ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

Access Control System Using Fingerprint

นายภาณุ ลภาพงษ์
นายยอดจิตต เยี่ยมยกกุล
นายสมิต โชควัฒนากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

Access Control System Using Fingerprint

นายภาณุ ลภาพงษ์
นายยอดจิตต เยี่ยมยกกุล
นายสมิต โชควัฒนากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ**
Access Control System Using Fingerprint

ผู้จัดทำ **นายภาณุ ฉภาพงษ์**
นายยอดจิตต เยี่ยมยกกุล
นายฉมิต โชควัฒนากุล

.....**อาจารย์ที่ปรึกษา**
(ผู้ช่วยศาสตราจารย์ เกียรติวรรณ ทรงสัตย์)

ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

Access Control System Using Fingerprint

By

นายภาณุ ฉภาพงษ์

MR.PANU LAPAPONG

นายยอดจิตต เยี่ยมยกกุล

MR.YODJIT YIAMYOKKUN

นายสมิต โชควัฒนากุล

MR.SMIT CHOAKWATTANAKUL

Advisor

ผู้ช่วยศาสตราจารย์ เกียรติวรรณ ทรงสัตย์

Asst. Prof. KIETTIWAN SONGSATAYA

Academic Year 2006

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการพัฒนาระบบควบคุมการเข้าออกด้วยลายนิ้วมือ โดยการประยุกต์ไมโครคอนโทรลเลอร์ตระกูล พีไอซี เป็นหน่วยประมวลผลกลางระบบที่ออกแบบขึ้นมานี้จะประกอบด้วยหน่วยการทำงานย่อยๆ 2 หน่วย ได้แก่ เซิร์ฟเวอร์โมดูล และ เทอร์มินอลโมดูล เซิร์ฟเวอร์โมดูลกับเทอร์มินอลโมดูล จะเชื่อมต่อระหว่างกันด้วยโครงข่ายสื่อสารมาตรฐานอาร์เอส 485 ซึ่งมันสามารถขยายขีดความสามารถในการติดต่อเซิร์ฟเวอร์ 1 ตัวต่อกับเทอร์มินอลได้สูงสุด 32 ตัว

ABSTRACT

This thesis presents the development of the Access Control System Using Fingerprint. The application of microcontroller PIC as the central processor each components is constructed. The designed system composes of two units, namely, sever module and terminal module. Sever module and terminal module are connected together by RS-485 network that can extend the ability to maximum reach up to 32 terminals

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์เกียรติวรรณ ทรงสัตย์ เป็นอย่างสูงที่ชี้แนะและเป็นที่ปรึกษาให้กับโครงงาน รวมทั้งความเอื้อเฟื้อ อุปกรณ์การทำงานเป็นอย่างดี

ขอขอบคุณท่านอาจารย์ภาควิชาวิศวกรรมระบบควบคุมทุกๆท่าน ที่ประสิทธิประสาทวิชาความรู้ และคอยแนะนำแนวทางการแก้ปัญหาต่างๆ รวมทั้งความรักความเมตตาที่ท่านมอบให้

ขอขอบคุณ คุณพ่อ คุณแม่ผู้ให้ความดูแลเลี้ยงดู อบรมสั่งสอนตลอดมา

ขอขอบคุณ พี่ๆเพื่อนๆและน้องๆทุกคนที่คอยเป็นกำลังใจ ให้ความช่วยเหลือให้ยืมอุปกรณ์และคำแนะนำที่ดีๆมีประโยชน์ยิ่ง

และสุดท้าย ขอขอบคุณ คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่เป็นสถานศึกษา ที่เป็นแหล่งสร้างทักษะความรู้และประสบการณ์ต่างๆที่สำคัญต่อผู้จัดทำ

คณะผู้จัดทำ

นายภาณุ ลภาพงษ์

นายยอดจิตต เยี่ยมยกกุล

นายสมิต โชควัฒนากุล

สารบัญ

สารบัญภาพ

รูปที่ **หน้าที่**

สารบัญตาราง

บทที่ 1
บทนำ

1.1 กล่าวนำ

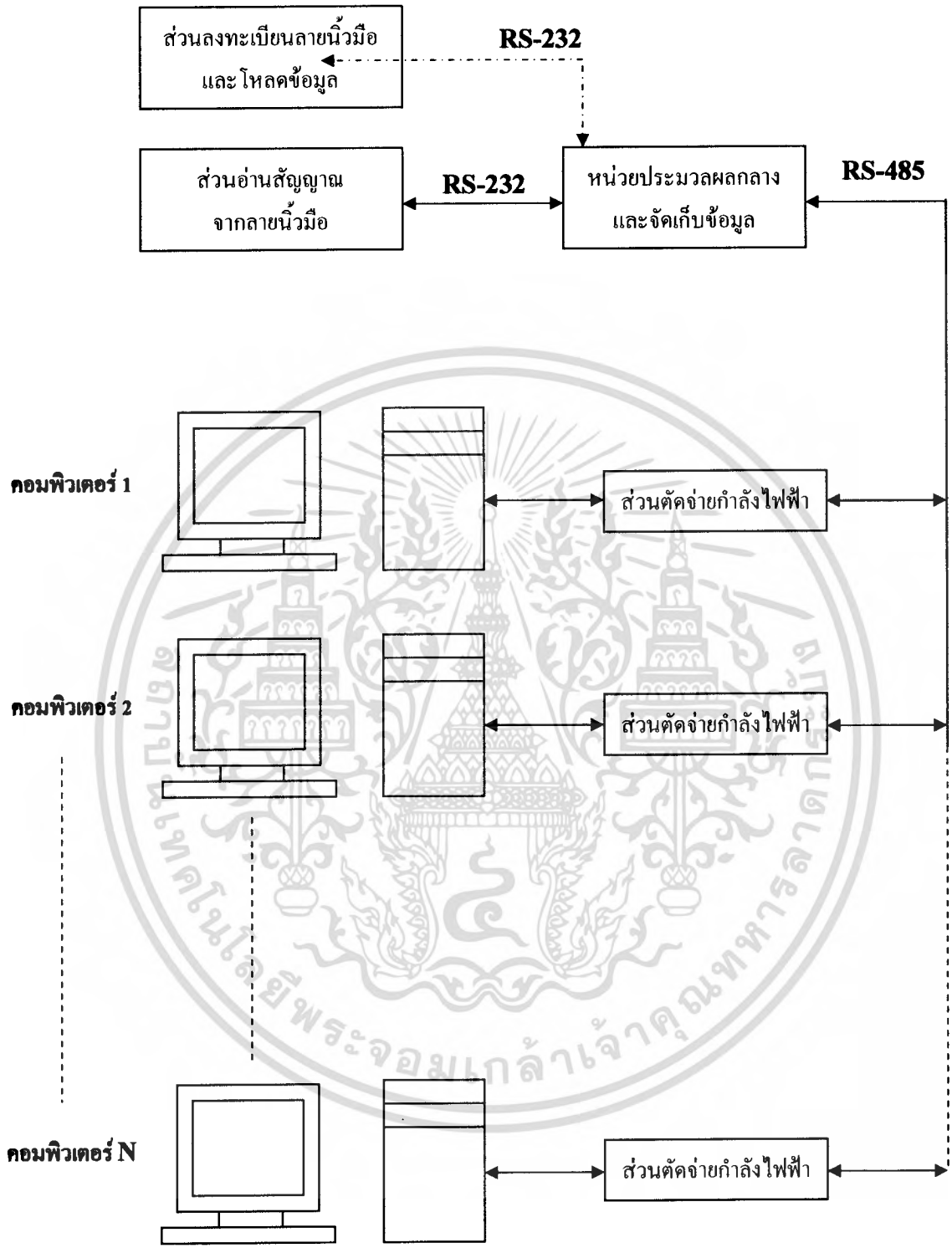
ปัจจุบันคอมพิวเตอร์มีบทบาทสำคัญอย่างมากในชีวิตประจันไม่เพียงแต่การใช้งานเครื่องเดียวตามลำพังแต่ยังได้มีการเชื่อมต่อโครงข่ายของคอมพิวเตอร์ขึ้นเพื่อเพิ่มประสิทธิภาพในการใช้งานดังนั้นจึงมีความจำเป็นอย่างยิ่งที่จะต้องมีระบบที่สามารถควบคุมตรวจสอบประมวลผลและรักษาความปลอดภัยของข้อมูลบนคอมพิวเตอร์ หรือการอนุญาตเฉพาะผู้ที่มีสิทธิ์ในการใช้เครื่องคอมพิวเตอร์นั้นๆเท่านั้น ซึ่งจะทำให้เกิดความสะดวกสบายและปลอดภัยมากยิ่งขึ้น ในกรณีนี้ระบบควบคุมการเข้าออกด้วยลายนิ้วมือจึงได้ถูกนำมาใช้และพัฒนาให้สามารถควบคุมการใช้งานห้องคอมพิวเตอร์และเครื่องคอมพิวเตอร์ได้

โดยในโครงงานนี้ได้เลือกนำเสนอระบบควบคุมการเข้าออกด้วยลายนิ้วมือ ซึ่งเป็นระบบที่ออกแบบไว้โดยใช้ไมโครคอนโทรลเลอร์ในการควบคุมผ่านมาตรฐานการติดต่อแบบอนุกรม RS-232 และ RS-485 โดยได้ใช้ข้อมูลจากตัวบุคคล (Human Data) ซึ่งได้จากการแสกนลายนิ้วมือนั่นเอง

โดยในการออกแบบระบบนั้นจะประกอบไปด้วยองค์ประกอบ 3ส่วนด้วยกัน ได้แก่

1. ส่วนตรวจสอบ และแปลงสัญญาณจากลายนิ้วมือ
2. ส่วนควบคุมเครื่องคอมพิวเตอร์หรือหน่วยจ่ายต่อกำลังไฟฟ้า
3. ส่วนประมวลผล และจัดเก็บข้อมูล

ซึ่งในการทำงานของระบบนั้น องค์ประกอบทั้ง 4 ส่วนจะถูกเชื่อมต่อถึงกัน โดยใช้มาตรฐานการติดต่อแบบอนุกรม RS-232 และ RS-485



รูปที่ 1.1 แผนผังระบบควบคุมการเข้าออกและปิด-เปิดเครื่องคอมพิวเตอร์

1.2 วัตถุประสงค์ในการออกแบบสร้างและพัฒนาระบบ

1.2.1 วัตถุประสงค์ในการออกแบบสร้างระบบควบคุมแสกนลายนิ้วมือได้แก่

- เพื่อออกแบบระบบตัวควบคุมการเข้าออกโดยใช้ไมโครคอนโทรลเลอร์ ตระกูล PIC เป็นหน่วยประมวลผล

- เพื่อใช้ติดต่อสื่อสารแบบอนุกรมผ่านครงข่าย RS-232 และ RS-485

- เพื่อนำเครื่องแสกนลายนิ้วมือมาประยุกต์ใช้ในการเก็บข้อมูลและอ่านข้อมูล

1.2.2 วัตถุประสงค์ในการพัฒนาระบบได้แก่

- เพื่อพัฒนาหรือแก้ไขโปรแกรมการติดต่อสื่อสารข้อมูล การข้อใช้บริการและการควบคุมการเข้าออก ระหว่างเซิร์ฟเวอร์ (Sever) และเทอร์มินอล (Terminal) หรือ เซิร์ฟเวอร์กับคอมพิวเตอร์ ให้มีประสิทธิภาพ

- เพื่อการออกแบบตัวเซิร์ฟเวอร์ (Sever) และ เทอร์มินอล (Terminal) ให้มีประสิทธิภาพการใช้งานตามวัตถุประสงค์ได้อย่างเหมาะสม

1.3 หลักการที่ใช้ในการออกแบบระบบ

หลักการที่ใช้ในการออกแบบระบบควบคุมการเข้าออกโดยใช้เครื่องแสกนลายนิ้วมือคือ

1.3.1 การอ่านข้อมูลจากเครื่องแสกนลายนิ้วมือ

1.3.2 การทำงานร่วมกันของไมโครคอนโทรลเลอร์ PIC จำนวน 2 ตัว

I.3.3 การใช้โครงข่าย RS-232 และ RS-485

1.3.4 การใช้ภาษาซี ในการโปรแกรมการทำงาน

1.4 รายละเอียดของรายงานนี้แบ่งส่วนที่ดำเนินการออกเป็น

บทที่ 1 เป็นบทกล่าวนำ บอกวัตถุประสงค์ และหลักการออกแบบ

บทที่ 2 กล่าวถึงทฤษฎีและหลักการต่างๆ ได้แก่รูปแบบรวมทั้งหลักการพื้นฐานของการติดต่อสื่อสาร การควบคุมส่วนอ่านลานิ้มมือ การติดต่อสื่อสารเบื้องต้นและโครงสร้างการติดต่อสื่อสารแบบอนุกรม

บทที่ 3 กล่าวถึงวิธีการสร้าง และออกแบบรวมทั้งโครงสร้างของเซิร์ฟเวอร์และเทอร์มินอล

บทที่ 4 เป็นการทดลองการทำงานของระบบในส่วนต่างๆ และผลการทดลอง

บทที่ 5 บทสรุป เป็นการสรุป และวิจารณ์ผลจากระบบที่ออกแบบ รวมทั้งแนวทางแก้ไขพัฒนาต่อไป

บทที่ 2
หลักการและทฤษฎี

2.1 การอ่านลายนิ้วมือ

ลายนิ้วมือของมนุษย์แต่ละคนนั้น เริ่มปรากฏขึ้นตั้งแต่เป็นตัวอ่อนอายุ 3 ถึง 4 เดือนในครรภ์มารดา ลายนิ้วมือเป็นผิวหนังส่วนที่มีร่อง (Furrow) และมีสัน (Ridge) ไว้ใช้สำหรับอำนวยความสะดวกในการหยิบจับวัตถุสิ่งของต่างๆ ร่อง (Furrow) และสัน (Ridge) ของลานิ้วมือมีลักษณะที่สำคัญ 2 ประการ คือ ไม่มีการเปลี่ยนแปลงรูปแบบตามกาลเวลา (แต่อาจเปลี่ยนขนาดได้) และการมีรูปแบบเฉพาะในแต่ละคน

ลายนิ้วมือของมนุษย์นั้น ไม่มีการเปลี่ยนแปลงรูปแบบตามกาลเวลา (Permanence) กล่าวคือ นับตั้งแต่วันแรกที่มนุษย์เกิดจวบจนกระทั่งวันที่มนุษย์ตาย ร่องรอยและสันของลายนิ้วมือจะปรากฏเช่นเดิมไม่สูญหาย แต่ขนาดของร่องรอยและสันนั้นอาจเปลี่ยนแปลงได้ตามขนาดของร่างกายซึ่งเปรียบเหมือนกับการที่เราวาดรูปภาพไว้บนลูกโป่ง ซึ่งไม่ว่าลูกโป่งจะเล็ก หรือถูกเป่าให้พองใหญ่อย่างไร รูปภาพที่วาดไว้บนลูกโป่งนั้นยังคงเป็นรูปภาพเดิม เพียงแต่รูปภาพนั้นจะมีขนาดที่ใหญ่ขึ้นเท่านั้นเอง

ลายนิ้วมือมีรูปแบบเฉพาะในแต่ละคน(Individuality) การเปรียบเทียบลายนิ้วมือมนุษย์มานานกว่าร้อยปี ผลการศึกษาพบว่า ไม่มีมนุษย์คนใดในโลกที่มีลายนิ้วมือเหมือนกันซึ่งสอดคล้องกับผลการศึกษาวิจัยเรื่องการใช้ลายนิ้วมือระบุตัวบุคคลของ Sir Francis Galton (1982) โดยใช้การแบ่งรายละเอียดรูปแบบของลายนิ้วมือออกเป็นส่วนๆ และหาความน่าจะเป็นของการซ้ำกันของลายนิ้วมือแต่ละส่วน แล้วนำความน่าจะเป็นของแต่ละส่วนมาคูณกันเพื่อหาความน่าจะเป็นทั้งหมด พบว่า ลายนิ้วมือของแต่ละบุคคลนั้นจะมีลักษณะเฉพาะ (Individuality) และไม่มีการเปลี่ยนแปลงรูปแบบ (Permanence) โอกาสที่บุคคลสองคนจะมีลายนิ้วมือเหมือนกันมีความน่าจะเป็นอยู่ที่ 1/64,000,000,000

ลายนิ้วมือของคนแต่ละคนนั้นมีลักษณะเฉพาะแม้แต่ คู่แฝดแท้ (Identical Twin) ก็ยังมีลายนิ้วมือที่แตกต่างกัน (แต่มีรูปแบบทางพันธุกรรมเหมือนกัน) รูปแบบของลายนิ้วมือ สามารถแบ่งออกได้เป็น 3 ประเภท ได้แก่ ลายนิ้วมือก้นหอย (Whorl) ลายนิ้วมือมัดหวาย (Loop) และลายนิ้วมือโค้ง (Arch)

รูปที่ 2.1 ลักษณะของลายนิ้วมือแบบต่างๆ

รูปแสดงลายนิ้วมือสามารถแบ่งย่อยให้ละเอียดขึ้นไปได้เป็นลายนิ้วมือแบบมัดหวายเอียงขวา(Right Loop) ลายนิ้วมือแบบมัดหวายเอียงซ้าย (Left Loop) ลายนิ้วมือโค้งสูงแบบกระโจม (Tented Arch) เป็นต้น

นอกจากนี้ยังได้มีการแบ่งสัดส่วนของลายนิ้วมือแบบต่างๆ ที่ปรากฏเห็นในมนุษย์แต่ละคนพบว่า ลายนิ้วมือแบบมัดหวาย(Loop) จะพบมากที่สุดคิดเป็นร้อยละ 65 รองลงมา ได้แก่ ลายนิ้วมือแบบก้นหอย (Whorl) คิดเป็นร้อยละ 30 และพบลายนิ้วลายนิ้วมือแบบโค้ง (Arch) น้อยที่สุดคิดเป็นร้อยละ 5

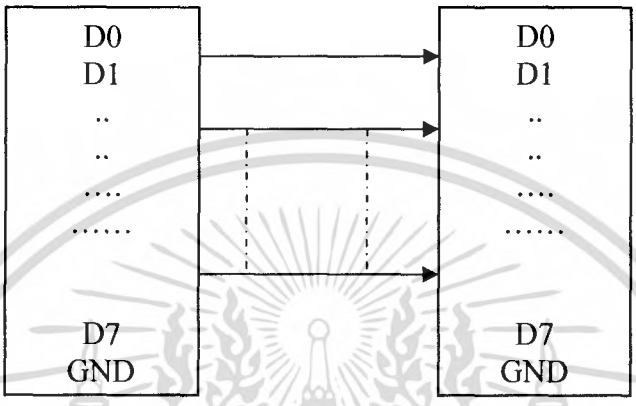
การแบ่งลายนิ้วมือออกเป็นหลายประเภทนี้เพื่อช่วยเพิ่มความรวดเร็วในการตรวจสอบลายนิ้วมือ แต่ไม่ได้เป็นสิ่งที่ใช้บอกความเหมือน หรือบอกความแตกต่างระหว่างลายนิ้วมือ ส่วนการศึกษาเปรียบเทียบลายนิ้วมือเพื่อหาความแตกต่างของแต่ละบุคคลนั้นจะพิจารณาจากสัน(Ridge) ของลายนิ้วมืออาทิ การสิ้นสุดของสัน (Ridge Ending), สันแบบลายจุด (Dot), สันที่แตกแขนง (Bifurcations) และรูปแบบต่างๆของสันที่เกิดขึ้น

ในการศึกษาเรื่องการควบคุมการเข้าออกด้วยลายนิ้วมือ ส่วนตรวจสอบ และแปลงสัญญาณจากลายนิ้วมือนับว่ามีความสำคัญมากเพราะ เป็นส่วนที่ใช้ในการรักษาความปลอดภัย เนื่องจากสัญญาณส่วนที่ได้จากส่วนนี้เป็นสัญญาณเฉพาะของแต่ละบุคคลไม่สามารถลอกเลียนแบบกันได้ ผู้ศึกษาได้ใช้อุปกรณ์คือ โมดูลแสกนลายนิ้วมือรุ่น MRB200 ผู้ศึกษายังได้ทำการเชื่อมต่อกับหน่วยประมวลผลกลางโดยผ่านมาตรฐาน RS-232 หรือ RS-485

2.2 พื้นฐานการรับส่งข้อมูล

การรับส่งข้อมูลในระบบคอมพิวเตอร์ทั่วไปจะหมายถึง การรับส่งข้อมูลเป็นจำนวนไบต์ๆให้กับอุปกรณ์ที่เกี่ยวข้องกับเครื่องคอมพิวเตอร์ ซึ่งอาจแบ่งประเภทของการรับส่งข้อมูลได้ 2 แบบ

แบบที่หนึ่ง การรับส่งข้อมูลแบบขนาน(Parallel) การรับส่งข้อมูลแบบขนานเป็นการรับส่งข้อมูลจำนวน 1 ไบต์ ออกทางพอร์ท ในเวลาเดียวกันระบบคอมพิวเตอร์ 1 ไบต์ จะมีจำนวน 8บิต คือ D0-D7 ถ้ามีการส่งข้อมูลแบบขนานจะใช้สายสัญญาณอย่างน้อย 9 เส้น คือสาย Data 8 เส้น และสาย Ground 1เส้น ดังรูป



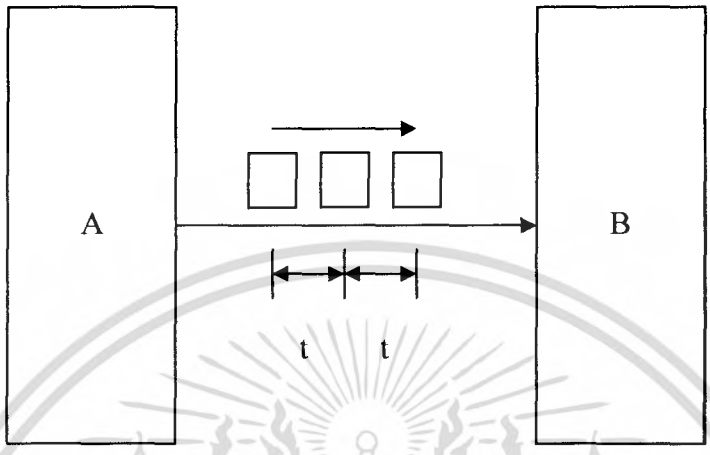
รูปที่ 2.2 การรับส่งข้อมูลแบบขนาน

แบบที่สอง การรับส่งข้อมูลแบบอนุกรม(Serial) คือ การรับส่งข้อมูลแต่ละบิต จนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์ คือ D0-D7 อาจต้องส่งบิต D0 ออกไปก่อนแล้วตามด้วย D1 ไปเรื่อยๆ จนถึง D7

การรับส่งทั้งสองแบบมีข้อดีข้อเสียแตกต่างกันคือ การรับส่งข้อมูลแบบขนานสามารถส่งข้อมูลได้เร็ว ส่งข้อมูลครั้งเดียวจะได้ข้อมูลครบ 1 ไบต์ แต่มีข้อจำกัดที่หากมีข้อมูลไปในที่ๆมีระยะทางไกลๆ จะทำให้สิ้นเปลืองสายสัญญาณมาก ขณะที่การส่งข้อมูลแบบอนุกรมสามารถส่งข้อมูลไปในระยะทางไกลๆ ได้ดีกว่าช่วยประหยัดสายสัญญาณ เนื่องจากการส่งข้อมูลแบบอนุกรมนั้นจะใช้สายอย่างน้อยเพียง 2 เส้น ได้แก่ สายสัญญาณ กับสายกราวด์ แต่มีข้อจำกัดที่ใช้เวลานานในการรับส่งข้อมูลเนื่องจาก เป็นการส่งทีละ บิต ซึ่งการศึกษาในส่วนนี้จะ กล่าวถึงพื้นฐานการรับส่งข้อมูลแบบอนุกรม โดยจะเน้นที่ตัว PIC เป็นสำคัญ

