login.SetFocus
memberdyn.AddNew
checkstatus = True
Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
```

Form2 - 4

```
'cancel1
Private Sub cmdcancel1_Click()
checkstatus = False
If memberdyn.EditMode = dbEditAdd Or memberdyn.EditMode = dbEditInProgress Then
   memberdyn.CancelUpdate
End If
login.Text = ""
Command1.Enabled = False
Call clear
memberdyn.MoveFirst
Call displayfield
r1.Enabled = True
rr1.Enabled = True
l1.Enabled = False
ll1.Enabled = False
Call enableoperation(True)
Call savecancel(False)
Call disablewrite
End Sub
'delete1
Private Sub cmddel1_Click()
On Error GoTo handleError
If MsgBox("Are you sure to delete all data of this member form database??", vbYesNo) = vbYes Then
 memberdyn.Delete
  Call ll1_Click
End If
Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
'edit1
Private Sub cmdedit1_Click()
On Error GoTo handleError
Call enableoperation(False)
Call savecancel(True)
r1.Enabled = False
rr1.Enabled = False
l1.Enabled = False
ll1.Enabled = False
login.SetFocus
Command1.Enabled = True
Call enablewrite
memberdyn.Edit
Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
'enable to write data
```

Form2 - 5

```
Private Sub enablewrite()
login.Locked = False
pass.Locked = False
name1.Locked = False
surname.Locked = False
End Sub
'disable to write data
Private Sub disablewrite()
login.Locked = True
pass.Locked = True
name1.Locked = True
surname.Locked = True
End Sub
'save1
Private Sub cmdsave1_Click()
On Error GoTo handleError
checkstatus = False
If login.Text = "" Then
    MsgBox "Please swipe your card to detect login name.", vbCritical
    login.SetFocus
ElseIf pass.Text = "" Then
    MsgBox "Please enter your password by Number 4 digits.", vbCritical
    pass.SetFocus
ElseIf name1.Text = "" Then
    MsgBox "What is your Name?  You don't tell me yet. ", vbQuestion
    name1.SetFocus
ElseIf surname.Text = "" Then
    MsgBox "What is your Surname?  You don't tell me yet. ", vbQuestion
    surname.SetFocus
Else
    surname.SetFocus
    memberdyn("ID") = login.Text
    memberdyn("name") = name1.Text
    memberdyn("surname") = surname.Text
    memberdyn("password") = pass.Text
    memberdyn("pic") = Text2.Text
memberdyn.Update
memberdyn.Requery
MsgBox "Your data have been updated in database already.", vbSystemModal, "Congratulation @^_^@"
memberdyn.MoveFirst
Call displayfield
r1.Enabled = True
rr1.Enabled = True
l1.Enabled = False
ll1.Enabled = False
Call enableoperation(True)
Call savecancel(False)
Call disablewrite
```

```
Command1.Enabled = False
End If
Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
ชิพสถาวะดีเทคการรูคบัตรใด้
Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
 Case comEventBreak
  MsgBox "A break was received", vbExclamation
 Case comEventCDTO
  MsgBox "CD (RLSD)Timeout", vbExclamation
 Case comEventCTSTO
  MsgBox "CTS Time out", vbExclamation
 Case comEventDSRTO
  MsgBox "DSR Time out", vbExclamation
 Case comEventFrame
  MsgBox "Framing Error", vbExclamation
 Case comEventOverrun
  MsgBox "Data loss", vbExclamation
 Case comEventRxOver
  MsgBox "Receive buffer overflow", vbExclamation
 Case comEventRxParity
  MsgBox "Parity Error", vbExclamation
 Case comEventTxFull
  MsgBox "Transmit buffer full", vbExclamation
 Case comEventDCB
  MsgBox "Unexpected error retrieving", vbExclamation
 ' Event
 Case comEvCD
  MsgBox "Change in the CD line", vbExclamation
 Case comEvCTS
  MsgBox "Change in the CTS line", vbExclamation
 Case comEvDSR
  MsgBox "Change in the DSR line", vbExclamation
 Case comEvRing
  MsgBox "Change in the ring indicator", vbExclamation
'*****************************************************************************
 Case comEvReceive
  str = MSComm1.Input
  check = Mid$(str, 1, 2)
  If check = "de" Then
   Call handleinput(str)
   Call seekdata
   End If
'*****************************************************************************
 Case comEvSend
```



```
  MsgBox "There are SThreshold number of characters in the transmit buffer"
 Case comEvEOF
  MsgBox "An EOF characters was found in the input stream", vbExclamation
 End Select
End Sub
Sub handleinput(str As Variant)
Dim a, b, data1 As Variant
 a = InStr(1, str, ";", 0)
 If a <> 0 Then
 b = InStr(1, str, "?", 0)
 data1 = Mid(str, a, (b - a) + 1)
 login.Text = data1
 Text1.Text = str
 Else: MsgBox "Data error!!!  Please insert card and try again", vbExclamation
 Text1.Text = str
End If
End Sub
Sub seekdata()
Dim count1 As Integer
Dim data2, data3 As String
Set finddyn = projectDb.OpenRecordset("SELECT * FROM member WHERE member.ID = " & "" & login.Text & "", dbOpenDynaset)
count1 = finddyn.RecordCount
If count1 <> 0 Then
pass.Text = finddyn.Fields("password")
name1.Text = finddyn.Fields("name")
surname.Text = finddyn.Fields("surname")
Text2.Text = finddyn("pic")
Image1.Picture = LoadPicture("picture\" & Text2.Text)
ElseIf checkstatus = False Then
        If MsgBox("Your card data was not found in database , Do you want to add new card? ", vbYesNo) = vbYes Then
        Call cmdadd1_Click
        data2 = Text1.Text
        a = InStr(1, data2, ";", 0)
          If a <> 0 Then
          b = InStr(1, data2, "?", 0)
          data3 = Mid(str, a, (b - a) + 1)
          login.Text = data3
          Else: MsgBox "Data error!!!  Please insert card and try again)", vbExclamation
          End If
          End If
    Else: data2 = Text1.Text
          a = InStr(1, data2, ";", 0)
          If a <> 0 Then
          b = InStr(1, data2, "?", 0)
          data3 = Mid(str, a, (b - a) + 1)
          login.Text = data3
          Else: MsgBox "Data error!!!  Please insert card and try again)", vbExclamation
          End If
```

```
End If
End Sub
Private Sub Timer1_Timer()
MSComm1.InBufferCount = 0
MSComm1.Output = code1
Timer1.Enabled = False
End Sub
```



```
Private Sub Check1_Click()
If Check1.Value = vbChecked Then
Text1.Enabled = False
Command1.Enabled = False
Text2.Enabled = True
Text3.Enabled = True
Command2.Enabled = True
Else: Text1.Enabled = True
Command1.Enabled = True
Text2.Enabled = False
Text3.Enabled = False
Command2.Enabled = False
End If
End Sub
Private Sub Command1_Click()
Dim filenumber As Integer
Dim code As String
filenumber = FreeFile(1)
Open "code.sys" For Input As #1
Line Input #1, code
If Text1.Text = code Then
Close
Form2.Show
Unload Me
Else: MsgBox("คุณใส่รหัสผ่านไม่ถูกต้อง ไม่สามารถโหลดหน้าจอฐานข้อมูลขึ้นมาได้")
Close
Text1 = ""
End If
End Sub
Private Sub Command2_Click()
Dim filenumber As Integer
Dim code As String
Dim code2 As String
filenumber = FreeFile(1)
Open "code.sys" For Input As #1
Line Input #1, code
If Text2.Text <> code Then
MsgBox("คุณใส่รหัสผ่านไม่ถูกต้อง ไม่สามารถทำการเปลี่ยนรหัสให้ท่านได้")
Text2 = ""
Text3 = ""
Text2.SetFocus
GoTo again
ElseIf Text3 = "" Then
    MsgBox("กรุณาใส่รหัสใหม่ที่ต้องการเปลี่ยนแปลงด้วย")
    Text3.SetFocus
    GoTo again
End If
code = Text3.Text
```

Form3 - 2

