ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

(Access Control System by fingerprint)

โดย

นายจิรยศ ชุติภรางค์กูล

นางสาวจิรารัตน์ เจนจิตรานนท์

นายเชิงชล ทองเจริญสุข

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของวิชาการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

Access Control System by fingerprint

ผู้จัดทำ นายจิรยศ ชุติภรางค์กูล 47010105

สาวจิรารัตน์ เจนจิตรานนท์ 47010110

นายเชิงชล ทองเจริญสุข 47010192

.....อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ เกียรติวรรณ ทรงสัตย์)

ระบบควบคุมการเข้าออกด้วยลายนิ้วมือ

โดย

นายจิรยศ ชุติภรางค์กูล

นางสาวจิรารัตน์ เจนจิตรานนท์

นายเชิงชล ทองเจริญสุข

อาจารย์ที่ปรึกษา

ผู้ช่วยศาสตราจารย์ เกียรติวรรณ ทรงสัตย์

ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการพัฒนาระบบควบคุมการเข้าออกด้วยลายนิ้วมือ โดยการประยุกต์ไมโครคอนโทรลเลอร์ตระกูล เอ็มซีเอส เป็นหน่วยประมวลผลกลาง การศึกษาการทำงานของเครื่องสแกนลายนิ้วมือ เพื่อนำมาใช้ในธุรกิจ โดยการเก็บลายนิ้วมือของพนักงาน เพื่อเช็คเวลาการทำงานของแต่ละคน เป็นเทคโนโลยีหนึ่งที่มีความสำคัญ เพราะการใช้เครื่องสแกนลายนิ้วมือนั้นจะไม่สามารถทำแทนกันได้เลย นอกจากนี้สามารถที่จะนำไปประยุกต์ใช้กับระบบอื่นได้อีกมากมาย

Access Control System By Fingerprint

By

Jirayos Chutiparangkul

Jirarat Jenjitranont

Chengchon Thongjaroensuk

Advisor

Asst.Prof. Kiettiwan Songsataya

Academic Year 2007

Abstract

This thesis presents the development of the Access Control System By Fingerprint. The application of microcontroller MCS As the central processor each components is constructed. There is an invention of fingerprint which will prevent checking in for other people. The record check in time of the employees by fingerprint will collect the information about users and the record check time can be applied to develop the program for other businesses

กิตติกรรมประกาศ

ขอขอบพระคุณอาจารย์เกียรติวรรณ ทรงสัตย์ ที่ได้กรุณาให้คำปรึกษาแนะนำให้กับโครงงาน รวมทั้งเอื้อเฟื้ออุปกรณ์ในการทำงาน และความช่วยเหลืออื่นๆ

ขอบคุณพี่ๆเพื่อนๆและน้องๆทุกคนที่ให้กำลังใจ สนับสนุนอุปกรณ์ ช่วยทำงาน กระตุ้น และให้คำแนะนำดีๆที่มีประโยชน์อย่างมาก

ขอขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เป็นสถานที่ศึกษา เสริมสร้างความรู้และประสบการณ์ที่สำคัญต่อผู้จัดทำ

สุดท้ายขอกราบขอบพระคุณคุณพ่อ คุณแม่ และครอบครัวที่ดูแล อบรมสั่งสอน คอยให้กำลังใจ รวมทั้งสนับสนุนในเรื่องของงบประมาณ ตลอดจนเป็นแรงบันดาลใจที่ดีที่ทำให้โครงงานเสร็จสมบูรณ์ได้

ผู้จัดทำ

นายจิรยศ ชุติภรางค์กูล

นางสาวจิรารัตน์ เจนจิตรานนท์

นายเชิงชล ทองเจริญสุข

สารบัญ

สารบัญ(ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

กล่าวนำ

การตรวจสอบเวลาการทำงานในปัจจุบัน มีวิธีที่จะสามารถยืนยันการเข้าออกของพนักงานได้หลายวิธี เช่น เซ็นชื่อ หรือ ตอกบัตร ซึ่งวิธีดังกล่าวไม่สามารถป้องกันการปลอมแปลงบุคคลอื่นซึ่งมาบันทึกข้อมูลแทนกันได้ แต่สำหรับเทคโนโลยีการสแกนลายนิ้วมือนั้น การยืนยันตัวบุคคลสามารถทำได้อย่างมีประสิทธิภาพที่ดีกว่า

ส่วนประกอบที่สำคัญของโครงงานประกอบไปด้วยส่วนที่สำคัญทั้งหมด 3 ส่วนด้วยกัน คือ ส่วนของโมดูลสแกนลายนิ้วมือ เป็นส่วนที่ใช้สำหรับทำการสแกนลายนิ้วมือ โดยจะเชื่อมต่อกับคอมพิวเตอร์ทางพอร์ตอนุกรม ในส่วนของคอมพิวเตอร์เป็นส่วนที่ใช้ในการแสดงผล และใช้ติดต่อกับฐานข้อมูล สุดท้ายส่วนฐานข้อมูล เป็นส่วนที่ใช้เก็บข้อมูลประวัติต่างๆเพื่อนำข้อมูลขึ้นเว็บเซอร์วิส(Web Service) ทำให้พนักงานภายในองค์การสามารถตรวจสอบเวลาการทำงานได้สะดวกยิ่งขึ้น

ในการออกแบบระบบนั้นจะประกอบไปด้วยองค์ประกอบ 3 ส่วนด้วยกันได้แก่

1. ส่วนของโมดูลสแกนลายนิ้วมือ เป็นส่วนที่ใช้สำหรับสแกนลายนิ้วมือเพื่อการบันทึกข้อมูลลายนิ้วมือ และเปรียบเทียบลายนิ้วมือ โดยในโครงงานนี้ได้ใช้โมดูลสแกนลายนิ้วมือรุ่น MRB200 ซึ่งจะเป็นเลนส์สแกนแบบสัมผัสโดยตรงกับพื้นผิวของเลนส์สแกน(Capacitive Sensor)
2. ส่วนของคอมพิวเตอร์ หรือโปรแกรมตรวจสอบข้อมูลการเข้าออก เป็นส่วนที่ใช้ในการแสดงข้อมูลประวัติ และเขียนข้อมูลประวัติที่ต้องการลงไปเก็บไว้ในฐานข้อมูล จะใช้ในการรับส่งคำสั่งการทำงานระหว่างโมดูลสแกนลายนิ้วมือกับคอมพิวเตอร์ โดยการรับส่งข้อมูลระหว่างโมดูลสแกนลายนิ้วมือกับคอมพิวเตอร์จะรับส่งข้อมูลผ่านทางพอร์ตอนุกรม โปรแกรมที่ใช้เขียนในโครงงานนี้ได้ใช้โปรแกรม PHP ในการรับส่งคำสั่งการทำงานทั้งหมดของระบบ
3. ส่วนของฐานข้อมูล ในส่วนนี้จะใช้ในการเก็บข้อมูลประวัติของบุคคลที่ต้องการ ซึ่งจะมีข้อมูลที่สำคัญในการเก็บประวัติบุคคล เช่น รหัสลายนิ้วมือ ชื่อ นามสกุล เวลา เป็นต้น ในโครงงานนี้ได้ใช้โปรแกรม MYSQL เป็นโปรแกรมฐานข้อมูล MYSQL เป็นโปรแกรมที่เหมาะสมกับองค์กรขนาดกลางที่มีข้อมูลไม่มากนัก และเป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์(Relationnal Database Management System) ซึ่งเป็นฟรีแวร์ทางด้านฐานข้อมูลจึงได้รับความนิยมอย่างมากในปัจจุบัน พร้อมทั้งยังสนับสนุนการใช้งานบนระบบปฏิบัติการ ตัวอย่างเช่น Unix Windows นอกจากนี้ยังทำงานร่วมกับ Java , C , C++ , PHP , ASP หรือ Perl ได้ ซึ่งในโครงงานนี้ได้ใช้ MYSQL ติดต่อกับ PHP

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1.2.1 เพื่อออกแบบระบบตัวควบคุมการเข้าออกโดยใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นหน่วยประมวลผล

1.2.2 เพื่อใช้ติดต่อสื่อสารแบบอนุกรมผ่านโครงข่าย RS-232

1.2.3 เพื่อนำเครื่องสแกนลายนิ้วมือมาประยุกต์ใช้ในการเก็บข้อมูลและอ่านข้อมูล

1.2.4 เพื่อพัฒนาหรือแก้ไขโปรแกรมการติดต่อสื่อสารข้อมูล การขอใช้บริการ และการควบคุมการเข้าออกระหว่างเซิร์ฟเวอร์(Sever) และเทอร์มินอล(Terminal) หรือเซิร์ฟเวอร์กับคอมพิวเตอร์ ให้มีประสิทธิภาพ

1.2.5 เพื่อการออกแบบตัวเซิร์ฟเวอร์(Sever) และเทอร์มินอล(Terminal) ให้มีประสิทธิภาพในใช้งานตามวัตถุประสงค์ได้อย่างเหมาะสม

1.3 หลักการที่ใช้ในการออกแบบระบบ

หลักการที่ใช้ในการออกแบบระบบควบคุมการเข้าออกโดยใช้เครื่องแสกนลายนิ้วมือคือ

1.3.1 การอ่านข้อมูลจากเครื่องแสกนลายนิ้วมือ

1.3.2 การทำงานร่วมกันของไมโครคอนโทรลเลอร์ MCS-51

1.3.3 การใช้โครงข่าย RS-232

1.3.4 การใช้ภาษาซี ในการโปรแกรมการทำงาน

1.4 รายละเอียดของรายงานนี้แบ่งส่วนที่ดำเนินการออกเป็น

บทที่ 1 บทกล่าวนำ บอกวัตถุประสงค์ และหลักการออกแบบ

บทที่ 2 กล่าวถึงทฤษฎีและหลักการต่างๆ ได้แก่รูปแบบรวมทั้งหลักการพื้นฐานของการติดต่อสื่อสาร การควบคุมส่วนอ่านลายนิ้วมือ การติดต่อสื่อสารเบื้องต้น และโครงสร้างการติดต่อสื่อสารแบบอนุกรม

บทที่ 3 กล่าวถึงวิธีการสร้าง และออกแบบรวมทั้งโครงสร้างของเซิร์ฟเวอร์และเทอร์มินอล

บทที่ 4 การทดลองการทำงานของระบบในส่วนต่างๆ และผลการทดลอง

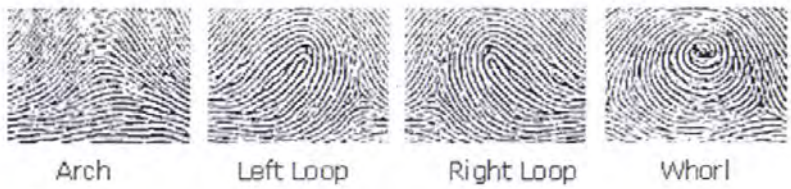
บทที่ 5 บทสรุป เป็นการสรุป และวิจารณ์ผลจากระบบที่ออกแบบ รวมทั้งแนวทางแก้ไขพัฒนาต่อไป

บทที่ 2
หลักการและทฤษฎี

2.1 การอ่านลายนิ้วมือ

ลายนิ้วมือของมนุษย์แต่ละคนนั้น เริ่มปรากฏขึ้นตั้งแต่เป็นตัวอ่อนอายุ 3 ถึง 4 เดือนในครรภ์มารดา ลายนิ้วมือเป็นผิวหนังส่วนที่มีร่อง(Furrow) และมีสัน(Ridge) ไว้ใช้สำหรับอำนวยความสะดวกในการหยิบจับวัตถุสิ่งของต่างๆ ร่อง(Furrow) และสัน(Ridge)ของลายนิ้วมือมีลักษณะที่สำคัญ 2 ประการคือ หนึ่งไม่มีการเปลี่ยนแปลงรูปแบบตามกาลเวลา นับตั้งแต่วันแรกที่มนุษย์เกิดจวบจนกระทั่งวันที่มนุษย์ตาย ร่องรอย และสันของลายนิ้วมือจะปรากฏเช่นเดิมไม่สูญหาย แต่ขนาดของร่องรอย และสันนั้นอาจเปลี่ยนแปลงได้ตามขนาดของร่างกาย และสองลายนิ้วมือมีรูปแบบเฉพาะในแต่ละคน(Individuality) การเปรียบเทียบลายนิ้วมือมนุษย์มีมานานกว่าร้อยปี ผลการศึกษาพบว่า ไม่มีมนุษย์คนใดในโลกที่มีลายนิ้วมือเหมือนกันซึ่งสอดคล้องกับผลการศึกษาวิจัยเรื่องการใช้ลายนิ้วมือระบุตัวบุคคลของ Sir Francis Galton (1982) โดยใช้การแบ่งรายละเอียดรูปแบบของลายนิ้วมือออกเป็นส่วนๆ และหาความน่าจะเป็นของการซ้ำกันของลายนิ้วมือแต่ละส่วน แล้วนำความน่าจะเป็นของแต่ละส่วนมาคูณกันเพื่อหาความน่าจะเป็นทั้งหมด พบว่าลายนิ้วมือของแต่ละบุคคลนั้นจะมีลักษณะเฉพาะ(Individuality) และไม่มีการเปลี่ยนแปลงรูปแบบ(Permanence) โอกาสที่บุคคลสองคนจะมีลายนิ้วมือเหมือนกันมีความน่าจะเป็นอยู่ที่ 1/64,000,000,000

ลายนิ้วมือของคนแต่ละคนนั้นมีลักษณะเฉพาะแม้แต่คู่แฝดแท้(Identical Twin) ก็ยังมีลายนิ้วมือที่แตกต่างกัน (แต่มีรูปแบบทางพันธุกรรมเหมือนกัน) รูปแบบของลายนิ้วมือสามารถแบ่งออกได้เป็น 3 ประเภท ได้แก่ ลายนิ้วมือก้นหอย(Whorl) ลายนิ้วมือมัดหวาย(Loop) และลายนิ้วมือโค้ง(Arch)



รูปที่ 2.1 ลักษณะของลายนิ้วมือแบบต่างๆ

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

นอกจากนี้ยังได้มีการแบ่งสัดส่วนของลายนิ้วมือแบบต่างๆที่ปรากฏเห็นในมนุษย์แต่ละคน พบว่า 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 และแปลงสัญญาณจากลายนิ้วมือนับว่ามีความสำคัญมาก เพราะเป็นส่วนที่ใช้ในการรักษาความปลอดภัยเนื่องจากสัญญาณส่วนที่ได้จากส่วนนี้เป็นสัญญาณเฉพาะของแต่ละบุคคลไม่สามารถลอกเลียนแบบกันได้ ผู้ศึกษาได้ใช้อุปกรณ์คือ โมดูลแสกนลายนิ้วมือรุ่น MRB200 ผู้ศึกษายังได้ทำการเชื่อมต่อกับหน่วยประมวลผลกลางโดยผ่านมาตรฐาน RS-232 หรือ RS-485

2.2 การรับส่งข้อมูล

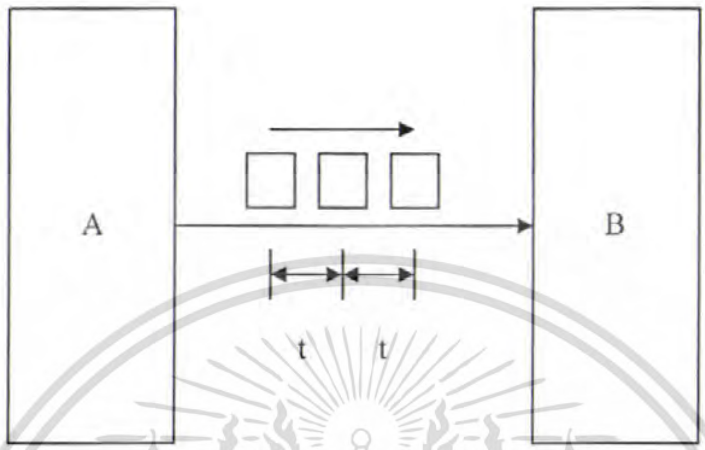
แบบที่หนึ่ง การรับส่งข้อมูลแบบขนาน(Parallel) การรับส่งข้อมูลแบบขนานเป็นการรับส่งข้อมูลจำนวน 1 ไบต์ออกทางพอร์ท ในเวลาเดียวกันระบบคอมพิวเตอร์ 1 ไบต์จะมีจำนวน 8 บิตคือ D0-D7 ถ้ามีการส่งข้อมูลแบบขนานจะใช้สายสัญญาณอย่างน้อย 9 เส้นคือสายสัญญาณ 8 เส้น และสายกราวน์ 1 เส้น

แบบที่สอง การรับส่งข้อมูลแบบอนุกรม(Serial) คือการรับส่งข้อมูลแต่ละบิตจนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์คือ D0-D7 อาจต้องส่งบิต D0 ออกไปก่อนแล้วตามด้วย D1 ไปเรื่อยๆจนถึง D7