2.2.1 การรับส่งข้อมูลแบบซิงโครนัส (Synchronous Input/Output) การรับส่งข้อมูลแบบนี้ไม่ว่าจะเป็นการส่งแบบอนุกรมหรือแบบขนาน ข้อมูลแต่ละไบต์ที่ถูกส่งออกไปจะมีช่วงเวลาห่าง

กันแน่นอน เช่น การส่งข้อมูลจาก A ไป B ดังรูปที่2.3 Data 1จะห่างจาก Data2 เป็นเวลา t และData 3 จะห่างจาก Data 2 เป็นเวลา t เช่นกัน ระบบแบบนี้เหมาะกับงานที่ไม่มีความยุ่งยากมาก



รูปที่ 2.3 การส่งข้อมูลแบบซิงโครนัส

2.2.2 การรับส่งข้อมูลแบบอะซิงโครนัส (Asynchronous Input / Output) การรับส่งข้อมูลแบบนี้ ข้อมูลที่ถูกส่งออกไปจะไม่มีเวลาที่แน่นอน ซึ่งจะอยู่กับความพร้อมของผู้ส่ง และผู้รับโดยจะมีสายสัญญาณตรวจสอบความพร้อมของระบบทั้งสองว่าพร้อมที่จะติดต่อกันหรือไม่ โดยสัญญาณที่เพิ่มขึ้นมาจากระบบแบบซิงโครนัส เรียกว่าสายสเตตัส (Status Line)

2.3 รูปแบบของการรับส่งข้อมูลแบบอนุกรม

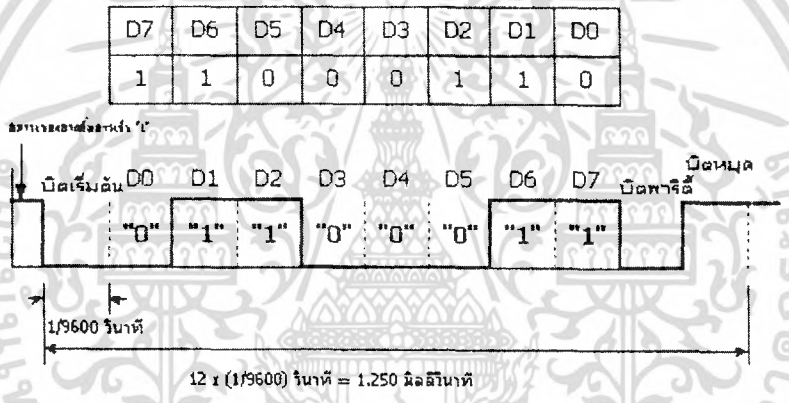
การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัสเป็นการรับส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตของข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริงเพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นซึ่งประกอบด้วยกัน 4 ส่วน คือ

ส่วนที่หนึ่ง บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นระดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสาร ขณะที่ยังไม่มีการส่งข้อมูล

ส่วนที่สอง บิตข้อมูล(Data bit) จะเริ่มจากบิตที่มีนัยสำคัญต่ำสุดก่อนหรือบิต LSB ก่อน โดยข้อมูลจะส่งอาจจะมีขนาด 5 บิต 6 บิต 7 บิต หรือ 8 บิตก็ได้

ส่วนที่สาม บิตแสดงสภาวะเลขคู่หรือเลขคี่(Parity bit) มีขนาด 1 บิต โดยบิตนี้จะนำไปต่อท้ายกับบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็นเลนหนึ่ง โดยการเลือกการส่งข้อมูลเป็นแบบ พาริตี้คู่ หรือพาริตี้คี่ ตัวอย่างถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลขหนึ่งเป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็นหนึ่งเพื่อจะได้จำนวนเลขหนึ่งเป็นคู่นั้นเอง ทำนองเดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็นหนึ่งรวมทั้งบิตพาริตี้ หนึ่งบิต ถ้ามีค่าหนึ่งเป็นจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง ซึ่งสามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้

ส่วนที่สี่ บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสถานะลอจิกเป็นหนึ่ง ซึ่งอาจมีจำนวน มากกว่า หนึ่งบิตก็ได้เช่น 1 บิต 1.5 บิต หรือ 2 บิต



รูปที่ 2.4 แสดงการส่งข้อมูลขนาด 8 บิตแบบอนุกรมพร้อมด้วยบิตเริ่มต้น บิตพาริตี้ บิตหยุด ด้วยความเร็ว 9600 บิตต่อวินาที

STOP	P	D7	D6	D5	D4	D3	D2	D1	D0	START

รูปที่ 2.5 บิตต่างๆของข้อมูลที่ส่งแบบอนุกรม

ถ้ามีการส่งข้อมูลแบบ 8 บิต จะต้องส่งบิต แรกออกไปก่อนเรียกว่า บิตเริ่มต้น (Start Bit) ถ้ามีการส่งข้อมูลหลายๆไบต์ออกมาบิตนี้จะเป็นตัวบอกว่า มีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้น

มักมีระดับลอจิกเป็นศูนย์ต่อจากบิตเริ่มต้นจะเป็นบิตข้อมูล D0 ถึง D7 จากนั้นจะตามด้วยบิตตรวจสอบความถูกต้อง (Parity Bit) ถ้าข้อมูล 8 บิตที่ส่งออกมาจำนวนบิตที่มีค่าเป็นหนึ่ง เป็นจำนวนคู่ บิตนี้จะมีค่าเป็นศูนย์ แต่ถ้าจำนวนของบิตที่มีค่าเป็นหนึ่งเป็นจำนวนคี่ บิตนี้จะมีค่าเป็นหนึ่ง จากนั้นข้อมูลที่ส่งออกไปจะตามด้วยบิตสิ้นสุดข้อมูล (Stop Bit) เพื่อเป็นการบอกว่าข้อมูลที่ส่งออกมา 8 บิตนั้นหมดแล้ว ตัวบิต Stop นั้นอาจมีจำนวนมากกว่า 1 บิตก็ได้เช่น 1.5 บิต 2 บิต

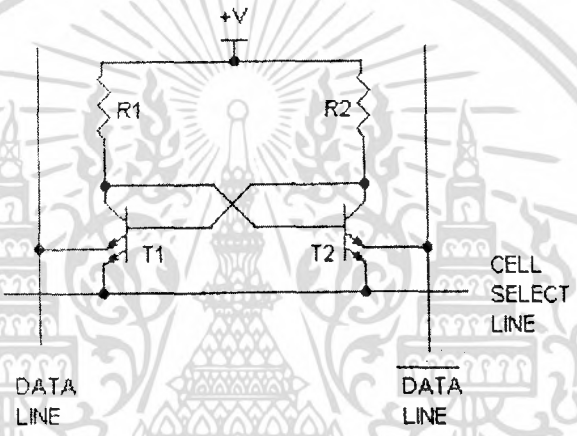
บิตตรวจสอบความถูกต้องหรือพาริตี้บิตจะมีสองลักษณะคือ พาริตี้คู่ (Even Parity) และพาริตี้คี่ (Odd Parity) ซึ่งเราสามารถเลือกได้ ถ้าหากระบุเป็นพาริตี้บิตคู่ หมายความว่าข้อมูลที่ส่งไปหรือไบต์นั้น มีจำนวนลอจิกนึ่งรวมกับพาริตี้เป็นจำนวนคู่บิต ส่วนถ้าระบุเป็นพาริตี้คี่ก็หมายความว่าจะมีจำนวนลอจิกหนึ่งของไบต์ข้อมูลที่ส่งไปรวมกับพาริตี้เป็นจำนวนคี่บิต ตัวอย่างเช่น ถ้าเราส่งไบต์ข้อมูลที่มีค่าเป็น B2H หรือ 10110010B ออกไป ถ้าระบุว่าเป็นพาริตี้คี่แล้ว ค่าของบิตพาริตี้จะต้องเป็น หนึ่งเพื่อให้มี จำนวนของ ลอจิกหนึ่งเป็น 5 บิต ซึ่งเป็นจำนวนคี่ เมื่อทางภาครับได้รับข้อมูลเข้ามาก็จะทำการตรวจสอบว่าข้อมูลทั้งหมดมีลอจิกหนึ่งเป็นจำนวนคู่หรือคี่ตามที่ได้ออกแบบการทำงานของระบบไว้หรือไม่ ซึ่งจะสามารถตรวจสอบความถูกต้องได้ในระดับหนึ่ง

ความเร็วในการสื่อสารแบบอนุรมนี้จะมีหน่วยเป็นบิตต่อวินาที(Bit Per Second: BPS) โดยเรียกกันว่าบอดเรต (Baud Rate) ซึ่งจะกำหนดค่ามาตรฐานในการใช้งานไว้หลายค่าเช่น 100 150 300 600 2400 4800 9600 19200 เป็นต้น

2.4 หน่วยความจำข้อมูล

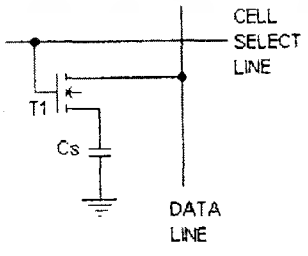
หน่วยความจำชนิดนี้ CPU สามารถอ่านและเขียนข้อมูลได้เรียกว่า RAM (Random Access Memory) เป็นหน่วยความจำที่ใช้เก็บข้อมูลจากการประมวลผลของ CPU ข้อมูลภายใน Ram จะคงอยู่ตลอดเวลาที่มีการแหล่งจ่ายไฟต่ออยู่กับหน่วยความจำ ตัวหน่วยความจำ RAM นี้ยังแบ่งออกได้เป็นสองชนิดใหญ่ๆดังนี้ คือ

2.4.1 สแตติกแรม (Static RAM) เป็นหน่วยความจำ RAM ที่สามารถอ่านและเขียนข้อมูลได้ และข้อมูลยังคงอยู่ตลอดถ้ามีไฟเลี้ยงโครงสร้างภายใน ในการเก็บข้อมูลแต่ละบิตจะสร้างเป็นฟลิปฟลอป ทำให้การเก็บข้อมูลแต่ละบิตจะต้องสร้างออกมาจากทรานซิสเตอร์ออกมาหลายตัวดังรูปที่ 2.10 แสดงโครงสร้างของสแตติกแรมในการเก็บข้อมูล 1 บิต

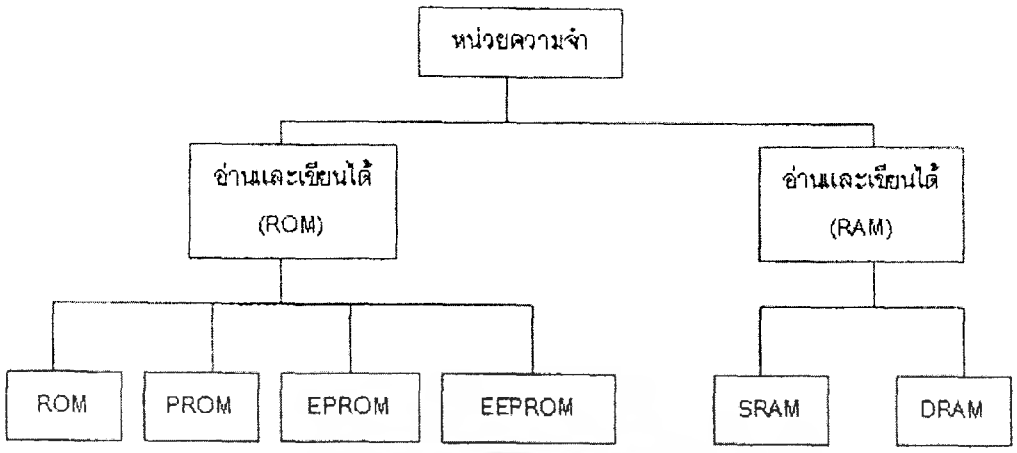


รูป 2.6 แสดงการเก็บข้อมูลแต่ละบทของสแตติกแรม

2.4.2 ไดนามิกแรม (Dynamic RAM) เป็นหน่วยความจำข้อมูลจะคงอยู่ตลอดไปถ้ามีไฟเลี้ยง แต่จะต้องมีการกระตุ้นหรือเขียนข้อมูลซ้ำ (Refresh) ตลอดเวลาด้วยวงจรพิเศษ เนื่องจากโครงสร้างภายในของไดนามิกแรมจะสร้างเป็นตัวเก็บประจุ ดังนั้นในการเก็บข้อมูล 1 บิต จึงต้องมีการเขียนข้อมูลซ้ำเพื่อให้ประจุคงอยู่ และเนื่องจากโครงสร้างภายในเป็นแบบตัวเก็บประจุ ทำให้ในการเก็บข้อมูล 1 บิตจะสร้างทรานซิสเตอร์ไม่กี่ตัว ซึ่งทำให้ความจุของข้อมูลต่อชิพสูงกว่าหน่วยความจำแบบสแตติกแรม



รูปที่ 2.7 แสดงโครงสร้างภายในของไดนามิกแรม

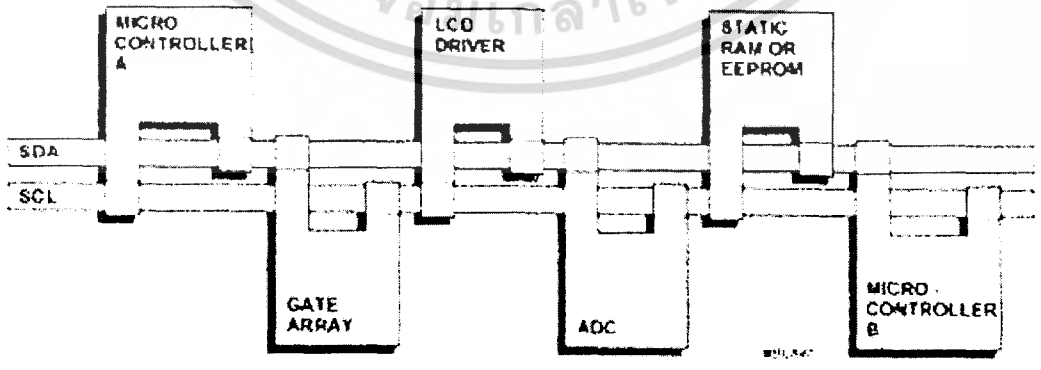


รูปที่ 2.8 แผนภาพของหน่วยความจำต่างๆ

2.5 การติดต่อกับอุปกรณ์โดยใช้ระบบบัสแบบ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ซึ่งได้รับการพัฒนาขึ้นจากบริษัท Philips ด้วยจุดมุ่งหมายคือต้องการให้ไอซี หรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่ง คือ สายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกา ที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูล และสายสัญญาณนาฬิกา ของอุปกรณ์แต่ละตัว ขนาน หรือพ่วงกันไป ส่วนการกำหนด แอดเดรสหรือตำแหน่งสำหรับติดต่อกับอุปกรณ์แต่ละตัวจะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรมหรือ SDA (Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อว่า สายสัญญาณอุกรมหรือ SCL (Serial Clock Line)

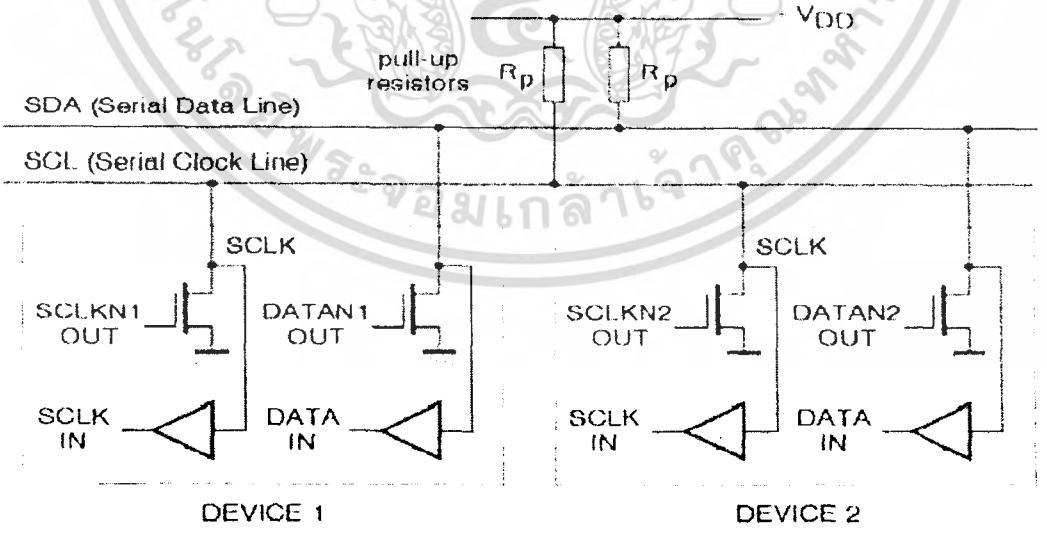


รูปที่ 2.9 ผังแสดงการเชื่อมต่ออุปกรณ์ต่างๆ บนระบบบัส I^2C

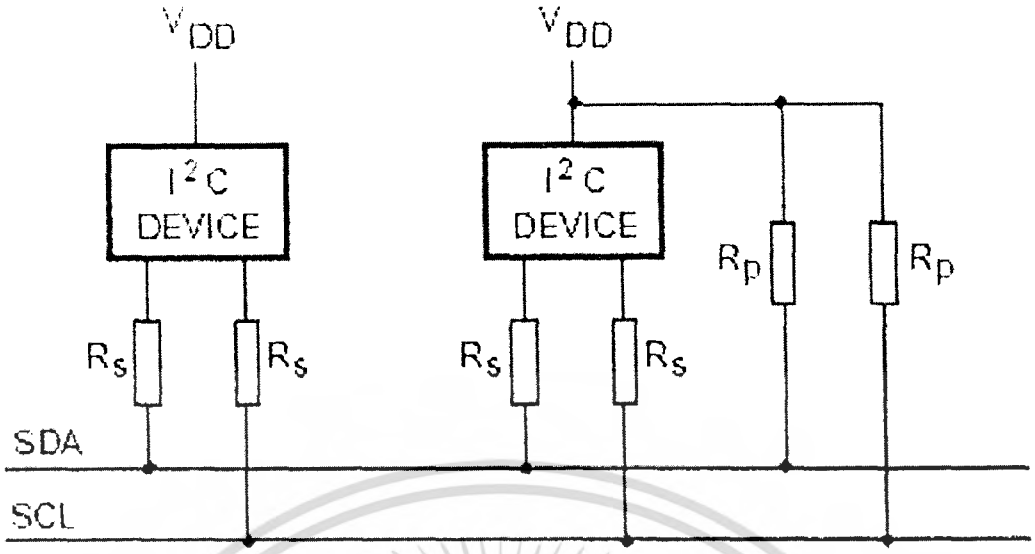
จากรูปที่ 2.9 แสดงผังของการเชื่อมต่ออุปกรณ์ต่างๆ บนบัส I^2C จะเห็นได้ว่าอุปกรณ์ที่ทำการเชื่อต่อบนบัส I^2C มีหลากหลาย เช่นไอซีขยายพอร์ตอินพุต (I/O Expander) ไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) และแปลงสัญญาณดิจิตอลเป็นอะนาลอก (DAC) ไอซี Real Time Clock (RTC) ไอซีขับโมดูล LCD หน่วยความจำ EEPROM และไมโครคอนโทรลเลอร์

2.5.1 คุณสมบัติทั่วไปของบัส I^2C สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (Bi-direct Line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลาเพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งานทั้งช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (Open-Drain) หรือคอลเล็กเตอร์เปิด (Open-Collector) ดังแสดงรายละเอียดที่รูป 2.14

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (Standard Mode) และสูงถึง400กิโลบิตต่อวินาทีในโหมดความเร็วสูง (Fast Mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7บิต (7-bit Addressing) หรือ 10 บิต (10-bit Addressing) ข้อเด่นอีกประการหนึ่งของบัส I^2C คือสามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้ โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกันกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือให้ต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป(Rp) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังรูปที่ 2.10



รูปที่ 2.10 แสดงวงจรเอาต์พุตของอุปกรณ์ในระบบ I^2C



รูปที่ 2.11 การต่อตัวต้านทาน Rs เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในบัส

I^2C ในกรณีที่อาจมีแรงดันไฟฟ้ากระฉากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDAและ SCL ของอุปกรณ์แต่ละตัวต้องมีตัวต้านทานต่ออนุกรมกับขา SDA และ SCL เรียกว่า Rs ก่อนต่อเข้าสู่บัส I^2C ดังแสดงในรูปที่ 2.15

2.5.2 หลักการของบัส I^2C บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้นดังนั้นได้กล่าวมาแล้ว คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (Protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดกันอยู่ และอุปกรณ์ใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะขออธิบายหน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

- อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูลเรียกว่าตัวส่ง (Transmitter)
- อุปกรณ์ที่เป็นผู้รับข้อมูลเรียกว่า ตัวรับ (Receiver) ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งอย่างเดียว
- อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (Master)
- อุปกรณ์ที่ถูกควบคุม หรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (Slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

หนึ่ง การถ่ายข้อมูลเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

สอง ในระหว่างการถ่ายข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

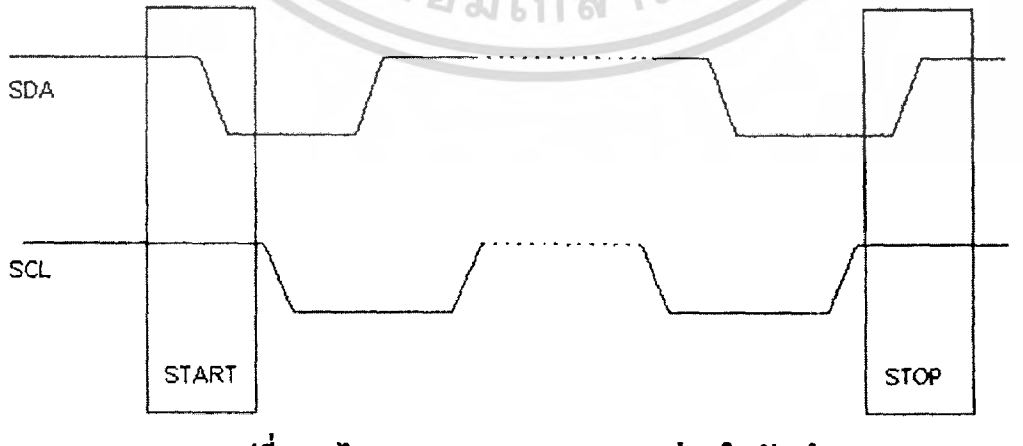
2.5.3 สภาวะที่เกิดขึ้นบนบัส I^2C มีด้วยกัน 5 สภาวะ ดังนี้

หนึ่ง บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายข้อมูลสามารถเริ่มต้นขึ้นได้

สอง เริ่มต้นการถ่ายข้อมูล(Start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START)

สาม หยุดการถ่ายทอดข้อมูล (Stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูงเรียกสภาวะที่เกิดขึ้นว่า สภาวะหยุด (STOP)

สี่ ข้อมูลดำรงอยู่บนบัส (Data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น “0” หรือ “1” ข้อมูลอาจเกิดการเปลี่ยนแปลงขึ้นได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำหน้าที่การควบคุมการถ่ายทอดข้อมูล และแปลความหมายเป็นสภาวะหยุด หรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น



รูปที่ 2.12 ไดอะแกรมเวลาแสดงสถานะต่างๆในบัส I^2C

ห้า การรับรู้ข้อมูล (Acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับ (Acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลครบถ้วน ส่วนอุปกรณ์ Master จะทำการส่งสัญญาณรับรู้พิเศษ ซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์ Slave ที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้น ก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

ในรูปที่ 2.16 เป็นไดอะแกรมเวลาที่แสดงถึงสภาวะต่างๆที่ต่ออยู่บนบัส I^2C ไม่ว่าจะเป็นสภาวะบัสว่าง เริ่มต้น ถ่ายข้อมูล รับรู้ และหยุดการถ่ายทอดข้อมูล

2.5.4การทำงานบนบัส I^2C ก่อที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อนโดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะต้องใช้การอ้างถึงแบบ 7 บิต หรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิต ก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้วก็จะเริ่มถ่ายทอดข้อมูลกันต่อไป

ดังนั้นหัวใจสำคัญอันดับแรกของการทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดของการอ้างถึงทั้ง 2 รูปแบบ