```
code2 = InputBox("กรุณาใส่รหัสที่ต้องการเปลี่ยนใหม่ เพื่อเป็นการยืนยันอีกครั้ง", "Confirm")
      If code2 <> code Then
        MsgBox("ท่านยืนยันรหัสผิด ไม่สามารถทำการเปลี่ยนรหัสให้ท่านได้ กรุณาเข้ามาใหม่อีกครั้ง")
        Text2 = ""
        Text3 = ""
        Text2.SetFocus
        GoTo again
      Else: Close
      End If
Open "code.sys" For Output As #1
code = code + Chr$(13)
Print #1, code
MsgBox("ขณะนี้ได้โปรแกรมทำการเปลี่ยนแปลงรหัสผ่านที่คุณต้องการเรียบร้อยแล้ว")
Text2 = ""
Text3 = ""
again: Close
End Sub
Private Sub Form_Load()
Text2.Enabled = False
Text3.Enabled = False
Command2.Enabled = False
End Sub
```

Form4 - 1

```
Dim str As Variant
Dim check As Variant
Dim code1 As Variant
Dim setcmd As String
Public projectWs As Workspace
Public projectDb As Database
Public memberdyn As Recordset
Public finddyn As Recordset
Public statusdyn As Recordset
Private Sub Command1_Click()
Dim mink As Variant
Dim i As Integer
mink = name1.Text
If Len(name1.Text) < 10 Then
  For i = Len(name1.Text) To 11
  mink = mink & " "
  Next i
Else: mink = name1.Text
End If
MSComm1.Output = mink
Timer2.Enabled = True
End Sub
```

เริ่มต้น โหลดฟอร์ม

```
Private Sub Form_Load()
 On Error GoTo handleError
 Unload Form2
 Unload Form7
 '*************************************************
 MSComm1.CommPort = 1
 MSComm1.Settings = "2400,e,7,1"
 MSComm1.InputLen = 0
 MSComm1.PortOpen = True
 MSComm1.RThreshold = 42
 MSComm1.InBufferCount = 0
 '*************************************************
 Dim x As Integer
 code1 = "M1"
 Timer5.Enabled = True
 Set projectWs = DBEngine.Workspaces(0)
 Set projectDb = projectWs.OpenDatabase("database.mdb", False, False)
 Set memberdyn = projectDb.OpenRecordset("SELECT * FROM member ORDER BY name", dbOpenDynaset)
 memberdyn.MoveFirst
 Call displayfield
 Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
Private Sub displayfield()
```

```
Dim temppics As String
'show data
login.Text = memberdyn.Fields("ID")
pass.Text = memberdyn.Fields("password")
name1.Text = memberdyn.Fields("name")
surname.Text = memberdyn.Fields("surname")
temppics = memberdyn.Fields("pic")
Image1.Picture = LoadPicture("picture\" & temppics)
End Sub
Private Sub cmdexit1_Click()
Unload Me
Form1.Show
End Sub
พิศสภาวะลิเทคการรูดบัตรได้
Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
 Case comEventBreak
  MsgBox "A break was received", vbExclamation
 Case comEventCDTO
  MsgBox "CD (RLSD)Timeout", vbExclamation
 Case comEventCTSTO
  MsgBox "CTS Time out", vbExclamation
 Case comEventDSRTO
  MsgBox "DSR Time out", vbExclamation
 Case comEventFrame
  MsgBox "Framing Error", vbExclamation
 Case comEventOverrun
  MsgBox "Data loss", vbExclamation
 Case comEventRxOver
  MsgBox "Receive buffer overflow", vbExclamation
 Case comEventRxParity
  MsgBox "Parity Error", vbExclamation
 Case comEventTxFull
  MsgBox "Transmit buffer full", vbExclamation
 Case comEventDCB
  MsgBox "Unexpected error retrieving", vbExclamation
 ' Event
 Case comEvCD
  MsgBox "Change in the CD line", vbExclamation
 Case comEvCTS
  MsgBox "Change in the CTS line", vbExclamation
 Case comEvDSR
  MsgBox "Change in the DSR line", vbExclamation
 Case comEvRing
  MsgBox "Change in the ring indicator", vbExclamation
'*************************************************************************
 Case comEvReceive
  str = MSComm1.Input
```



```
 check = Mid$(str, 1, 2)
 If check = "de" Then
   Call handleinput(str)
   Call seekdata_de
 End If
 If check = "en" Then
   Call handleinput(str)
   Call seekdata_en
 End If
 If check = "ps" Then                    'password pass
   Call writestatus_de
 End If
'*************************************************************************
 Case comEvEOF
  MsgBox "An EOF characters was found in the input stream", vbExclamation
End Select
End Sub
Sub handleinput(str As Variant)
Dim a, b, data1 As Variant
 a = InStr(1, str, ";", 0)
 If a <> 0 Then
 b = InStr(1, str, "?", 0)
 data1 = Mid(str, a, (b - a) + 1)
 login.Text = data1
 Text1.Text = str
 Else: MsgBox "Data error!!!  Please insert card and try again", vbExclamation
 Text1.Text = str
End If
End Sub
Sub seekdata_en()
Dim count1 As Integer
Dim temp As String
Set finddyn = projectDb.OpenRecordset("SELECT * FROM member WHERE member.ID = " & "'" & login.Text & "'", dbOpenDynaset)
count1 = finddyn.RecordCount
If count1 <> 0 Then
code1 = "cv"
Timer3.Enabled = True
pass.Text = finddyn.Fields("password")
name1.Text = finddyn.Fields("name")
surname.Text = finddyn.Fields("surname")
temp = finddyn.Fields("pic")
Image1.Picture = LoadPicture("picture\" & temp)
Call writestatus_en
Else: code1 = "ci"
Timer3.Enabled = True
End If
End Sub
Sub seekdata_de()
```

Form4 - 4

```
Dim count1 As Integer
Dim codepass As Variant
Set finddyn = projectDb.OpenRecordset("SELECT * FROM member WHERE member.ID = " & "'" & login.Text & "'", dbOpenDynaset)
count1 = finddyn.RecordCount
If count1 <> 0 Then
code1 = "cv"
Timer3.Enabled = True
pass.Text = finddyn.Fields("password")
name1.Text = finddyn.Fields("name")
surname.Text = finddyn.Fields("surname")
Timer1.Enabled = True
Else: code1 = "ci"
Timer3.Enabled = True
End If
End Sub
Private Sub Timer1_Timer()
Call Command1_Click
Timer1.Enabled = False
End Sub
Private Sub Timer2_Timer()
MSComm1.Output = pass.Text
Timer2.Enabled = False
End Sub
Private Sub Timer3_Timer()
MSComm1.Output = code1
Timer3.Enabled = False
End Sub
Private Sub Timer4_Timer()
date1.Caption = Format(Date, "Dd/Mm/Yy")
time1.Caption = Format(Time, "Hh : Mm : Ss")
End Sub
Sub writestatus_en()
Set statusdyn = projectDb.OpenRecordset("SELECT * FROM status", dbOpenDynaset)
statusdyn.AddNew
statusdyn("date") = Date
statusdyn("ID") = login.Text
statusdyn("time") = Time
statusdyn("in_out") = "OUT"
statusdyn.Update
statusdyn.Requery
End Sub
Sub writestatus_de()
Set statusdyn = projectDb.OpenRecordset("SELECT * FROM status", dbOpenDynaset)
statusdyn.AddNew
statusdyn("date") = Date
statusdyn("ID") = login.Text
statusdyn("time") = Time
statusdyn("in_out") = "IN"
```

Form4 - 5