การรับส่งทั้งสองแบบมีข้อดีข้อเสียแตกต่างกันคือ การรับส่งข้อมูลแบบขนานสามารถส่งข้อมูลได้เร็ว ส่งข้อมูลครั้งเดียวจะได้ข้อมูลครบ 1 ไบต์ แต่มีข้อจำกัดที่หากมีข้อมูลไปในที่ๆมีระยะทางไกลๆ จะทำให้สิ้นเปลืองสายสัญญาณมาก ขณะที่การส่งข้อมูลแบบอนุกรมสามารถส่งข้อมูลไปในระยะทางไกลๆได้ดีกว่า ช่วยประหยัดสายสัญญาณเนื่องจากการส่งข้อมูลแบบอนุกรมนั้นจะใช้สายอย่างน้อยเพียง 2 เส้น ได้แก่ สายสัญญาณ กับสายกราวด์ แต่มีข้อจำกัดที่ใช้เวลานานในการรับส่งข้อมูลเนื่องจากเป็นการส่งทีละบิต

2.2.1 การรับส่งข้อมูลแบบซิงโครนัส(Synchronous Input/Output) การรับส่งข้อมูลแบบนี้ไม่ว่าจะเป็นการส่งแบบอนุกรมหรือแบบขนาน ข้อมูลแต่ละไบต์ที่ถูกส่งออกไปจะมีช่วงเวลาห่างกันแน่นอน เช่น การส่งข้อมูลจาก A ไป B ดังรูปที่ 2.2 สัญญาณ 1 จะห่างจาสัญญาณ 2 เป็นเวลา

t และสัญญาณ 3 จะห่างจากสัญญาณ 2 เป็นเวลา t เช่นกัน ระบบแบบนี้เหมาะกับงานที่ไม่มีความยุ่งยากมาก



รูปที่ 2.2 การส่งข้อมูลแบบซิงโครนัส

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

2.3 รูปแบบของการรับส่งข้อมูลแบบอนุกรม

การสื่อสารข้อมูลอนุกรมแบบอะซิงโครนัสเป็นการรับส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตของข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริงเพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นซึ่งประกอบด้วยกัน 4 ส่วนคือ

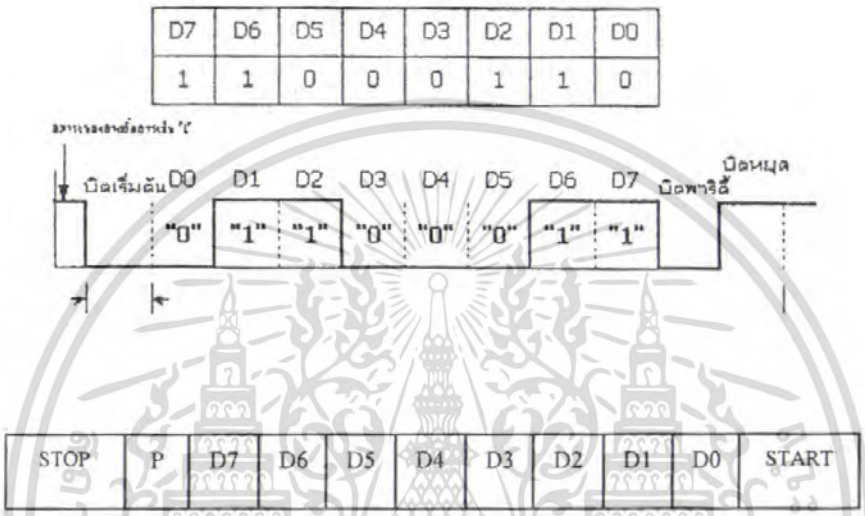
ส่วนที่หนึ่ง บิตเริ่มต้น(Start bit) จะมีขนาด 1 บิต จะเป็นระดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสารขณะที่ยังไม่มีการส่งข้อมูล

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 ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็นเลขหนึ่ง โดยการเลือกการส่งข้อมูลเป็นแบบพาริตี้คู่ หรือพาริตี้คี่ ตัวอย่างถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลขหนึ่งเป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็นหนึ่งเพื่อจะได้จำนวนเลขหนึ่งเป็นคู่นั้นเอง ทำนอง

เดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็นหนึ่งรวมทั้งบิตพาริตี้หนึ่งบิต ถ้ามีค่าหนึ่งเป็นจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง ซึ่งสามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้

ส่วนที่สี่ บิตสุดท้ายหรือบิตหยุด(Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูลโดยจะทำให้ขาข้อมูลมีสถานะลอจิกเป็นหนึ่ง ซึ่งอาจมีจำนวนมากกว่าหนึ่งบิตก็ได้เช่น 1 บิต 1.5 บิต หรือ 2 บิต



รูปที่ 2.3 บิตต่างๆของข้อมูลที่ส่งแบบอนุกรม

ถ้ามีการส่งข้อมูลแบบ 8 บิต จะต้องส่งบิตแรกออกไปก่อนเรียกว่า บิตเริ่มต้น(Start Bit) ถ้ามีการส่งข้อมูลหลายๆไบต์ออกมาบิตนี้จะเป็นตัวบอกว่า มีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้นมักมีระดับลอจิกเป็นศูนย์ต่อจากบิตเริ่มต้นจะเป็นบิตข้อมูล D0 ถึง D7 จากนั้นจะตามด้วยบิตตรวจสอบความถูกต้อง(Parity Bit) ถ้าข้อมูล 8 บิตที่ส่งออกมาจำนวนบิตที่มีค่าเป็นหนึ่งเป็นจำนวนคู่ บิตนี้จะมีค่าเป็นศูนย์ แต่ถ้าจำนวนของบิตที่มีค่าเป็นหนึ่งเป็นจำนวนคี่ บิตนี้จะมีค่าเป็นหนึ่ง จากนั้นข้อมูลที่ส่งออกไปจะตามด้วยบิตสิ้นสุดข้อมูล(Stop Bit) เพื่อเป็นการบอกว่าข้อมูลที่ส่งออกมา 8 บิตนั้นหมดแล้ว ตัวบิต Stop นั้นอาจมีจำนวนมากกว่า 1 บิตก็ได้เช่น 1.5 บิต 2 บิต

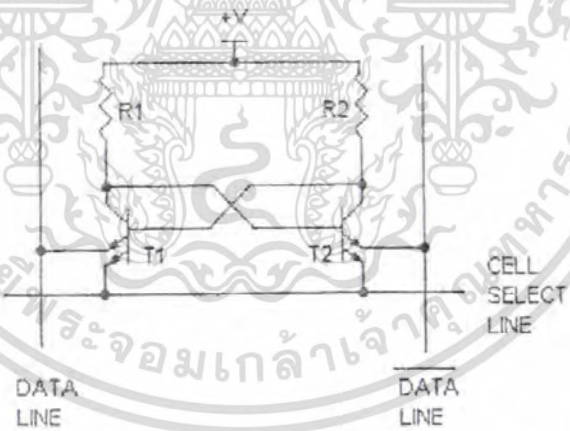
บิตตรวจสอบความถูกต้องหรือพาริตี้บิตจะมีสองลักษณะคือ พาริตี้คู่(Even Parity) และพาริตี้คี่(Odd Parity) ซึ่งเราสามารถเลือกได้ ถ้าหากระบุเป็นพาริตี้บิตคู่ หมายความว่าข้อมูลที่ส่งไปหรือไบต์นั้นมีจำนวนลอจิกนึ่งรวมกับพาริตี้เป็นจำนวนคู่บิต ส่วนถ้าระบุเป็นพาริตี้คี่ก็หมายความว่าจะมีจำนวนลอจิกหนึ่งของไบต์ข้อมูลที่ส่งไปรวมกับพาริตี้เป็นจำนวนคี่บิต ตัวอย่างเช่น ถ้าเราส่งไบต์ข้อมูลที่มีค่าเป็น B2H หรือ 10110010B ออกไป ถ้าระบุว่าเป็นพาริตี้คี่แล้ว ค่าของบิตพาริตี้

จะต้องเป็นหนึ่งเพื่อให้มีจำนวนของลอจิกหนึ่งเป็น 5 บิตซึ่งเป็นจำนวนคี่ เมื่อทางภาครับได้รับข้อมูลเข้ามาก็จะทำการตรวจสอบว่าข้อมูลทั้งหมดมีลอจิกหนึ่งเป็นจำนวนคู่หรือคี่ตามที่ได้ออกแบบการทำงานของระบบไว้หรือไม่ ซึ่งจะสามารถตรวจสอบความถูกต้องได้ในระดับหนึ่ง ความเร็วในการสื่อสารแบบอนุกรมนี้จะมีหน่วยเป็นบิตต่อวินาที(Bit Per Second: BPS) โดยเรียกกันว่าบอดเรต(Baud Rate) ซึ่งจะกำหนดค่ามาตรฐานในการใช้งานไว้หลายค่าเช่น 100 150 300 600 2400 4800 9600 19200 เป็นต้น

2.4 หน่วยความจำข้อมูล

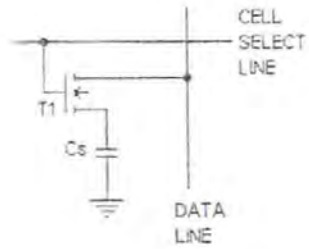
หน่วยความจำชนิดนี้ระบบควบคุมส่วนกลาง(Central Processing Unit : CPU) สามารถอ่านและเขียนข้อมูลได้เรียกว่าแรม(Random Access Memory) เป็นหน่วยความจำที่ใช้เก็บข้อมูลจากการประมวลผลของระบบควบคุมส่วนกลางข้อมูลภายในแรมจะคงอยู่ตลอดเวลาที่มีแหล่งจ่ายไฟต่ออยู่กับหน่วยความจำ ตัวหน่วยความจำแรมนี้ยังแบ่งออกได้เป็นสองชนิดใหญ่ๆดังนี้คือ

2.4.1 สแตติกแรม(Static RAM) เป็นหน่วยความจำแรมที่สามารถอ่านและเขียนข้อมูลได้ และข้อมูลยังคงอยู่ตลอดถ้ามีไฟเลี้ยงโครงสร้างภายใน ในการเก็บข้อมูลแต่ละบิตจะสร้างเป็นฟลิปฟลอป ทำให้การเก็บข้อมูลแต่ละบิตจะต้องสร้างออกมาจากทรานซิสเตอร์ออกมาหลายตัวดังรูปที่ 2.4 ซึ่งแสดงโครงสร้างของสแตติกแรมในการเก็บข้อมูล 1 บิต



รูปที่ 2.4 แสดงการเก็บข้อมูลแต่ละบทของสแตติกแรม

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



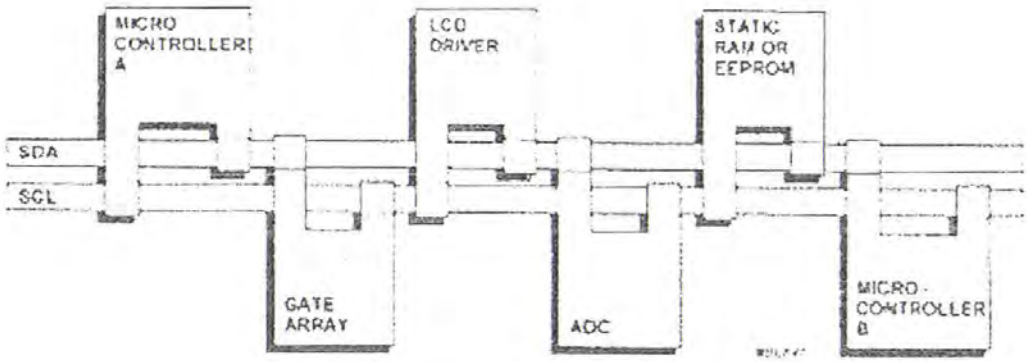
รูปที่ 2.5 แสดงโครงสร้างภายในของไดนามิกแรม

รูปที่ 2.6 แผนภาพของหน่วยความจำต่างๆ

2.5 การติดต่อกับอุปกรณ์โดยใช้ระบบบัสแบบ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึงการติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ซึ่งได้รับการพัฒนาขึ้นจากบริษัท Philips ด้วยจุดมุ่งหมายคือต้องการให้ไอซี หรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูล และสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัว ขนาน หรือพ่วงกันไป ส่วนการกำหนดแอดเดรส หรือตำแหน่งสำหรับติดต่อกับอุปกรณ์แต่ละตัวจะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือSDA(Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อว่า สายสัญญาณอนุกรม หรือSCL(Serial Clock Line)

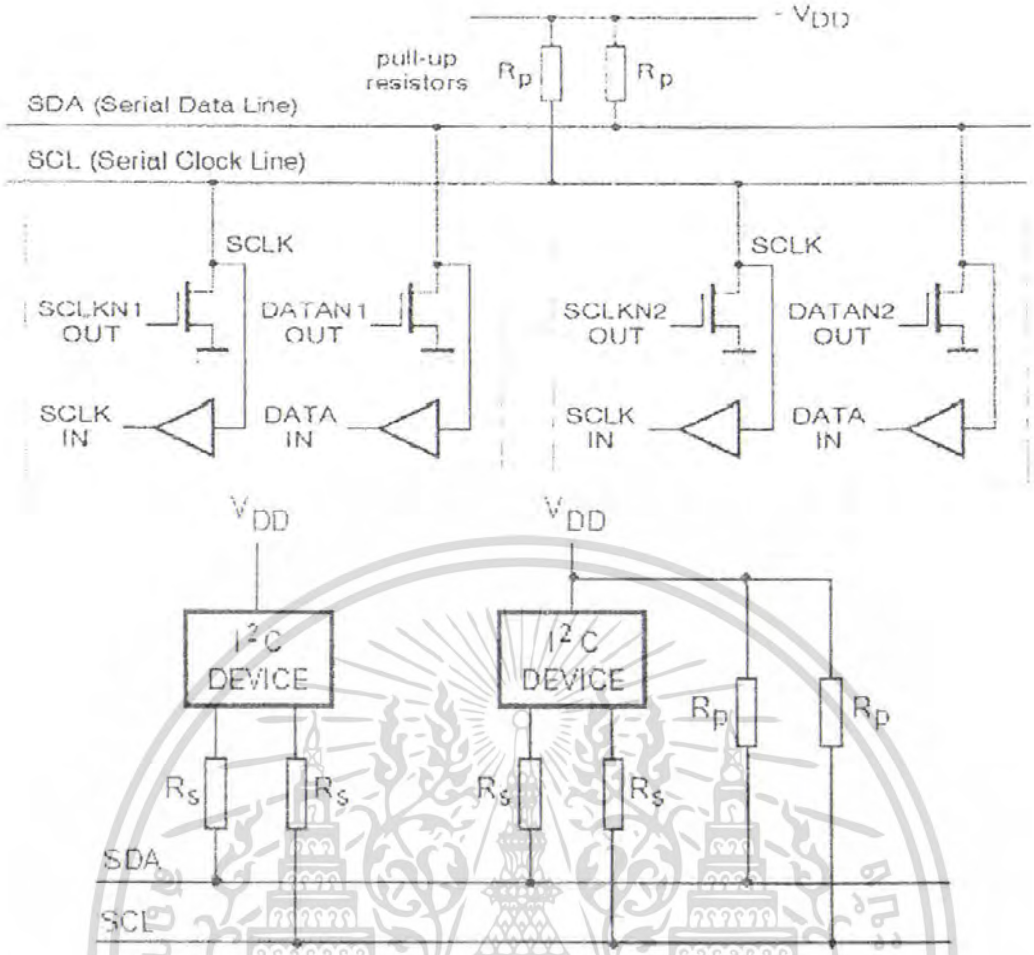


รูปที่ 2.7 ผังแสดงการเชื่อมต่ออุปกรณ์ต่างๆบนระบบบัส I^2C

จากรูปที่ 2.7 แสดงผังของการเชื่อมต่ออุปกรณ์ต่างๆบนบัส I^2C จะเห็นได้ว่าอุปกรณ์ที่ทำการเชื่อมต่อบนบัส 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 และSCL 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



รูปที่ 2.8 การต่อตัวต้านทาน Rs เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในบัส

I^2C ในกรณีที่อาจมีแรงดันไฟฟ้าระลากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องมีตัวต้านทานต่ออนุกรมกับขา SDA และSCL เรียกว่า Rs ก่อนต่อเข้าสู่บัส I^2C

2.5.2 หลักการของบัส I^2C บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้นดังนั้นได้กล่าวมาแล้ว คือSDA และSCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบการติดต่อบนบัส หรือเรียกว่าโปรโตคอล(Protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดกันอยู่ และอุปกรณ์ใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะอธิบายหน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

- อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูลเรียกว่าตัวส่ง(Transmitter)
- อุปกรณ์ที่เป็นผู้รับข้อมูลเรียกว่าตัวรับ(Receiver) ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งอย่างเดียว
- อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่ามาสเตอร์(Master)
- อุปกรณ์ที่ถูกควบคุม หรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่าสเลฟ(Slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