2.5.5 การอ้างถึงแบบ 7 บิต (7-bit addressing)

รูปที่ 2.13 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อหรือข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูปที่ 2.17 ใน 7 บิตบนรวมทั้ง MSB ด้วย จะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็นบิตกำหนดแอดเดรสคงที่ (Fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถดัดแปลงแก้ไขได้ถัดมาอีก 3 บิต เป็นบิตกำหนดแอดเดรสที่สามารถกำหนดโปรแกรมได้ (Programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียน

ข้อมูลกับอุปกรณ์สเลฟตัวนั้นหาก LSB เป็น “0” หมายถึงการเขียนข้อมูลไปยังอุปกรณ์นั้นถ้าเป็น “1” จะเป็นการอ่านข้อมูลจาดอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่างไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุตบิตใดเป็นเอาท์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลเพื่อกำหนดให้การทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลไบต์ต่อมาคือ ข้อมูลควบคุม(Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไปยกตัวอย่างไอซีขยายพอร์ต มีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาท์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (Data) หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ ตอบกลับมาด้วยทุกครั้งเพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

2.5.6 การอ้างถึงแบบ 10 บิต ในการอ้างถึงแบบนี้ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกันแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อ เช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดจ้อมูลสามารถดำเนินต่อไปได้

2.5.7 อุปกรณ์ที่ใช้การเชื่อมต่อแบบบัส I^2C ในปัจจุบันบัส I^2C ได้รับความนิยมเพิ่มมากขึ้นเรื่อยๆ ด้วยข้อดีที่ชัดเจนคือ ใช้สายสัญญาณเพียง 2 เส้นเท่านั้นและการขยายระบบไมโครคอนโทรเลอร์ ที่มีจำนวนอินพุตและหน่วยความจำ จำกัดสามารถทำได้ง่ายขึ้นด้วยระบบบัส I^2C เมื่อเป็นเช่นนี้ จึงมีอุปกรณ์ เพอริเฟอรัล ที่ใช้การเชื่อมต่อแบบบัส I^2C มากมายจากหลายผู้ผลิตออกมาให้ได้ใช้งานกันดังมีตัวอย่างดังต่อไปนี้

ไอซีขยายพอร์ตอินพุตเอาท์พุต (I/O Expander) PCF8574 PCF8582 PCF8584

ไอซีหน่วยความจำอีอีพรอมอนุกรม(Serial EEPROM) 24Cxx PCF8570 PCF72/73 PCF8582

ไอซี ADC/DAC PCF8591

ไอซีรีลไทม์คล็อก (Real-Time clock: RTC) PCF8583

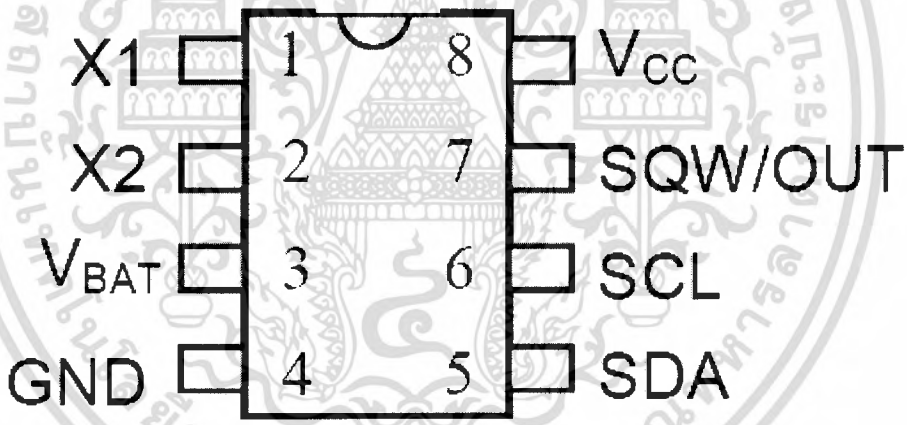
ไอซี LCD โมดูล (LCD driver): PCF8466 PCF8576 PCF8577/78 PCF8579 SAA1064

2.6 การใช้งานไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (DS1307)

DS1307 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก ผู้ผลิตคือ ดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีหน้าที่สร้างฐานเวลาจริงให้แก่ระบบไมโครคอนโทรลเลอร์โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที นาที ชั่วโมง วันที่ วันในสัปดาห์ เดือน และปี โดยสามารถ ปรับวันเดือนปีให้ตรงตามปฏิทินได้อย่างถูกต้องรวมถึงการกำหนดวันในปีอธิกสุรทินด้วย คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

- เป็นไอซีรีลไทม์คล็อกให้ข้อมูลตั้งแต่ วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรทินด้วย สามารถให้ข้อมูลเวลาได้อย่างเที่ยงตรงถึงปี ค.ศ. 2100
- ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี

2.6.1 รายละเอียดขาต่อใช้งานของ DS1307 ในรูปที่ 2.14 แสดงการจัดขาของ DS1307 แต่ละขามีหน้าที่ละการใช้งานดังนี้ Vcc GND (ขา 8, 4) ต่อไฟเลี้ยง +5V



รูปที่ 2.14 การจัดขาไอซีของ DS1307

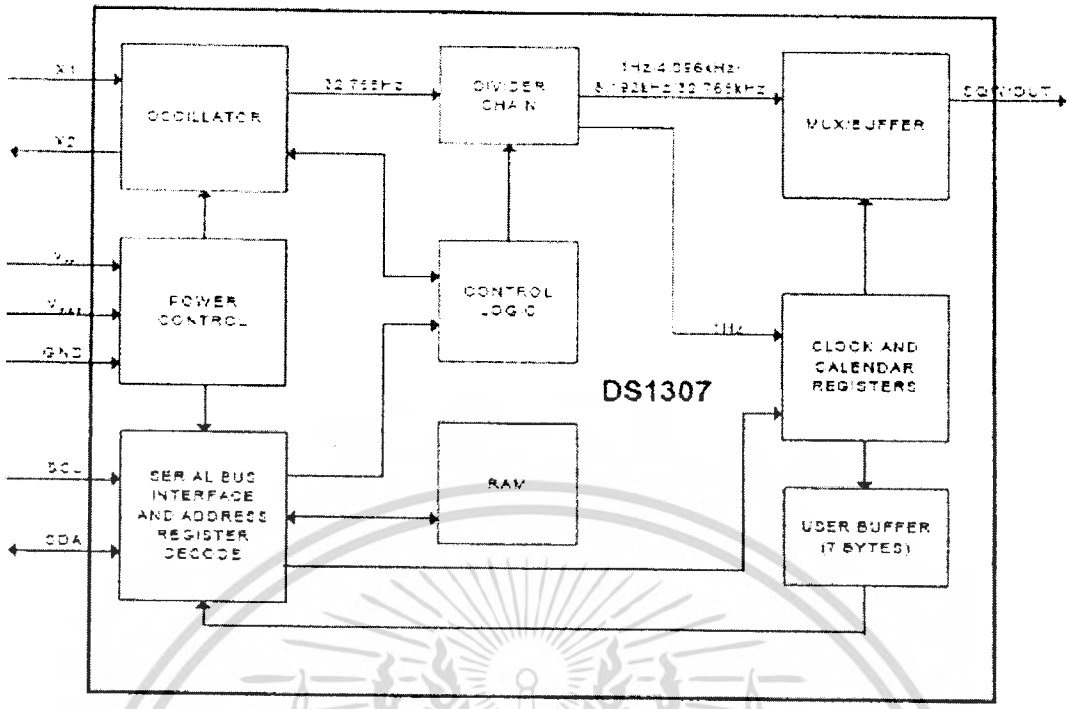
VBAT (ขา3) ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS1307 ให้คงอยู่ต่อไป แม้ว่าไม่มีไฟเลี้ยงจ่าให้กับ DS1307 ชนิดของแบตเตอรี่ที่เหมาะสมคือ แบตเตอรี่แบบลิเธียม ซึ่งมีความจุ 40 mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นาน 10 ปีที่อุณหภูมิ 25 องศาเซลเซียส SDA, SCL (ขา 5 และ 6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบ I^2C ในการใช้งานต้องต่อตัวต้านทานพูลอัพที่ขานี้ด้วย SQW/OUT (ขา7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกความถี่ได้ 1 kHz, 4.096 kHz, 8.192 kHz และ 32 kHz ในการใช้งานต้องต่อตัวต้านทานพูลอัพที่ขานี้ด้วย X1, X2 (ขา 1 และ ขา2) ใช้ต่อกับคริสตอลมาตรฐาน

32.768 kHz เพื่อใช้เป็น ฐานเวลาในการสร้างค่าเวลาจริงในการใช้งาน และแต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆประมาณ 15 pF คร่อมกับขากราวด์ด้วย

2.6.2 การทำงานของ DS1307 ไอซี DS1307 จัดการเชื่อมต่อในแบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์ สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งาน จึงต้องกำหนดรูปแบบตามที่กำหนด วงจรออสซิเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริงในขณะที่ DS1307 ทำงาน ที่ขาSQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลาในกรณีที่การอีเนเบิลวงจรกำหนดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่าคือ 1Hz 4.096KHz 8.192kHz และ 32 kHz พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาไว้หน่วยความจำนอนวอละไทป์ ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 3 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25*VBAT ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า 1.25*VBAT หรือประมาณ 3.75V ในกรณีที่ใช้ VBAT 3V

หากไฟเลี้ยงมีค่าต่ำกว่า VBAT ไอซีจะเข้าสู่โหมดสำรองข้อมูลกรแสต่ำทันที จะไม่มีการส่งพัลส์สัญญาณออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานอยู่เพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานต่อไป



รูปที่ 2.15 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่วงทางการสื่อสารระหว่าง DS1307 กับอุปกรณ์มาสเตอร์ ผู้ใช้สามารถเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลาและหน่วยความจำใช้งานทั่วไปได้โดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบบัส I^2C

2.6.3 การจัดสรรหน่วยความจำใน DS1307 ในรูปที่ 2.16 แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลา ที่ใช้เก็บข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07 เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ซึ่งแสดงรายละเอียดของรีจีสเตอร์ค่าเวลา และรีจีสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามต้องการ โดยไม่จำเป็นต้องอ่านออกทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงเวลาในรูปของชั่วโมง สามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 ชั่วโมง โดยกำหนดที่บิตที่ 6 ของแออดเดรส 02H และเมื่อเลือก 12 ชั่วโมงที่บิตที่ 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น “1” หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงเวลาหลังเที่ยงวัน โดยในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้แสดงค่า 2 ของเลขฐานสิบในหน่วยชั่วโมง

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00-59
01H	0	10 Minutes			Minutes				Minutes	00-59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1-12 +AM/PM 00-23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01-07
04H	0	0	10 Date		Date				Date	01-31
05H	0	0	0	10 Month	Month				Month	01-12
06H	10 Year				Year				Year	00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H-FFH

รูปที่ 2.16 การจัดหน่วยความจำแรมภายใน DS1307

2.6.4 รีจีสเตอร์ควบคุม มีแอดเดรสอยู่ที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

หนึ่ง OUT (Output Control): ใช้ในการควบคุมระดับลอจิกที่ขา SQW/OUT ในกรณีที่ดิสเอเบิลการกำหนดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น “1” ที่ขา SQW/OUT ก็จะเป็น “1” ถ้าบิตนี้เป็น “0” ที่ขา SQW/OUT ก็จะเป็น “0”

สอง SQWE (Square Wave Enable): ใช้ในการอีนาเบิลวงจรกำหนดสัญญาณสี่เหลี่ยมที่ขา SQW/OUT ถ้าต้องการให้มีสัญญาณสี่เหลี่ยมออกมา ให้กำหนดบิตนี้เป็น “1”

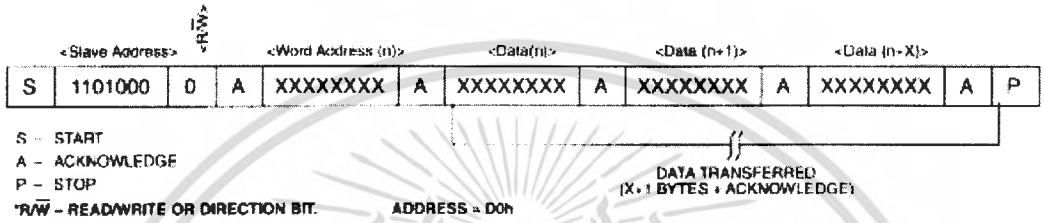
สาม RS1 RS2 (Rate Select): ใช้ในการเลือกความถี่ของสัญญาณสี่เหลี่ยมที่ออกจากขา SQW/OUT ดังมีรายละเอียดดังต่อไปนี้

RS1	RS0	ค่าความถี่ของสัญญาณสี่เหลี่ยม
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

ตารางที่ 2.1 แสดงความถี่ของสัญญาณจากขา SQW/OUT

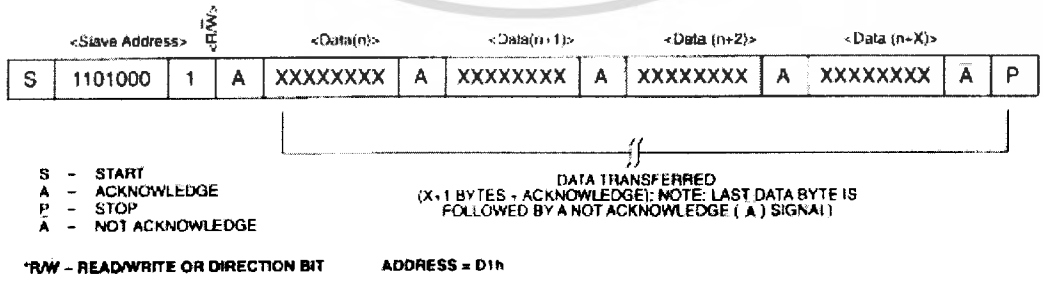
2.6.5 โหมดการทำงานของ DS 1307 มีด้วยกัน 2โหมด คือ โหมดเขียนข้อมูล และโหมดอ่านข้อมูลในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งานโหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อตั้งค่าเวลาใหม่ และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเข้าข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูลจากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูลต่อไป

- โหมดการเขียนข้อมูล มีรูปแบบดังในรูปที่ 2.17 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (START: S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียนนั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมาคือส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียนหลังจากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรบจาก DS1307 ทุกครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (STOP: P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล



รูปที่ 2.17 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล

- โหมดการอ่านข้อมูล มีรูปแบบแสดงในรูปที่ 2.18 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้น แล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบกลับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรส หรือ 1 ไบต์โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้า ด้วยโหมดการเขียนข้อมูล วิธีการง่ายๆคือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้งตามด้วยเลือกโหมดการอ่านข้อมูลที่ออกจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้



รูปที่ 2.18 รูปแบบข้อข้อมูลสำหรับติดต่อกับ DS1307ในโหมดการอ่านข้อมูล

2.7 หน่วยความจำEEPROM แบบ I^2C

ในที่นี้เราได้เลือกใช้หน่วยความจำ EEPROM เบอร์ 24LC512 โดยมีขนาดหน่วยความจำภายในขนาด 64Kb ซึ่งทำหน้าที่หลักในการจัดเก็บข้อมูลในส่วนของลายนิ้วมือ และข้อมูลแสดงการเข้าใช้งาน หรือเลิกใช้งานเครื่องคอมพิวเตอร์ย้อนหลัง ซึ่งมีข้อดีคือไม่จำเป็นต้องใช้ไฟเลี้ยง IC ในการจัดเก็บข้อมูลในขณะปิดเครื่อง โดยสามารถเขียนหรืออ่านได้นับล้านครั้ง

2.7.1 รายละเอียดขาต่อใช้งานของ 24LC512 ในรูปที่2.19 เป็นการแสดงการจัดขาของ 24LC512 แต่ละขามีหน้าที่และการใช้งานดังนี้

รูปที่ 2.19 การจักการขาของ 24LC512

- A0, A1, A2 (ขา1-3) ทั้ง3ขานี้เป็นขาที่มีไว้เพื่อเซตค่าแอดเดรส ของ 24LC512
- Vss (ขา4) ใช้สำหรับต่อ Ground
- SDA, SCL (ขา5และขา 6) ใช้สำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบ I^2C
- WP (ขา7) ขานี้คือ Write Protect ถ้าขานี้เป็นลอจิกสูงจะเป็นการป้องกันการเขียนจะทำให้สามารถอ่านข้อมูลจาก 24LC512 ได้อย่างเดียว แต่ถ้าให้ขานี้เป็นลอจิกต่ำ หรือปล่อยขานี้ลอยไว้จะเป็นการเขียนข้อมูลหรืออ่านข้อมูลตามปกติ
- Vcc (ขา8) ขานี้ไว้ใช้สำหรับต่อไฟเลี้ยงเข้ากับตัว IC ซึ่งระดับของไฟเลี้ยงนั้นจะอยู่ระหว่าง 2.5ถึง 5.5 V

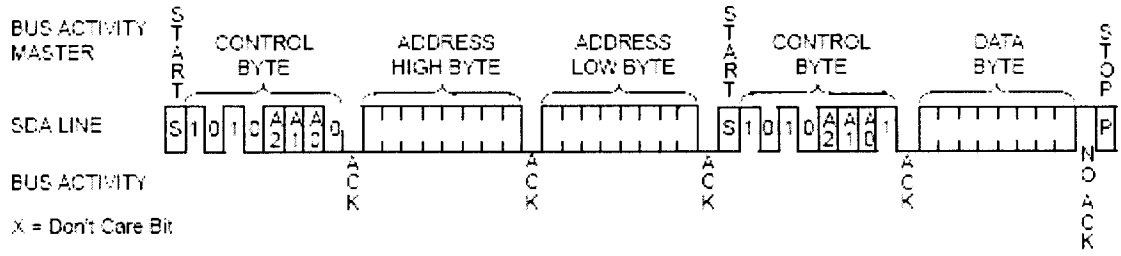
2.7.2 โหมดการทำงานของ 24LC512

- โหมดการเขียนข้อมูล มีรูปแบบดังในรูปที่ 2.20 เรียกว่าการเขียนแบบ Byte Write เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ ทำงานกำหนดสภาวะเริ่มต้น (START: S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1010000 จากนั้นรอการตอบรับจาก 24LC512 ต่อมาให้ส่งแอดเดรส HIGH BYTE แล้วรอสภาวะตอบกลับจากนั้นให้ส่งแอดเดรส LOW BYTE จากนั้นให้รอสภาวะตอบกลับ ขั้นตอนต่อมาคือ เริ่มส่งข้อมูลที่ต้องการเขียนลงไป ซึ่งได้ครั้งละแอดเดรสจากนั้นก็ให้รอสภาวะตอบกลับแล้วจึงส่งสภาวะหยุด (STOP: P) เป็นการสิ้นสุดการเขียนข้อมูลขนาด 1 ไบต์ ลงไปใน 24LC512 จากนั้นถ้าต้องการเขียนเพิ่มก็ทำการเพิ่มแอดเดรสไปเรื่อยๆ

จะมีการเขียนข้อมูลลงไปใน 24LC512 อีกรูปแบบหนึ่งคือการเขียนแบบ Page Write คือการส่งข้อมูล I^2C นั้นจะเหมือนแบบ Byte Write แต่ต่างกันที่หลังจากรอสภาวะตอบกลับจากการส่งข้อมูลแอดเดรส LOW BYTE แล้วให้ส่งข้อมูล และรอสภาวะตอบกลับ หลังจากนั้นแทนที่จะส่งสภาวะหยุด (STOP: P) ให้ส่งข้อมูลไปเรื่อยๆ ซึ่งการเขียนข้อมูลลักษณะนี้ จะเขียนได้ครั้งละ 64 BYTE จากนั้นจึงส่งสภาวะหยุด (STOP: P) ไปที่ 24LC512

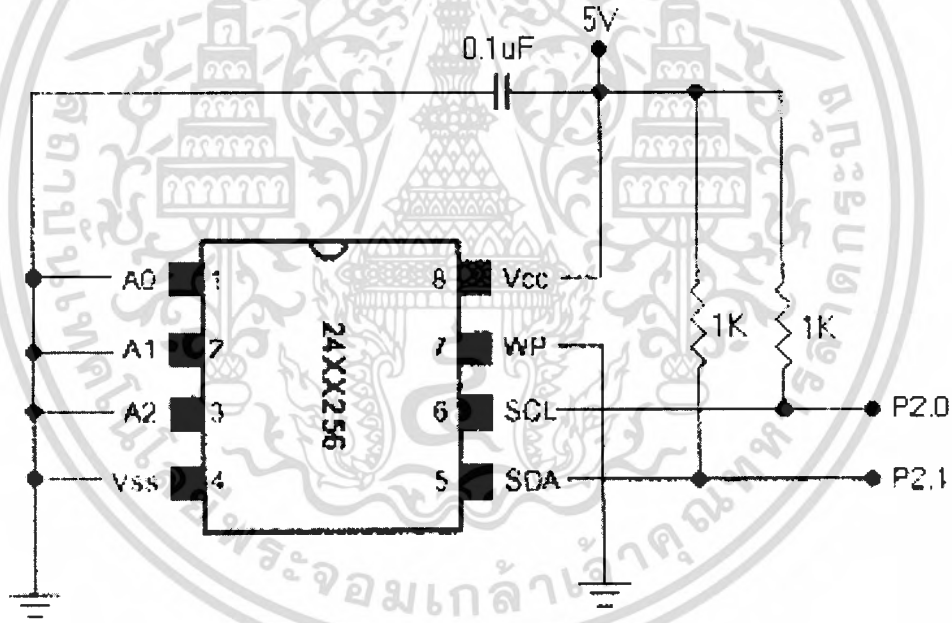
รูปที่ 2.20 รูปแบบข้อข้อมูลสำหรับติดต่อกับ 24LC512 ในโหมดการเขียนข้อมูลแบบ Byte Write

- โหมดการอ่านข้อมูล มีรูแบบแสดงในรูปที่ 2.21 โหมดการอ่านข้อมูลมีหลายรูปแบบในที่นี้เราเลือกใช้โหมดการอ่านข้อมูลแบบกำหนด ADDRESS เป็นหลักซึ่งมีรูปแบบการทำงานดังนี้ คือ เริ่มส่งสภาวะ (START: S) จากนั้นส่งข้อมูลกำหนดแอดเดรสคือ 1010001 แล้วรอสภาวะตอบกลับ หลังจากนั้นเอาตัวแปรมารับข้อมูลที่ส่งออกมาจากแอดเดรสนั้นๆ และหลังจากนั้นจะไม่มีสภาวะตอบกลับจาก 24LC512 แล้วส่งให้สภาวะหยุด (STOP: P) ไป



รูปที่ 2.21 รูปแบบของข้อมูลสำหรับติดต่อกับ 24LC512 ในโหมดการอ่านข้อมูลแบบ Random Read

2.7.3 การเชื่อมต่อ 24LC512 กับไมโครคอนโทรลเลอร์ ในการเชื่อมต่อ 24LC512 เข้ากับไมโครคอนโทรลเลอร์นั้น จำเป็นที่จะต้องกำหนดแอดเดรสของตัว 24LC512 ใช้คุณสมบัติ ของการเชื่อมต่อแบบ I^2C ซึ่งเป็นการใช้สายสัญญาณเพียง 2 เส้น คือ SCL และ SLC