```
statusdyn.Update
statusdyn.Requery
End Sub
Private Sub Timer5_Timer()
MSComm1.InBufferCount = 0
MSComm1.Output = code1
Timer5.Enabled = False
End Sub
```

Form5 - 1

```
Private Sub Command1_Click()
Unload Form5
End Sub
Private Sub Command2_Click()
Dim temppic As String
temppic = Form2.Text2.Text
Form2.Text2.Text = temppic
Unload Form5
End Sub
Private Sub Dir1_Change()
File1.Path = Dir1.Path
End Sub
Private Sub Drive1_Change()
Dir1.Path = Drive1.Drive
End Sub
Private Sub File1_Click()
Form2.Text2 = File1.FileName
Form2.Image1.Picture = LoadPicture("picture\" & Form2.Text2.Text)
End Sub
```

Form6 - 1

```
Dim setcmd As String
Dim keeprecord, i As Variant
Public pkwWs As Workspace
Public pkwDb As Database
Public statusdyn As Recordset
Private Sub cmdexit1_Click()
Unload Me
End Sub
Private Sub Command1_Click()
DataGrid2.Enabled = True
DataGrid2.Visible = True
ado2.Visible = True
setcmd = "SELECT date,time,in_out,name,surname FROM status LEFT JOIN member  ON status.ID=member.ID  WHERE status.date = "
setcmd = setcmd & "'" & Format(datetime, "Dd/Mm/Yy") & "'" & " ORDER BY date"
ado2.Enabled = False
ado2.RecordSource = setcmd
ado2.Enabled = True
ado2.Refresh
End Sub
Private Sub Form_Load()
Call listall
Set pkwWs = DBEngine.Workspaces(0)
Set pkwDb = pkwWs.OpenDatabase("database.mdb", False, False)
Set statusdyn = pkwDb.OpenRecordset("SELECT * FROM status ORDER BY date", dbOpenDynaset)
datetime.Value = Format(Date, "Dd,Mm,Yy")
If statusdyn.RecordCount = 0 Then
remove.Enabled = False
End If
End Sub
Private Sub Command2_Click()
Call listall
End Sub
Sub listall()
DataGrid2.Enabled = True
DataGrid2.Visible = True
ado2.Visible = True
setcmd = "SELECT date,time,in_out,name,surname FROM status LEFT JOIN member  ON status.ID=member.ID ORDER BY date"
ado2.Enabled = False
ado2.RecordSource = setcmd
ado2.Enabled = True
ado2.Refresh
End Sub
Private Sub remove_Click()
Set statusdyn = pkwDb.OpenRecordset("SELECT * FROM status ORDER BY date", dbOpenDynaset)
keeprecord = statusdyn.RecordCount
statusdyn.MoveFirst
For i = 1 To keeprecord
statusdyn.Delete
```

Form6 - 2

```
statusdyn.MoveNext
Next i
Timer1.Enabled = True
End Sub
Private Sub Timer1_Timer()
ado2.Refresh
Timer1.Enabled = False
remove.Enabled = False
End Sub
```

Form7 - 1

```
Dim str As Variant
Dim check As Variant
Dim code1 As Variant
Dim data1, data2 As String
Dim I1, I2, I3, I4, I5, I6, I7, I8 As String
Private Sub cmdexit1_Click()
Unload Me
End Sub
Private Sub Command9_Click()
Form8.Show
End Sub
Private Sub Form_Load()
'เริ่มต้น โหลดฟอร์ม
 On Error GoTo handleError
 Unload Form4
 Unload Form2
 '*****************************************
 MSComm1.CommPort = 1
 MSComm1.Settings = "2400,e,7,1"
 MSComm1.InputLen = 0
 MSComm1.PortOpen = True
 MSComm1.RThreshold = 12
 MSComm1.InBufferCount = 0
 '*****************************************
 I1 = "off"
 I2 = "off"
 I3 = "off"
 I4 = "off"
 I5 = "off"
 I6 = "off"
 I7 = "off"
 I8 = "off"
 code1 = "M2"
 Timer1.Interval = 1000
 Timer1.Enabled = True
Exit Sub
handleError:
MsgBox Error(Err.Number)
End Sub
Private Sub Command1_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z1"
Call time_show
If I1 = "off" Then
Command1.Picture = LoadPicture("lon.ico")
I1 = "on"
ElseIf I1 = "on" Then
```

Form7 - 2

```
Command1.Picture = LoadPicture("loff.ico")
l1 = "off"
End If
End Sub
Private Sub Command2_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z2"
Call time_show
If l2 = "off" Then
Command2.Picture = LoadPicture("lon.ico")
l2 = "on"
ElseIf l2 = "on" Then
Command2.Picture = LoadPicture("loff.ico")
l2 = "off"
End If
Timer1.Interval = 100
Timer1.Enabled = True
End Sub
Private Sub command3_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z3"
Call time_show
If l3 = "off" Then
Command3.Picture = LoadPicture("lon.ico")
l3 = "on"
ElseIf l3 = "on" Then
Command3.Picture = LoadPicture("loff.ico")
l3 = "off"
End If
End Sub
Private Sub command4_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z4"
Call time_show
If l4 = "off" Then
Command4.Picture = LoadPicture("lon.ico")
l4 = "on"
ElseIf l4 = "on" Then
Command4.Picture = LoadPicture("loff.ico")
l4 = "off"
End If
End Sub
Private Sub command5_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
```

Form7 - 3

```
code1 = "Z5"
Call time_show
If l5 = "off" Then
Command5.Picture = LoadPicture("lon.ico")
l5 = "on"
ElseIf l5 = "on" Then
Command5.Picture = LoadPicture("loff.ico")
l5 = "off"
End If
End Sub
Private Sub command6_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z6"
Call time_show
If l6 = "off" Then
Command6.Picture = LoadPicture("lon.ico")
l6 = "on"
ElseIf l6 = "on" Then
Command6.Picture = LoadPicture("loff.ico")
l6 = "off"
End If
End Sub
Private Sub command7_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z7"
Call time_show
If l7 = "off" Then
Command7.Picture = LoadPicture("lon.ico")
l7 = "on"
ElseIf l7 = "on" Then
Command7.Picture = LoadPicture("loff.ico")
l7 = "off"
End If
End Sub
Private Sub command8_Click()
MSComm1.RThreshold = 0
Timer2.Enabled = True
code1 = "Z8"
Call time_show
If l8 = "off" Then
Command8.Picture = LoadPicture("lon.ico")
l8 = "on"
ElseIf l8 = "on" Then
Command8.Picture = LoadPicture("loff.ico")
l8 = "off"
End If
```