หนึ่ง การถ่ายข้อมูลเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

สอง ในระหว่างการถ่ายข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

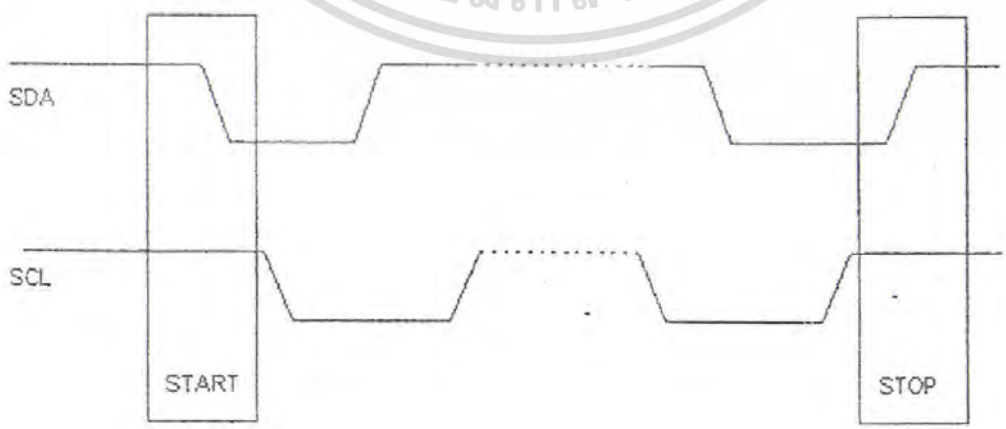
2.5.3 สภาวะที่เกิดขึ้นบนบัส I^2C มีด้วยกัน 5 สภาวะ ดังนี้

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 2.9 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

2.5.4 การทำงานบนบัส 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 2.10 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อหรือข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูปที่ 2.10 ใน 7 บิตบนรวมทั้ง MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็นบิตกำหนดแอดเดรสคงที่ (Fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถดัดแปลงแก้ไขได้ถัดมาอีก 3 บิต เป็นบิตกำหนดแอดเดรสที่สามารถกำหนดโปรแกรมได้ (Programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นหาก LSB เป็น "0" หมายถึงการเขียนข้อมูลไปยังอุปกรณ์นั้นถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่างไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต

บิตใดเป็นเอาท์พุต 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 เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

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

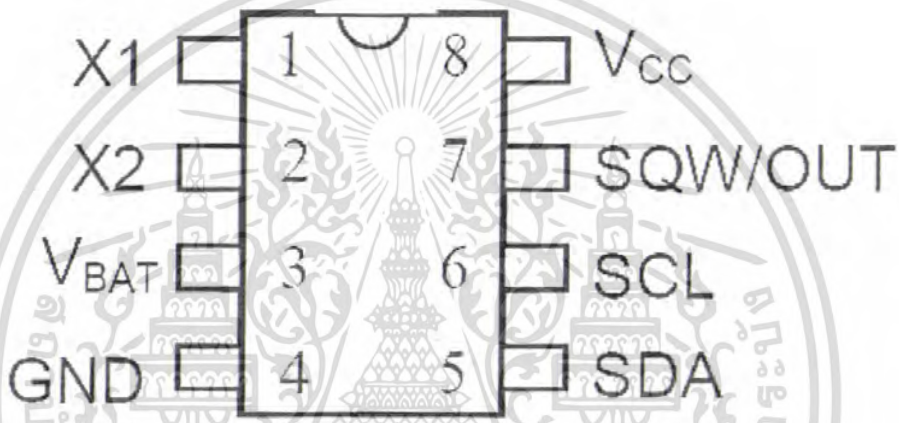
2.6 การใช้งานไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (DS1307)

DS1307 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก ผู้ผลิตคือ ดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีหน้าที่สร้างฐานเวลาจริงให้แก่ระบบไมโครคอนโทรลเลอร์โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที นาที ชั่วโมง วันที่ วันใน

สัปดาห์ เดือน และปี โดยสามารถ ปรับวันเดือนปีให้ตรงตามปฏิทินได้อย่างถูกต้องรวมถึงการกำหนดวันในปีอธิกสุรทินด้วย คุณสมบัติทางเทคนิคที่สำคัญมีดังนี้

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

2.6.1 รายละเอียดขาต่อใช้งานของ DS1307 ในรูปที่ 2.11 แสดงการจัดขาของ DS1307 แต่ละขามีหน้าที่ละการใช้งานดังนี้ Vcc GND (ขา 8, 4) ต่อไฟเลี้ยง +5V



รูปที่ 2.11 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

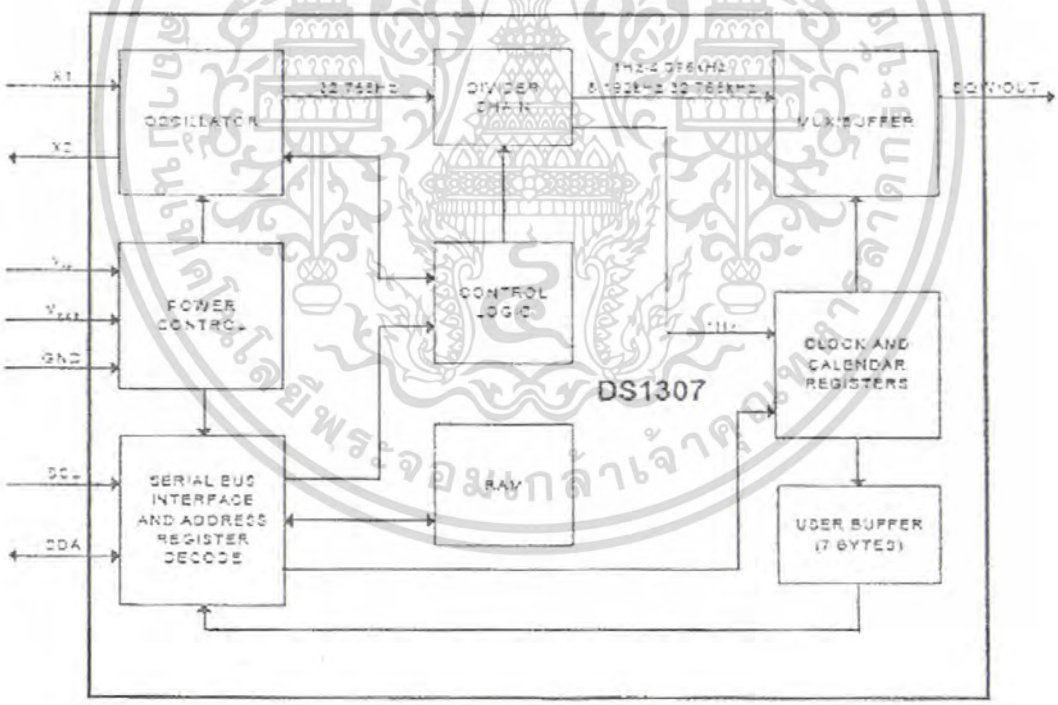
ในการใช้งานต้องต่อตัวต้านทานพูลอัพที่ขานี้ด้วย SQW/OUT (ขา7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกความถี่ได้ 1 kHz, 4.096 kHz, 8.192 kHz และ 32 kHz ในการใช้งานต้องต่อตัวต้านทานพูลอัพที่ขานี้ด้วย X1, X2 (ขา 1 และ ขา2) ใช้ต่อกับคริสตอลมาตรฐาน 32.768 kHz เพื่อใช้เป็น ฐานเวลาในการสร้างค่าเวลาจริงในการใช้งาน 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 2.12 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307

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

2.6.3 การจัดสรรหน่วยความจำใน DS1307 ในรูปที่ 2.12 แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลา ที่ใช้เก็บข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07 เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ซึ่งแสดงรายละเอียดของรีจีสเตอร์ค่าเวลา และรีจีสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามต้องการโดยไม่จำเป็นต้องอ่านออกทั้งหมดก็ได้ 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
		24	PM/AM							00–23
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 2.13 การจัดหน่วยความจำแรมภายใน DS1307

2.6.4 รีจีสเตอร์ควบคุม มีแออดเดรสอยู่ที่ 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

ตารางที่ 2.1 แสดงความถี่ของสัญญาณจากขา SQW/OUT

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

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

- โหมดการเขียนข้อมูล เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (START: S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียนนั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมาคือส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียน หลังจากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็ เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรบจาก DS1307 ทุกครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (STOP: P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล

รูปที่ 2.14 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล

โหมดการอ่านข้อมูล มีรูปแบบแสดงในรูปที่ 2.14 เริ่มต้นการทำงานเหมือนกับ โหมดการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้น แล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบกลับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์ครั้งละ 1 แอดเดรส หรือ 1 ไบต์โดยแอดเดรสที่

เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้า ด้วยโหมดการเขียนข้อมูล วิธีการง่ายๆคือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้งตามด้วยเลือกโหมดการอ่านข้อมูลที่ออกจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้

2.7 หน่วยความจำEEPROM แบบ I^2C

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

2.8 MCS-51

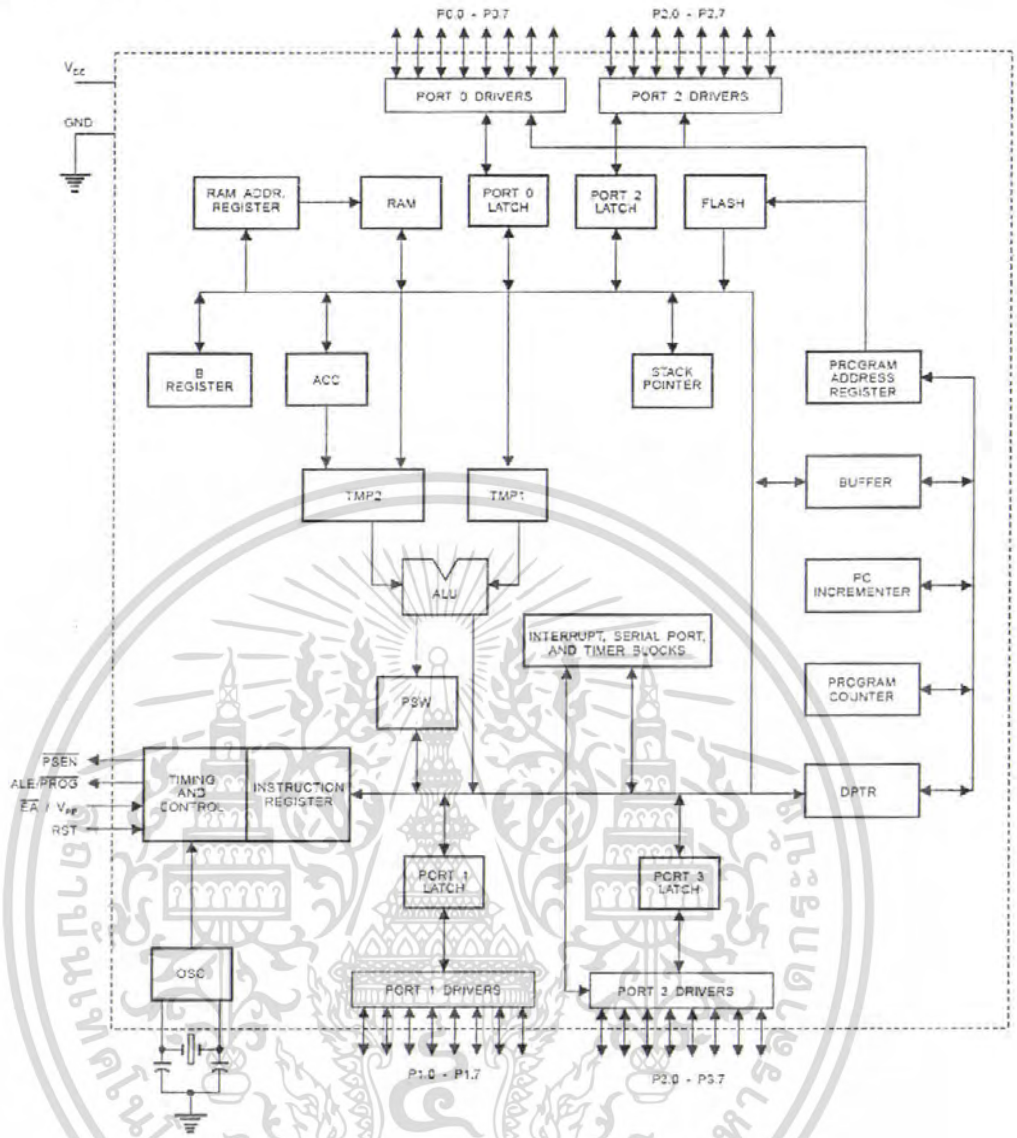
2.8.1 คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

1. ต้องการแหล่งจ่ายไฟ +5V ชุดเดียว
2. มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ8031 ส่วนในเบอร์ 8032 ไม่มีหน่วยความจำ และในเบอร์ 8052 มีหน่วยความจำถึง 8 กิโลไบต์
3. มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256 ไบต์
4. มีหน่วยความจำสำหรับโปรแกรมและข้อมูล (Program Memory) และ Data MeMory แยกจากกันอย่างละ 64 กิโลไบต์
5. คำสั่งที่ใช้ ใช้เวลาน้อยสุดประมาณ 1μS เมื่อทำงานที่ความถี่ 12 MHz
6. มี Timer/Counter ขนาด 16 บิต 2 ชุด
7. รับอินเตอร์รัพได้ 6 แหล่ง 5 เวกเตอร์
8. มีพอร์ตสำหรับส่งข้อมูลอนุกรม (UART) 2 พอร์ต ทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมด
9. มีคำสั่งในการทำ AND,ORหรือCOMPLEMENT ได้ทั้งแบบ 8 บิต และ 1 บิต

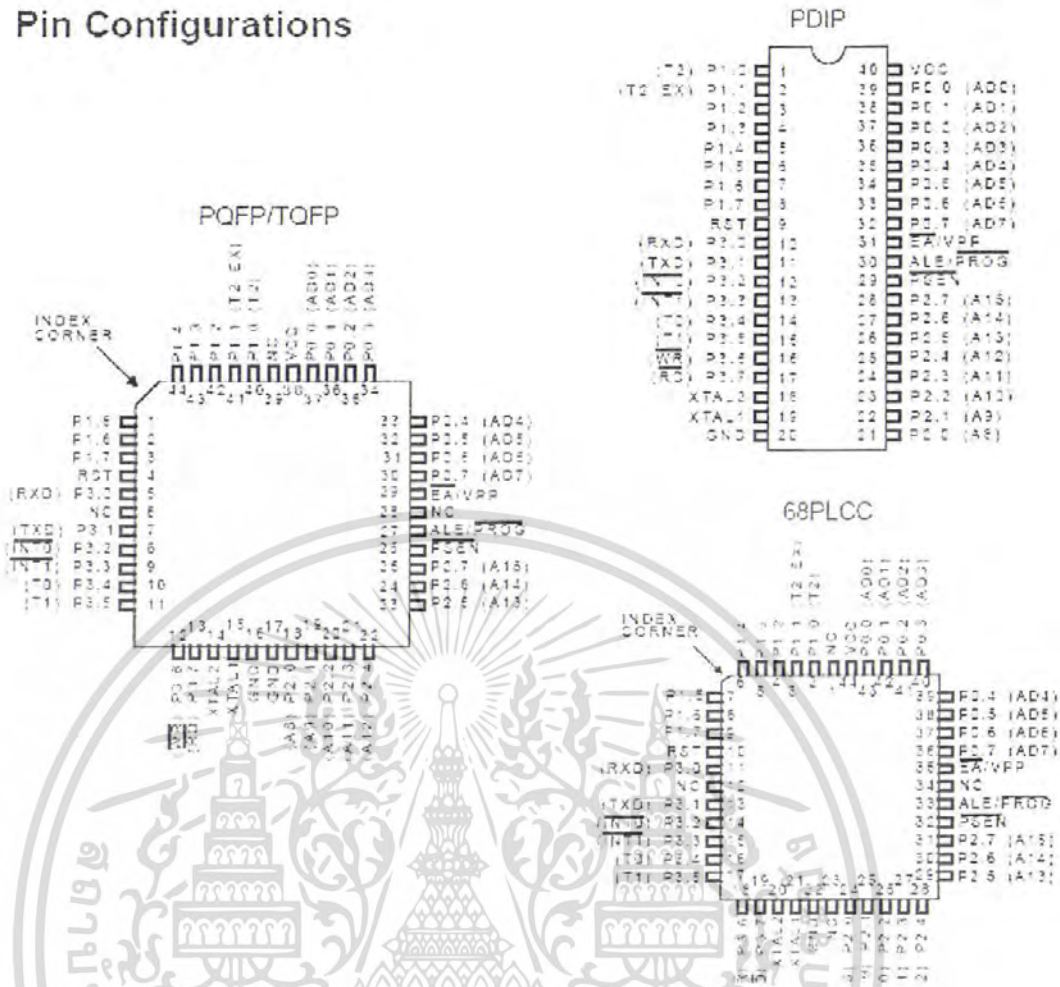
2.8.2 โครงสร้างภายในของ 8051

MCS-51 ใช้เทคโนโลยีในการผลิตแบบ NMOS และ CMOS เบอร์ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวกที่จะเขียนโปรแกรมด้วยภาษาเบสิค โครงสร้างภายในสำหรับเบอร์ 8051 แสดงดังรูปที่ 2.15 และการจัดวางขาดังรูปที่ 2.16