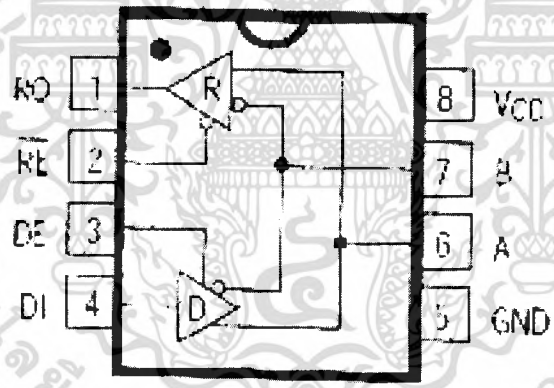


รูปที่ 2.22 แสดงการเชื่อมต่อ 24LC256 เข้ากับไมโครคอนโทรลเลอร์

2.8 มาตรฐานการสื่อสารข้อมูลอนุกรมแบบอาร์เอส 485

อาร์เอส 485 เป็นมาตรฐานการสื่อสารข้อมูลแบบสมดุลพัฒนามาจาก มาตรฐาน อาร์เอส 422A (RS422A) เพื่อให้ตัวรับและตัวส่งจำนวนมากคู่ สามารถใช้คู่สายในการรับส่งสัญญาณร่วมกันได้ (Multipoint multiple drivers and receivers) ซึ่งในกรณีของ อาร์เอส 422A คู่สายสัญญาณสามารถรับส่งได้ 1 คู่ และมีตัวรับได้ไม่เกิน 10 ชุด และมีตัวส่งเพียง 1 ชุด ในกรณี อาร์เอส 485 สามารถใช้ตัวรับ 32 ชุด และตัวส่ง 32 ชุดร่วมกันได้ภายในคู่สัญญาณ 1 คู่ โดยทั่วไป อาร์เอส485 มีคุณลักษณะเฉพาะทางไฟฟ้าของตัวรับ และตัวส่งคล้ายตัวรับ และตัวส่งเพียงหนึ่งชุด แต่ในกรณีของ อาร์เอส 422A ไม่จำกัดโปรโตคอลที่จะนำไปใช้งานกับ ระบบที่พัฒนาขึ้น โดยขึ้นอยู่กับผู้พัฒนาระบบเองว่าจะเลือกใช้โปรโตคอลแบบไหนมาใช้ นอกจากนี้ตัวรับและตัวส่งมีราคาไม่สูง ทำให้ อาร์เอส 485 ถูกนำมาประยุกต์กันอย่างแพร่หลายสำหรับการเชื่อมต่อ

2.9.1 รายละเอียดขาต่อการใช้งาน RS-485ในรูปที่ 2.23 แสดงการจัดขาของRS-485 ซึ่งแต่ละขามีหน้าที่และการใช้งานดังนี้



รูปที่ 2.23 การจัดการขาของ RS-485

- R0 (ขาที่1): Receiver Output ใช้ต่อกับขา RxD ของไมโครคอนโทรลเลอร์

- ขา RE, DE (ขาที่2, 3): Receiver Output Enable และ Driver Output Enable ถ้าต้องการให้ RS-485 ตัวไหนเป็นตัวส่งต้องให้ทั้ง 2 ขานี้เป็น 1 และเมื่อต้องการให้ตัวไหนเป็นตัวรับต้องให้ทั้งสิงขานี้เป็น 0

- DI (ขาที่4): Driver input ใช้ต่อกับขา TxD ของไมโครคอนโทรลเลอร์

- GND (ขาที่5) ใช้ต่อกับ Ground

- A (ขาที่6): No inverting Receiver Input and No inverting Driver Output ใช้ต่อกับโครงข่าย RS-485

- B (ขาที่7): Inverting Receiver Input and Inverting Driver Output ใช้ต่อกับโครงข่ายของ RS-485

- Vcc (ขาที่8): ใช้ต่อกับไฟเลี้ยง 5V

2.8.2 คุณลักษณะเฉพาะของ อาร์เอส 485

คุณลักษณะของตัวส่ง

- ตัวส่ง 1ตัวสามารถขับโหลดได้ถึง 32 ชุด (ตัวรับ 1 ตัว ตัว ส่ง 1 ตัว) และค่าความต้านทานรวมระหว่างคู่สายมากกว่า 60 โอห์ม

- เอาท์พุตของตัวส่งในสภาวะออฟ มีกระแสรั่วไม่เกิน 100 ไมโครแอมป์ในช่วงแรงดันไฟฟ้าโหมดร่วมระหว่าง -7โวลต์ ถึง 7 โวลต์

- เอาท์พุตของตัวส่งในสภาวะออฟ มีกระแสรั่วไม่เกิน 700 ไมโครแอมป์ในช่วงแรงดันไฟฟ้าโหมดร่วมระหว่าง -7โวลต์ ถึง 7 โวลต์

- เอาท์พุตของตัวส่ง ให้แรงดันไฟฟ้าเอาท์พุต 1.5 โวลต์ ถึง 5 โวลต์ ในช่วงแรงดันไฟฟ้าโหมดร่วม ระหว่าง -7โวลต์ ถึง 12 โวลต์

- ตัวส่งมีวงจรป้องกันตัวเองที่ส่วนเอาท์พุต และในกรณีที่ตัวส่งหลายๆตัวส่งข้อมูลออกมาพร้อมๆกัน

คุณลักษณะของตัวรับ

- ค่าความต้านทานที่อินพุตมีค่าสูง โดยมีค่าไม่น้อยกว่า 12 กิโลโอห์ม

- ตัวรับมีค่าแรงดันไฟฟ้าอินพุต โหมดร่วม ระหว่าง -7โวลต์ ถึง 12 โวลต์

- ตัวรับสามารถตอบสนองต่อ สัญญาณที่แตกต่างจากสัญญาณโหมดร่วมได้ +200 มิลลิโวลต์ และ-200มิลลิโวลต์

บทที่ 3
โครงสร้าง และการออกแบบ

3.1 โครงสร้างและการออกแบบระบบ

3.1.1 บอร์ดหลัก

บอร์ดหลักนั้นจะทำหน้าติดต่อกับผู้ใช้งานในการตรวจสอบลายนิ้วมือ และทำการส่งคำสั่งไปให้กับหน่วยตัด-จ่ายกำลังไฟฟ้า ผ่านทางอนุกรม RS485 และติดต่อกับ PC เพื่อทำการบันทึกข้อมูลเข้าใช้และลงทะเบียนลายนิ้วมือ ผ่านอนุกรม RS232 ส่วนโครงสร้างในส่วนของบอร์ดหลักนั้น จะประกอบไปด้วยองค์ประกอบต่างๆดังนี้

1.1) หน่วยประมวลผลกลาง ซึ่งทำหน้าที่ประมวลผลข้อมูลต่างๆที่รับมาจากอุปกรณ์ตัวอื่นๆ และส่งคำสั่งไปเพื่อควบคุมอุปกรณ์ทั้งหมดบนบอร์ด โดยในโครงงานนี้ใช้ Microcontroller ตระกูล PIC เบอร์ 16F877 ขนาด 40 ขา

1.2) หน่วยแสดงผล ใช้จอ LCD ขนาด 16*2 ทำหน้าที่แสดงผลข้อมูลต่างๆให้ผู้ใช้ได้รับทราบโดยจะติดต่อกับหน่วยประมวลผลกลางแบบ 4Bit

1.3) โมดูลสแกนลายนิ้วมือ ทำหน้าที่สแกนและเปรียบเทียบลายนิ้วมือของผู้ใช้และส่งข้อมูล ID ของผู้ผู้ใช้ให้กลับหน่วยประมวลผลกลางผ่านพอร์ตอนุกรม

1.4) หน่วยเก็บข้อมูลใช้ ใช้หน่วยความจำ EEPROM เบอร์ 24LC512 ซึ่งมีขนาดหน่วยความจำ64K Byte จำนวน 2 ตัว ติดต่อกับหน่วยประมวลผลกลางผ่าน Bus I2C

1.5) IC สร้างฐานเวลาจริง เบอร์ DS1307 ใช้ในการอ้างอิงเวลาในการลงชื่อเข้าใช้ของผู้ใช้งาน กับหน่วยประมวลผลกลางผ่าน Bus I2C

1.6) RS232 และ RS485 Transceiver ใช้ IC เบอร์ MAX232 และ SN 75176 ตามลำดับ

ขั้นตอนการทำงานของส่วนบอร์ดหลักโดยคร่าวจะเป็นดังนี้ ในสภาวะปกติเครื่องจะ Stand by รอให้ผู้ใช้เอานิ้วทาบที่โมดูลแสกนลายนิ้วมือ เมื่อผู้ใช้ทาบนิ้วลงบนโมดูล โมดูลก็จะทำการสแกนลายนิ้วมือของผู้ใช้ หากข้อมูลตรงกันกับลายนิ้วมือที่มีการลงทะเบียนไว้แล้ว ก็จะตรวจสอบว่าผู้ใช้คนนี้มีการใช้งานคอมอยู่แล้วหรือไม่ หากมีการใช้งานคอมอยู่แล้ว ก็จะส่งคำสั่งปิดคอมไปให้กับหน่วยตัด-จ่ายกำลังไฟฟ้าเบอร์ที่ที่ผู้ใช้ใช้งานอยู่ หากไม่มีการใช้งานคอมอยู่ก็จะให้ผู้ใช้เลือกคอมที่ต้องการจะเปิด แล้วจึงส่งคำสั่งเปิดคอมไปให้กับหน่วยตัด-จ่ายกำลังไฟฟ้าเบอร์ที่เลือก เมื่อส่งคำสั่งเสร็จแล้วก็จะรอรับการตอบกลับจาก หน่วยตัด-จ่ายกำลังไฟฟ้า หากได้รับข้อมูลว่าทำการเปิดหรือปิดเรียบร้อยแล้ว ก็จะทำการบันทึกข้อมูลของผู้ใช้ลงใน EEPROM

ในส่วนการทำงานในโหมดการลงทะเบียนและการบันทึกข้อมูลการเข้าใช้งานนั้นจะต้องทำการติดต่อกับ PC ผ่าน RS 232 โดยจะติดต่อกับโปรแกรมบน PC ที่เขียนขึ้นด้วยโปรแกรม Visual Basic

ในส่วนของการเก็บข้อมูลนั้นหน่วยประมวลผลกลางจะทำการจัดเก็บข้อมูลไว้ใน EEPROM 24LC512 ซึ่งมีขนาดหน่วยความจำ 64 kByte จำนวน 2 ตัว โดยผู้ใช้ 1 คนจะมีเนื้อที่ที่ใช้จัดเก็บข้อมูลคนละ 256 Byte และสามารถรองรับผู้ใช้งานได้ถึง 510 คน โดยการจัดเก็บข้อมูลเข้าใช้งานนั้นจะจัดเก็บหมายเลขของเครื่องที่ใช้งานและวันเวลาที่เข้าใช้โดยอ้างอิงจาก ไอซีเรียลไทม์คล็อก DS1307 การเข้าใช้งานของผู้ใช้ 1 ครั้งจะใช้เนื้อที่ในการจัดเก็บข้อมูลทั้งสิ้น12 Byte

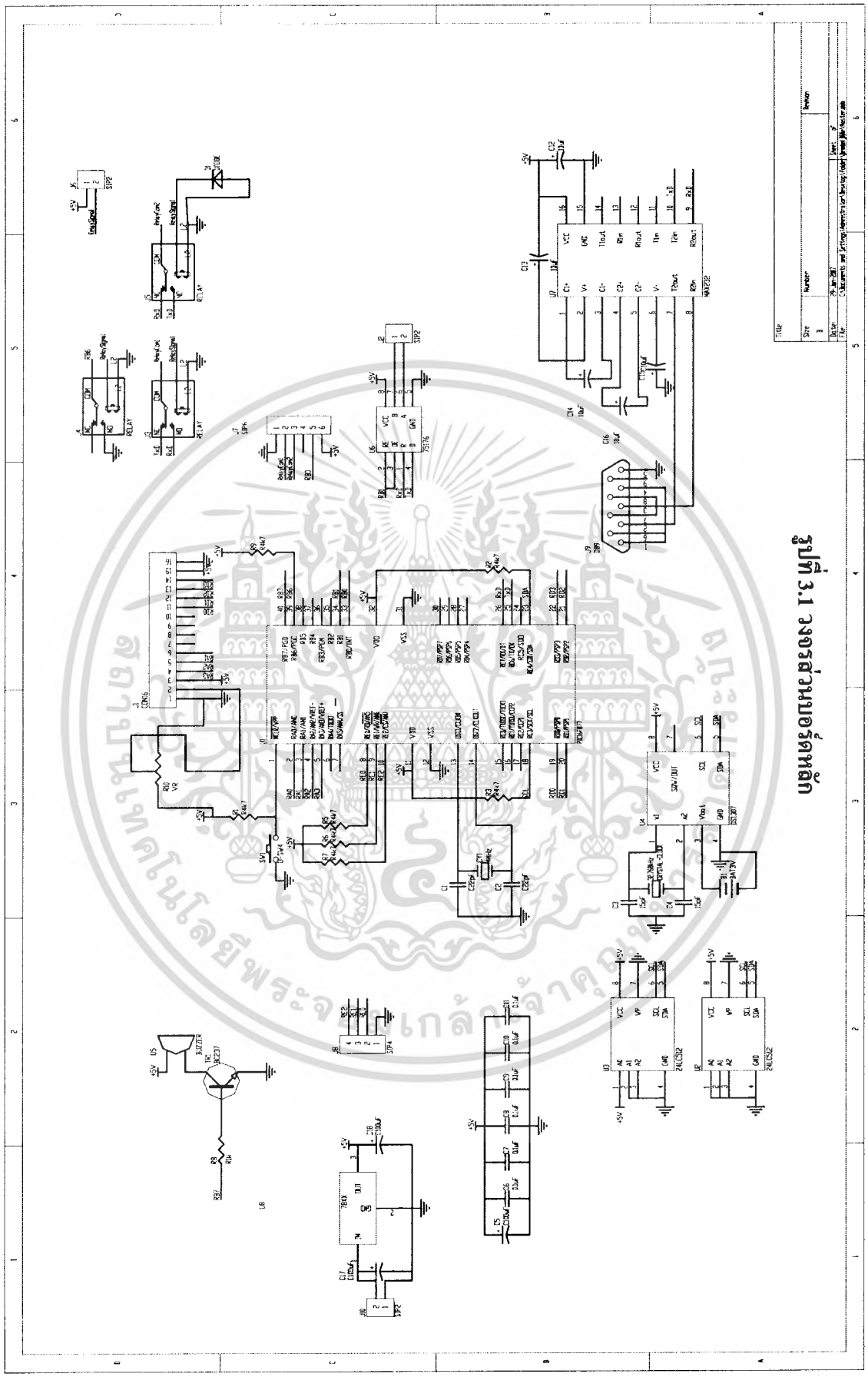
3.1.2 หน่วยตัด –จ่ายกำลังไฟฟ้า

หน่วยตัด –จ่ายกำลังไฟฟ้า จะทำหน้าที่ตัด-จ่ายไฟฟ้าให้กับเครื่องคอมพิวเตอร์ที่ต้องการควบคุมการเปิดปิด โดยจะรับคำสั่งมาจากบอร์ดหลักผ่านอนุกรม RS-485 ซึ่งในหน่วยตัด – จ่ายกำลังไฟฟ้านี้จะประกอบด้วยส่วนประกอบหลักๆคือ

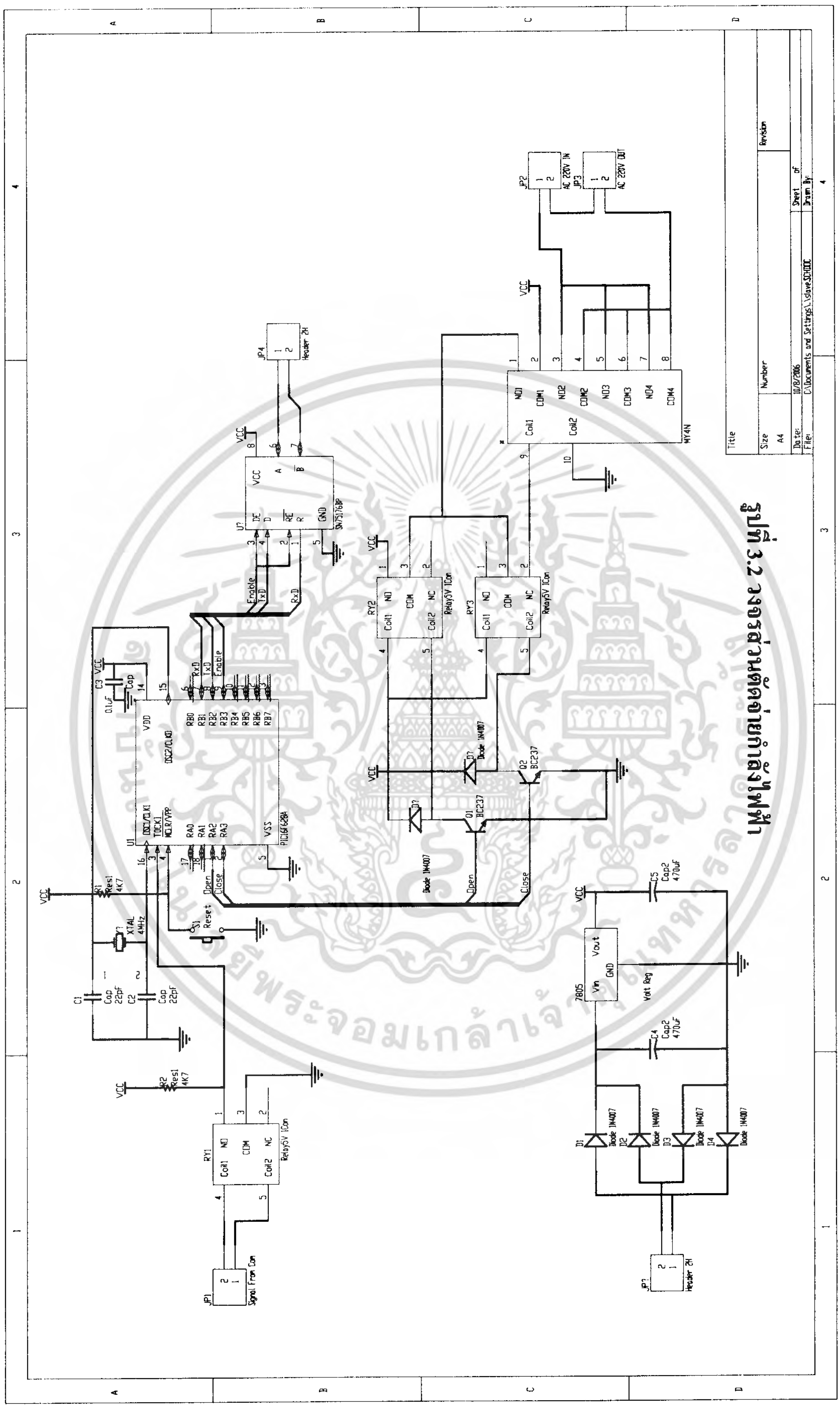
2.1) Microcontroller ตระกูล PIC เบอร์ 16F628 ซึ่งจะทำหน้าที่รับ-ส่งและประมวลผลคำสั่งจากบอร์ดหลัก

2.2) วงจรควบคุมการทำงานของ Relay ใช้ในการควบคุมการจ่ายไฟ AC 220V ด้วยไฟ DC 5 V และใช้ในการตรวจจับสัญญาณจากคอมพิวเตอร์ที่เปิดใช้งานอยู่ผ่านทางไฟ 5V ของ Port USB เพื่อเช็คดูว่ามีคอมพิวเตอร์เปิดอยู่หรือเปล่า

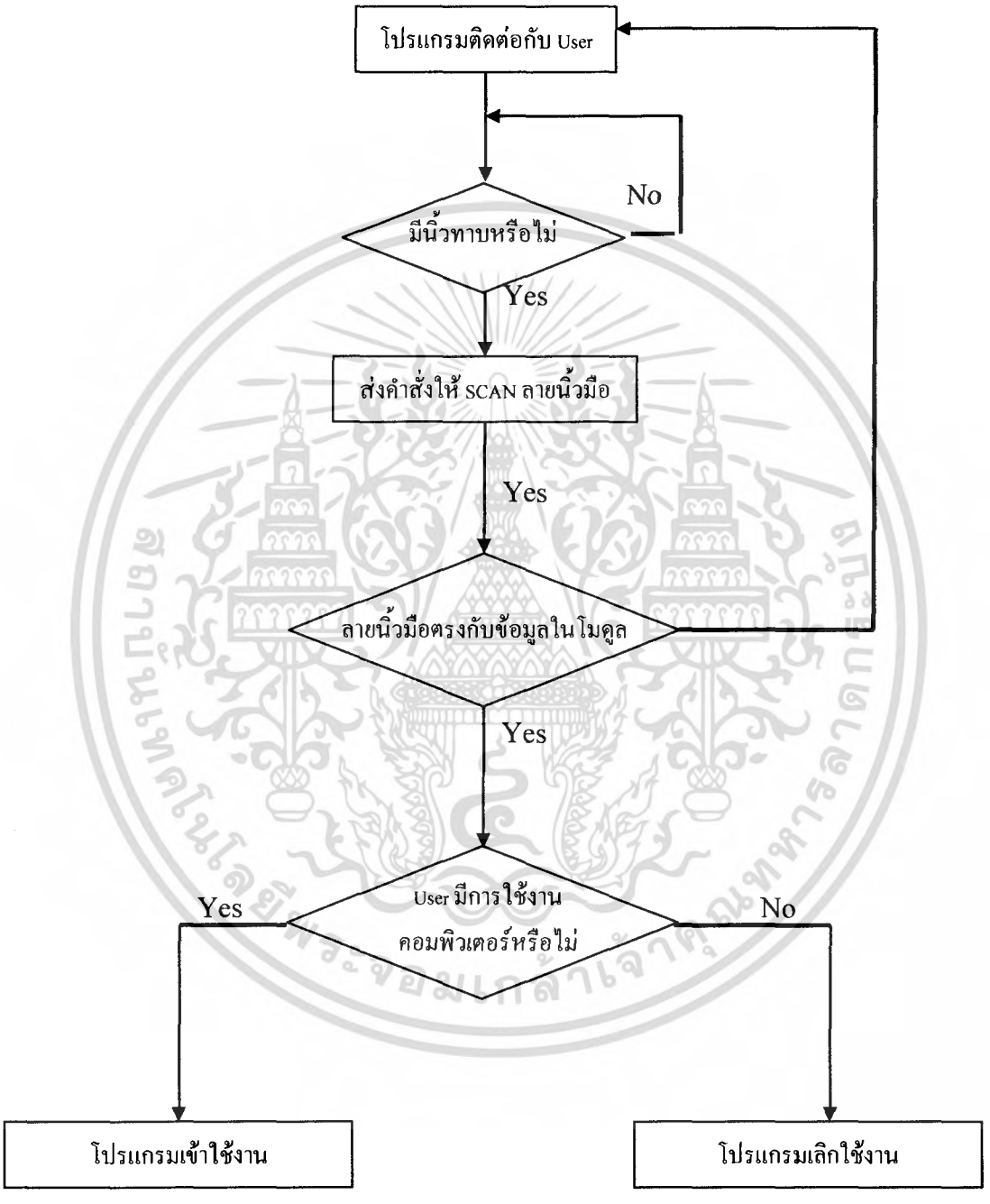
การทำงานของหน่วยตัด-จ่ายกำลังไฟฟ้านั้น จะรอคำสั่งเปิด-ปิดคอมพิวเตอร์มาจาก บอร์ดหลักและตรวจเช็คสัญญาณ 5V จาก Port USB ของคอมพิวเตอร์เพื่อตรวจเช็คว่าไม่มีการเปิดหน่วยตัด-จ่ายกำลังไฟฟ้าอยู่ เนื่องจากโดยปกติแล้วหากคอมพิวเตอร์เปิดอยู่ที่พอร์ต USB จะมีแรงดันไฟฟ้า 5V ตลอดเวลา หากมีการเปิดใช้คอมพิวเตอร์อยู่ระบบจะไม่ทำการตัด-จ่ายไฟฟ้าและส่งข้อมูลไปยังบอร์ดหลักว่าไม่สามารถเปิด-ปิดคอมพิวเตอร์เครื่องนี้ในเวลานี้ได้ หากคอมพิวเตอร์ไม่ได้เปิดอยู่ก็จะทำการ ตัด-จ่ายกำลังไฟฟ้าให้กับคอมพิวเตอร์แล้วส่งข้อมูลไปยังบอร์ดหลักว่าได้ทำการ ตัด-จ่ายกำลังไฟฟ้าให้กับคอมพิวเตอร์เรียบร้อยแล้ว