```
End Sub
```

ชนิดสภาวะดีเทคการรูดบัตรได้

```
Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
 Case comEventBreak
  MsgBox "A break was received", vbExclamation
 Case comEventCDTO
  MsgBox "CD (RLSD)Timeout", vbExclamation
 Case comEventCTSTO
  MsgBox "CTS Time out", vbExclamation
 Case comEventDSRTO
  MsgBox "DSR Time out", vbExclamation
 Case comEventFrame
  MsgBox "Framing Error", vbExclamation
 Case comEventOverrun
  MsgBox "Data loss", vbExclamation
 Case comEventRxOver
  MsgBox "Receive buffer overflow", vbExclamation
 Case comEventRxParity
  MsgBox "Parity Error", vbExclamation
 Case comEventTxFull
  MsgBox "Transmit buffer full", vbExclamation
 Case comEventDCB
  MsgBox "Unexpected error retrieving", vbExclamation
 ' Event
 Case comEvCD
  MsgBox "Change in the CD line", vbExclamation
 Case comEvCTS
  MsgBox "Change in the CTS line", vbExclamation
 Case comEvDSR
  MsgBox "Change in the DSR line", vbExclamation
 Case comEvRing
  MsgBox "Change in the ring indicator", vbExclamation
'*************************************************************************
Case comEvReceive
  str = MSComm1.Input              'xx;_____?
  check = Mid$(str, 1, 2)
  If check = "op" Then
  Call handleinput(str)
  End If
'*************************************************************************
  Case comEvEOF
  MsgBox "An EOF characters was found in the input stream", vbExclamation
 End Select
End Sub
Sub handleinput(str As Variant)
Dim a, b As Variant
Dim op As String
```



```
 a = InStr(1, str, ";", 0)
 If a <> 0 Then
 b = InStr(1, str, "?", 0)
 data1 = Mid(str, a, (b - a) + 1)
 Text1.Text = data1
 Else: MsgBox "Data error!!!  Please insert card and try again", vbExclamation
End If
'1
op = Mid$(data1, 2, 1)
If op = 1 Then
Command8.Picture = LoadPicture("lon.ico")
l8 = "on"
Else: Command8.Picture = LoadPicture("loff.ico")
l8 = "off"
End If
'2
op = Mid$(data1, 3, 1)
If op = 1 Then
Command7.Picture = LoadPicture("lon.ico")
l7 = "on"
Else: Command7.Picture = LoadPicture("loff.ico")
l7 = "off"
End If
'3
op = Mid$(data1, 4, 1)
If op = 1 Then
Command6.Picture = LoadPicture("lon.ico")
l6 = "on"
Else: Command6.Picture = LoadPicture("loff.ico")
l6 = "off"
End If
'4
op = Mid$(data1, 5, 1)
If op = 1 Then
Command5.Picture = LoadPicture("lon.ico")
l5 = "on"
Else: Command5.Picture = LoadPicture("loff.ico")
l5 = "off"
End If
'5
op = Mid$(data1, 6, 1)
If op = 1 Then
Command4.Picture = LoadPicture("lon.ico")
l4 = "on"
Else: Command4.Picture = LoadPicture("loff.ico")
l4 = "off"
End If
'6
```