Block Diagram



รูปที่ 2.15 โครงสร้างภายใน MCS-51



รูปที่ 2.16 การจัดวางขา MCS-51

2.8.3 พอร์ตของ 8051

8051 เป็นไมโครคอนโทรลเลอร์ขนาด 40 ขาซึ่งมีขาต่างๆดังนี้

-ไฟเลี้ยง(ขา40) ต่อกับ +5V

-ไฟเลี้ยง(ขา20) เป็นขา GND

-พอร์ต 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (P0.0-P0.7) ใช้งานได้ 2 หน้าที่แอดเดรสและข้อมูลออกไปให้หน่วยความจำภายนอกเมื่อทำการเขียนข้อมูลลงในหน่วยความจำภายนอกเมื่อทำการเขียนข้อมูลลงในหน่วยความจำภายนอกและอีกหน้าที่หนึ่งคือเป็นพอร์ตอินพุตและเอาต์พุต ถ้าต้องการทำงานเป็นอินพุตต้องส่งลอจิก 1 ไปยังพอร์ตนี้

-พอร์ต 1 (ขา 1-8) มีทั้งหมด 8 บิตคือ (P1.0-P1.7) มีโครงสร้างคล้าย พอร์ต 0 แต่จะใช้ความต้านทานภายในพูลอัพแทน (Internal Pull Up Register)

-พอร์ต 2 (ขา 21-28) มีทั้งหมด 8 บิต คือขา (P2.0-P2.7) มีโครงสร้างคล้าย พอร์ต 0 โดยมี FET ตัวล่างตัวเดียว ส่วนด้านบนใช้ความต้านทานพูลอัพแทน พอร์ตนี้ทำงาน 2 หน้าที่ คือสามารถใช้เป็นพอร์ตสำหรับส่งแอดเดรส 8 บิตบน (A8-A15) และเป็นอินพุต,เอาต์พุตพอร์ตใช้งานทั่วไปเมื่อจะใช้งานเป็นอินพุตพอร์ต ต้องส่งลอจิก 1 มาที่พอร์ตนี้ก่อน

-พอร์ต 3 (ขา 10-17) มีทั้งหมด 8 บิต คือขา (P3.0-P3.7) มีโครงสร้างคล้ายพอร์ต 1 พอร์ตนี้ทำหน้าที่เป็นอินพุตและเอาต์พุตพอร์ต ถ้าจะให้พอร์ตนี้เป็นอินพุตพอร์ตก็ให้ส่งลอจิก 1 มาที่พอร์ตนี้ก่อนและอีกหน้าที่หนึ่งคือ ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไป สัญญาณต่างๆ มีดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม

P3.1/RXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

P3.2/INT0 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.3/INT1 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.4/T0 (Timer/Counter 0 External input) ขารับสัญญาณเข้าไปในวงจร
ยัง Timer/Counter 0 ที่ทำหน้าที่นับจำนวนไซเคิลของสัญญาณ T1นี้หรือสัญญาณนาฬิกาก็ได้ ซึ่งมีการทำงานเหมือนกับ T0

P3.5/T1 (Timer/Counter1 External input) ขารับสัญญาณเข้าไปยัง Timer/Counter1 ซึ่งมีการทำงานเหมือนกับ T0

P3.6/WR (External Data Memory write strobe) ขาสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

P3.7/RD (External Data Memory Read strobe) ขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอก

-ALE (ขา 30) เป็นขาส่ง สโตรบสำหรับใช้ในการแลตช์แอดเดสไบต์ต่ำ (A0-A7) ที่ส่งออกมาจาก (พอร์ต 0) สัญญาณนี้จะแอคทีฟทุกๆ ครั้งใน 1 แมชชีนไซเคิล (1/16 ของสัญญาณนาฬิกา)

-PSEN (ขา 29) เป็นขาที่ส่งสโตรบำหรับอ่านข้อมูลจาก Program Memory ภายนอก (หน่วยความจำประเภท ROM EPROM) สัญญาณนี้จะส่งออกมา 2 ครั้ง ในแต่ละแมชชีนไซเคิล แต่ถ้าเป็นการอ่าน Internal Program Memory จะไม่มีสัญญาณนาฬิกาออกมาจากขานี้

-EA (ขา 30) ถ้าป้อนลอจิก 0 เข้าที่ขานี้ CPU จะอ่านค่าจาก Program Memory ภายนอกชิพเท่านั้นแต่ถ้าถูกป้อนด้วยลอจิก 1 ก็จะอ่านโปรแกรมภายในชิพ

-RST (ขา 9) เป็นขารีเซ็ทCPU จะรีเซ็ทก็ต่อเมื่อป้อนลอจิก 1 เข้าที่ขานี้นานอย่างน้อย 2 แมชชีนไซเคิล เมื่อ CPU ถูกรีเซ็ทค่าต่างๆ ในรีจีสเตอร์ใดๆ จะมีคำสั่งดังตารางที่ 2.1

-XTAL1 (ขา 19) ใช้ต่อคริสตัลภายนอกโดยเป็นอินพุตเข้าสู่วงจรออสซิเลเตอร์

-XTAL2 (ขา 18) ใช้ต่อคริสตัลภายนอกโดยเป็นเอาต์พุตของวงจรออสซิเลเตอร์

2.8.4 การแบ่งประเภทของหน่วยความจำ

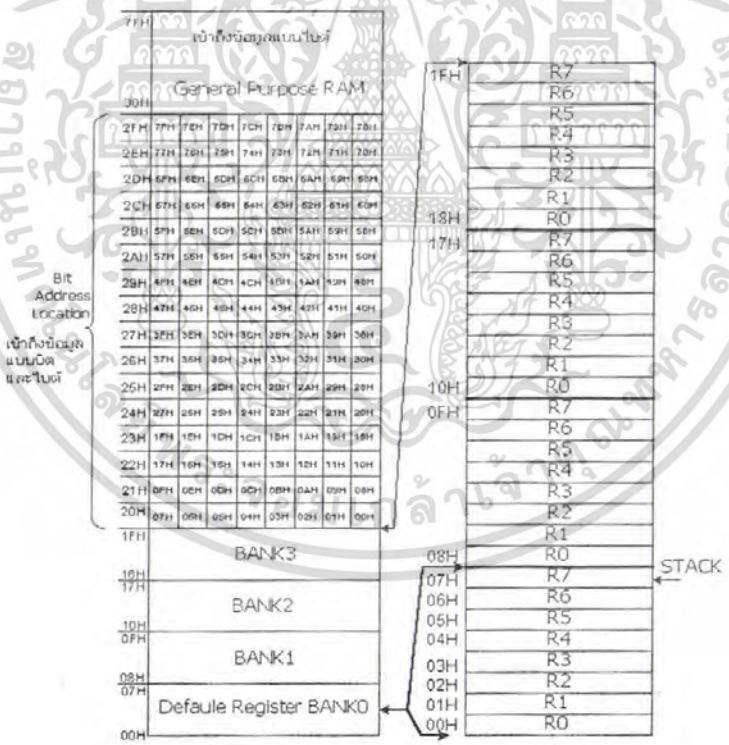
หน่วยความจำที่ใช้กับ MCS-51 มีอยู่ด้วยกัน 2 ชนิดคือ Program Memory และ Data Memory Program Memory ซึ่งเป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุในชิพ 8051 ส่วนที่เป็น Program Memory ก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง แต่ถ้าเป็นเบอร์ 8052 ก็คือ ROM ขนาด 8 กิโลไบต์

Data Memory เป็นหน่วยความจำ ที่ใช้เก็บข้อมูลหน่วยความจำนี้ สามารถเขียนข้อมูลลงไปและอ่านข้อมูลออกมาได้ ซึ่งเป็นหน่วยความจำภายในชิพมีเพียง 128 ไบต์ สำหรับเบอร์ 8051 และ 256 ไบต์ สำหรับเบอร์ 8052 ส่วนหน่วยความจำภายนอกชิพมี 64 กิโลไบต์

พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม (Direct and Indirect Address Area) พื้นที่ 128 ไบต์ ล่างสุดจะแบ่งเป็น 3 ส่วน

1. รีจิสเตอร์แบงค์ (Register Banks 0-3) ตั้งแต่ตำแหน่ง (00H-1FH) จะเป็นส่วนของรีจิสเตอร์แบงค์ มี0-3 โดยแบ่งเป็นแบงค์ละ 8 ไบต์รวมแล้วได้ 32 ไบต์ ถ้าCPUทำงานอยู่ที่แบงค์ 3 เมื่อถูกรีเซ็ทก็จะกลับมาทำงานที่แบงค์ 0 เสมอ และ SP จะมาเริ่มที่ตำแหน่ง 07H ทันที
2. บริเวณหน่วยความจำที่ใช้คำสั่งเขียนอ่านเกี่ยวกับบิตได้ (Bit Addressable Area) พื้นที่ตั้งแต่แอดเดรส (20H-7FH) จำนวน 16 ไบต์ หรือถ้ารับนับเป็นบิตจะได้เท่ากับ 128 บิต ซึ่งตำแหน่งบิต 00,01,02,03,04,06,07 ก็คือตำแหน่งหน่วยความจำตำแหน่งหน่วยความจำตำแหน่ง 20H ที่บิต 0,1,2,3,4,5,6,7

ตารางที่ 2.2 แสดงตำแหน่งของหน่วยความจำภายใน



3. บริเวณหน่วยความจำที่ใช้งานทั่วไป (Scratch Pad Area) พื้นที่ตั้งแต่ (30H-7FH) จะเขียนข้อมูลได้ทีละไบต์เท่านั้น ไม่สามารถใช้คำสั่งเกี่ยวกับบิตได้
   - ช่วง 78H-37H คือช่วง SCRATCH PAD AREA
   - ช่วง 28H-27H คือช่วง BIT ADDRESSABLE SEGMENT
   - ช่วง 18H-07H คือช่วง REGISTER BANK

2.8.5 ส่วนการสร้างฐานเวลาและการนับ (Timer/Counter)

โดยพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 จะมี Timer ให้เราใช้งานได้อย่างน้อย 2 ตัวคือ T0 (Timer 0) และ T1 (Timer 1) ซึ่งสามารถใช้งานได้อย่างอิสระ เมื่อใช้งานเป็น Timer ก็คือการนับจำนวนพัลส์สัญญาณนาฬิกาภายใน สำTimer 0 และ Timer 1 สามารถแยกเป็นรีจีสเตอร์ขนาด 8 บิต คือ TH0 กับ TL0 สำหรับรีจีสเตอร์ T0 และ TH1 กับ TL1 สำหรับรีจีสเตอร์ T1 และรีจีสเตอร์ที่สำคัญในการควบคุมการทำงานของ Timer นี้ คือ TMOD (Timer mode control register) และ TCON (Timer/Counter control register) โดยที่รีจีสเตอร์ TCON สามารถอ้างถึงได้ในระดับบิต TMOD เป็นรีจีสเตอร์กำหนดการทำงานว่าเป็นตัวนับหรือจับเวลาและโหมดการทำงาน ไม่สามารถเข้าถึงในระดับบิต

ตารางที่ 2.3 แสดงตำแหน่งรีจิสเตอร์ของ TMOD

Timer/Counter Mode Control Register :TMOD

ตารางที่ 2.4 แสดงหน้าที่ของแต่ละตำแหน่งของรีจิสเตอร์

ชื่อบิต	ตำแหน่ง	ความหมาย
GATE1	TMOD.7	บิตควบคุม GATE สำหรับ Timer 1 (บิตนี้เซตวงจรจำทำงานเมื่อ /INT1 เป็น High)
C/T1	TMOD.6	บิตกำหนดการทำงานว่าเป็น Timer หรือ Counter (1=Counter, 0=Timer)
M1	TMOD.5	บิตบนกำหนดโหมดการทำงานของ Timer 1
M0	TMOD.4	บิตล่างกำหนดโหมดการทำงานของ Timer 1
GATE0	TMOD.3	บิตควบคุม GATE สำหรับ Timer 0
C/T0	TMOD.2	บิตกำหนดการทำงานว่าเป็น Timer หรือ Counter
M1	TMOD.1	บิตบนกำหนดโหมดการทำงานของ Timer 0
M0	TMOD.0	บิตล่างกำหนดโหมดการทำงานของ Timer 0

จากตารางด้านบน จะเห็นว่าบิตของ TMOD จะแบ่งออกเป็นสองส่วนอย่างชัดเจนคือ 4 บิตบนจะเป็นของ T1 และ 4 บิตล่างจะเป็นของ T0 เมื่อเราใช้งานเป็นตัวจับเวลา(Timer) รีจีสเตอร์จะทำการเพิ่มค่าขึ้นที่ละหนึ่งในทุก ๆ แมชชีนไซเคิลของการทำงานของ CPU นั้นคือการจับเวลาเป็นการนับหน่วยเวลาที่ถูกสร้างมาจากวงจรออสซิลเลเตอร์ของ CPU นั้นเอง โดยการคำนวณค่าระยะเวลาของ หนึ่งแมชชีนไซเคิลจะใช้เวลาเท่ากับ 12 คาบเวลาของสัญญาณนาฬิกา(คาบเวลาของออสซิลเลเตอร์จำนวน 12 คาบ)

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์ = ความถี่ของสัญญาณนาฬิกา/12

1 แมชชีนไซเคิล = 1/ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์

ตัวอย่างเช่น ไมโครคอนโทรลเลอร์ ทำงานที่สัญญาณนาฬิกา 12 MHz (คริสตอล) หนึ่งแมชชีนไซเคิลจะเท่ากับดังนี้

1 แมชชีนไซเคิล = 1/(12MHz/12)

= 1/(1MHz)

= 1 ไมโครวินาที

<u>การกำหนดโหมดของ Timer</u>

ในการกำหนดโหมดการทำงานของ Timer เราจะกำหนดในบิต M0 และ M1 ซึ่งมีรายละเอียดการกำหนดดังนี้

ตารางที่ 2.5 แสดงโหมดของ Timer

M1	M0	โหมดการทำงาน	ความหมาย
0	0	0	จับเวลา ในโหมด 0 โดยใช้ 13 บิต คือ TH0 หรือ TH1 เป็นตัวนับขนาด 8 บิต และใช้ TL0 และ TL1 ขนาด 5 บิต
0	1	1	จับเวลา ในโหมด 1 โดยใช้ 16 บิต โดยที่ TH0 หรือ TH1 จะเก็บค่าไบต์บน และ TL0 หรือ TL1 จะเก็บค่าไบต์ล่าง
1	1	2	จับเวลา ในโหมด 2 โดยใช้ 8 บิต คือ TL0 หรือ TL1 และจะใช้ TH0 หรือ TH1 เป็นตัวเก็บค่าเริ่มต้นของการนับ เมื่อเกิดโอเวอร์โฟลว์จาก FFH เป็น 00H ระบบจะนำค่าใน TH0 หรือ TH1 มาใส่ให้กับ TL0 หรือ TL1 โดยอัตโนมัติที่ใช้

1	1	3	เรียกว่า Auto Reload จับเวลา ในโหมด 3 นี้จะใช้งานได้เพียง Timer 0 เท่านั้น คือ TL0 และ TH0 เมื่อเกิดโอเวอร์โฟลว์บิต TF0 และ TF1 จะถูกเซ็ต ควบคุมโดย รีจีสเตอร์ TCON

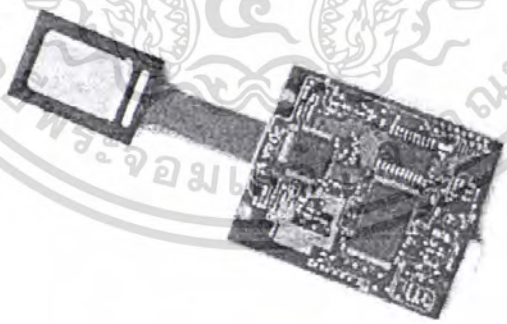
TCON เป็นรีจิสเตอร์ ใช้สำหรับควบคุมทำงานของ Timer/Counter และระดับการตอบรับอินเตอร์รัพท์ ซึ่งสามารถอ้างอิงระดับบิตได้

ตารางที่ 2.6 แสดงรีจิสเตอร์ TCON

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF1	*TR1*	*TF0*	*TR0*	IE1	IT1	IE0	IT0