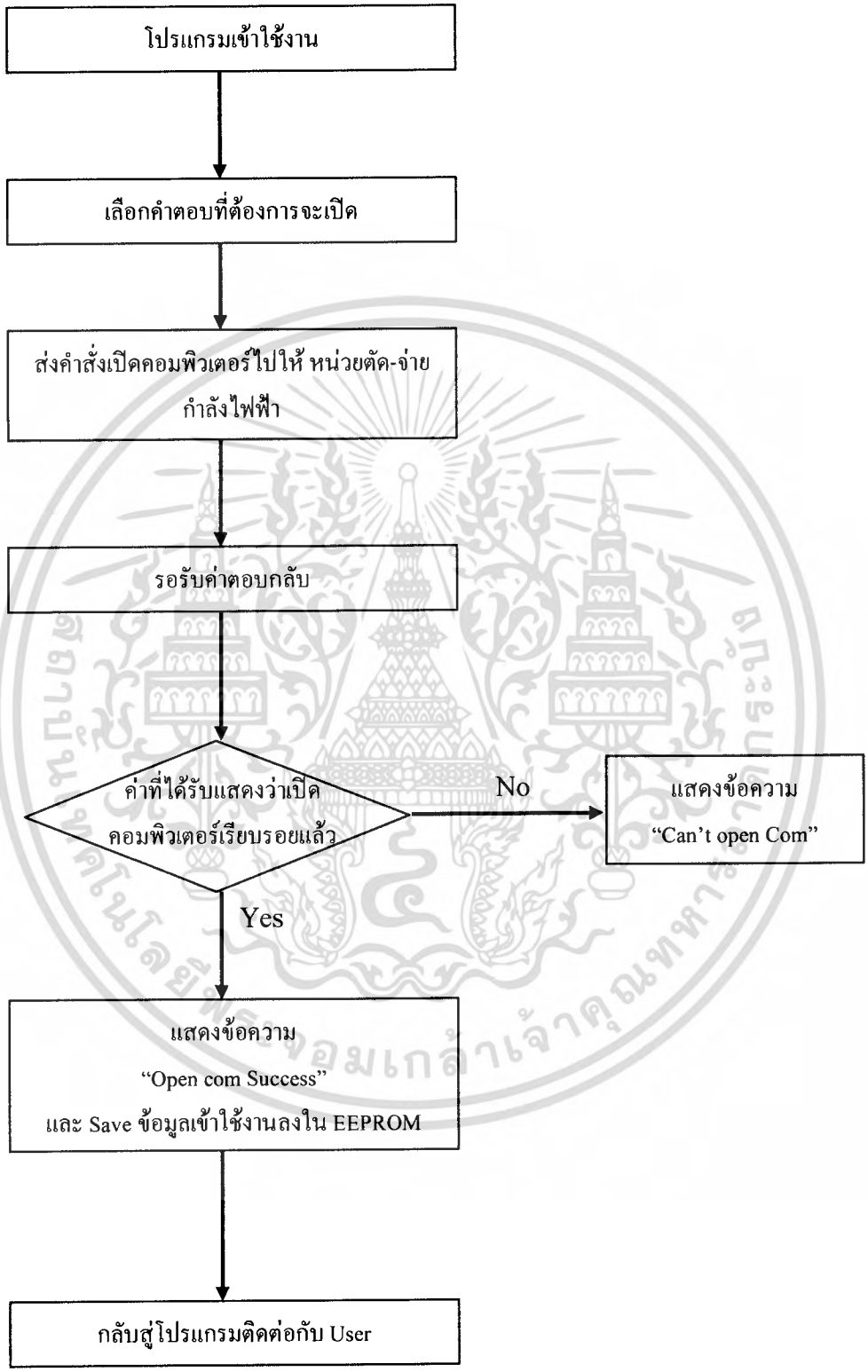
รูปที่ 3.1 วงจรส่วนของบอร์ดหลัก



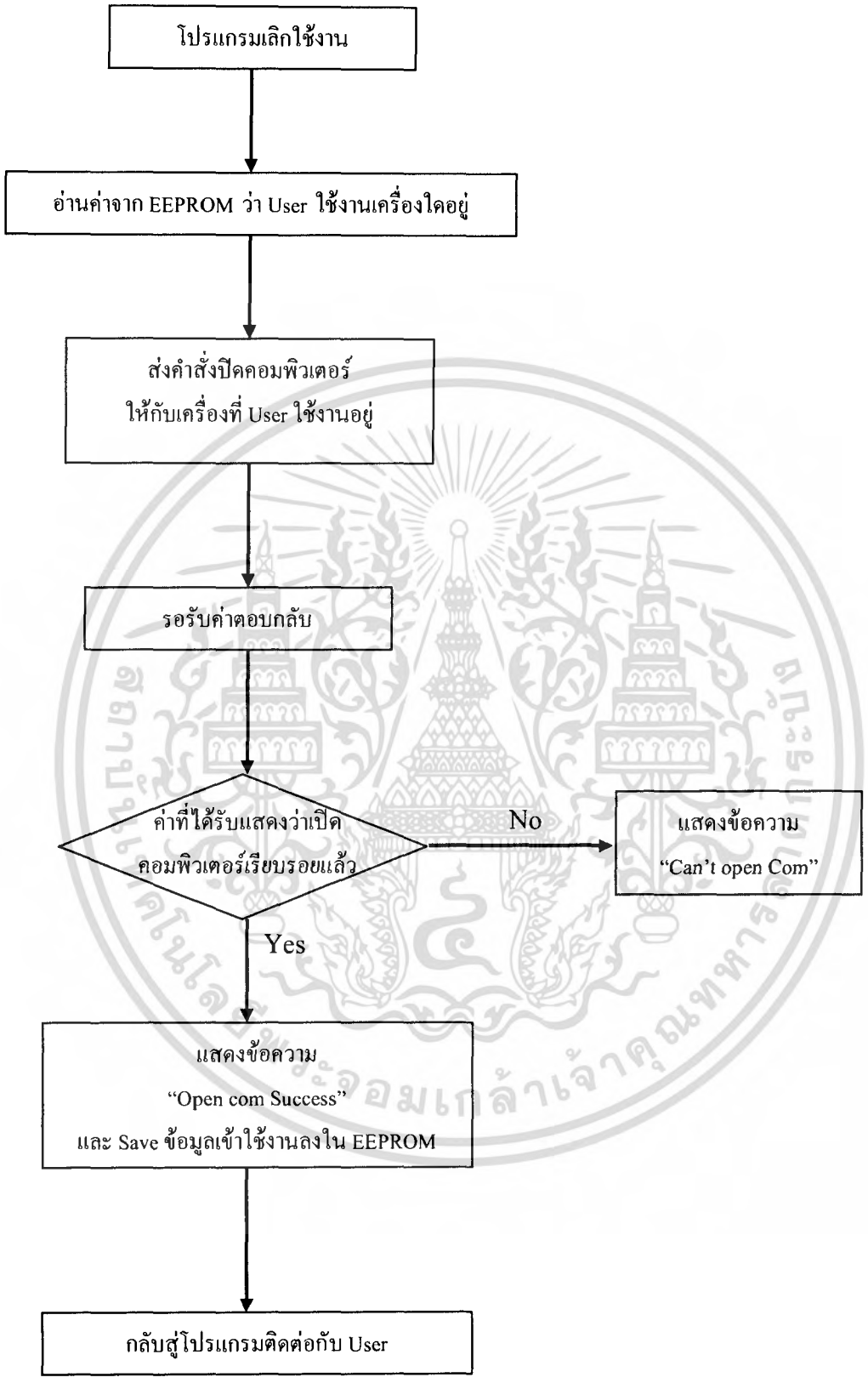
รูปที่ 3.2 วงจรส่วนตัดต่อจ่ายกำลังไฟฟ้า



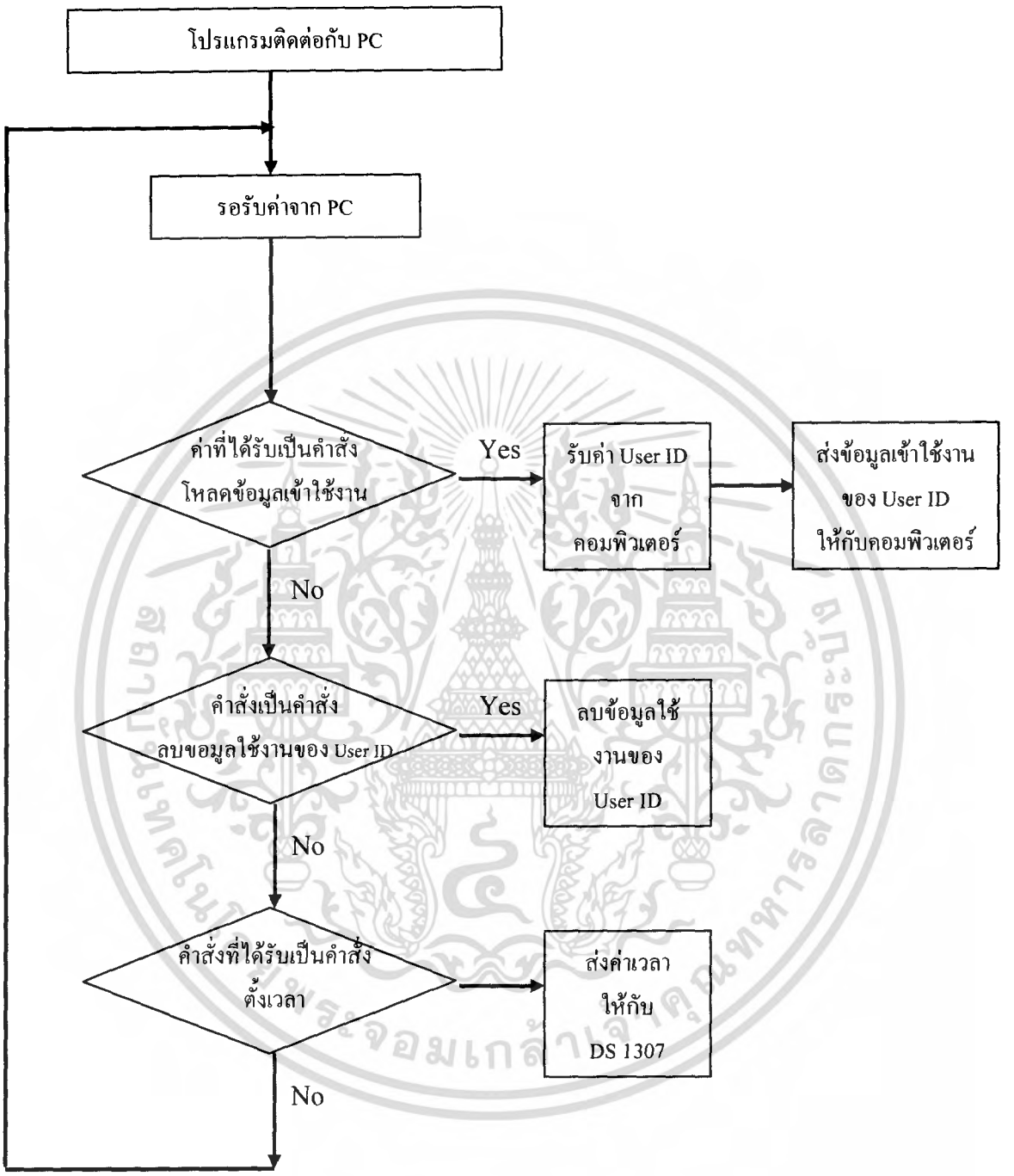
รูปที่ 3.3 แสดงแผนผังโปรแกรมติดต่อกับ User



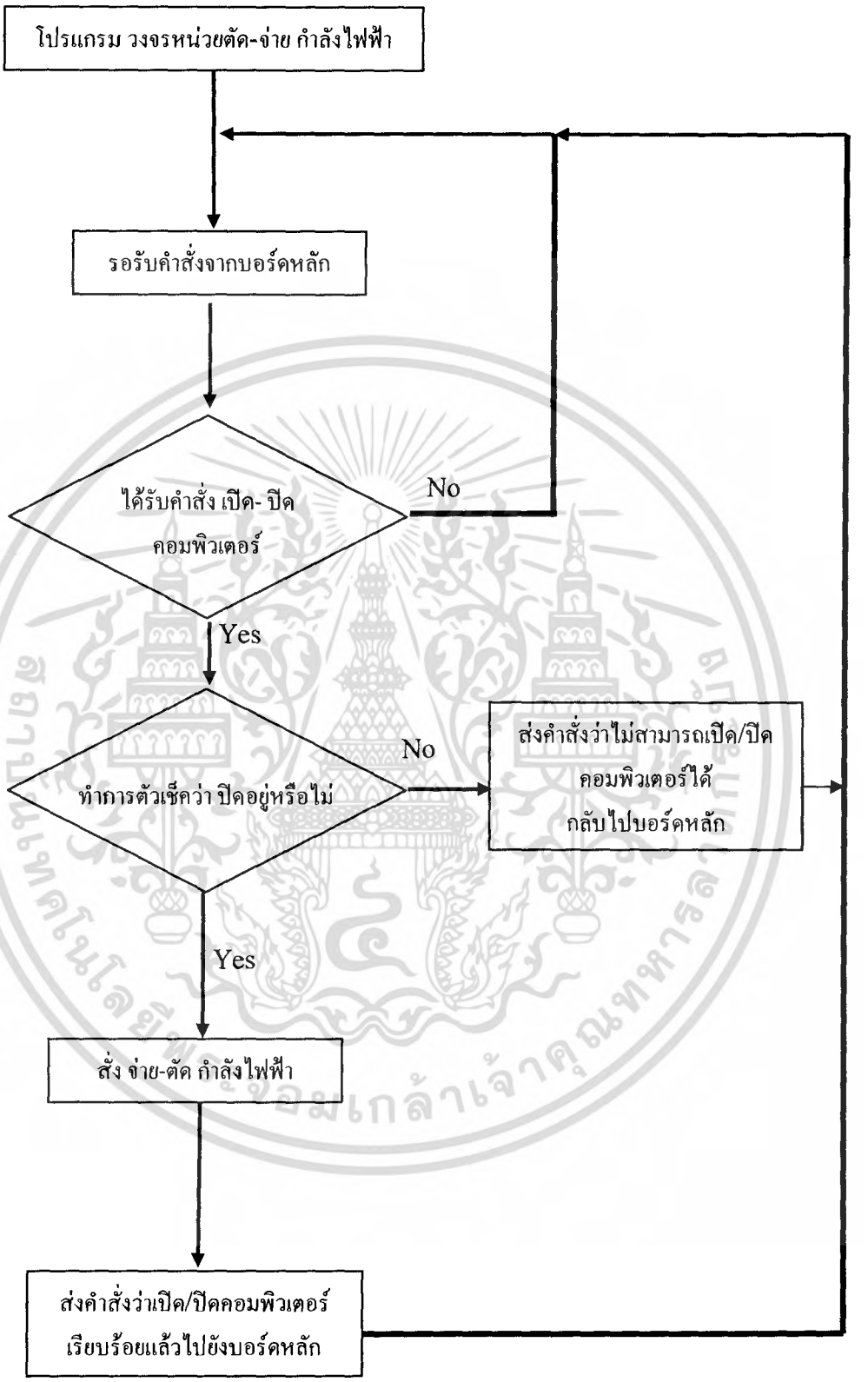
รูปที่ 3.4 แสดงผังผังโปรแกรมเข้าใช้งาน



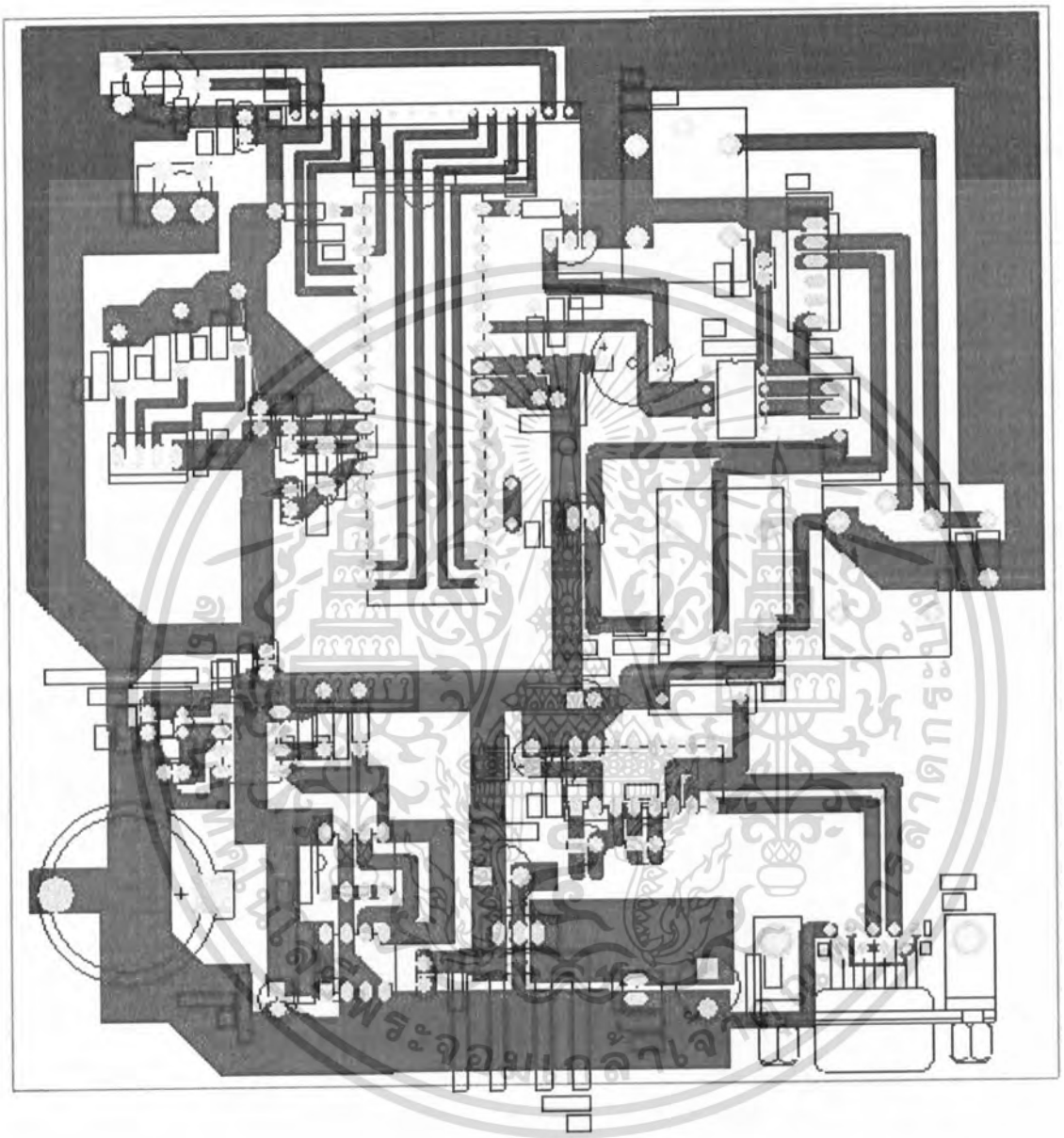
รูปที่ 3.5 แสดงแผนผังโปรแกรมเลิกใช้งาน



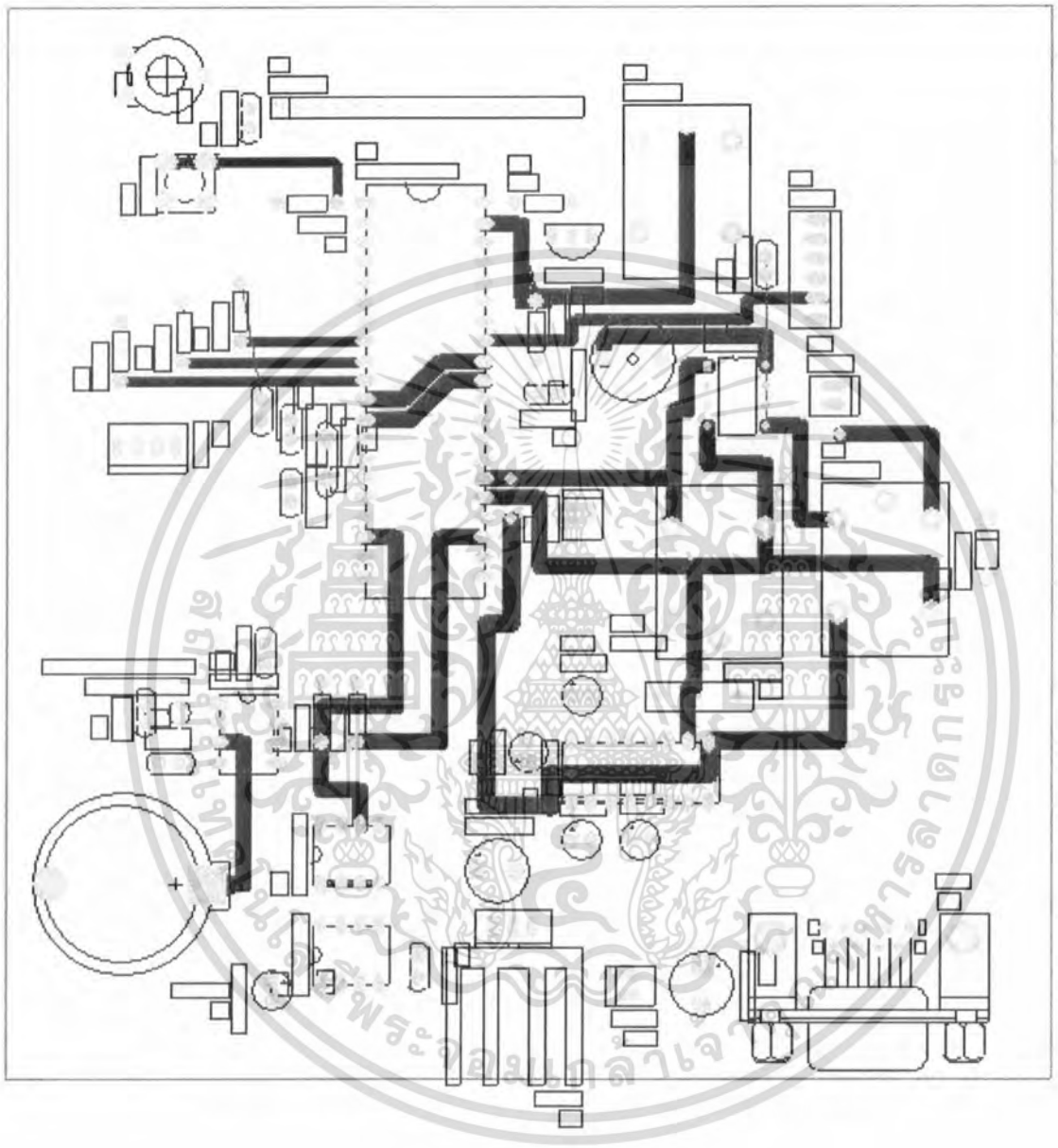
รูปที่ 3.6 แสดงแผนผังโปรแกรมติดต่อกับ PC



รูปที่ 3.7 แสดงแผนผังโปรแกรม วงจรหน่วยตัด-จ่าย กำลังไฟฟ้า



รูปที่ 3.8 ลายวงจรของบอร์ดหลักด้านล่าง



รูปที่ 3.9 ลายวงจรของบอร์ดหลักด้านบน

บทที่ 4
การทดลอง และผลการทดลอง

การทดลองได้ทำการทดลองโดยแยกเป็นส่วนของ วงจรบอร์ดหลัก ส่วนของวงจรตัดจ่ายกำลังไฟฟ้า และส่วนของภาคแสดงผลบนคอมพิวเตอร์

การทดลอง และ ผลการทดลอง

การทดลองได้ทำการทดลองโดยแบ่งเป็นส่วนต่างๆดังนี้ วงจรบอร์ดหลัก การติดต่อกับโมดูลสแกนลายนิ้วมือ หน่วยตัด-จ่ายกำลังไฟฟ้า และส่วนของโปรแกรมบนคอมพิวเตอร์