```
op = Mid$(data1, 7, 1)
If op = 1 Then
Command3.Picture = LoadPicture("lon.ico")
l3 = "on"
Else: Command3.Picture = LoadPicture("loff.ico")
l3 = "off"
End If
'7
op = Mid$(data1, 8, 1)
If op = 1 Then
Command2.Picture = LoadPicture("lon.ico")
l2 = "on"
Else: Command2.Picture = LoadPicture("loff.ico")
l2 = "off"
End If
'8
op = Mid$(data1, 9, 1)
If op = 1 Then
Command1.Picture = LoadPicture("lon.ico")
l1 = "on"
Else: Command1.Piciure = LoadPicture("loff.ico")
l1 = "off"
End If
End Sub
Private Sub Timer1_Timer()
MSComm1.Output = code1
Timer1.Enabled = False
End Sub
Sub time_show()
Timer1.Interval = 50
Timer1.Enabled = True
End Sub
Private Sub Timer2_Timer()
 MSComm1.InBufferCount = 0
 MSComm1.RThreshold = 12
 Timer2.Enabled = False
End Sub
```

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

90-MILLIMETER COMPATIBLE
SWIPE READER
TECHNICAL REFERENCE MANUAL

Part Number 99831083-2

OCTOBER 1999

MAGTEK
20725 South Annalee Avenue
Carson, CA 90746
Phone: (310) 631-8602
FAX: (310) 631-3956
Technical Support: (888) 624-8350
www.magtek.com

Copyright © 1999
MAG-TEK, Inc.
Printed in the United States of America

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Mag-Tek, Inc.

Mag-Tek is a registered trademark of Mag-Tek, Inc.

REVISIONS

Rev Number	Date	Notes
1	Sep 92	Initial Release
2	22 Oct 99	Formatted manual; Added IEC requirements to Specifications; Changed values in Specifications; Clarified Connector information; Added correct drawings; Clarified Card Present Signal

Limited Warranty

Mag-Tek, Inc. (hereinafter "Mag-Tek") warrants this Mag-Tek product IN ITS ENTIRETY, to be in good working order for a period of 90 days from the date of purchase from Mag-Tek. Should this product fail to be in good working order at any time during this warranty period, Mag-Tek will, at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and products become the property of Mag-Tek. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, misuse, abuse, or non-Mag-Tek modification of the product.

Limited Warranty service may be obtained by delivering the product during the warranty period to Mag-Tek (20725 S. Annalee Ave., Carson, CA 90746). If this product is delivered by mail, you agree to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location and to use the original shipping container or equivalent.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO A PERIOD OF 90 DAYS FROM THE DATE OF PURCHASE, AND NO WARRANTIES, WHETHER EXPRESS OR IMPLIED, WILL APPLY AFTER THIS PERIOD. EXCEPT AS PROVIDED IN THE PRECEDING SENTENCE. EACH PURCHASER UNDERSTANDS THAT THE MAG-TEK PRODUCT IS OFFERED AS IS.

IF THIS PRODUCT IS NOT IN GOOD WORKING ORDER AS WARRANTED ABOVE, YOUR SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. IN NO EVENT WILL MAG-TEK BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE SUCH PRODUCT, EVEN IF MAG-TEK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

FCC Warning Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Canadian DOC Statement

This digital apparatus does not exceed the Class A limits for radio noise for digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de las classe A prescrites dans le Réglement sur le brouillage radioélectrique édicté par les ministère des Communications du Canada.

TABLE OF CONTENTS

FIGURES

TABLES

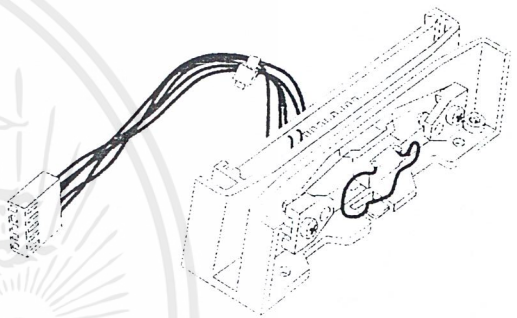


Figure 1-1. 90-millimeter Compatible Swipe Reader

SECTION 1. FEATURES AND SPECIFICATIONS

The 90-millimeter OEM Swipe Reader has a TTL level interface and is designed for use in retail, access control, and time and attendance environments. This Reader is in compliance with industry specifications, including ANSI/ISO Standards 7810, 7811-1 through -6, 7812, 7813, and AAMVA. The Reader can be customized. Bidirectional read capability is available.

CONFIGURATIONS

	Part Number	Read	Color
Integral Electronics, Single Track without Cover	21045001 21045002	Track 1 Track 2	Black Black
Integral Electronics, Dual Tracks without Cover	21045011	Tracks 1 & 2	Black
Integral Electronics, 3 Tracks without Cover	21045019	Tracks 1,2, & 3	Black

SPECIFICATIONS

IEC: Meets or Exceeds Requirements for:	IEC 1000-4-2 ESD (Electro Static Discharge) IEC 1000-4-3 Radiated EMC Field (2X requirement) IEC 1000-4-4 Electrical Fast Transient Burst requirement (transmission on I/O cable)
Flammability	Meets UL94V-0
Recording Method	Two-Frequency Coherent Phase (F2F)
Speed	Card speed through the unit may vary from: 2-125 ips at 75 bpi 2-60 ips at 210 bpi