Timer/counter control register : TCON (Bit addressable)

2.9 คุณสมบัติของโมดูลสแกนลายนิ้วมือรุ่น MRB200 มีดังนี้



รูปที่ 2.17 โมดูลสแกนลายนิ้วมือรุ่น MRB200

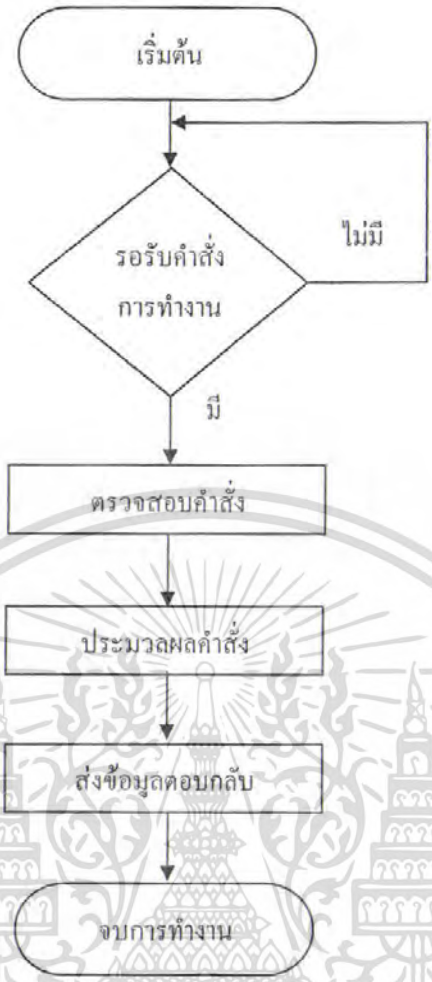
1. เลนส์สแกนเป็นแบบชนิด Capacitive Sensor
2. เก็บบันทึกลายนิ้วมือได้สูงสุด 1,000 ลายนิ้วมือ
3. สามารถวางนิ้วมือเอียงได้ 45 องศากับเลนส์สแกน
4. ความเร็วในการสแกนเพื่อตรวจสอบลายนิ้วมือน้อยกว่า 2 วินาที

5. ตั้งระดับความสำคัญของลายนิ้วมือได้ 2 รูปแบบ คือ User และ Manager
6. ช่วงระดับแรงดันการทำงานอยู่ในช่วง 4 โวลท์ ถึง 6.5 โวลท์
7. ใช้กระแส 170 มิลิแอมป์ ที่แรงดัน 5 โวลท์เมื่อมีการตรวจจับลายนิ้วมือ
8. ใช้กระแสเพียง 80 ไมโครแอมป์ ที่แรงดัน 5 โวลท์เมื่อเข้าสู่โหมด Standby
9. ค่าความผิดพลาดในการยืนยันบุคคลน้อยกว่า 0.001 เปอร์เซ็นต์
10. เชื่อมต่อกับคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม
11. ความเร็วสูงสุดในการส่งข้อมูล 38,400 บิตต่อวินาที
12. ทำงานในช่วงอุณหภูมิ 30 องศา ถึง 50 องศา
13. มีโหมดควบคุมการประหยัดพลังงาน

รูปที่ 2.18 ตำแหน่งขาของโมดูลสแกนลายนิ้วมือรุ่น MRB200 ที่ใช้เชื่อมต่อกับอุปกรณ์ภายนอก

ตารางที่ 2.7 รายละเอียดในการทำงานในแต่ละขาของโมดูลสแกนลายนิ้วมือรุ่น MRB200

ขาที่	คุณสมบัติ
1	กราวน์
2	RX : เพื่อรับคำสั่งหรือข้อมูลต่างๆ จากพอร์ต RS-232
3	TX : เพื่อส่งคำสั่งหรือข้อมูลต่างๆ ออกทางพอร์ต RS-232
4	FINGER ON : ใช้ตรวจสอบสถานะของเลนส์สแกน - สถานะลอจิก0 หมายถึงไม่มีลายนิ้วมือวางอยู่บนเลนส์สแกน - สถานะลอจิก1 หมายถึง มีลายนิ้วมือวางอยู่บนเลนส์สแกน
5	SLEEP : ขาควบคุมโหมดการประหยัดพลังงาน - สถานะลอจิก0 หมายถึง เลือกใช้โหมดการประหยัดพลังงาน - สถานะลอจิก1 หมายถึง ยกเลิกใช้โหมดการประหยัดพลังงาน
6	แรงดันไฟเลี้ยง โดยมีระดับการทำงานในช่วง 4 โวลท์ ถึง 6.5 โวลท์



รูปที่ 2.19 หลักการทำงานของโมดูลบอร์ดสแกนลายนิ้วมือรุ่น MRB200

จากโฟลว์ ชาร์ตข้างบน เป็นหลักการทำงานของโมดูลบอร์ดสแกนลายนิ้วมือรุ่นMRB200 ซึ่งการทำงานของโมดูลจะเริ่มจากทำการรอรับคำสั่งการทำงานจากคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์จากตำแหน่งขา RX หากมีคำสั่งการทำงานเข้ามาโมดูลสแกนลายนิ้วมือจะทำการตรวจสอบคำสั่งที่ได้รับว่าเป็นคำสั่งการทำงานลักษณะใด จากนั้นจะทำการประมวลผลจากคำสั่งที่ได้รับและนำข้อมูลที่ได้จากการประมวลผลส่งกลับมาออกมาทางตำแหน่งขา TX เพื่อส่งข้อมูลที่ได้จากการประมวลผลไปยังคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์

2.10 รูปแบบของชุดคำสั่งในการเขียนโปรแกรมรับส่งกับโมดูลสแกนลายนิ้วมือ รุ่น MRB200

ในการส่งข้อมูล(Data) หรือ คำสั่ง(Command) จากคอมพิวเตอร์ไปยังตัวโมดูลสแกนลายนิ้วมือรุ่น MRB200 จะส่งข้อมูลผ่านพอรต์อนุกรมของคอมพิวเตอร์ โดยข้อมูลหรือคำสั่งจะมีรูปแบบเป็นชุดคำสั่ง หรือ แพกเกจ ซึ่งใน 1 ชุดคำสั่งจะประกอบด้วยชุดข้อมูลจำนวน 8 ไบต์ ซึ่งจะเป็นรูปแบบ เลขฐาน 16 ทั้งหมด ดังตารางที่ 1

ตารางที่ 2.8 รูปแบบชุดคำสั่งที่ใช้รับส่งข้อมูลเพื่อการสั่งงานโมดูลสแกนลายนิ้วมือรุ่น MRB200

ไบต์ที่ 1	ไบต์ที่ 2	ไบต์ที่ 3	ไบต์ที่ 4	ไบต์ที่ 5	ไบต์ที่ 6	ไบต์ที่ 7	ไบต์ที่ 8
0xFE	0x00	0xXX	0xXX	0xXX	0xXX	0xXX	0xFD

ไบต์ที่ 1 หมายถึง ตำแหน่งเริ่มต้นของชุดคำสั่ง โดยจะมีข้อมูล FE เป็นตัวกำหนด

ไบต์ที่ 2 หมายถึง ตำแหน่งนี้จะเป็นแบงก์ว่าง ซึ่งจะมีข้อมูล คือ 00

ไบต์ที่ 3 หมายถึง ตำแหน่งแสดงสถานะการทำงานในโหมดต่างๆ เช่น

02 หมายถึง โหมดบันทึกลายนิ้วมือ

12 หมายถึง โหมดตรวจสอบลายนิ้วมือ

20 หมายถึง โหมดลบลายนิ้วมือ

ไบต์ที่ 4 และ 5 หมายถึง รหัสลายนิ้วมือ หรือ ตำแหน่งลายนิ้วมือ เช่น

00 01 หมายถึง รหัสลายนิ้วมือที่ 1

00 0F หมายถึง รหัสลายนิ้วมือที่ 15

02 00 หมายถึง รหัสลายนิ้วมือที่ 200

ซึ่งในรหัสหรือตำแหน่งลายนิ้วมือนั้น ตัวโมดูลสแกนลายนิ้วมือ จะให้เราใส่รหัสหรือตำแหน่งลายนิ้วมือได้สูงสุดที่รหัส 3999

ไบต์ที่ 6 หมายถึง ระดับความสำคัญของลายนิ้วมือ

03 หมายถึง Manager หรือผู้จัดการ

02 หมายถึง User หรือผู้ใช้งานทั่วไป

ในระดับความสำคัญของลายนิ้วมือ จะมีประโยชน์ในกรณีที่นำไปประยุกต์ใช้งานที่เกี่ยวกับการตรวจสอบระดับความสำคัญของบุคคลหรือลายนิ้วมือ ซึ่งไบต์นี้เราสามารถทำการกำหนดระดับความสำคัญของลายนิ้วมือได้ และจะช่วยให้เราสามารถตรวจสอบระดับความสำคัญของลายนิ้วมือได้

ไบต์ที่ 7 หมายถึง การตรวจเช็คความผิดพลาดในการรับส่งข้อมูลหรือการตรวจเช็คข้อมูล โดยการนำข้อมูลในไบต์ที่ 3 ไบต์ที่ 4 ไบต์ที่ 5 และไบต์ที่ 6 มาทำการเอกซ์คูลซีฟออร์ (Xor) ก็จะได้ข้อมูลตำแหน่งไบต์ที่ 7

ไบต์ที่ 8 หมายถึง ตำแหน่งสิ้นสุดข้อมูลของชุดคำสั่ง โดยจะมีข้อมูล FD เป็นตัวกำหนด

ตัวอย่างคำสั่งที่ใช้ในการสั่งงานโมดูลสแกนลายนิ้วมือ

FE 00 02 00 01 02 01 FD หมายถึง การบันทึกลายนิ้วมือลายนิ้วที่ 0001 โดยบันทึกในรูปแบบ User หรือผู้ใช้งานทั่วไป

FE 00 20 00 01 00 21 FD หมายถึง การลบลายนิ้วมือในตำแหน่งหรือรหัสลายนิ้วที่ 0001

FE 00 21 00 01 02 23 FD หมายถึง เป็นคำสั่งที่ใช้ในการลบลายนิ้วมือทั้งหมดในตัวโมดูล

จากตัวอย่างคำสั่งข้างต้นเป็นเพียงส่วนหนึ่งของคำสั่งที่ใช้ในการติดต่อกับโมดูล ซึ่งโดยแท้จริงแล้วคำสั่งที่ใช้ในการติดต่อกับโมดูลนั้นมีอยู่หลายคำสั่งด้วยกัน

ต่อไปจะทำการอธิบายหลักการเขียนโปรแกรมสั่งงานโมดูลสแกนลายนิ้วมือ โดยในที่นี้จะขอยกตัวอย่างหลักการเขียนโปรแกรมในคำสั่งที่สำคัญ 2 คำสั่ง คือ คำสั่งบันทึกข้อมูลลายนิ้วมือ และคำสั่งการตรวจสอบข้อมูลลายนิ้วมือ

ตัวอย่างการใช้งานคำสั่งบันทึกข้อมูลลายนิ้วมือ

เพื่อความเข้าใจที่ดียิ่งขึ้นและสามารถนำหลักการดังกล่าวไปใช้งานจริง ผมจะทำการยกตัวอย่างขั้นตอนการเขียนโปรแกรมในคำสั่งบันทึกข้อมูลลายนิ้วมือ โดยจะทำการกำหนดค่าต่างๆ ในตัวอย่างดังนี้ คือ บันทึกรหัสลายมือที่รหัส 1234 และ กำหนดระดับความสำคัญของลายนิ้วมือเป็นแบบ Manager

วิธีการให้เริ่มจากให้เราทำการเปลี่ยนรหัสลายมือในรูปแบบเลขฐานสิบให้อยู่ในรูปแบบเลขฐานสิบหก จากตัวอย่างที่เราได้กำหนดรหัสลายนิ้วมือ คือ **1234** เมื่อเราทำการแปลงค่าให้อยู่ในรูปแบบเลขฐานสิบหกจะได้ค่าเป็น **04D2** จากนั้นนำค่าที่ได้จากแปลงมาแบ่งออกเป็น 2 ไบต์ ซึ่งจะได้ ข้อมูลที่ได้ทำการแบ่ง คือ **04** และ **D2** โดยจะนำค่าที่ได้จากแบ่งข้อมูลของรหัสลายนิ้วมือไปใส่ในแพจเกจในตำแหน่งไบต์ที่ 4 และไบต์ที่ 5 เมื่อเราได้ค่าต่างๆ ครบแล้วจากนั้นให้นำค่าที่ได้มาจัดให้อยู่ในรูปแบบคำสั่งดังตารางที่ 1 ซึ่งจะได้ค่าต่างๆ ที่จะทำการบันทึกข้อมูลลายนิ้วมือในตัวอย่างนี้ดังตารางที่ 2.9

ตารางที่ 2.9 รูปแบบชุดคำสั่งที่ใช้สั่งงานโมดูลให้บันทึกลายนิ้วมือในรหัสลายนิ้วมือที่ 1234

ไบต์ที่ 1	ไบต์ที่ 2	ไบต์ที่ 3	ไบต์ที่ 4	ไบต์ที่ 5	ไบต์ที่ 6	ไบต์ที่ 7	ไบต์ที่ 8
0xFE	0x00	0x02	0x04	0xD2	0x03	0xD7	0xFD

จากตารางที่ 2.8 จะสังเกตข้อมูลในไบต์ที่ 7 ซึ่งข้อมูลที่แสดงอยู่นั้นเป็นการนำข้อมูลจากตำแหน่งไบต์ที่ 3 จนถึงไบต์ที่ 6 มาทำการเอกซ์คูลซีฟออร์ (Xor)

จากนั้นนำค่าที่ได้จากตารางที่ 2 ส่งออกไปให้กับโมดูลสแกนลายนิ้วมือ ซึ่งขบบวนการส่งข้อมูลและรับข้อมูลที่โมดูลส่งกลับมาจะมีการรับส่งด้วยกันทั้งหมด 6 ครั้งด้วยกัน โดยมีขบวนการทำงาน 7 ขั้นตอน ดังนี้

1. ทำการส่งค่าข้อมูลที่ได้จากตารางที่ 2 โดยส่งออกไปให้กับโมดูลสแกนลายนิ้วมือ ซึ่งมีข้อมูล คือ FE 00 02 04 D2 03 D7 FD

2. หลังจากนั้นให้ทำการสแกนลายนิ้วมือ (โดยจะต้องวางลายนิ้วมือที่เลนส์สแกนเพื่อทำการตรวจสอบจนกว่าจะสิ้นสุดขบวนการตรวจสอบข้อมูลลายนิ้วมือ) จากนั้นจะมีข้อมูลที่โมดูลส่งกลับมา โดยข้อมูลนี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลง ไม่ว่าเราจะบันทึกข้อมูลตำแหน่งหรือรหัสลายนิ้วมือใดก็ตาม ซึ่งข้อมูลที่โมดูลตอบกลับมา คือ FE 00 42 00 00 00 42 FD

3. เมื่อเราได้รับข้อมูลดังกล่าวที่โมดูลส่งกลับมา ให้เราทำการส่งข้อมูลออกไปให้กับโมดูลสแกนลายนิ้วมือเป็นครั้งที่สอง โดยให้เราทำการเปลี่ยนข้อมูลในไบต์ที่ 3 ให้เป็นข้อมูลใหม่ โดยมีข้อมูลในไบต์ที่ 3 คือ **04** ซึ่งข้อมูลในไบต์นี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลงไม่ว่าเราจะบันทึก ข้อมูลตำแหน่งหรือรหัสลายนิ้วมือใดก็ตาม จากนั้นให้เราทำการเอกซ์คูลซีฟออร์ (Xor) อีกครั้ง เพื่อให้ได้ข้อมูลที่ตำแหน่งไบต์ที่ 7 ซึ่งข้อมูลที่ได้ คือ **D1** โดยข้อมูลในตำแหน่งในไบต์อื่นๆ ให้คงค่าเดิมเอาไว้ จากนั้นนำค่าที่ได้ทั้งหมดส่งออกไปให้กับตัวโมดูลสแกนลายนิ้วมืออีกครั้ง ซึ่งมีข้อมูล คือ **FE 00 04 04 D2 03 D1 FD** ; เป็นไบต์ที่เราได้ทำการเปลี่ยนข้อมูล

4. หลังจากนั้นจะมีของข้อมูลที่โมดูลส่งกลับมา โดยข้อมูลนี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลงไม่ว่าเราจะบันทึกข้อมูลตำแหน่งหรือรหัสลายนิ้วมือใดก็ตาม ซึ่งข้อมูลที่โมดูลตอบกลับมา คือ **FE 00 44 00 00 00 44 FD**