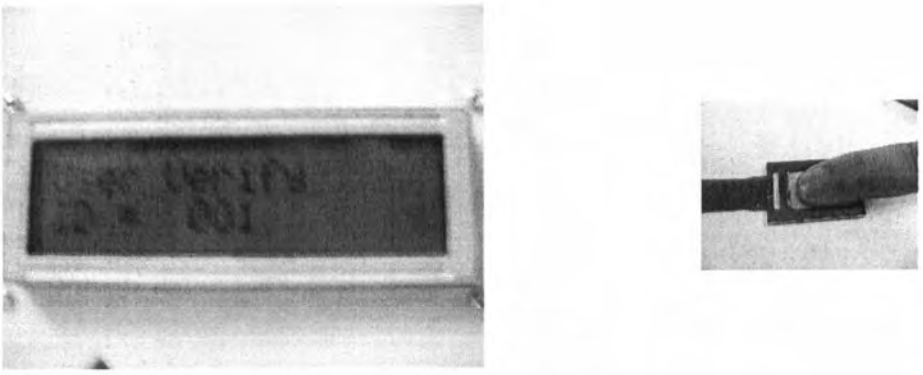
4.1 การทดลองของส่วนวงจรบอร์ดหลัก

ในการทดลองของวงจรบอร์ดหลักนั้น ได้ทำการทดลองโดยการต่อวงจรและเขียนโปรแกรมควบคุม Microcontroller ให้ทำการติดต่อกับอุปกรณ์ต่างๆคือ IC RTC DS1307 หน่วยความจำ EEPROM จอ LDC โมดูลสแกนลายนิ้วมือ และทำการทดสอบการติดต่อสื่อสารกับคอมผ่าน RS-232และทำการทดลองติดต่อกับส่วนตัดจ่ายกำลังไฟฟ้า ผ่านทาง RS-485 ด้วย

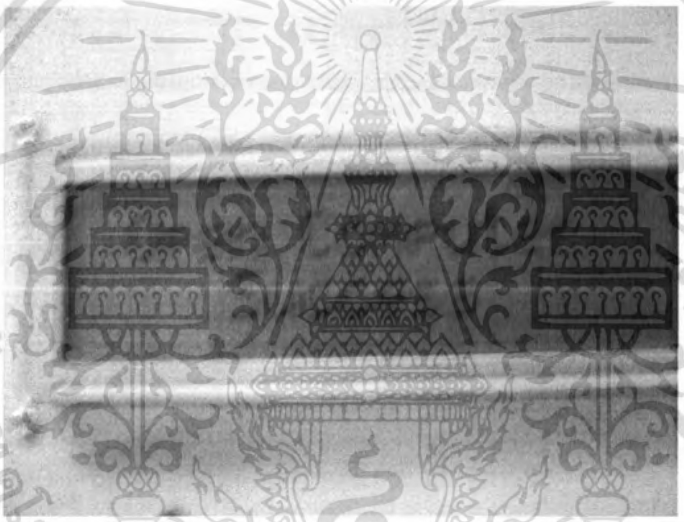
โดยมีผลการทดลองดังนี้

รูปที่ 4.1 การอ่านค่าจาก DS1307 และแสดงผลบนจอ LCD

รูปที่ 4.2 การติดต่อกับ โมดูลสแกนลายนิ้วมือเมื่อมีนิ้วมาทาบ



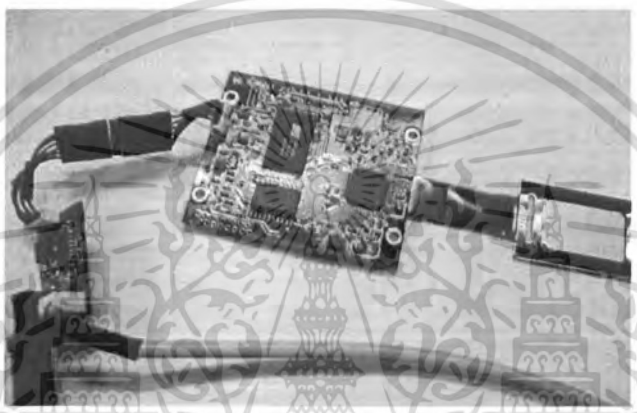
รูปที่ 4.3 แสดง การรับค่า User ID จากโมดูลสแกนลายนิ้วมือและแสดงผ่าน LCD



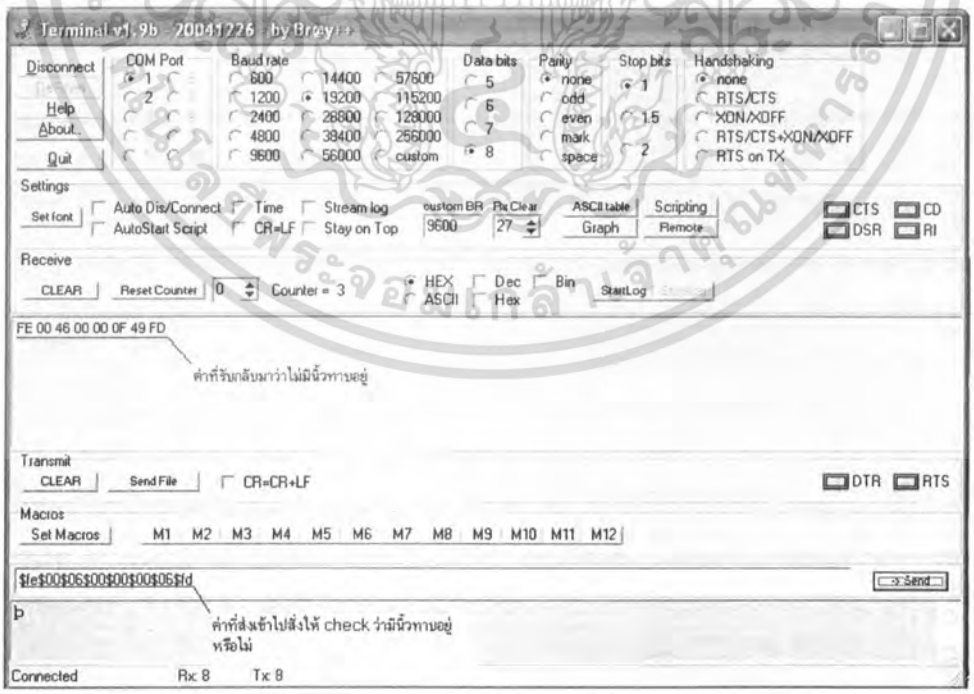
รูปที่ 4.4 การทดลองติดต่อกับหน่วยตัดจ่ายกำลังไฟฟ้า โดยส่งและรอรับค่าว่าเปิดคอมเรียบร้อยแล้ว

4.2 การทดลองการติดต่อกับโมดูลสแกนลายนิ้วมือ

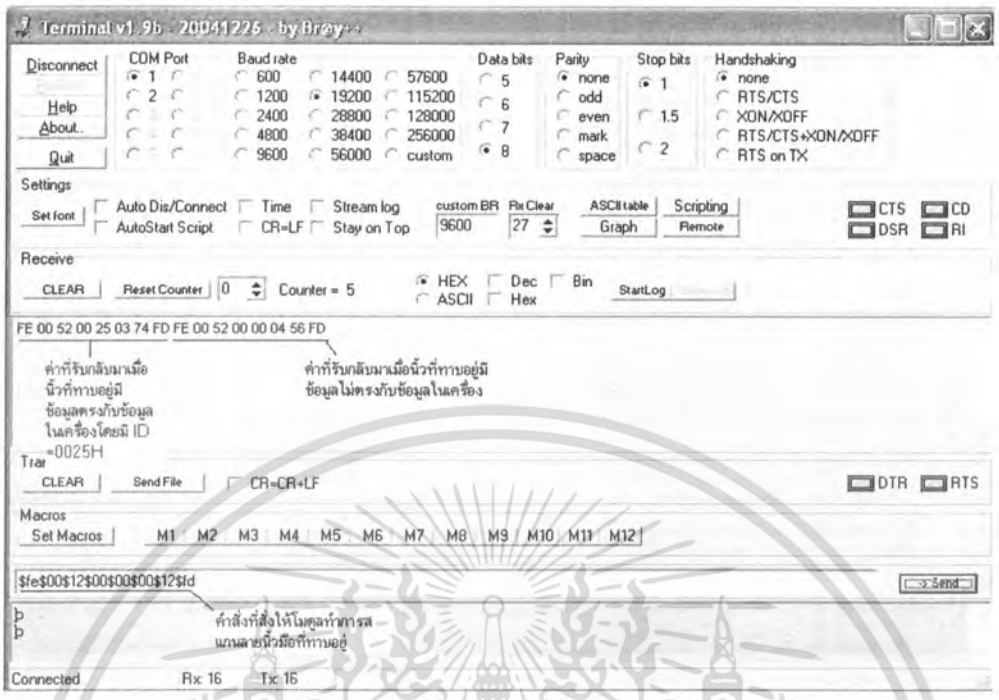
การทดลองนี้เป็นการทดลอง ส่งข้อมูลให้กับโมดูลสแกนลายนิ้วมือ โดยส่งคำสั่งในชุดคำสั่งที่ได้จาก Data sheet และดูผลที่โมดูล ตอบกลับมาด้วยคอมพิวเตอร์ผ่านทาง โปรแกรม Terminal ซึ่งเป็นโปรแกรมที่ใช้ในการส่งและรับค่าผ่านทาง port อนุกรม โดยชุดคำสั่ง จะมีขนาด 8 Byte โดยมี Byte แรก (Byte ที่ 0) เป็น Byte Start โดยจะมีค่า 0xFE และ Byte สุดท้ายเป็น Byte Stop โดยจะมีค่า 0xFD



รูปที่ 4.5 แสดงการต่อโมดูลลายนิ้วมือกับคอมผ่านพอร์ตอนุกรม



รูปที่ 4.6 แสดงค่าที่รับได้จากโมดูลเมื่อสั่งให้โมดูลตรวจว่ามี นิ้วทาบอยู่หรือไม่



รูปที่ 4.7 แสดงค่าที่ได้จากโมดูลเมื่อสั่งให้เปรียบเทียบข้อมูลลายนิ้วมือกับนิ้วที่ทาบอยู่

รูปที่ 4.8 สั่งลบข้อมูลลายนิ้วมือ

4.3 การทดลองส่วนของวงจรตัวจ่ายกำลังงานไฟฟ้า

ในการทดลองในส่วนนี้ จะใช้บอร์ดหลักในการสั่งควบคุมโดยผ่านทางมาตรฐานการส่งข้อมูลอนุกรมแบบ RS- 485 ซึ่งใช้สายสัญญาณเพียง 2 เส้น และมีลักษณะเด่นคือระยะทางของสายนั้นใช้ได้ยาวมากหลายร้อยฟุต ทั้งยังสามารถรับจำนวนโหลดได้มากอีกด้วย

ในส่วนของการทดลองได้ทำการทดลองดังนี้คือ

- การทดลองโดยการให้บอร์ดหลักสั่งการ Relay ให้จ่ายกระแสไฟ โดยในการทดลองนี้ส่วนของตัวตัดจ่ายกำลังไฟฟ้าจะคอยรับข้อมูลจากบอร์ดหลักผ่านทาง RS-485 ซึ่งถ้าข้อมูลที่ได้รับตรงกับของตัวจ่ายกำลังงานไฟฟ้าและเป็นสัญญาณสั่งให้เปิดคอม ก็จะทำการเช็คเงื่อนไขในส่วนของสัญญาณจากคอมว่าคอมปิดอยู่หรือเปล่า(เป็นสัญญาณ 5VDC จาก Port USB) ถ้าเงื่อนไขเรียบร้อยดีก็จะทำการสั่งให้ Relay On เพื่อจะนำไปควบคุมการจ่ายไฟ 220VAC ต่อไป หากมีคำสั่งให้ปิดคอมก็จะทำเช่นเดียวกันแต่จะเป็นการสั่งให้ Relay Off

รูปที่ 4.9 แสดงส่วนของวงจรตัดจ่ายกำลังไฟฟ้าเมื่อมีคำสั่งเปิดคอม

รูปที่ 4.10 แสดงส่วนของวงจรตัดจ่ายกำลังไฟฟ้าเมื่อมีคำสั่งปิดคอม

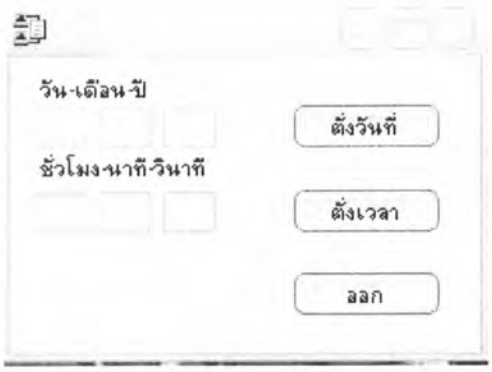
4.4 การทดลองในส่วนของภาคแสดงผลบนคอมพิวเตอร์

ตัวโปรแกรมใช้งานบนวินโดว์ สร้างขึ้นโดยใช้โปรแกรม Visual Basic โดยมีหน้าหลักสำหรับดูข้อมูลการใช้งา น และสามารถ Lock in เพื่อแก้ไข และเพิ่มรายชื่อ สมาชิกผู้เข้าใช้งาน อีกทั้งยังสามารถตั้งเวลาให้กับโมดูล ได้อีกด้วย

รูปที่ 4.11 แสดงหน้าต่างหลักของโปรแกรมโดยสามารถ เข้าไปยังหน้าต่าง ของโปรแกรมได้ทุกหน้าต่างและเมื่อใส่รหัสผ่าน ก็จะสามารถ เข้าไปยังหน้าต่างแก้ไขและ เพิ่มรายชื่อได้

รูปที่ 4.12 แสดงหน้าต่าง ดูข้อมูลการเข้าใช้ของนักศึกษาโดยสามารถแสดงแบบได้ ดังนี้

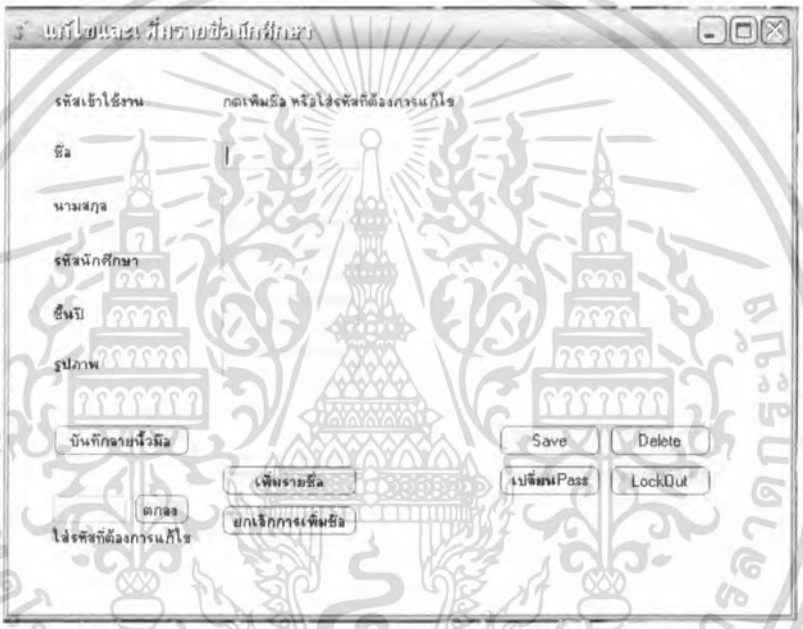
1. **ตามรหัสเข้าใช้งาน โดยการ คลิกที่รหัส แล้วใส่หมายเลข ตามด้วยกดแสดง**
2. **ตามเครื่องที่เข้าใช้ โดยการ คลิกที่เครื่อง แล้วใส่หมายเลข ตามด้วยกดแสดง**
3. **แสดงทั้งหมด โดยการ คลิกที่แสดงทั้งหมด แล้วกดแสดง**



รูปที่ 4.13 แสดงหน้าต่าง การตั้งเวลาให้อุปกรณ์

รูปที่ 4.14 แสดงหน้าต่างการ UPDATE ข้อมูลจากอุปกรณ์กดเริ่มเพื่อทำการดึงข้องมูลจากอุปกรณ์ แล้วโปรแกรมจะทำการบันทึกลงคอมพิวเตอร์ และลบข้อมูลในอุปกรณ์ เอง โดยอัตโนมัติ

รูปที่ 4.15 แสดงหน้าต่างบันทึกลายนิ้วมือ
กดเริ่มเพื่อทำการบันทึกลายนิ้วมือ



รูปที่ 4.16 แสดงหน้าต่างแก้ไขและเพิ่มรายชื่อเข้าให้งาน
กดเพิ่มลายชื่อและพิมพ์ข้อมูลแล้วทำการSaveก็จะเก็บข้อมูลของนักศึกษาเรียบร้อยแล้ว
และยังสามารถแก้ไขข้อมูลได้โดย ใส่รหัสแล้วกดตกลง ก็จะแสดงข้อมูลออกมาให้แก้ไขแล้ว

รูปที่ 4.17 แสดงหน้าต่าง การลบลายนิ้วมือทั้งหมด
จะต้องใส่รหัสผ่านแล้วกดตกลงจึงจะสามารถลบลายนิ้วมือได้

บทที่ 5
บทสรุป วิจารณ์ ปัญหาที่พบ และการพัฒนา

บทสรุป

จากการศึกษาและทดลองลงมือปฏิบัติโครงงานระบบควบคุมการเข้าออกด้วยลายนิ้วมือ (Access Control by Finger Print) ในภาคเรียนที่ 2 ปีการศึกษา 2549 พบว่า การดำเนินงานคืบหน้าไปด้วยดี สามารถออกแบบกล่องควบคุมให้สามารถใช้งานได้ง่าย อีกทั้งยังสามารถบำรุงรักษาได้อย่างสะดวก ลังมีการพัฒนาในด้านโปรแกรมการใช้งานให้ผู้ใช้สามารถเรียกใช้งานและเรียกดูข้อมูลต่างๆที่จำเป็นได้โดยง่าย อีกทั้งยังสามารถแสดงผลและเก็บค่าเป็นฐานข้อมูลได้อีกด้วย

จากการปฏิบัติงานจริงในโครงงานนี้ทำให้นักศึกษาได้ทราบถึงปัญหาต่างๆที่เกิดขึ้นจากการลงมือปฏิบัติ รวมทั้งเป็นการฝึกการแก้ปัญหา ซึ่งจะทำให้สามารถนำไปประยุกต์ใช้กับชีวิตจริงและหน้าที่การงานต่อไป

บทวิจารณ์

ในระบบของเรานั้นได้นำไอซีฐานเวลาจริง (Real Time Clock) และหน่วยความจำแบบ EEPROM มาใช้งานด้วย ทำให้ระบบของเราสามารถทราบ และตรวจสอได้ภายหลังว่ามีใครเข้ามาใช้งานคอมพิวเตอร์เมื่อใด เครื่องไหน ทำให้ระบบของเรามีความสามารถที่จะตรวจเช็ค และสามารถทราบถึงผู้ที่เข้ามาใช้งานแล้วไม่ทำตามกฎระเบียบ อันพึงจะเกิดความเสียหายแก่ทรัพย์สิน

ปัญหาที่พบ

ในการสแกนลายนิ้วมือนั้นจำเป็นต้องวางนิ้วมือให้แม่นยำและได้มุมที่ถูกต้อง ไม่เช่นนั้นในบางครั้งโมดูลสแกนลายนิ้วมืออาจมองเห็นว่าเป็นคนละลายนิ้วมือได้ทั้งที่เป็นลายนิ้วมือเดียวกัน

แนวทางการพัฒนาต่อ

ในอนาคตเมื่อนำไปติดตั้งใช้งานจริง อาจจำเป็นต้องเพิ่มขนาดของหน่วยความจำเพื่อรองรับการเก็บลายนิ้วมือเพิ่มขึ้น ทำให้สามารถรองรับจำนวนผู้ใช้งานได้มากขึ้นด้วย

เอกสารอ้างอิง

1. ฉันทวุฒิ พืชผล และพิชิต สันติรุจานนท์ " คู่มือเรียน Visual Basic 6 " ซีเอ็ดยูเคชั่น ,560 หน้า, พิมพ์ครั้งที่ 11 , 2547
2. อภิชาติ ภู่พลับ " เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาร์ดแวร์ด้วย Visual Basic" บริษัท "Dev Book", 240 หน้า , 2546

ภาคผนวก ก.

การใช้งานเครื่องแสกนลายนิ้วมือ

การใช้งานเครื่องแสกนลายนิ้วมือ สามารถทำได้โดยเชื่อมต่อส่วนของบอร์ดหลักเข้ากับเครื่องคอมพิวเตอร์เพื่อทำการลงทะเบียน บันทึกลายนิ้วมือ โดยทำการเลื่อนสวิซส์ด้านข้างลงเพื่อเข้าสู่โหมดการ อัปโหลดข้อมูล จากนั้นก็สามารถใช้งานเครื่องแสกนลายนิ้วมือได้ตามปกติ

โหมดการลงทะเบียนลายนิ้วมือ (Enrollment)

ในโหมดการทำงานนี้เป็นโหมดการทำงานเพื่อทำการลงทะเบียน โดยการให้ผู้ใช้งานกรอกประวัติ ลงตามแบบฟอร์มที่กำหนด หลังจากนั้นทำการบันทึกลายนิ้วมือ เพื่อจัดเก็บข้อมูลขอผู้ใช้งาน

โหมดการเข้าใช้งานเครื่องคอมพิวเตอร์(Access)

เมื่อผู้ใช้งานต้องการจะทำการ Login เข้าใช้งานก็สามารถทำได้โดยการทาบนิ้วมือลงบนกล่องควบคุม จากนั้น ทำการเลือกเครื่องคอมพิวเตอร์ที่ต้องการใช้งาน จากนั้นก็สามารถใช้งานเครื่องคอมพิวเตอร์ได้ตามปกติ

โหมดการเลิกใช้งานเครื่องคอมพิวเตอร์ (Exit)

เมื่อผู้ใช้งาน ต้องการจะ Logout เพื่อยกเลิกการใช้งาน ต้องทำการ Turn Off เครื่องคอมพิวเตอร์ก่อนทุกครั้ง เพื่อเป็นการบอกถึงสภาวะที่ต้องการเลิกใช้งานแล้ว หลังจากนั้นทำการ Logout โดยการทาบนิ้วมือลงบนกล่องควบคุมอีกครั้ง เป็นอันเสร็จสิ้นกระบวนการใช้งาน

โหมดการโอนถ่ายข้อมูลไปสู่เครื่องคอมพิวเตอร์ (Upload To Com)

เมื่อต้องการติดต่อ เพื่อดึงข้อมูลการเข้าใช้งานจากเครื่องแสกนลายนิ้วมือ โดยการติดต่อกล่องควบคุมกับเครื่องคอมพิวเตอร์ที่มีโปรแกรมการใช้งาน พร้อมทั้งเลื่อนสวิซต์ไปยังตำแหน่ง อัปโหลด ข้อมูล จากนั้น ก็สามารถเรียกดูข้อมูลต่างๆได้ตามปกติ

ภาคผนวก ข.

MRB200 FINGERPRINT MODULE FOR G6103 DEVELOPER'S MANUAL

V1.0.0.2

Editor by Zack

2004-10-17

Chapter 1 Introduction

Fingerprint module is the latest module for G6102. Adopting encryption techniques and safety measure for IC card, this system encrypts the data to ensure the security of data and the validity of card, which assures the validity of cardholder and supports vary touch and no touch card such as memory card, and CPU card, including contact and contactless card.

Products application

This system can be applied to the stored information on IC card and contactless card such as fingerprint temporary residence permit, fingerprint ID card, fingerprint admission, traffic duty, social security, driver school, fingerprint finance card, electronic wallet and some other system which need to authenticate the card offline. Together with the perfect technical support and product upgrading service, we can ensure the benefit for our development partner and system integrator.