Power Requirements	Single Track: 2.4 to 5.5VDC at 1mA, typical Dual Track: 2.4 to 5.5VDC at 2mA, typical 3 Tracks: 2.4 to 5.5VDC at 3mA, typical
Output Signal Levels	V_{ol} = 0.4V at 2mA V_{oh} = V_{cc} -0.5V at -2mA
Operating Temperature	-30°C to 70°C
Operating Humidity	10% to 90% relative humidity
Life	300,000 passes Single Track 1,000,000 passes Multi-Track
Dimensions	Length: 3.54" (90.0mm) Height: 0.95" (24.13mm) Width: 0.88" (22.4mm)
Cable Length:	Single Track: 6" (150mm) Dual Track: 4" (101.6mm) 3 Track 5" (127mm)
Connector	See Section 2, Connectors
Colors available	Black, Standard

REFERENCE DOCUMENT

I/O Interface for TTL Swipe Readers, Technical Reference Manual, P/N 99875148

SECTION 2. INSTALLATION

This section consists of installation and checkout of the Reader.

MOUNTING

The dimensions for mounting without the cover are shown in Figure 2-1.

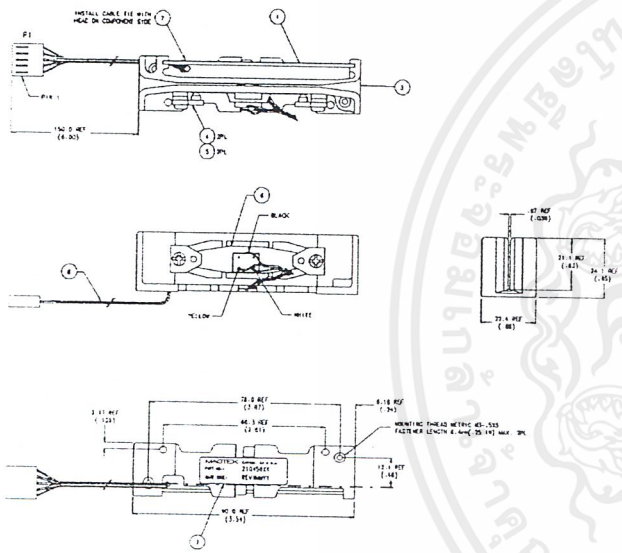


Figure 2-1. Reader Mounting Dimensions

CONNECTORS

Single Track I/O Connector is shown in Table 2-1, and the Dual Track I/O Connector is shown in Table 2-2.

Table 2-1. I/O Connector for Single Track, 5 Pin

	Pin Number	Color	Signal
Connector for Single Track, 5 Pin	1	Red	V_{cc}
Molex 5 Pin	2	Black	GND
22-01-2951	3	Orange	$\overline{\text{DATA}}$
0.100 inch Contact Spacing	4	Brown	$\overline{\text{STROBE}}$
	5	Green	$\overline{\text{CARD PRESENT}}$

Note: V_{cc} = 2.4 to 5.5 VDC

Table 2-2. I/O Connector for Dual Track, 7 Pin

	Pin Number	Color	Signal
Connector for Dual Track, 7 Pin	1	Red	V_{cc}
	2	Black	GND
	3	Orange	$\overline{\text{DATA (Tk 2)}}$
Molex 7 Pin	4	Brown	$\overline{\text{STROBE (Tk 2)}}$
22-01-2071	5	Green	$\overline{\text{CARD PRESENT}}$
0.100 inch Contact Spacing	6	Yellow	$\overline{\text{STROBE (Tk 1)}}$
	7	White	$\overline{\text{DATA (Tk 1)}}$

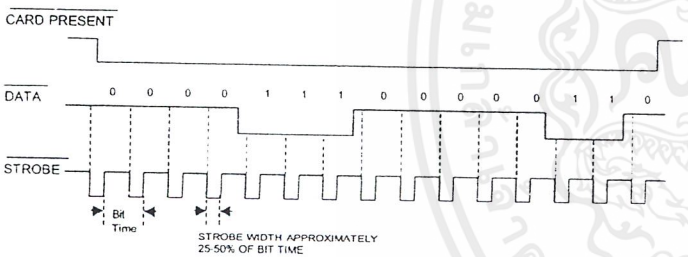
Note: V_{cc} = 2.4 to 5.5 VDC

Table 2-3. I/O Connector for 3 Track, 9 Pin

	Pin Number	Color	Signal
Connector for 3 Track, 9 Pin	1	Red	V_{cc}
	2	Black	GND
	3	Yellow	$\overline{\text{DATA (Tk 2)}}$
	4	Green	$\overline{\text{STROBE (Tk 2)}}$
	5	White	$\overline{\text{CARD PRESENT}}$
Molex 9 Pin	6	Blue	$\overline{\text{STROBE (Tk 1)}}$
22-01-2091	7	Brown	$\overline{\text{DATA (Tk 1)}}$
0.100 inch Contact Spacing	8	Gray	$\overline{\text{STROBE (Tk 3)}}$
	9	Orange	$\overline{\text{DATA (Tk 3)}}$

Note: V_{cc} = 2.4 to 5.5 VDC

TIMING



Notes:
1. Time out of the $\overline{\text{CARD PRESENT}}$ signal occurs approximately 150 ms after the last strobe transition.
2. $\overline{\text{DATA}}$ is valid 1.0μ sec before the negative edge of $\overline{\text{STROBE}}$.
3. 16 or 17 head flux reversals for high density configuration.

Figure 2-2. Timing

DATA

The Data signal is valid while the strobe is low. If the Data signal is high, the bit is a zero. If the Data signal is low, the bit is a one.

STROBE

The Strobe signal indicates when Data is valid. It is recommended that Data be loaded by the user with the leading edge (negative) of the Strobe.

CARD PRESENT

Card Present will go low after 14/15 flux reversals from the head. Card Present will return high 150 milliseconds after the last flux reversal.

When no card is being moved through the unit, the Data, Strobe, and Card Present signals are high. The signal timing diagram shown above represents the data along with other signals that are generated during the reading process.

MAGTEK

Application Note - Character Conversion

Track 1 Character Set

	Card Data							ASCII			
Character	P	B_6	B_5	B_4	B_3	B_2	B_1	P0	P1	EP	OP
Space	1	0	0	0	0	0	0	20	A0	A0	20
! (ND)	0	0	0	0	0	0	1	21	A1	21	A1
" (ND)	0	0	0	0	0	1	0	22	A2	22	A2
# (OG)	1	0	0	0	0	1	1	23	A3	A3	23
$	0	0	0	0	1	0	0	24	A4	24	A4
% (SS)	1	0	0	0	1	0	1	25	A5	A5	25
& (ND)	1	0	0	0	1	1	0	26	A6	A6	26
' (ND)	0	0	0	0	1	1	1	27	A7	27	A7
(	0	0	0	1	0	0	0	28	A8	28	A8
)	1	0	0	1	0	0	1	29	A9	A9	29
* (ND)	1	0	0	1	0	1	0	2A	AA	AA	2A
+ (ND)	0	0	0	1	0	1	1	2B	AB	2B	AB
, (ND)	1	0	0	1	1	0	0	2C	AC	AC	2C
-	0	0	0	1	1	0	1	2D	AD	2D	AD
.	0	0	0	1	1	1	0	2E	AE	2E	AE
/	1	0	0	1	1	1	1	2F	AF	AF	2F
0	0	0	1	0	0	0	0	30	B0	30	B0
1	1	0	1	0	0	0	1	31	B1	B1	31
2	1	0	1	0	0	1	0	32	B2	B2	32
3	0	0	1	0	0	1	1	33	B3	33	B3
4	1	0	1	0	1	0	0	34	B4	B4	34
5	0	0	1	0	1	0	1	35	B5	35	B5
6	0	0	1	0	1	1	0	36	B6	36	B6
7	1	0	1	0	1	1	1	37	B7	B7	37
8	1	0	1	1	0	0	0	38	B8	B8	38
9	0	0	1	1	0	0	1	39	B9	39	B9
: (ND)	0	0	1	1	0	1	0	3A	BA	3A	BA
; (ND)	1	0	1	1	0	1	1	3B	BB	BB	3B
< (ND)	0	0	1	1	1	0	0	3C	BC	3C	BC
= (ND)	1	0	1	1	1	0	1	3D	BD	BD	3D
> (ND)	1	0	1	1	1	1	0	3E	BE	BE	3E
? (ES)	0	0	1	1	1	1	1	3F	BF	3F	BF