5. จากนั้นเมื่อเราได้รับข้อมูลดังกล่าวที่โมดูลส่งกลับมา ให้เราทำการส่งข้อมูลออกไปให้กับโมดูลสแกนลายนิ้วมือเป็นครั้งที่สาม โดยให้เราทำการเปลี่ยนข้อมูลในไบต์ที่ 3 ให้เป็นข้อมูลใหม่ โดยมีข้อมูลในไบต์ที่ 3 คือ **03** ซึ่งข้อมูลในไบต์นี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลงไม่ว่าเราจะบันทึก ข้อมูลตำแหน่งหรือรหัสลายนิ้วมือใดก็ตาม จากนั้นให้เราทำการเอกซ์คูลซีฟออร์ (Xor) อีกครั้ง เพื่อให้ได้ข้อมูลที่ตำแหน่งไบต์ที่ 7 ซึ่งข้อมูลที่ได้ คือ **D1** โดยข้อมูลในตำแหน่งในไบต์อื่นๆ ให้คงค่าเดิมเอาไว้ จากนั้นนำค่าที่ได้ทั้งหมดส่งออกไปให้กับตัวโมดูลสแกนลายนิ้วมืออีกครั้งซึ่งมีข้อมูล คือ **FE 00 03 04 D2 03 D6 FD** ; เป็นไบต์ที่เราได้ทำการเปลี่ยนข้อมูล

6. หลังจากนั้นจะมีข้อมูลที่โมดูลส่งกลับมา ซึ่งข้อมูลในขั้นตอนนี้จะเป็นข้อมูลที่ช่วยบอกให้เราทราบว่าขบวนการบันทึกลายนิ้วมือเสร็จสมบรูณ์ โดยข้อมูลนี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลง ไม่ว่าเราจะบันทึกข้อมูลตำแหน่งหรือรหัสลายนิ้วมือใดก็ตาม ซึ่งข้อมูลที่โมดูลตอบกลับมา คือ **FE 00 43 00 00 00 43 FD**

7. หลังจากนั้นเมื่อเราได้รับข้อมูลดังกล่าวที่โมดูลส่งกลับมาก็เป็นการเสร็จสิ้นขบวนการบันทึกลายนิ้วมือ จากนั้นเราก็สามารถยกนิ้วมือออกจากเลนส์สแกนได้

โมดูลตรวจสอบข้อมูลแล้วตอบกลับมามีสองกรณีคือ

1. กรณีที่ไม่มีข้อมูลลายนิ้วมือ

ในกรณีที่ไม่มีข้อมูลลายนิ้วมือนั้นอยู่ในตัวโมดูลหรือโมดูลตรวจสอบข้อมูลดังกล่าวไม่พบตัวโมดูลก็จะทำการส่งข้อมูลกลับมาดังข้อมูลในรูปแบบตารางที่ 4 ซึ่งหากเราได้รับข้อมูลดังตารางที่ 4 ก็จะทำให้เราสามารถตรวจเช็คได้ว่าไม่มีข้อมูลลายนิ้วมือดังกล่าวในตัวโมดูลนี้

ตารางที่ 2.10 รูปแบบชุดคำสั่ง ในกรณีที่มีไม่ข้อมูลลายนิ้วมือตอบกลับ

ไบต์ที่ 1	ไบต์ที่ 2	ไบต์ที่ 3	ไบต์ที่ 4	ไบต์ที่ 5	ไบต์ที่ 6	ไบต์ที่ 7	ไบต์ที่ 8
0xFE	0x00	0x12	0x00	0x00	0x00	0x12	0xFD

2. กรณีที่มีข้อมูลลายนิ้วมือ

ในกรณีที่มีข้อมูลลายนิ้วมือนั้นอยู่ในตัวโมดูล ตัวโมดูลก็จะทำการส่งข้อมูลกลับมาดังข้อมูลในรูปแบบตารางที่ 5 จากนั้นก็สามารถตัวเช็คได้ว่าเป็นรหัสลายนิ้วมือที่เท่าไร โดยการนำข้อมูลในไบต์ที่ 4 และไบต์ที่ 5 มาต่อกัน แล้วทำการแปลงข้อมูลให้อยู่ในรูปแบบเลขฐานสิบ ก็จะทำให้เราทราบได้ว่าลายนิ้วมือที่ได้ทำการสแกนเป็นรหัสลายนิ้วมือที่เท่าไร

ตารางที่ 2.11 รูปแบบชุดคำสั่งที่โมดูลสแกนลายนิ้วมือตอบกลับ ในกรณีที่มีข้อมูลลายนิ้วมือ

ไบต์ที่ 1	ไบต์ที่ 2	ไบต์ที่ 3	ไบต์ที่ 4	ไบต์ที่ 5	ไบต์ที่ 6	ไบต์ที่ 7	ไบต์ที่ 8
0xFE	0x00	0x12	0x00	0x00	0x00	0x12	0xFD

เช่นในกรณีที่ข้อมูลในไบต์ที่ 4 และไบต์ที่ 5 มีข้อมูล คือ **04** และ **D2** จากนั้นให้เรานำข้อมูลมาต่อกันจะได้เป็นข้อมูล คือ **04D2** หลังจากนั้นให้ทำการแปลงข้อมูลดังกล่าวให้อยู่ในรูปแบบเลข ฐานสิบ จะได้ข้อมูลที่ทำการแปลงแล้ว คือ **1234** ซึ่งจะทำให้เราทราบได้ทันทีว่าลายนิ้วมือที่ได้ทำการสแกนเป็นรหัสลายนิ้วมือที่ 1234 นั้นเอง

2.11 โครงสร้างของภาษา SQL

ภาษา SQL (สามารถอ่านออกเสียงได้ 2 แบบ คือ "เอสคิวแอล" (SQL) หรือ "ซีเควล" (Sequel)) ย่อมาจาก Structured Query Language หรือภาษาในการสอบถามข้อมูล เป็นภาษาทางด้านฐานข้อมูล ที่สามารถสร้างและปฏิบัติการกับฐานข้อมูลแบบสัมพันธ์(relational database)โดยเฉพาะ และ เป็นภาษาที่มีลักษณะคล้ายกับภาษาอังกฤษ ภาษา SQLถูกพัฒนาขึ้นจากแนวคิดของ relational calculus และ relational algebra เป็นหลัก ภาษา SQL เริ่มพัฒนาครั้งแรกโดย almaden research center ของบริษัท IBM โดยมีชื่อเริ่มแรกว่า "ซีเควล" (Sequel) ต่อมาได้เปลี่ยนชื่อเป็น"เอสคิวแอล" (SQL) หลังจากนั้นภาษา SQL ได้ถูกนำมาพัฒนาโดยผู้ผลิตซอฟแวร์ด้านระบบจัดการฐานข้อมูลเชิงสัมพันธ์จนเป็นที่นิยมกันอย่างแพร่หลายในปัจจุบัน โดยผู้ผลิตแต่ละรายก็พยายามที่จะพัฒนาระบบจัดการฐานข้อมูลของตนให้มีลักษณะเด่นเฉพาะขึ้นมา ทำให้รูปแบบการใช้คำสั่ง SQL มีรูปแบบที่แตกต่างกันไปบ้าง เช่น ORACLE ACCESS SQL Base ของ Sybase INGRES หรือ SQL Server ของ Microsoft เป็นต้น

ดังนั้นในปี ค.ศ. 1986 ทางด้าน American National Standards Institute (ANSI) จึงได้กำหนดมาตรฐานของ SQL ขึ้น อย่างไรก็ดี โปรแกรมฐานข้อมูลที่ขายในท้องตลาด ได้ขยาย SQL ออกไปจนเกินข้อกำหนดของ ANSI โดยเพิ่มคุณสมบัติอื่นๆ ที่คิดว่าเป็นประโยชน์เข้าไปอีกแต่โดยหลักทั่วไปแล้วก็ยังปฏิบัติตามมาตราฐานของ ANSI ในการอธิบายคำสั่งต่างๆของภาษา SQL ในหนังสือเล่มนี้จะอธิบายคำสั่งที่เป็นรูปแบบคำสั่งมาตราฐานของภาษา SQL โดยทั่วไป

2.11.1ประเภทของคำสั่งของภาษา SQL

ภาษา SQL เป็นภาษาที่ใช้งานได้ตั้งแต่ระดับเครื่องคอมพิวเตอร์ส่วนบุคคลพีซีไปจนถึงระดับเมนเฟรม ประเภทของคำสั่งในภาษา SQL (The subdivision of sql) แบ่งออกเป็น 3 ประเภท คือ1. ภาษาสำหรับการนิยามข้อมูล (Data Definition Language : DDL) ประกอบด้วยคำสั่งที่ใช้ในการกำหนดโครงสร้างข้อมูลว่ามีคอลัมน์อะไร แต่ละคอลัมน์เก็บข้อมูลประเภทใด รวมถึงการเพิ่มคอลัมน์ การกำหนดดัชนี การกำหนดวิวหรือตารางเสมือนของผู้ใช้ เป็นต้น2. ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language : DML) ประกอบด้วยคำสั่งที่ใช้ในการเรียกใช้ข้อมูล การเปลี่ยนแปลงข้อมูล การเพิ่มหรือลบข้อมูล เป็นต้น

2.11.2 ชนิดของข้อมูลที่ใช้ในภาษา SQL

ในภาษา SQL การบรรจุข้อมูลลงในคอลัมน์ต่าง ๆ ของตารางจะต้องกำหนดชนิดของข้อมูล (data type) ให้แต่ละคอลัมน์ ชนิดของข้อมูลนี้จะแสดงชนิดของค่าที่อยู่ในคอลัมน์ ค่าทุกค่าในคอลัมน์ที่กำหนดจะต้องเป็นชนิดเดียวกัน เช่น ในตารางลูกค้าคอลัมน์ที่เป็นรายชื่อลูกค้า จะต้องเป็นตัวหนังสือ ในขณะที่คอลัมน์จำนวนเงินที่ลูกค้าซื้อสินค้าเป็นตัวเลขชนิดของข้อมูลของแต่ละคอลัมน์จะขึ้นกับลักษณะของข้อมูลแต่ละคอลัมน์ ซึ่งแบ่งได้ดังนี้ชนิดข้อมูลพื้นฐานในภาษา SQL ดังนี้2.1 ตัวหนังสือ(character) ในภาษา SQL จะใช้- ตัวหนังสือแบบความยาวคงที่(fixed-length character) จะใช้ char (n) หรือ character(n) แทนประเภทของข้อมูลที่เป็นตัวหนังสือใดๆที่มีความยาวของข้อมูลคงที่โดยมีความยาว n ตัวหนังสือประเภทนี้จะมีการจองเนื้อที่ตามความยาวที่คงที่ตามที่กำหนดไว้ ชนิดของข้อมูลประเภทนี้จะเก็บความยาวของข้อมูลได้มากที่สุดได้ 255 ตัวอักษร- ตัวหนังสือแบบความยาวไม่คงที่(variable-length character) จะใช้ varchar (n) แทนประเภทของข้อมูลที่เป็นตัวหนังสือใดๆที่มีความยาวของข้อมูลไม่คงที่ โดยมีความยาว n ตัวหนังสือประเภทนี้จะมีการจองเนื้อที่ตามความยาวของข้อมูล ชนิดของข้อมูลประเภทนี้จะเก็บความยาวของข้อมูลได้มากที่สุดได้ 4000 ตัวอักษร2.2 จำนวนเลข(numeric)- จำนวนเลขที่มีจุดทศนิยม(decimal) ในภาษา SQL จะใช้ dec(m,n) หรือ decimal(m,n) เป็นประเภทข้อมูลที่เป็นจำนวนเลขที่มีจุดทศนิยมโดย m คือจำนวนตัวเลขทั้งหมด (รวมจุดทศนิยม) และ n คือจำนวนตัวเลขหลังจุดทศนิยม- จำนวนเลขที่ไม่มีจุดทศนิยมในภาษา SQL จะใช้ int หรือ integer เป็นเลขจำนวนเต็มบวกหรือลบขนาดใหญ่ เป็นตัวเลข 10 หลัก ที่มีค่าตั้งแต่ –2,147,483,648 ถึง +2,147,483,647 และในภาษา SQL จะใช้ smallint เป็นประเภทข้อมูลที่เป็นเลขจำนวนเต็มบวกหรือลบขนาดเล็ก เป็นตัวเลข 5 หลัก ที่มีค่าตั้งแต่ –32,768 ถึง + 32,767 ตัวเลขจำนวนเต็มประเภทนี้จะมีการจองเนื้อที่น้อยกว่าแบบ integer- เลขจำนวนจริง ในภาษา SQL อาจใช้ number(n)แทนจำนวนเลขที่ไม่มีจุดทศนิยมและจำนวนเลขที่มีจุดทศนิยม2.3 ข้อมูลในลักษณะอื่นๆ

- วันที่และเวลา(Date/Time) เป็นชนิดวันที่หรือเวลาในภาษา SQL จะใช้ **date** เป็นข้อมูลวันที่ ซึ่งจะมีหลายรูปแบบให้เลือกใช้ เช่น yyyy-mm-dd (1999-10-31) dd.mm.yyyy(31. 10.1999) หรือ dd/mm/yyyy (31/10/1999)

2.11.3 ลักษณะการใช้งานของภาษา SQL

ภาษา SQL เป็นส่วนประกอบหนึ่งของ DBMS มักพบใน DBMS เชิงสัมพันธ์หลายตัวและเป็นที่นิยมใช้ในปัจจุบัน ภาษา SQL ง่ายต่อการเรียนรู้ การใช้งานในภาษา SQLแบ่งเป็น 2 ลักษณะ คือ ภาษา SQL ที่โต้ตอบได้ (interactive SQL)และภาษา SQL ที่ฝังในโปรแกรม (embedded SQL)3.1 ภาษา SQL ที่โต้ตอบได้ ใช้เพื่อปฏิบัติงานกับฐานข้อมูลโดยตรง เป็นการใช้คำสั่งภาษา SQLสั่งงานบนจอภาพ โดยเรียกดูข้อมูลได้โดยตรงในขณะที่ทำงาน เพื่อให้ได้ผลลัพธ์ที่นำไปใช้ได้

2.11.4 ภาษา SQL ที่ฝังในโปรแกรม

เป็นภาษา SQL ที่ประกอบด้วยคำสั่งต่าง ๆ ของ ภาษา SQL ที่ใส่ไว้ในโปรแกรมที่ส่วนมากแล้วเขียนด้วยภาษาอื่น เช่น โคบอล ปาสคาล ภาษาซี ลักษณะของคำสั่ง SQL จะแตกต่างจากภาษาอื่นๆ ในแง่ที่ว่า SQL ไม่มีคำสั่งที่เกี่ยวกับการควบคุม(control statement)เหมือนภาษาอื่น เช่น if..then…else for…do หรือ loop หรือ while ทำให้มีข้อจำกัดในการเขียนชุดคำสั่งงาน การใช้ภาษา SQL ฝังในโปรแกรมอื่นจะทำให้ภาษา SQL มีความสามารถและมีประสิทธิภาพมากยิ่งขึ้น ผลลัพธ์ของคำสั่งที่เกิดจากภาษา SQL ที่ฝังในโปรแกรมจะถูกส่งผ่านไปให้กับตัวแปรหรือพารามิเตอร์ที่ใช้โดยโปรแกรมที่ภาษา SQL ไปฝังตัวอยู่ เช่นwhile not end-of-file(input) dobeginreadin(id-num, salesperson,loc,comm);EXEC SQL INSERT INTO SALESTABVALUES(:id-num,:salesperson,:loc,:comm);end;จากตัวอย่างถ้าใช้คำสั่งINSERT INTO SALESTABVALUES (:id-num,:salesperson,: loc, :comm);เพียงอย่างเดียว จะทำให้คำสั่งนี้ใส่ค่า id-num salesperson loc comm ใส่ค่าได้เพียงครั้งเดียว แต่เมื่อนำคำสั่งนี้มาใส่ไว้ในภาษาปาสคาลข้างต้นจะทำให้คำสั่งดังกล่าวมีความสามารถสูงขึ้นคือคำสั่งนี้จะสามารถทำงานซ้ำ(loop) โดยใส่ค่าต่างๆลงในตัวแปรเพื่อให้ทำซ้ำกันหลายๆครั้ง โดยจากตัวอย่างส่วนของโปรแกรมภาษาปาสคาลจะกำหนดลูปวนซึ่งจะอ่านค่าจากแฟ้มข้อมูลแล้วเก็บค่านั้นไว้ในตัวแปร id-num, salesperson, loc, comm ของตาราง SALESTAB การอ่านค่าแล้วเก็บค่าไว้ในตัวแปรจะทำซ้ำจนกระทั่งข้อมูลหมดจากแฟ้มข้อมูลทั้งภาษา SQL ที่โต้ตอบได้และภาษา SQL ที่ฝังในโปรแกรมจะมีลักษณะของคำสั่งที่ใช้งานเหมือนกันจะต่างกันแต่เพียงภาษา SQL ที่ฝังในโปรแกรมจะมีวิธีการเชื่อมโยงกับภาษาอื่น ๆ