Features

- It is a portable device can be operated in an easy way, which results from its integrative structure design. So it is fit for the mobile case especially.
- It is compatible with multi-protocol for smart cards. For example, it can operate the card compatible with ISO7816, ISO14443-A/B or ISO15693.
- Improve the safety by SAM card key management method, and it expands the scope of the device, makes the update easier and safer.
- Open developing platform supports redevelopment for different application.
- Footprint identification function is realized by the latest independent footprint identification module of our corporation, which has the advanced dermal detecting function.
- Low power consumption but long halt time results from its power saving design. As a portable terminal device, it is an important feature.
- A Li-ion battery is used, which can be recharged.
- There is a large storage for identification system, and it can be extended according to user's need.

Technical Parameters

Footprint Compare Time	<0.01second
Compare Method	1:1 , 1:N
Storage Capacity	More than 800
Error footprints pass rate	<0.01‰~0.001‰
Right footprints are rejected rate	<0.1%~0.01%

Chapter 2 How to Develop It

Before you are going to develop the application for G6102 with fingerprint module, you should master the basic way for G6102's development. Fingerprint module is a daughter board for G6102, as a appendant device. It communicates with G6102 by a serial port. No library functions for you to operate fingerprint module, and you just send some commands to operate it, which will give you a introduction in the next chapter. While in the last chapter, we will give you some useful functions to help you to know how to use the fingerprint module smoothly.

At the following chapters, we just give you an introduction about fingerprint module. If you want to know how to use the basic G6102 and contactless module's library functions, please refer to other documents. Further questions please contact with our technical support department.

Chapter 3 Commands for Fingerprint Module

MRB200 fingerprint module is the daughter board for G6102 in fact. It communicates with G6102's main board by asynchronous serial port. Main board sends all kinds of commands (CMD) to fingerprint module to operate it, and fingerprint module will apply the command and answer to main board the result of the operation (ACK).

The parameters for the serial port communication between them show as following:

- 19200bps (Default);
- No checksum;
- One starting bit;
- One end bit;

A commands (CMD) consists of 8 bytes or more bytes. So there are two types of command format. The one is a group of basic commands consist of 8 bytes, the other is a group of commands consist of more than 8 bytes and the first byte would be 0x3X.

Basic commands format:

The first byte is the head byte, and it must be 0xFE.

The second byte is the device number, and it would be 0x00 normally.

The third byte is the command code.

The fourth and fifth bytes are parameter code: P1, P2.

The sixth byte is assistant parameter code: P3.

The seventh byte is check sum, which used to save the ^ (Bit operation symbol) value from the second byte to the sixth byte.

The eighth byte is the end byte, and it must be 0xFD.

Answer (ACK) consists of 8 bytes or more bytes normally, and the format is described as following:

The first byte is the head byte, and it must be 0xFE.

The second byte is the device number, and it would be 0x00 normally.

The third byte is the answer code.

The fourth and fifth bytes are parameter code: P1, P2.

The sixth byte is the answer parameter code AP.

The seventh byte is check sum, which used to save the ^ value from the second byte to the sixth byte.

The eighth byte is the end byte, and it must be 0xFD.

3.1 User Power

The database of the MRB200 fingerprint module supports three level user power. That is, administrator, common user and temporary user. It will give you a convenience way to manage the users. The developer can give the different power to different user according to his requests.

For example, if MRB200 module is used to a fingerprint identify lock, it can give different power to users. Common users can open the door, unlock and so on. The administrator can modify, search and delete information from the fingerprint library except the common operation.

MRB200 fingerprint module will set a default administrator for the first user if the fingerprint library is blank.

1 Basic Commands

1.1 CMD_GET_USER_SUM_DB

- Function

  Query the total fingerprint's number in the module.

- CMD Parameters

  CODE = 0x05、P1 = 0x00、P2 = 0x00、P3 = 0x00

- ACK Parameters

  HEAD + CH + 0x45 + SUM_H + SUM_L + AP + CHK + END

  SUM = The total number of registered fingerprint. The number will be saved as a hex format and use two bytes. The fourth byte will be high byte and the fifth byte will be low byte.

  AP must equal EW_SUCCESS, which is defined by a head file.

1.2 CMD_GET_USER_RIGHT_DB

- Function

  Send user ID to confirm the user's power.

- CMD Parameters

  CODE = 0x00、P1 = UserID_H、P2 = UserID_L、P3 = 0x00

  NOTE: [P1,P2] is the user ID number, P1 is high byte, P2 is low byte. For example, user ID is 0x1234, so P1 = 0x12, P2 = 0x34.

- ACK Parameters

  HEAD + CH + 0x40 + 0x00+ 0x00 +AP + CHK + END

  AP = User power. If:

  - AP=EW_NO_USER　　User is not occurred in the library.
  - AP=EW_GUEST_USER　　User is a temporary user.
  - AP=EW_NORMAL_USER　　User is a common user.
  - AP=EW_MASTER_USER　　User is the administrator.

1.3 CMD_VERIFY_DB

- Function

  Get the user's fingerprint and compare it with the fingerprint selected by command in the fingerprint library. It will confirm if it is the same fingerprint. User must put the finger on the fingerprint collection in five seconds after he send the command, or the module will return the timeout error.

- CMD Parameters

  CODE = 0x01、P1 = UserID_H、P2 = UserID_L、P3 = 0x00

- ACK Parameters

  HEAD + CH + 0x41 + 0x00 + 0x00 + AP + CHK + END

  AP = Return Value. If:

  - AP=EW_TIME_OUT Timeout error.
  - AP=EW_SUCCESS Compare successfully.
  - AP=EW_FAIL Unknown error.
  - EW_FAIL_BMF Collect fingerprint failed.
  - EW_FAIL_FEA Get eigenvalue failed.
  - EW_FAIL_MATCH Compare failed.
  - EW_NO_USER User is not occurred.

1.4 CMD_REG_START_DB

- Function

  MRB200 require user input fingerprint for three times to ensure the exact collection in the fingerprint register process. This command will finish the first collection.

- CMD Parameters

  CODE = 0x02、P1 = UserID_H、P2 = UserID_L、P3 = User Power

  NOTE: UserID != 0

- ACK Parameters

  HEAD + CH + 0x42 + 0x00 + 0x00 + AP + CHK + END

  AP = Return information. If:

  - AP = EW_TIME_OUT Collection timeout error. User must put his finger on the fingerprint collection in five seconds after command is send, or it will return timeout error.
  - AP=EW_SUCCESS Success. It should sent CMD_REG_SECOND_DB right now for the second collection.
  - EW_FAIL Unknown error.
  - EW_FAIL_ID The user ID should not be 0. This error probably is made by UserID = 0.
  - EW_FAIL_FEA The module can't get eigenvalue from the collection fingerprint. The possible reason is the collection is not a fingerprint or the fingerprint is collected is bad.
  - EW_FAIL_BMF Collection failed.
  - EW_FULL The capacity of the fingerprint's library is full.

1.5 CMD_REG_SECOND_DB

- Function

The second fingerprint collection in the fingerprint register process. It should be send after the command CMD_REG_START_DB. A timeout error will be send if user couldn't put his finger on the fingerprint collection in five seconds after the command is send.

If the last command is not CMD_REG_START_DB, it will return register failed right now.

If the user ID or power is not same as the corresponding CMD_REG_START_DB command's parameters, it will use the parameters in the command CMD_REG_START_DB.

- CMD Parameters

CODE = 0x04、P1 = UserID_H、P2 = UserID_L、P3 = User power

- ACK Parameters

HEAD + CH + 0x44 + 0x00 + 0x00 + AP + CHK + END

AP = Return value. If :

- EW_TIME_OUT Timeout, while it can send this command again.
- EW_SUCCESS Success. It should send command CMD_REG_END_DB right now.
- EW_FAIL Register failed. And if you want to collect a fingerprint again. You have to send command CMD_REG_START_DB again.
- EW_FAIL_REG Last command is not CMD_REG_START_DB
- EW_FAIL_FEA Get eigenvalue failed
- CMD_FAIL_MATCH Fingerprint is not match between this one and last one.

1.6 CMD_REG_END_DB

- Function

The third fingerprint collection in the fingerprint register process. It should be send after the command CMD_REG_SECOND_DB. A timeout error will be send if user couldn't put his finger on the fingerprint collection in five seconds after the command is send. Module will compare the fingerprint collect this time and the ones of last two times. If they are the same, it will save it into the fingerprint library, and then return a successful sign.

If the last command is not CMD_REG_SECOND_DB, it will return register failed right now.

If the user ID or power is not same as the corresponding CMD_REG_START_DB command's parameters, it will use the parameters in the command CMD_REG_START_DB

- CMD Parameters

  CODE = 0x03、P1 = UserID_H、P2 = UserID_L、P3 = User Power

- ACK Parameters

  HEAD + CH + 0x43 + 0x00+ 0x00 + AP + CHK + END

  AP = Return value. If:

  - EW_TIME_OUT Timeout, while it can send this command again.
  - EW_SUCCESS Success.
  - EW_FAIL Register failed. While it can send this command again
  - EW_FAIL_REG Last command is not CMD_REG_SECOND_DB
  - EW_FAIL_FEA Get eigenvalue failed
  - CMD_FAIL_MATCH Fingerprint is not match between this one and last one.
  - CMD_FAIL_FLASH Save fingerprint information failed.

1.7 CMD_REG_DELETE_DB

- Function

  Delete the user specified by user number and return the result.

  [NOTE] Developers should think about the problem of user's power, because this command will change the fingerprint database. Developers should check the user's power before this command is send in their program.

- CMD Parameters

  CODE = 0x20、P1 = UserID_H、P2 = UserID_L、P3 = 0x00

- ACK Parameters

  HEAD + CH + 0x60 + 0x00 + 0x00 + AP + CHK + END

  AP = Return Value. Always return:

  - EW_SUCCESS Delete successfully. The return value just ensure the user is not existed when ACK is returned, while it's not sure if the user has registered.

1.8 CMD_REG_ALLDEL_DB

- Function

  Delete the all users that their power is P3, then return the result of the value.

  [NOTE] Developers should think about the problem of user's power, because this command will change the fingerprint database. Developers should check the user's power before this command is send in their program.

- CMD Parameters

  CODE = 0x21、P1 = 0x00、P2 = 0x00、P3 = User Power

- P3 = EW_ALL_USER All user;
- P3 = EW_GUEST_USER Temporary user;
- P3 = EW_NORMAL_USER Common user;
- P3 = EW_MASTER_USER Administrator;

- ACK Parameters

HEAD + CH + 0x61 + 0x00 + 0x00 + AP + CHK + END

AP = Return value. It always returns:

- AP = EW_SUCCESS Delete successfully. The return value just ensure the user is not existed when ACK is returned, while it's not sure if the user has registered

1.9 CMD_GET_USER_NUMBER_DB

- Function

Get P3(It is a number parameter) users' number and their power level from the user pointed by P1, P2 in the fingerprint library.

- CMD Parameters

CODE = 0x10、[P1, P2] = Get from which user、P3 = Want to get how many users

For example: Return five users' number and their power level from No. 10 user. It should be:

P1=0x00, P2=0x10. P3=0x05;

NOTE: P3 should not more than 30. If P3 is more than 30, module will let it equal 30 forcedly.

- ACK Parameters

HEAD + CH + 0x50 + 0x00 + Length + AP + CHK + END

Length: The total length of the data including users' number and their power level.

AP = EW_SUCCESS It is a fixed value

And then, module will send the data including users' number and their power level and other three bytes code, whose format is listed as following:

HEAD +Data (Length bytes) + CHK + END

Three bytes store one user's number and his level. For example, the second, third and fourth bytes store the first user's information. The fifth, sixth and seventh bytes store the second user's information, and so on until the No. P3 user.

1.10 CMD_GET_VALUE

- Function

Collect a fingerprint and pick up the fingerprint eigenvalue. User must put his finger onto the collecting unit in five seconds after this command is sent, or the module will return a

TIMEOUT error. After module finishes picking up the fingerprint eigenvalue, it will return ACK, and then return the fingerprint eigenvalue.

- CMD Parameters

  CODE = 0x26、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x66 + Length_H + Length_L + AP + CHK + END

  Length = [Length_H, Length_L] is the length of eigenvalue.

  AP = Status. If:

  - AP = EW_TIME_OUT Timeout error.
  - AP = EW_SUCCESS Success.
  - AP = EW_FAIL Failed.

  (1) EW_FAIL_BMF Collect failed (2) EW_FAIL_FEA Fail to pick up the eigenvalue

  (3) EW_FAIL_MATCH, Compare failed.

  If it is successful, the module will send the fingerprint eigenvalue, whose total length Length + 3.

  HEAD + Fingerprint eigenvalue data(Length bytes) + CHK + END

1.11 CMD_IDENTIFY_DB

- Function

  Collect a fingerprint, and compare with the all of the fingerprint in the fingerprint library, then return the result and user's number. User must put his finger onto the collecting unit in five seconds after this command is sent, or the module will return a TIMEOUT error.

- CMD Parameters

  CODE = 0x12、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x52 + P1 + P2 + AP + CHK + END

  The fourth and fifth bytes [P1, P2] = User number. For example, the user's number is 0x1234, so P1=0x12, P2=0x34.

  The sixth byte AP = User power level. If:

  - AP = EW_NO_USER The user is not occured
  - AP = EW_TIME_OUT Timeout error
  - AP = EW_GUEST_USER Temporary user
  - AP = EW_NORMAL_USER Common user
  - AP = EW_MASTER_USER Administrator

1.12 CMD_FROM_VALUE_DB

- Function

Send a fingerprint eigenvalue and user's number to the module. The module will save the eigenvalue to the fingerprint library according to the user's number and return the processing result and user's number.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. Then the eigenvalue is send, the module saves it according to the user's number pointed by command. At last, the module will return the ACK.

- CMD Paramters

CODE = 0x31、[P1, P2] = (The length of the fingerprint eigenvalue+3)（P1 HSB，P2 LSB)、P3 = 00

Then send the fingerprint eigenvalue:

HEAD + UserID_H + UserID_L + User power + Fingerprint eigenvalue + CHK + END

- ACK Format

HEAD + CH + 0x71 + P1 + P2 + AP + CHK + END

The fourth, fifth bytes [P1, P2] = User number. For example, the user number is 0x1234, then P1=0x12H，P2=0x34H。

The sixth byte AP = Processing result. If:

- AP=EW_SUCCESS Success;
- AP=EW_FULL The user's capacity is full;
- AP=EW_FAIL Failed

1.13 CMD_TO_VALUE_DB

- Function

Get the fingerprint eigenvalue in the fingerprint library according to the user's number pointed in the command.

- CMD Parameters

CODE = 0x22、[P1, P2] = 用户号，P3 = 0x00

- ACK Format

HEAD + CH + 0x62 + P1 + P2 + AP + CHK + END

The fourth and fifth bytes is [P1,P2] = The length of the eigenvalue.

The sixth byte AP = Result. If:

- AP=EW_NO_USER The user does not exist;
- AP=EW_GUEST_USER Temporary User;
- AP=EW_NORMAL_USER Common User;

- AP=EW_MASTER_USER Administrator;

If the user exists, then it sends the eigenvalue:

HEAD + Eigenvalue + CHK + END

1.14 CMD_FROM_VERIFY_DB

- Function

Send a fingerprint eigenvalue and user's number to the module. The module will compare the eigenvalue with the one in the library according to the user's number, then return the processing result.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. Then the eigenvalue is send, the module saves it according to the user's number pointed by command. At last, the module will return the ACK.

- CMD Paramters

CODE = 0x31、[P1, P2] = (The length of the fingerprint eigenvalue+3)（P1 HSB，P2 LSB)、P3 = 00

Then send the fingerprint eigenvalue:

HEAD + UserID_H + UserID_L + User power + Fingerprint eigenvalue + CHK + END

- ACK Format

HEAD + CH + 0x73 + 0x00 + 0x00 + AP + CHK + END

The fourth byte AP = Processing result. If:

- AP=EW_SUCCESS Success;
- AP=EW_NO_USER The user doesn't exist;
- AP=EW_FAIL Failed

1.15 CMD_FROM_VERIFY

- Function

Send a fingerprint eigenvalue to the module. The module will send a command to fingerprint collector to collect a fingerprint, and then compare the eigenvalue with the one collected just now,. At last return the processing result.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. After the eigenvalue is send, the module will send a command to fingerprint collector, and collect the fingerprint eigenvalue in five seconds. Then compare them and return the ACK.

- CMD Parameters

CODE = 0x31、[P1, P2] = (The length of the fingerprint eigenvalue+3)（P1 HSB，P2 LSB)、P3 = 00

Then send the fingerprint eigenvalue:

HEAD + 0x00 + 0x00 + 0x00 + Fingerprint eigenvalue + CHK + END

- ACK Format

HEAD + CH + 0x75 + 0x00 + 0x00 + AP + CHK + END

The fourth byte AP = Processing result. If:

- AP=EW_SUCCESS Success;
- AP=EW_TIMEOUT Timeout error;
- AP=EW_FAIL Failed

1.16 CMD_FROM_IDENTIFY_DB

- Function

Send a fingerprint eigenvalue to fingerprint module, and it will compare the eigenvalue with all of the eigenvalue in the library. At last, return the processing result and the user's number.

Send the command and the module will blank the buffer. Then send the eigenvalue. The module will compare it with all of the eigenvalue in the library and then return the ACK.

- CMD Parameters

CODE = 0x34、[P1,P2] = (The length of fingerprint eigenvalue+3)（P1 HSB, P2 LSB）、P3 = 0x00

Then send the fingerprint eigenvalue:

HEAD + 0x00 + 0x00 + 0x00 + （[P1,P2]-3） Fingerprint eigenvalue + CHK + END

- ACK Format

HEAD + CH + 0x74 + P1 + P2 + AP + CHK + END

The fourth, fifth bytes [P1, P2] = User number. For example, the user number is 0x1234, then P1=0x12H，P2=0x34H。

The sixth byte AP = Processing result. If:

- AP=EW_NO_USER The user does not exist, and [P1, P2] is meaningless;
- AP=EW_GUEST_USER Temporary User;
- AP=EW_NORMAL_USER Common User;
- AP=EW_MASTER_USER Administrator;

1.17 CMD_GET_IMAGE

- Function

Collect a piece of fingerprint image by the module, and get the data of the image outline.

- CMD Parameters

  CODE = 0x27、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x67 + Length_H + Length_L + AP + CHK + END

  Length = Data Length, Length_H HSB, Length_L LSB

  AP = Return Value. If:

  - AP=EW_TIME_OUT Timeout error.
  - EW_TIME_OUT_BMF Collect failed
  - AP=EW_SUCCESS Success
  - AP=EW_FAIL Failed

  If it is successful, the module will return the fingerprint image outline right now. The format is following:

  HEAD + Length (data) + END

  Image outline's properties: 256 Gray scale (8 bits), Width: 256 pixel, High: 256 pixel

  Data format: Send the image data line by line. Every byte is represent two pixels. HSB is the former pixel's HSB and LSB is the later pixel's HSB. That is, the precision of the image is 4 bits. The LSB of pixel needs to be filled by user.

1.18 CMD_SET_BAUD

- Function

  Set the baud rate for communication. The fingerprint module's baud rate is 19200bps after it is powered on. If the user want to change its baud rate, this command should be sent in the baud rate as 19200bps. After a success sign is returned, it represents that the module has set the baud rate successfully. The user can communicate with the module by new baud rate until the next time the module is powered on.

- CMD Parameters

  CODE = 0x0F、P1 = P2 = 0x00、P3 = New baud rate

  - P3 = EW_BAUD_9600, Baud rate is 9600bps
  - P3 = EW_BAUD_19200, Baud rate is 19200bps (Default value)
  - P3 = EW_BAUD_38400, Baud rate is 38400bps
  - P3 = EW_BAUD_57600, Baud rate is 57600bps
  - P3 = EW_BAUD_115200, Baud rate is 115200bps(Not command to use this value. Too fast speed would make the module instable.

- ACK Format

  HEAD + CH + 0x4F + 0x00 + 0x00 + AP + CHK + END

The sixth byte : AP = Old baud rate.

- P3 = EW_BAUD_9600， Baud rate is 9600bps
- P3 = EW_BAUD_19200， Baud rate is 19200bps (Default value)
- P3 = EW_BAUD_38400， Baud rate is 38400bps
- P3 = EW_BAUD_57600， Baud rate is 57600bps
- P3 = EW_BAUD_115200，Baud rate is 115200bps(Not command to use this value. Too fast speed would make the module instable.

1.19 CMD_PROCESS_IMAGE

- Function

  Get the eigenvalue data of the fingerprint image saved in the module's image buffer.

  [NOTE] The fingerprint image is collected last time, and the module will make mistakes if the image data is picked up or compared, because these operation will change the content of the buffer.

- CMD Parameters

  CODE = 0x2D、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x6D + Length_H + Length_L + AP + CHK + END

  Length = [Length_H, Length_L] The length of the eigenvalue

  AP = Return value. If:

  - AP = EW_SUCCESS Success
  - AP = EW_FAIL failed
  - EW_FAIL_FEA Failed to pick up the eigenvalue

  After then, the fingerprint eigenvalue is sent, the format is following:

  HEAD + Length(fingerprint eigenvalue data) + CHK + END

1.20 CMD_TEST_COMM

- Function

  Test if the communication works well.

- CMD Parameters

  CODE = 0x2C、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x6C + 0x00 + 0x00 + AP + CHK + END

  AP = Return Value. If:

  - AP = EW_SUCCESS Communication is OK.

- AP = Others Communication error.

Communication error will result from: (1) No response for a long time. (2) The return value of the module is not compatible with the communication protocol, which includes the number of the data and the definition of the data.

Reasons for these error result from possibility: (1) The baud rate is different from the control baud rate. (2) The module can't work well.

1.21 CMD_TEST_FINGER

- Function

  Confirm it the finger is stay on the sensor.

- CMD Parameters

  CODE = 0x06、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x46 + 0x00 + 0x00 + AP + CHK + END

  AP = Return value. If:

  - AP = EW_SUCCESS The finger is on the sensor.
  - AP = EW_TIME_OUT The finger is not on the sensor
  - EW_TIME_OUT_BMF Collect failed

1.22 CMD_SERIAL_PROG_UPGRADE

- Function

Send executable file and the its save address to the module, which is not more than 264 bytes. It is convenience for program update in the serial flash. (20bit = 11-bit page address (0x800 pages per chip) + 9-bit byte address (264 bytes per page))

After sending command to fingerprint module, continue to send data for a moment. If the module receives the command, it will blank the buffer, and ready to receive the data. When it finishes receiving the data, it will save them according to the command's parameters and returns ACK.

The executive address (byte) is 0x27FFE ~ 0x27FFF and Flash address is (word) 0x2007FF. It is the bit-OR'ing check value. If it is not correct, the program can't run smoothly. The only function of it is to update program.

- CMD Parameters

  CODE = 0x3B、[P1, P2] = (Program data length+3)(P1 HSB, P2 LSB)、P3 =PageAddr

  [P1,P2]=The data length will be send (byte) ≤ (264+3) . If it is more than 264+3, it can't not operate Serial Flash, and return UPGRADE_LEN_ERROR

if [P1,P2] ≤3, it can be erased according to page. (The format is just like the method specified in this document.

For example: If [P1,P2]=1, so it can send command HEAD + DEVICE + CODE + P1 + P2 + P3 + CHK + END at first, and then send HEAD+DEVICE+Any Value+CHK+END。

P3=The 11bit page address in the Serial Flash. address，and the const list is in the page 125 ~ 185. So only 8 bits is enough

	page	Flash IP adress(byte)	(byte)
Program data	0	0x00000~0x00108	0x020000~0x020108
	.....	.....	.....
	124	0x0F800~0x0F908	
Const It msut let the program 'const_move' to finish the copy task located in 0x20100.	125	0x0FA00	Limited by const_move
	.....		
	185	0x17200	

Then send data:

HEAD + DEVICE + ByteAddr_H + ByteAddr_L + Data + CHK + END

ByteAddr_H is the bit 8 of the beginning data byte address, ByteAddr_L is the bit7~bit0 of the beginning data byte address. Or it doesn't process the flash operation and return EW_FULL.

- ACK Format

  HEAD + CH + 0x7A + 0x00 + 0x00 + AP + CHK + END

  The sixth byte AP = Processing result. If:

  - AP=EW_SUCCESS Success
  - AP=EW_FAIL Failed
  - AP=EW_FULL The number of the data check sum or offset can't content the requests.