	Card Data							ASCII			
Character	P	B_6	B_5	B_4	B_3	B_2	B_1	P0	P1	EP	OP
@ (ND)	0	1	0	0	0	0	0	40	C0	C0	40
A	1	1	0	0	0	0	1	41	C1	41	C1
B	1	1	0	0	0	1	0	42	C2	42	C2
C	0	1	0	0	0	1	1	43	C3	C3	43
D	1	1	0	0	1	0	0	44	C4	44	C4
E	0	1	0	0	1	0	1	45	C5	C5	45
F	0	1	0	0	1	1	0	46	C6	C6	46
G	1	1	0	0	1	1	1	47	C7	47	C7
H	1	1	0	1	0	0	0	48	C8	48	C8
I	0	1	0	1	0	0	1	49	C9	C9	49
J	0	1	0	1	0	1	0	4A	CA	CA	4A
K	1	1	0	1	0	1	1	4B	CB	4B	CB
L	0	1	0	1	1	0	0	4C	CC	CC	4C
M	1	1	0	1	1	0	1	4D	CD	4D	CD
N	1	1	0	1	1	1	0	4E	CE	4E	CE
O	0	1	0	1	1	1	1	4F	CF	CF	4F
P	1	1	1	0	0	0	0	50	D0	50	D0
Q	0	1	1	0	0	0	1	51	D1	D1	51
R	0	1	1	0	0	1	0	52	D2	D2	52
S	1	1	1	0	0	1	1	53	D3	53	D3
T	0	1	1	0	1	0	0	54	D4	D4	54
U	1	1	1	0	1	0	1	55	D5	55	D5
V	1	1	1	0	1	1	0	56	D6	56	D6
W	0	1	1	0	1	1	1	57	D7	D7	57
X	0	1	1	1	0	0	0	58	D8	D8	58
Y	1	1	1	1	0	0	1	59	D9	59	D9
Z	1	1	1	1	0	1	0	5A	DA	5A	DA
[(ND)	0	1	1	1	0	1	1	5B	DB	DB	5B
\ (ND)	1	1	1	1	1	0	0	5C	DC	5C	DC
] (ND)	1	1	1	1	1	0	0	5D	DD	DD	5D
^ (FS)	0	1	1	1	1	1	0	5E	DE	DE	5E
_ (ND)	1	1	1	1	1	1	1	5F	DF	5F	DF

rack 2 and 3 Character Set

	Card Data					ASCII			
haracter	P	B_4	B_3	B_2	B_1	P0	P1	EP	OP
0	1	0	0	0	0	30	B0	30	B0
1	0	0	0	0	1	31	B1	B1	31
2	0	0	0	1	0	32	B2	B2	32
3	1	0	0	1	1	33	B3	33	B3
4	0	0	1	0	0	34	B4	B4	34
5	1	0	1	0	1	35	B5	35	B5
6	1	0	1	1	0	36	B6	36	B6
7	0	0	1	1	1	37	B7	B7	37

	Card Data					ASCII				
Character	P	B_4	B_3	B_2	B_1	P0	P1	EP	OP	Hex Character
8	0	1	0	0	0	38	B8	B8	38	
9	1	1	0	0	1	39	B9	39	B9	
: (AS)	1	1	0	1	0	3A	BA	3A	BA	A
; (SS)	0	1	0	1	1	3B	BB	BB	3B	B
< (ND)	1	1	1	0	0	3C	BC	3C	BC	C
= (FS)	0	1	1	0	1	3D	BD	BD	3D	D
> (ND)	0	1	1	1	0	3E	BE	BE	3E	E
? (ES)	1	1	1	1	1	3F	BF	3F	BF	F

= Parity
P = Even Parity
P = Odd Parity
) = Parity bit set to 0
= Parity bit set to 1
S = Start Sentinel
S = End Sentinel

FS = Field Separator
AS = Account Separator (Track 3 only)
ND = Character Not Defined by Credit Card Standards
OG = Optional Graphic

MAGTEK
20725 South Annalee, Carson, CA. 90746
Phone: (310) 631-8602, Help Line (888)624-8350
Fax (310) 631-3956
http://www.magtek.com
P/N 99875065-2, 1/00 © Copyright 2000, Mag-Tek, Inc.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

GlobalOptoisolator™

VDE	UL	CSA	SETI	SEMKO	DEMKO	NEMKO	BABT

6-Pin DIP Optoisolators Logic Output

H11L1*
[IF(on) = 1.6 mA Max]
H11L2
[IF(on) = 10 mA Max]
*Motorola Preferred Device

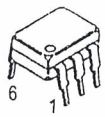
The H11L1 and H11L2 have a gallium arsenide IRED optically coupled to a high–speed integrated detector with Schmitt trigger output. Designed for applications requiring electrical isolation, fast response time, noise immunity and digital logic compatibility.

- Guaranteed Switching Times — t_{on}, t_{off} < 4 µs
- Built–In On/Off Threshold Hysteresis
- High Data Rate, 1 MHz Typical (NRZ)
- Wide Supply Voltage Capability
- Microprocessor Compatible Drive
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

Applications

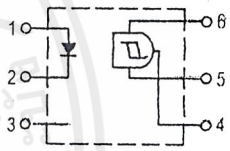
- Interfacing Computer Terminals to Peripheral Equipment
- Digital Control of Power Supplies
- Line Receiver — Eliminates Noise
- Digital Control of Motors and Other Servo Machine Applications
- Logic to Logic Isolator
- Logic Level Shifter — Couples TTL to CMOS

STYLE 5 PLASTIC



STANDARD THRU HOLE
CASE 730A–04

SCHEMATIC



PIN 1. ANODE
2. CATHODE
3. NC
4. OPEN COLLECTOR OUTPUT
5. GND
6. V_{CC}

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	6	Volts
Forward Current — Continuous — Peak Pulse Width = 300 µs, 2% Duty Cycle	I_F	60 1.2	mA Amp
LED Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	120 1.41	mW mW/°C
OUTPUT DETECTOR			
Output Voltage Range	V_o	0–16	Volts
Supply Voltage Range	V_{CC}	3–16	Volts
Output Current	I_O	50	mA
Detector Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	150 1.76	mW mW/°C
TOTAL DEVICE			
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.94	mW mW/°C
Maximum Operating Temperature(2)	T_A	–40 to +85	°C
Storage Temperature Range(2)	T_{stg}	–55 to +150	°C
Soldering Temperature (10 s)	T_L	260	°C
Isolation Surge Voltage (Pk ac Voltage, 60 Hz, 1 Second Duration)(1)	V_{ISO}	7500	Vac(pk)

1. Isolation surge voltage is an internal device dielectric breakdown rating.
   For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.
2. Refer to Quality and Reliability Section in Opto Data Book for information on test conditions.

Preferred devices are Motorola recommended choices for future use and best overall value.
GlobalOptoisolator is a trademark of Motorola, Inc.

REV 1

© Motorola, Inc. 1995

MOTOROLA