2.12 PHP (PHP Hypertext Preprocessor)

PHP เกิดในปี 1994 โดย Rasmus Lerdorf โปรแกรมเมอร์ชาวสหรัฐอเมริกาได้คิดค้นสร้างเครื่องมือที่ใช้ในการพัฒนาเว็บส่วนตัวของเขา โดยใช้ข้อดีของภาษา C และ Perl เรียกว่า Personal Home Page และได้สร้างส่วนติดต่อกับฐานข้อมูลชื่อว่า Form Interpreter (FI) รวมทั้งสองส่วนเรียกว่า PHP/FI ซึ่งก็เป็นจุดเริ่มต้นของ PHP มีคนที่เข้ามาเยี่ยมชมเว็บไซต์ของเขาแล้วเกิดชอบจึง

ติดต่อขอเอาโค้ดไปใช้บ้าง และนำไปพัฒนาต่อ ในลักษณะของ Open Source ภายหลังมีความนิยมขึ้นเป็นอย่างมากภายใน 3 ปีมีเว็บไซต์ที่ใช้ PHP/FI ในติดต่อฐานข้อมูลและแสดงผลแบบ ไดนามิก และอื่นๆ มากกว่า 50000 ไซต์

PHP เป็นภาษาสคริปต์ที่ประมวลผลที่ฝั่งเซิร์ฟเวอร์ แล้วส่งผลลัพธ์ไปแสดงผลที่ฝั่งไคลเอ็นต์ผ่านบราวเซอร์เช่นเดียวกับ CGI และ ASP ต่อมาเมื่อมีผู้ใช้มากขึ้นจึงมีการร้องขอให้มีการพัฒนาประสิทธิภาพของ PHP/FI ให้สูงขึ้น Rasmus Lerdorf ก็ได้ผู้ที่มาช่วยพัฒนาอีก 2 คนคือ Zeev Suraski และ Andi Gutmans ชาวอิสราเอล ซึ่งปรับปรุงโค้ดของ Lerdorf ใหม่โดยใช้ C++ ต่อมาก็มีเพิ่มเข้ามาอีก 3 คน คือ Stig Bakken รับผิดชอบความสามารถในการติดต่อ Oracle, Shane Caraveo รับผิดชอบดูแล PHP บน Window 9x/NT, และ Jim Winstead รับผิดชอบการตรวจ ความบกพร่องต่างๆ และได้เปลี่ยนชื่อเป็น Professional Home Page

PHP3 ได้ออกสู่สายตาของนักโปรแกรมเมอร์เมื่อ มิถุนายน 1998 ที่ผ่านมาในเวอร์ชั่นนี้มีคุณสมบัติเด่นคือสนับสนุนระบบปฏิบัติการทั้ง Window 95/98/ME/NT, Linux และเว็บเซร์ฟเวอร์อย่าง IIS, PWS, Apache, OmniHTTPd สนับสนุน ระบบฐานข้อมูลได้หลายรูปแบบเช่น SQL Server, MySQL, mSQL, Oracle, Informix, ODBC

เวอร์ชั่นล่าสุดในปัจจุบันคือ PHP4 ซึ่งได้เพิ่ม Functions การทำงานในด้านต่างๆ ให้มากและง่ายขึ้นโดย Zend ซึ่งมี Zeev และ Andi Gutmans ได้ร่วมก่อตั้งขึ้น (http://www.zend.com) ในเวอร์ชั่นนี้จะเป็น compile script ซึ่งในเวอร์ชั่นหน้านี้จะเป็น embed script interpreter ในปัจจุบันมีคนใช้ PHP สูงกว่า 5,100,000 sites แล้วทั่วโลก ผู้พัฒนาได้ตั้งชื่อของง PHP ใหม่ว่า PHP: Hypertext Preprocessor ซึ่งหมายถึงมีประสิทธิภาพระดับโปรเฟสเซอร์สำหรับไฮเปอร์เท็กซ์

ความสามารถของ PHP นั้นในความสามารถพื้นฐานที่ภาษาสคริปต์ทั่วๆไปมีนั้น PHP ก็มีความสามารถทำได้ทัดเทียมเช่นเดียวกันเช่น การรับข้อมูลจากฟอร์ม, การสร้าง Content ในลักษณะ Dynamic, รับส่ง Cookies, สร้าง, เปิด, อ่าน และปิดไฟล์ในระบบ, การรองรับระบบจัดการฐานข้อมูลมากมายดังนี้

Adabas D	Ingres	Oracle (OCI7 and OCI8)
Dbase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

โดยผู้จัดทำโครงงานนี้เลือกมาใช้ในบทความนี้คือ MySQL เหตุที่เลือกตัวนี้คือ เป็นที่นิยมกว้างขว้างและประเด็นหนึ่งที่จะต้องพิจารณาเพราะ MySQL จัดเป็น Software ประเภท Freeware รองรับ OS ได้หลายระบบด้วยกัน

เนื่องจากว่า PHP ไม่ได้เป็นส่วนหนึ่งของตัว Web Server ดังนั้นถ้าจะใช้ PHP ก็จะต้องดูก่อนว่า Web server นั้นสามารถใช้สคริปต์ PHP ได้หรือไม่ ยกตัวอย่างเช่น PHP สามารถใช้ได้กับ Apache WebServer และ Personal Web Server (PWP) สำหรับระบบปฏิบัติการ Windows 95/98/NT ในกรณีของ Apache เราสามารถใช้ PHP ได้สองรูปแบบคือ ในลักษณะของ CGI และ Apache Module ความแตกต่างอยู่ตรงที่ว่า ถ้าใช้ PHP เป็นแบบโมดูล PHP จะเป็นส่วนหนึ่งของ Apache หรือเป็นส่วนขยายในการทำงานนั่นเอง ซึ่งจะทำงานได้เร็วกว่าแบบที่เป็น CGI เพราะว่า ถ้าเป็น CGI แล้ว ตัวแปลชุดคำสั่งของ PHP ถือว่าเป็นแค่โปรแกรมภายนอก ซึ่ง Apache จะต้องเรียกขึ้นมาทำงานทุกครั้ง ที่ต้องการใช้ PHP ดังนั้น ถ้ามองในเรื่องของประสิทธิภาพในการทำงาน การใช้ PHP แบบที่เป็นโมดูลหนึ่งของ Apache จะทำงานได้มีประสิทธิภาพมากกว่า

2.13 Web Services

เว็บเซอร์วิส (Web Services) เป็นการ "บริการ" ที่เป็นระบบซอฟต์แวร์ที่ออกแบบมาเพื่อสนับสนุนการทำงาน ระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ผ่านระบบเครือข่าย โดยที่ภาษาที่ใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์ คือภาษาเอ็กซ์เอ็มแอล (XML) ตัวอย่างเช่น การบริการในการเช็คราคาหุ้นของตลาดหุ้นหลาย ๆ ที่และอ่านข่าวจากแหล่งข่าว ๆ หลายที่โดยให้เฉพาะข่าวของบริษัทที่ผู้ขอใช้บริการสนใจ ผู้ให้บริการเว็บเซอร์วิสหนึ่งอาจจะเป็นผู้ขอบริการเว็บเซอร์วิสอื่น ยกตัวอย่างเช่น เว็บเซอร์วิสที่ให้บริการข้อมูลก่อนการซื้อขายหุ้น อาจจะเป็นผู้ขอใช้บริการของเว็บเซอร์วิสที่ให้บริการการให้ข่าว ความสามารถของเว็บเซอร์วิสที่ทำให้โปรแกรมคุยกับโปรแกรมได้นั้น เป็นจุดแข็งของเว็บเซอร์วิส ที่สามารถจะเชื่อมบริการหลายๆอันเข้าด้วยกัน แนวความคิดนี้ได้ถูกนำมาวางแผนและนำเสนอมาตรฐานที่จะทำให้เว็บเซอร์วิส ติดต่อกันได้อย่างมีประสิทธิภาพเช่น การใช้เอกสารภาษา WSDL (Web Services Description Language) ซึ่งเป็นภาษา XML ประเภทหนึ่ง WSDL (Web Services Description Language) ที่มาอธิบายการเรียกใช้เว็บเซอร์วิสซึ่งเปรียบเสมือนการอ่านคู่มือการใช้งานโปรแกรมนั่นเอง แต่ทว่ามีข้อแตกต่างกันตรงที่ไม่เฉพาะมนุษย์เท่านั้นที่สามารถเข้าใจคู่มือนั่น โปรแกรมที่สามารถอ่านเอกสารภาษา XML เข้าใจสามารถที่จะเข้าใจเอกสาร WSDL ได้เช่นกัน ซึ่งจากคุณสมบัตินี้ช่วยทำให้การเรียกใช้เว็บเซอร์วิสเป็นไปได้อย่างอัตโนมัติ

นอกจาก XML จะถูกใช้ในการเป็นภาษาในการอธิบายการเรียกใช้เว็บเซอร์วิสแล้ว XML ยังเป็นภาษาที่ใช้ในการบันทึกข้อมูลระหว่างผู้ให้บริการและผู้ขอใช้การบริการเว็บเซอร์วิส รูปแบบของข้อมูล XML ที่ใช้ในการติดต่อนี้เรียกว่าSOAP (Simple Object Access Protocol) เนื่องจากข้อมูลที่ติดต่ออยู่ในรูปแบบ XML ทำให้โปรแกรมต่าง ๆ สามารถติดต่อกันได้ ถึงแม้ว่าอาจจะถูก

พัฒนาและเรียกใช้บนแพลตฟอร์มที่แตกต่างกัน หรือใช้ภาษาที่แตกต่างกันในการพัฒนา ทั้งนี้เนื่องจาก XML เป็นภาษาอักขระ (text) ซึ่งระบบปฏิบัติการทุกระบบสามารถเข้าใจ นอกจากนี้การที่ XMLมีแท็ก (tag) และรูปแบบโครงสร้างที่อธิบายข้อมูลด้วยตัวมันเอง ทำให้การเข้าใจและการจัดการข้อมูล SOAP messages นั้นสามารถทำได้โดยโปรแกรมและช่วยทำให้การติดต่อระหว่าง ผู้ให้บริการและผู้ใช้เว็บเซอร์วิสเป็นไปได้อย่างอัตโนมัติ

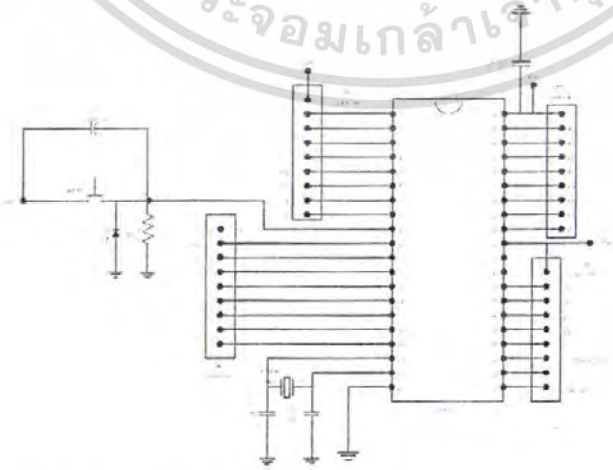
บทที่3
โครงสร้าง และการออกแบบ

เนื่องจากการทำงานในภาคเรียนที่ 2 ปีการศึกษา 2550 พบว่า โมดูลสแกนลายนิ้วมือรุ่น MRB 200 ได้เกิดปัญหาขึ้นสันนิษฐานว่าเกิดจากการเสื่อมตามเวลา ทำให้โหมดการลงทะเบียนลายนิ้วมือ (Enrollment) ที่จะจดจำลายนิ้วมือทั้งหมด 3 ครั้งเกิดอาการขัดข้องไม่สามารถบันทึกได้ ทั้งนี้ผู้จัดทำโครงงานนี้ได้พยายามหาทางซ่อมโมดูลรุ่น MRB 200 อย่างสุดความสามารถแล้ว รวมถึงการติดต่อบริษัทผู้ขายโดยตรง พบว่า โมดูลรุ่นนี้ไม่มีอะไหล่เปลี่ยนและไม่มีการผลิตอีกแล้ว จึงทำให้โครงงานไม่เป็นไปตามที่ผู้จัดคิดหวัง ทางอาจารย์ที่ปรึกษาได้ให้คำปรึกษาและวิธีแก้ไขปัญหาคือ ผู้จัดได้จำลอง Microcontroller ตระกูล MCS-51 ขึ้นมาจำลอง โมดูล Fingerprint รุ่น MRB 200 ซึ่ง โปรโตคอล รับ-ส่ง ข้อมูลระหว่าง Microcontroller หลัก กับ Fingerprint รุ่น MRB 200 เป็นไปตามโปรโตคอลของ โมดูลรุ่น MRB 200 ตามที่ผู้จัดได้อธิบายหลักการทำงานของโปรโตคอลอย่างละเอียดในบทที่ 2 ข้างต้นแล้ว

3.1 การออกแบบบอร์ดหลัก

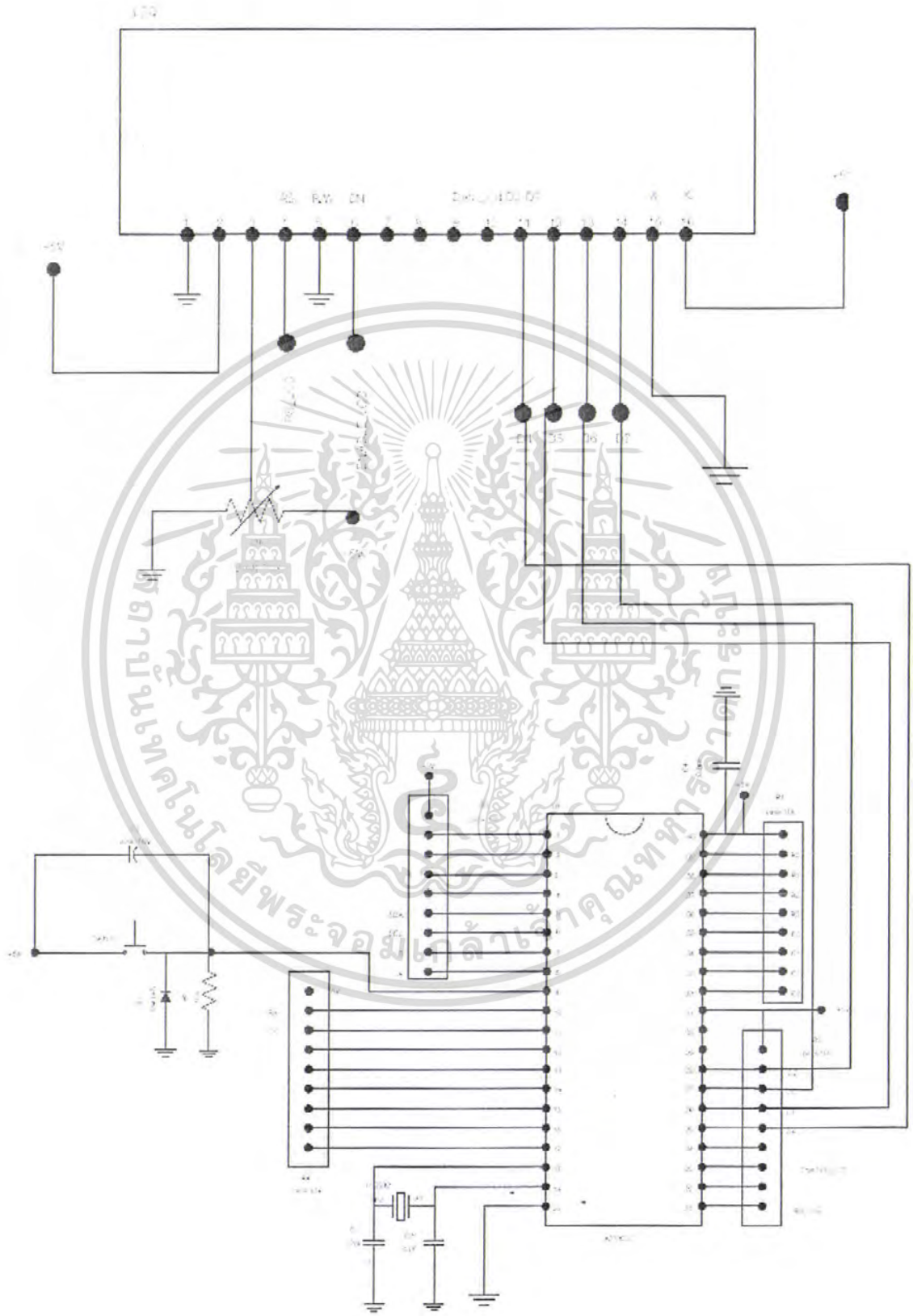
บอร์ดหลักนั้นจะทำหน้าติดต่อกับผู้ใช้งานในการตรวจสอบลายนิ้วมือ และติดต่อกับ PC เพื่อทำการบันทึกข้อมูลเข้าใช้และลงทะเบียนลายนิ้วมือ ผ่านอนุกรม RS232 ส่วนโครงสร้างในส่วนของบอร์ดหลักนั้น จะประกอบไปด้วยองค์ประกอบต่างๆดังนี้