1.23 Note

- All of the fingerprint eigenvalue send to or got from the fingerprint module are encrypted. It need to the special API and develop kit support by our company.
- For simplify, the module will not judge if the user is occurred when it adds the new user. It should be judged by another way.

2 Internal Commands

2.1 CMD_SET_REG

- Function

  Set BCT100 control register. Be unavailable in this version.

2.2 CMD_GET_VERSION

- Function

  Check the version information of the module. Be unavailable in this version.

2.3 CMD_IDLE

- Function

  Let the module run in the low power resuming status (5mA & 5V). After the module entering the status, all of the commands will process the same operation defined as following but CMD_EXIT_IDLE, and it will get the same return value.

- CMD Parameters

  CODE = 0x07、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x47 + 0x00 + 0x00 + AP + CHK + END

  ➢ AP = EW_SUCCESS Success. The module will return ACK before it entering the status, so it always returns success.

2.4 CMD_EXIT_IDLE

- Function

  Let the module quit the low power resuming status.

- CMD Parameters

  CODE = 0x08、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x48 + 0x00 + 0x00 + AP + CHK + END

  ➢ AP = EW_SUCCESS Success. The module will return ACK before it entering the status, so it always returns success.

2.5 CMD_PROG_UPGRADE

- Function

  Send executable file and the address where to save it to the module, and update the

program in the flash. The program should be less than 2K words.

After sending command to fingerprint module, continue to send data in a moment. If the module receives the command, it will blank the buffer, and ready to receive the data. When it finishes receiving the data, it will save them according to the command's parameters and returns ACK.

The executive address (byte) is 0x27FFE ~ 0x27FFF and Flash address is (word) 0x2007FF. It is the bit-OR'ing check value. If it is not correct, the program can't run smoothly. The only function of it is to update program.

- CMD Parameters

CODE = 0x3A、[P1, P2] = (Program data length+3)（P1 HSB, P2 LSB）、P3 = Save sector 's number

[P1,P2]=The data length will be send (byte) ≤（0x1000+3）. If it is more than 0x1000+3, it can't not operate Serial Flash, and return EW_FULL

if [P1,P2] ≤2, it only erases sector. (It still sends the data according to the format).

For example: If [P1,P2]=1, so it can send command HEAD + DEVICE + CODE + P1 + P2 + P3 + CHK + END at first, and then send HEAD + DEVICE + Any Value + CHK + END。

P3=The internal sector number in the flash. 0≤P3≤13, or it returns EW_FAIL. And:

	page	Flash address(byte)	Executive address (byte)
Program data	0	0x200000~0x2007FF	0x020000~0x02FFF
	……	……	……
	7	0x203800~0x203FFF	0x027000~0x027FFF
Const data It must let the program 'const_move' to finish copying task located in 0x20100h.	8	0x205000	Limited by const_move in the program
	……		
	13	0x207800	

Then send data:

HEAD + DEVICE + Offset_H + Offset_L + Data + CHK + END

Offset is the initial data's offset in the sector (count by word). 0x0000≤Offset≤ 0x800-(ceil(([P1,P2] –3) / 2)). Or it doesn't process the flash operation and return EW_FULL.

- ACK Format

HEAD + CH + 0x7A + 0x00 + 0x00 + AP + CHK + END

The sixth byte AP = Processing result. If:

- AP=EW_SUCCESS Success
- AP=EW_FAIL Failed
- AP=EW_FULL The number of the data or offset can't content the request

2.6 CMD_FROM_IMAGE

- Function

  Send the fingerprint image to DSP. Be unavailable in this version.

2.7 CMD_GET_LAST_ERROR

- Function

  Get the exact error code of the last one.

- CMD Parameters

  CODE = 0x09、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x49 + P1 + P2+ AP + CHK + END

  The sixth byte AP = Processing Result. If:

  - AP=EW_SUCCESS Success. P1 = HSB of the error code. P2 = LSB of the error code.
  - AP=EW_FAIL Failed.

2.8 CMD_GET_FLG

- Function

  Get the value marked by program.

- CMD Parameters

  CODE = 0x0B、P1 = P2 = P3 = 0x00

- ACK Format

  HEAD + CH + 0x4B + P1 + P2+ AP + CHK + END

  The sixth byte AP = Processing Result. If:

  - AP=EW_SUCCESS Success. P1 = HSB of the program mark. P2 = LSB of the program mark.
  - AP=EW_FAIL Failed

Affix 1 The corresponding definition

1. Fingerprint Module command (CMD) Definitions

```
#define CMD_GET_USER_SUM_DB          0x05
#define CMD_GET_USER_RIGHT_DB        0x00
#define CMD_VERIFY_DB                0x01
#define CMD_REG_START_DB             0x02
#define CMD_REG_SECOND_DB            0x04
#define CMD_REG_END_DB               0x03
#define CMD_REG_DELETE_DB            0x20
#define CMD_REG_ALLDEL_DB            0x21
#define CMD_GET_USER_NUMBER_DB       0x10
#define CMD_GET_VALUE                0x26
#define CMD_IDENTIFY_DB              0x12
#define CMD_TO_VALUE_DB              0x22
#define CMD_FROM_VALUE_DB            0x31
#define CMD_FROM_VERIFY_DB           0x33
#define CMD_FROM_IDENTIFY_DB         0x34
#define CMD_FROM_VERIFY              0x35
#define CMD_GET_IMAGE                0x27
#define CMD_SET_BAUD                 0x0F
#define CMD_GET_VERSION              0x2A
#define CMD_PROCESS_IMAGE            0x2D
#define CMD_TEST_COMM                0x2C
#define CMD_TEST_FINGER              0x06
```

2. Fingerprint Module Answer Code (ACK) Definitions

```
#define ACK_GET_USER_SUM_DB          0x45
#define ACK_GET_USER_RIGHT_DB        0x40
#define ACK_VERIFY_DB                0x41
#define ACK_REG_START_DB             0x42
#define ACK_REG_SECOND_DB            0x44
#define ACK_REG_END_DB               0x43
#define ACK_REG_DELETE_DB            0x60
#define ACK_REG_ALLDEL_DB            0x61
#define ACK_GET_USER_NUMBER_DB       0x50
#define ACK_GET_VALUE                0x66
```

```
#define ACK_IDENTIFY_DB             0x52
#define ACK_TO_VALUE_DB             0x62
#define ACK_FROM_VALUE_DB           0x71
#define ACK_FROM_VERIFY_DB          0x73
#define ACK_FROM_IDENTIFY_DB        0x74
#define ACK_FROM_VERIFY             0x75
#define ACK_GET_IMAGE               0x67
#define ACK_SET_BAUD                0x4F
#define ACK_GET_VERSION         0x6A
#define ACK_PROCESS_IMAGE           0x6D
#define ACK_COMM_TEST               0x6C
#define ACK_TEST_FINGER             0x46
```

3. Basic Answer Information Definitions

```
#define EW_SUCCESS          0x00
#define EW_FAIL             0x01
#define EW_FULL         0x02
#define EW_ROLLED_USER      0x03
#define EW_NO_USER          0x04
#define EW_TIME_OUT         0x0F
```

4. User Information Definitions

```
#define EW_GUEST_USER       0x01
#define EW_NORMAL_USER      0x02
#define EW_MASTER_USER      0x03
#define EW_ALL_USER         0x04
```

5. Transport Speed – Baud Rate Definitions

```
#define EW_BAUD_9600        0x01
#define EW_BAUD_19200       0x02
#define EW_BAUD_38400       0x03
#define EW_BAUD_57600       0x04
#define EW_BAUD_115200      0x05
```

Affix 2 Example

```
//ShenZhen GrandLand

//Fingerprint application based on EH0318 handpos machine

//Demo source code for Verifying one's fingerprint

//just open Modem Vcc, and then using Uart functions to communication with fingerprint module

//you must be familar with the communication protocol between G6102 and fingerprint module

//the following code is just an example: how to power on the finger module and

//how to send or recieve data beteen POS and EWAYTEK finger module

/*************************************************************************
**********************/

#include <console.h>

#include <mcard.h>

//Please read the document for G6102

/*************************************************************************
*/

//define constants


#define WR_CARD_SUCCESS 0x33

#define WR_CARD_FAIL     0x32

#define VERI_CARD_SUCCESS 0x31

#define VERI_CARD_FAIL     0x30

#define FIRST_FINGER_START_ADDR     0x072

#define SECOND_FINGER_START_ADDR        0x174

//define constants

#define MAX_TEMP_USED_LENGTH 256

#define USER_INFO_LEN 559


#define DATA_HEAD 0xfe

#define DATA_CH     0x00

#define DATA_END 0xfd


#define FINGER_DATA_LEN 256
```

```
#define DELETE_RECORD_CODE    0x87
#define UPLOAD_RECORD_CODE    0x88
#define UPLOAD_RECORD_ACK     0x89
#define DOWNLOAD_FINGER_CODE   0x98
#define DOWNLOAD_FINGER_ACK     0x99

#define COM_TEST_CODE            0x2C
#define COM_TEST_ACK           0x6C
#define EW_RETURN_SUCCESS          0x00
#define EW_RETURN_FAIL           0x01
#define EW_TIME_OUT             0x0f

#define FINGER_VERIFY_CODE              0x35
#define FINGER_VERIFY_ACK               0x75
#define GET_VALUE_CODE                  0x26
#define GET_VALUE_ACK                   0x66

#define IC_START_ADDR       0x020

/*************************************************************************
**********************/
//Function: UART send a char
//input: ch( char that will be send to UART)
//output:none
void UARTSendChar(char ch)
{
 typ_UART_stat_word Usw;
 int i;

 UART_send_char(ch);
       do{
   Usw.l_word = UART_stat();
```

```
 } while (Usw.bits.out_busy);
 return ;
}
//
/*************************************************************************
************************/
//Function: UART send a string array
//input: pBuf(the pointer points to the string array that will   be sent to UART)
//output:none
void UARTSendStr(char *pBuf)
{
 typ_UART_stat_word Usw;
 int i;

 for (i=0;pBuf[i];i++)
 {
       UART_send_char(pBuf[i]);
       do{
          Usw.I_word = UART_stat();
       } while (Usw.bits.out_busy);
 }
 return ;
}
//
/*************************************************************************
************************/
//Function: receive bytes from UART
//input: iLen (received bytes length)
//output: if success return 1 or return 0
int ReceiveUART(short iLen ,char   *pReceiveBuff)
{
 typ_msg_word msg;                  //system message
 typ_UART_stat_word Usw;            //Uart state
```

```
int iCounter;
unsigned char ch;

iCounter=0;
SPT_set(640);

while(1)
{
    msg.s_word = sys_msg(SM_STAY_AWAKE);
    if (msg.bits.comm_data)          //comm data detecting
    {
        do{
            ch = (unsigned char)UART_get_char();
            pReceiveBuff[iCounter++]=ch;
            if (iCounter>iLen-1) break;
                Usw.l_word = UART_stat();
        }while(Usw.bits.buff_data_available);

    }
    if(iCounter==iLen)
    {
        delay_n_ms(1);
        return 1;
    }

    if (msg.bits.time_out)
    {
        SPT_set(640);
        return 0;
    }
}
}
//
```

```
/*************************************************************************
**********************/
//Function: clr received buffer after opening UART
//input:none
//output:none
void ClrUART(void)
{
 int i;
 for(i=0;i<8;i++)
 {
        delay_n_ms(100);
      UART_get_char();
 }
 //
}
//
/*************************************************************************
**********************/


/*************************************************************************
**********************/
//Function: UART send a finger command
//input: pBuf(the pointer points to the command array) ,
//       chCode(CMD or ACK code) ,chLenHI(data lengh high byte),
//       chLenLO(data lengh low byte) ,chAP(ACK return)
//output:none
void UARTSendFingerCmd(char *pBuf, char chCode,char chLenHI,char chLenLO,char
chAP)
{
 typ_UART_stat_word Usw;
 int i;
 int iCheckSUM;
 //
```

```
pBuf[0x00]=DATA_HEAD;
pBuf[0x01]=DATA_CH;
pBuf[0x02]=chCode;
pBuf[0x03]=chLenHI;
pBuf[0x04]=chLenLO;
pBuf[0x05]=chAP;
//
iCheckSUM=0;
for(i=0x01;i<0x01+5;i++)
{
      iCheckSUM=iCheckSUM^pBuf[i];
}
pBuf[0x06]=iCheckSUM;
//
pBuf[0x07]=DATA_END;
//
for (i=0x00;i<0x00+8;i++)
{
      UART_send_char(pBuf[i]);
      do{
         Usw.l_word = UART_stat();
      } while (Usw.bits.out_busy);
}
return ;
}
//
/*************************************************************************
**********************/
//Function: UART send finger_value
//input: pBuf(the pointer points to the string array reprents finger values), iLen(length of the
array)
//output:none
void UARTSendFingerValue(char *pBuf,short iLen)
{
```

```
typ_UART_stat_word Usw;
int i;
int iCheckSUM=0;

for(i=0x000 ;i<0x000 + iLen;i++)
{
    iCheckSUM=iCheckSUM^pBuf[i];
}
//
UART_send_char(DATA_HEAD);
UART_send_char(0x00);
UART_send_char(0x00);
UART_send_char(0x00);
//
for (i=0x000 ;i<0x000 + iLen;i++)
{
    UART_send_char(pBuf[i]);
    do{
      Usw.l_word = UART_stat();
    } while (Usw.bits.out_busy);
}
//
UART_send_char(DATA_END);
return ;
}
//
/*************************************************************************
**********************/
/*************************************************************************/
//Function : finger verification, comm with DSP and return a verification result
//input:none
//output: 1(fail) or 0(success)
unsigned short VerifyFinger(void)
```

```
{
 //initialization UART

 UART_init(UART_MODEM_ON|UART_ON|UART_8_DATA_BITS|UART_BAUD_1920
0);
 UART_fcntl(UART_fcntl(UART_F_INQ)|UART_F_NO_CTS);
 delay_n_ms(100);
     ClrUART();


 UARTSendFingerCmd(g_SendCmdBuff,
                          FINGER_VERIFY_CODE,
                          0x00,
                          FINGER_DATA_LEN + 0x03,
                          0x00);
 UARTSendFingerValue(g_SendDataBuff, FINGER_DATA_LEN);


 clear_console();
 move_cursor(3,3);
 puts("Pls. put your finger");


 if(ReceiveUART(0x08,g_ReceBuff))
 {
      if (g_ReceBuff[0]= =DATA_HEAD && g_ReceBuff[2]= =FINGER_VERIFY_ACK
&& g_ReceBuff[7]= =DATA_END)
      {
           if (g_ReceBuff[5]==EW_RETURN_SUCCESS)
           {
                move_cursor(3,3);
                puts("Compare Successfully");
                delay_n_ms(1000);
               return 0;
           }


           if (g_ReceBuff[5]==EW_RETURN_FAIL)
```

```
            {
                move_cursor(3,3);
                puts("Compare failed");
                delay_n_ms(1000);
                return 1;
            }

            if (g_ReceBuff[5]==EW_TIME_OUT)
            {
                move_cursor(3,3);
                puts("Timeout");
                delay_n_ms(1000);
                return 1;
            }
            //
        }
        else
        {
            move_cursor(2,3);
            puts("Communication parameters error");
            delay_n_ms(1000);
            return 1;
        }
    }
    else
    {
        move_cursor(3,3);
        puts("Serial port error");
        delay_n_ms(1000);
        return 1;
      }
     UART_init(UART_OFF);
    //
```

```
}
//
/*************************************************************************/
/*************************************************************************/
//Function: write finger values into IC card from DSP
//input: none
//output: if success return 1, or return 0
short WriteFingerIntoCard(void)
{
 //initialization UART
 UART_init(UART_MODEM_ON|UART_ON|UART_8_DATA_BITS|UART_BAUD_1920
0);
 UART_fcntl(UART_fcntl(UART_F_INQ)|UART_F_NO_CTS);
 delay_n_ms(100);
 ClrUART();

 UARTSendFingerCmd(g_SendCmdBuff,GET_VALUE_CODE,0x00,0x00,0x00);

 clear_console();
 move_cursor(3,3);
 puts("Pls. put your finger");
 //
    //256 bytes finger values and ACK command(8bytes) +HEAD,END,CHK
 if(ReceiveUART(FINGER_DATA_LEN + 0x08 + 0x03 ,g_ReceBuff))
 {
       if ( g_ReceBuff[0]= =DATA_HEAD &&
             g_ReceBuff[2]= =GET_VALUE_ACK &&
             g_ReceBuff[7]= =DATA_END)
       {
            if (g_ReceBuff[5]==EW_RETURN_SUCCESS)
            {
                 memcpy(g_TempBuff,g_ReceBuff + 0x08 + 0x02 -
0x01 ,FINGER_DATA_LEN);
                 if(!WriteCard( FIRST_FINGER_START_ADDR,
```

```
                                    FINGER_DATA_LEN,
                                    g_TempBuff))
                {
                    clear_console();
                    move_cursor(3,3);
                    puts("Write card failed");
                    delay_n_ms(1000);
                    return 0;
                }
                else
                {
                    clear_console();
                    move_cursor(3,3);
                    puts("Finish writing");
                    return 1;
                }
                }
            else
            {
                clear_console();
                move_cursor(3,3);
                puts("Collect failed");
                return 0;
            }
        }
        else
        {
            clear_console();
            move_cursor(1,3);
            puts("Communication parameters error");
            return 0;
        }
    }
```

```
else
{
        clear_console();
        move_cursor(3,3);
        puts("Serial port error");
        return 0;
}
UART_init(UART_OFF);
}
/*****************************************************************************/
```

Affix 3 Commands List

[NOTE] All of the data is the hex data in the following list

Command Name	Command(CMD)		Answer(ACK)	
	Content	Format	Return Value	Format HEAD+CH+CODE+XX+XX+AP+CHK+END
		HEAD | CH | CODE | P1 | P2 | P3 | CHK | END		
CMD_GET_USER_SUM_DB	Query the user's fingerprint number in the module	FE | CH | 05 | 00 | 00 | 00 | CHK | FD	Return the total number of the fingerprint.	FE+CH+45+XX+XX+AP+CHK+FD
CMD_GET_USER_RIGHT_DB	Query the user's power	FE | CH | 00 | P1 | P2 | 00 | CHK | FD P1P2 is the user's number	AP=User power	FE+CH+40+00+00+AP+CHK+FD
CMD_VERIFY_DB	Compare the fingerprint collected with the one in the module.	FE | CH | 01 | P1 | P2 | 00 | CHK | FD P1P2 is the user's number	AP=Return processing result.	FE+CH+41+00+00+AP+CHK+FD
CMD_REG_START_DB	Collect the fingerprint for the first time	FE | CH | 02 | P1 | P2 | P3 | CHK | FD P1P2 is the user's number and P3 is the user's power	AP= Return processing result.	FE+CH+42+00+00+AP+CHK+FD
CMD_REG_SECOND_DB	Collect the fingerprint for the second time.	FE | CH | 04 | P1 | P2 | P3 | CHK | FD P1P2 is the user's number and P3 is the user's power	AP= Return processing result.	FE+CH+44+00+00+AP+CHK+FD
CMD_REG_END_DB	Collect the fingerprint for the third time	FE | CH | 03 | P1 | P2 | P3 | CHK | FD P1P2 is the user's number and P3 is the user's power	AP= Return processing result.	FE+CH+43+00+00+AP+CHK+FD

CMD_REG_DELETE_DB	Delete the appointed user.	FE | CH | 20 | P1 | P2 | 00 | CHK | FD P1P2 is the user's number.	AP= Return processing result.	FE+CH+60+00+00+AP+CHK+FD
CMD_REG_ALLDEL_DB	Delete all of the users	FE | CH | 21 | 00 | 00 | P3 | CHK | FD P3 is the user's power	AP= Return processing result.	FE+CH+61+00+00+AP+CHK+FD
CMD_GET_USER_NUMBER_DB	Show the user's number and his power who has registered.	FE | CH | 10 | P1 | P2 | P3 | CHK | FD P1P2 is the beginning user's record number. P3 is the fixed return number, witch should be less than 30.	Return the user's number P1+P2 and his power AP.	FE+CH+50+00+XX+AP+CHK+FD FE+$P1_1$+$P2_1$+AP_1+。。。 +$P1_{p3}$+$P2_{p3}$+AP_{p3}+CHK+FD P1+P2 is the user's number Ap is user's power The length form $P1_1$ to AP_{p3} is XX
CMD_GET_VALUE	Collect the fingerprint image and get fingerprint eigenvalue.	FE | CH | 26 | 00 | 00 | 00 | CHK | FD	AP= Return processing result.	FE+CH+66+00+XX+AP+CHK+FD FE+XX (eigenvalue) +CHK+FD
CMD_IDENTIFY_DB	Collect the fingerprint image and compare it with all of the fingerprints in the library.	FE | CH | 12 | 00 | 00 | 00 | CHK | FD	Return the compare result AP and user's number P1+P2	FE+CH+52+P1+P2+AP+CHK+FD
CMD_FROM_VALUE_DB	Send the eigenvalue and save it to the fingerprint library.	FE | CH | 31 | XX | XX | 00 | CHK | FD FE+P1+P2+P+(XXXX-3) fingerprint eigenvalue +CHK+FD P1P2 is the user's number and P3 is the user's power XXXX is the length of the transported characters.	Return processing result.	FE+CH+71+P1+P2+AP+CHK+FD P1P2 is user's number and AP is the result.

CMD_TO_VALUE_DB	Get the appointed user's eigenvalue in the module database.	FE | CH | 22 | P1 | P2 | 00 | CHK | FD P1P2 is the user's number.	Return processing result and user's number	FE+CH+62+00+XX+AP+CHK+FD FE+XX (eigenvalue)+CHK+FD
CMD_FROM_VERIFY_DB	Send fingerprint eigenvalue to the module and compare it with the appointed one in the library.	FE | CH | 33 | XX | XX | 00 | CHK | FD FE+P1+P2+P3+（XXXX-3）fingerprint eigenvalue +CHK+FD P1P2 is the user's number and P3 is the user's power XXXX is the length of the transported characters.	Return processing result AP.	FE+CH+73+ 00+00+AP+CHK+FD
CMD_FROM_VERIFY	Send the fingerprint eigenvalue to the module and compare it with the one collected by the module.	FE | CH | 35 | XX | XX | 00 | CHK | FD FE+00+00+00+（XXXX-3）fingerprint eigenvalue +CHK+FD XXXX is the length of the transported characters.	Return processing result AP.	FE+CH+75+ 00+00+AP+CHK+FD
CMD_FROM_ IDENTIFY_DB	Send the fingerprint eigenvalue to the module and compare it with all of the ones in the library.	FE | CH | 34 | XX | XX | 00 | CHK | FD FE+00+00+00+（XXXX-3）fingerprint eigenvalue +CHK+FD XXXX is the length of the transported characters	Return processing result AP and the user's number P1+P2	FE+CH+74+P1+P2+AP+CHK+FD
CMD_GET_IMAGE	Collect the fingerprint image and get its outline.	FE | CH | 27 | 00 | 00 | 00 | CHK | FD	Return processing result AP.	FE+CH+67+XX+XX +AP+CHK+FD FE+XXXX (Image value)+FD No check for image
CMD_SET_BAUD	Set transport speed.	FE | CH | 0F | 00 | 00 | P3 | CHK | FD P3：New transport speed	AP=Return the original speed.	FE+CH+4F+00+00+AP+CHK+FD

CMD_GET_VERSION	Query the module's version information.	FE | CH | 2A | 00 | 00 | 00 | CHK | FD	Return the version information.	FE+CH+6A+00+XX+AP+CHK+FD FE+00XX (version information)+CHK+FD
CMD_PROCESS_IMAGE	Get the images' fingerprint eigenvalues in the module.	FE | CH | 2D | 00 | 00 | 00 | CHK | FD	Return processing result AP.	FE+CH+6D+00+XX+AP+CHK+FD FE+XX (eigenvalue)+CHK+FD
CMD_TEST_COMM	Test if the communication works well.	FE | CH | 2C | 00 | 00 | 00 | CHK | FD	Return processing result AP.	FE+CH+6C+00+00+AP+CHK+FD