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)(1)

Characteristic		Symbol	Min	Typ(1)	Max	Unit
INPUT LED						
Reverse Leakage Current (V_R = 3 V, R_L = 1 MΩ)		I_R	—	0.05	10	μA
Forward Voltage (I_F = 10 mA) (I_F = 0.3 mA)		V_F	— 0.75	1.2 0.95	1.5 —	Volts
Capacitance (V_R = 0 V, f = 1 MHz)		C	—	18	—	pF
OUTPUT DETECTOR						
Operating Voltage		V_{CC}	3	—	15	Volts
Supply Current (I_F = 0, V_{CC} = 5 V)		$I_{CC(off)}$	—	1	5	mA
Output Current, High (I_F = 0, V_{CC} = V_O = 15 V)		I_{OH}	—	—	100	μA
COUPLED						
Supply Current (I_F = $I_{F(on)}$, V_{CC} = 5 V)		$I_{CC(on)}$	—	1.6	5	mA
Output Voltage, Low (R_L = 270 Ω, V_{CC} = 5 V, I_F = $I_{F(on)}$)		V_{OL}	—	0.2	0.4	Volts
Threshold Current, ON (R_L = 270 Ω, V_{CC} = 5 V)	H11L1 H11L2	$I_{F(on)}$	— —	1.2 —	1.6 10	mA
Threshold Current, OFF (R_L = 270 Ω, V_{CC} = 5 V)	H11L1 H11L2	$I_{F(off)}$	0.3 0.3	0.75 —	— —	mA
Hysteresis Ratio (R_L = 270 Ω, V_{CC} = 5 V)		$\frac{I_{F(off)}}{I_{F(on)}}$	0.5	0.75	0.9	
Isolation Voltage(2) 60 Hz, AC Peak, 1 second, T_A = 25°C		V_{ISO}	7500	—	—	Vac(pk)
Turn–On Time	R_L = 270 Ω(3) V_{CC} = 5 V, I_F = $I_{F(on)}$ T_A = 25°C	t_{on}	—	1.2	4	μs
Fall Time		t_f	—	0.1	—	
Turn–Off Time		t_{off}	—	1.2	4	
Rise Time		t_r	—	0.1	—	

1. Always design to the specified minimum/maximum electrical limits (where applicable).
2. For this test, IRED Pins 1 and 2 are common and Output Gate Pins 4, 5, 6 are common.
3. R_L value effect on switching time is negligible.

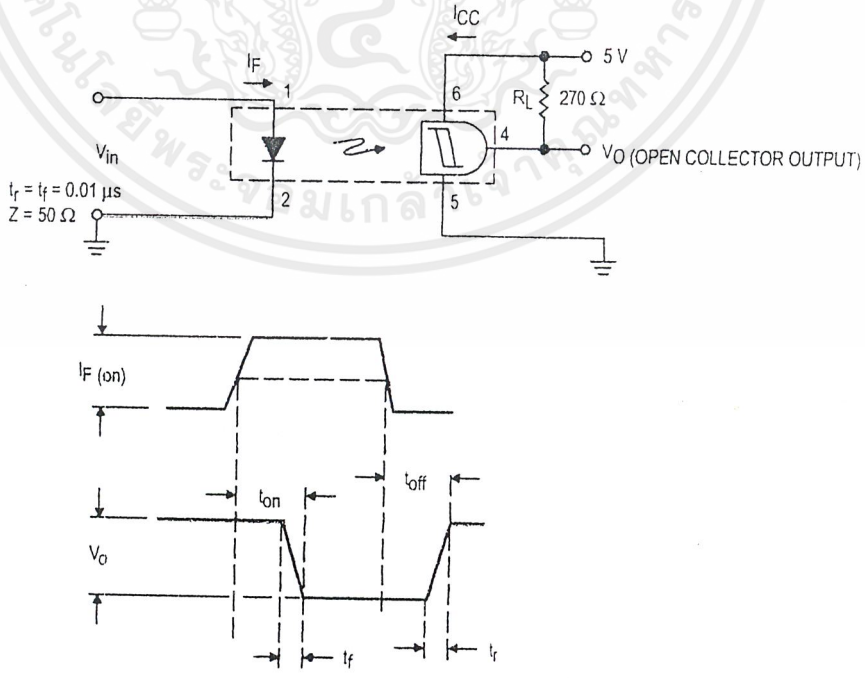


Figure 1. Switching Test Circuit

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ในเชิงพาณิชย์ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TYPICAL CHARACTERISTICS

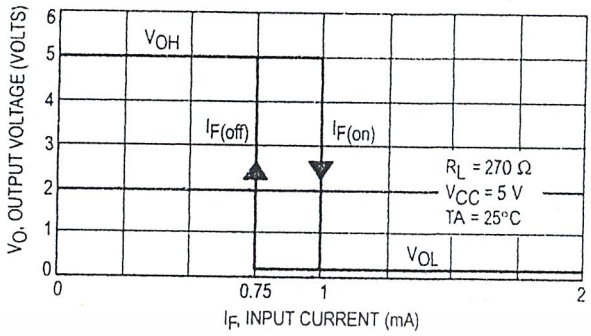


Figure 2. Transfer Characteristics for H11L1

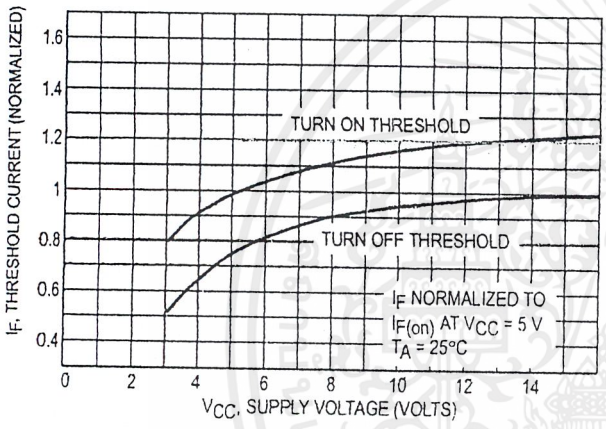


Figure 3. Threshold Current versus Supply Voltage

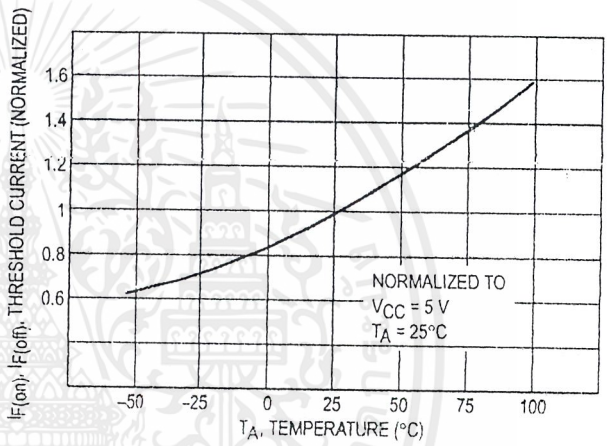


Figure 4. Threshold Current versus Temperature

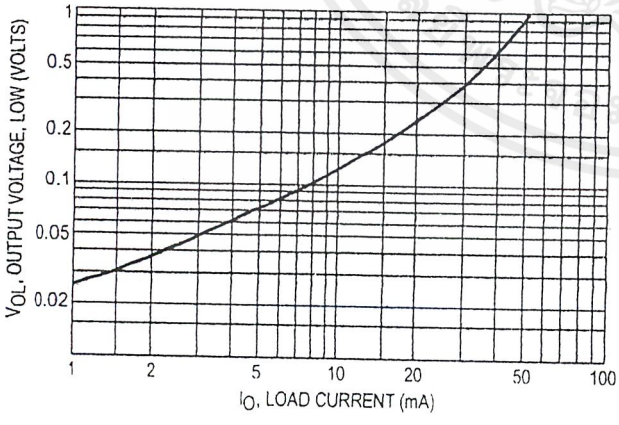


Figure 5. Output Voltage, Low versus Load Current

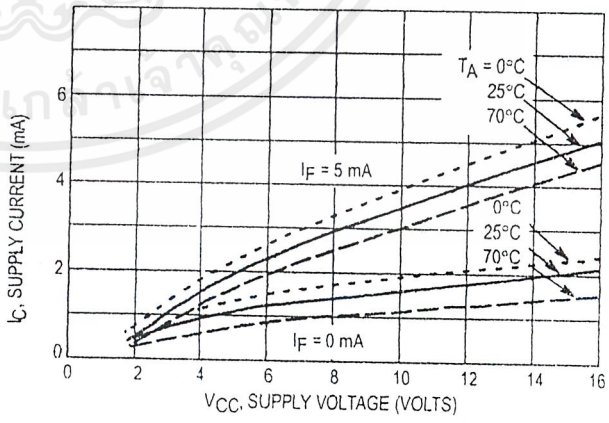
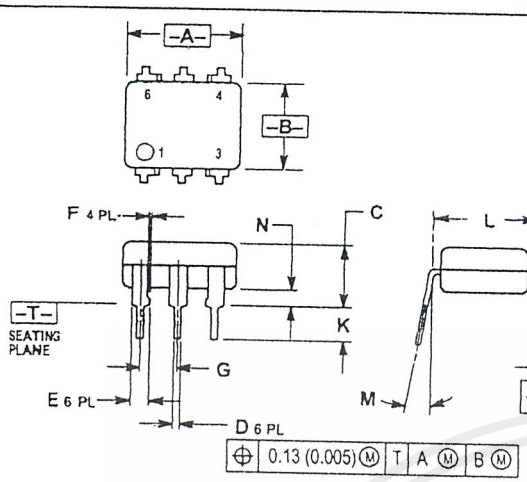


Figure 6. Supply Current versus Supply Voltage

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PACKAGE DIMENSIONS

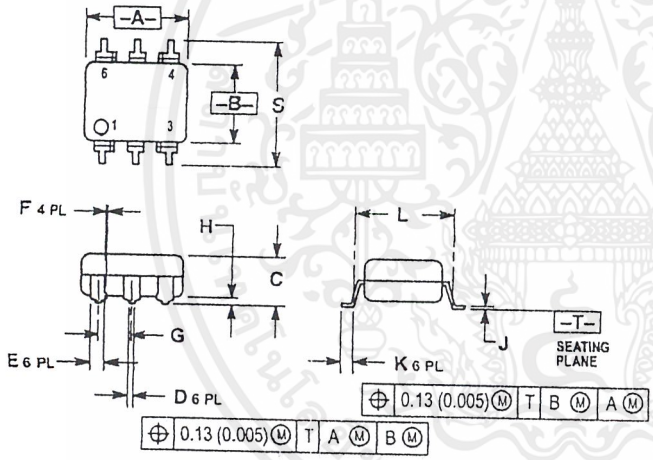


STYLE 5:
PIN 1. ANODE
2. CATHODE
3. NC
4. OUTPUT
5. GROUND
6. V_{CC}

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
J	0.008	0.012	0.21	0.30
K	0.100	0.150	2.54	3.81
L	0.300 BSC		7.62 BSC	
M	0°	15°	0°	15°
N	0.015	0.100	0.38	2.54

CASE 730A–04
ISSUE G



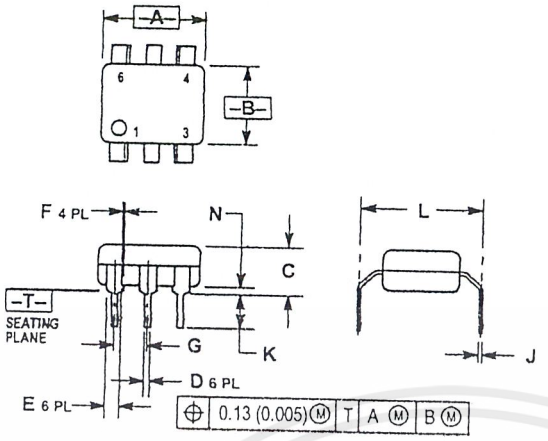
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
H	0.020	0.025	0.51	0.63
J	0.008	0.012	0.20	0.30
K	0.006	0.035	0.16	0.88
L	0.320 BSC		8.13 BSC	
S	0.332	0.390	8.43	9.90

*Consult factory for leadform option availability

CASE 730C–04
ISSUE D

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.320	0.350	8.13	8.89
B	0.240	0.260	6.10	6.60
C	0.115	0.200	2.93	5.08
D	0.016	0.020	0.41	0.50
E	0.040	0.070	1.02	1.77
F	0.010	0.014	0.25	0.36
G	0.100 BSC		2.54 BSC	
J	0.008	0.012	0.21	0.30
K	0.100	0.150	2.54	3.81
L	0.400	0.425	10.16	10.80
N	0.015	0.040	0.38	1.02

*Consult factory for leadform option availability

CASE 730D–05
ISSUE D

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

1. กฤษดา ใจเย็น , ชัยวัฒน์ ลิ้มพรจิตรวิไล , " เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม " , อินโนเวตีฟ เอ็กเพอริเมนต์
2. ชัยวัฒน์ ลิ้มพรจิตรวิไล , วรพจน์ กรแก้ววัฒนกุล , " เรียนรู้และปฏิบัติการไมโคร-คอนโทรลเลอร์ " , อินโนเวตีฟ เอ็กเพอริเมนต์
3. ธาริน สิทธิธรรมชารี , สุรสิทธิ์ คิวประสพศักดิ์ , " คู่มือการเขียนโปรแกรม Microsoft Visual Basic Version 6.0 ฉบับเพื่อการประยุกต์ใช้งาน " , ซัคเซส มีเดีย
4. ธาริน สิทธิธรรมชารี , " คู่มือการเขียนโปรแกรม Microsoft Visual Basic Version 6.0 ฉบับเพื่อการใช้งานจริง " , ซัคเซส มีเดีย
5. ธีรวัฒน์ ประกอบผล , " การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ " , แซทโฟร์ พริ้นติ้ง , 2541
6. ศุภชัย สมพานิช , " เรียนรู้และฝึกฝนการเขียนโปรแกรมใช้งานฐานข้อมูลด้วย Visual Basic " , อินโฟเพรส , 2543
7. วันสุระ ศรีใสดี , " ประยุกต์/อินเตอร์เฟส ไมโครคอนโทรลเลอร์ ภาคปฏิบัติการ " , สำนักพิมพ์ดวงกมล , 2542
8. ARC Electronics , " RS 232 DATA INTERFACE " , www.arcelect.com/rs232.htm
9. C.Denis Mee , Eric D. Daniel , " Magnetic recording "
10. C.Denis Mee , Eric D. Daniel," Magnetic recording handbook : technology and applications "
11. MAGTEK , " I/O INTERFACE FOR TTL SWIPE READERS TECHNICAL REFERENCE MANUAL " , MAGTEK , 1999

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้