3.1.1 หน่วยประมวลผลกลาง ซึ่งทำหน้าที่ประมวลผลข้อมูลต่างๆที่รับมาจากอุปกรณ์ตัวอื่นๆ และส่งคำสั่งไปเพื่อควบคุมอุปกรณ์ทั้งหมดบนบอร์ด โดยในโครงงานนี้ใช้ Microcontroller ตระกูล MCS-51 ขนาด 40 ขา



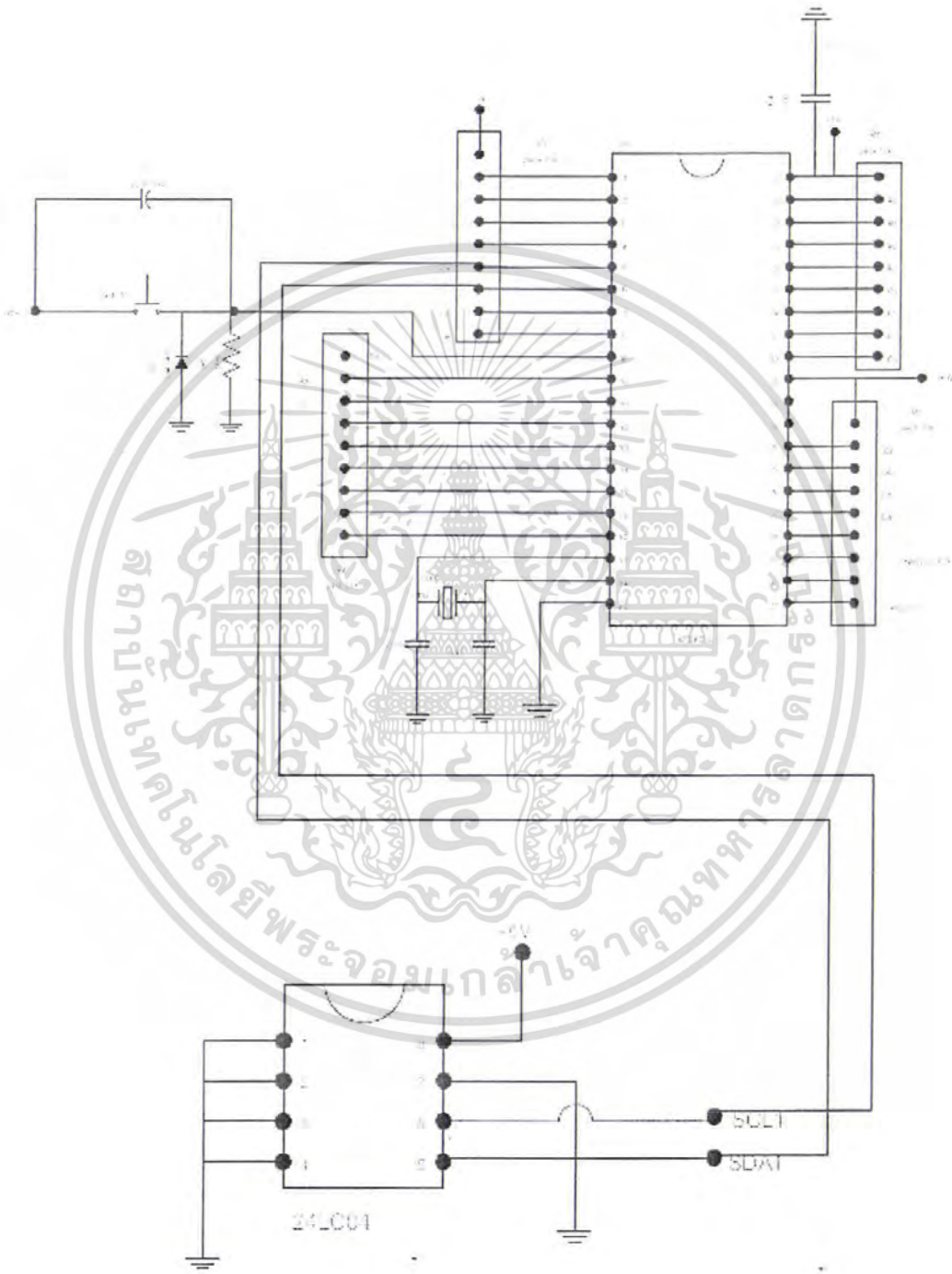
รูปที่ 3.1 แสดง Microcontroller ตระกูล MCS-51 ขนาด 40 ขา

3.1.2 หน่วยแสดงผล ใช้จอ LCD ขนาด 16*2 ทำหน้าที่แสดงผลข้อมูลต่างๆให้ผู้ใช้ได้รับทราบโดยจะติดต่อกับหน่วยประมวลผลกลางแบบ 4 Bit



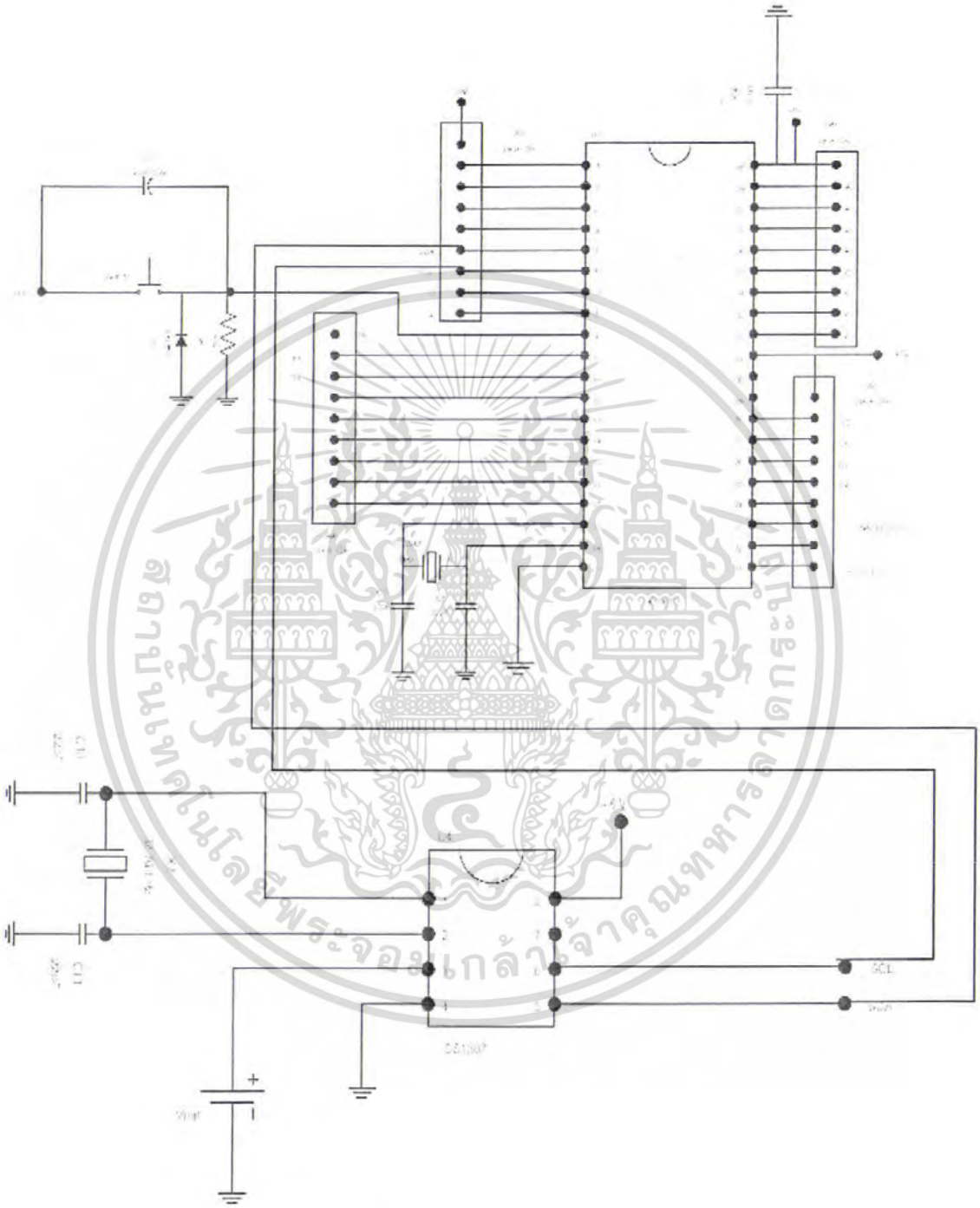
รูปที่ 3.2 แสดงการต่อระหว่าง Microcontrollerกับ LCD

3.1.3 หน่วยเก็บข้อมูลใช้ ใช้หน่วยความจำ EEPROM เบอร์ 24LC04B ซึ่งมีขนาดหน่วยความจำ



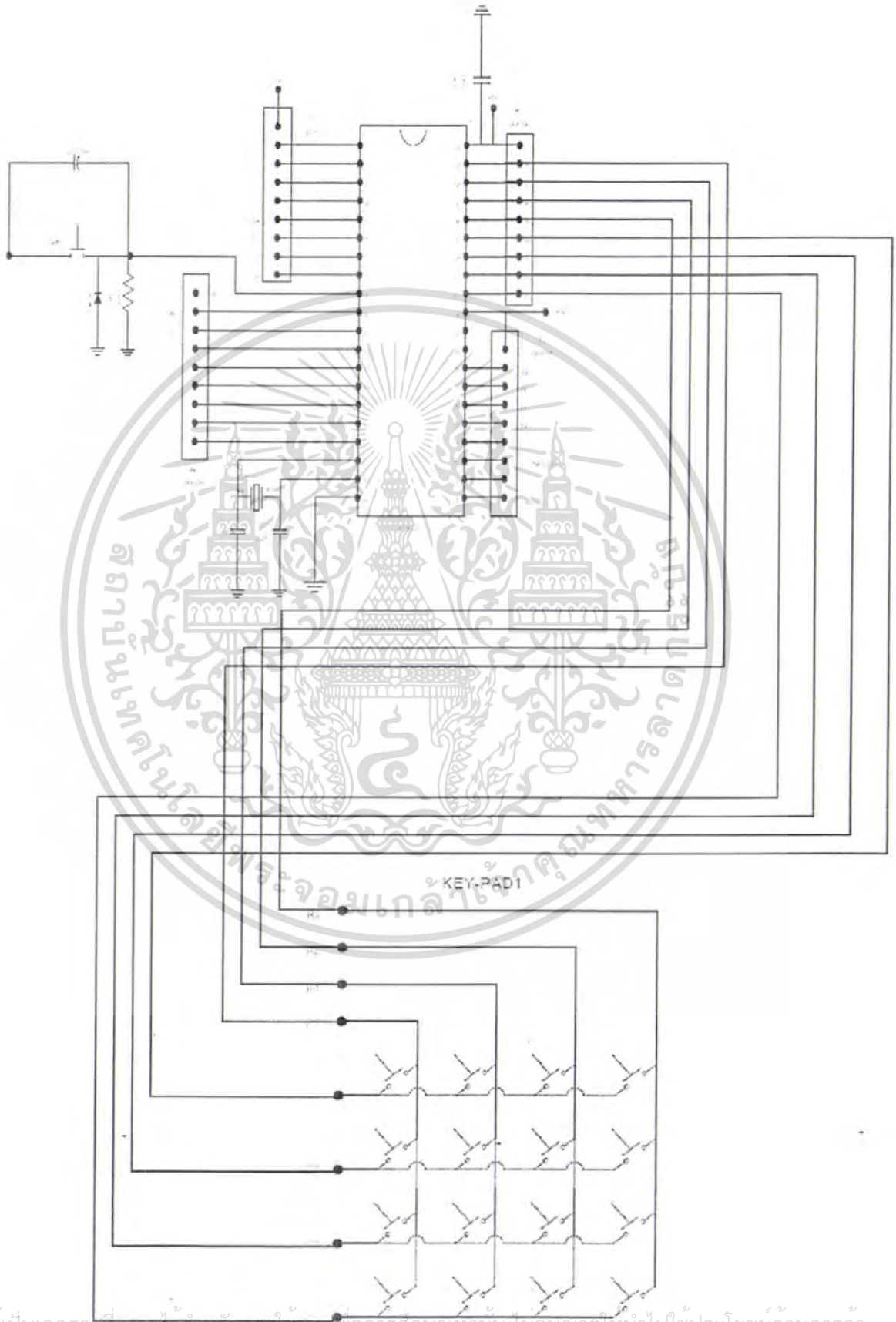
รูปที่ 3.3 แสดงการต่อระหว่าง Microcontrollerกับ EEPROM

3.1.4 IC สร้างฐานเวลาจริง เบอร์ DS1307 ใช้ในการอ้างอิงเวลาในการลงชื่อเข้าใช้ของผู้ใช้งาน กับหน่วยประมวลผลกลางผ่าน Bus I2C



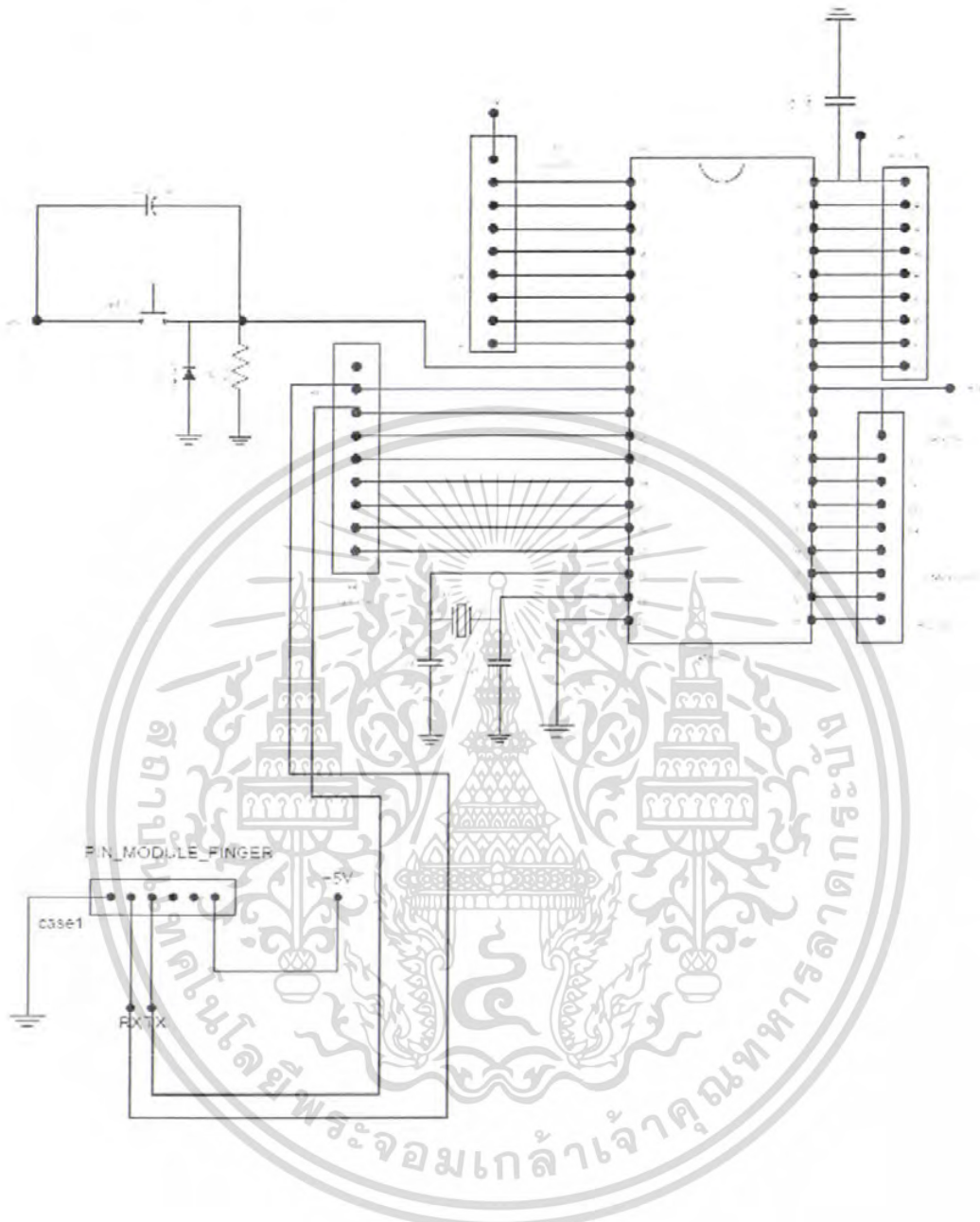
รูปที่ 3.4 แสดงการต่อระหว่าง Microcontrollerกับ DS1307

3.1.5 Keypad 4*4 จำนวน 1 ตัว



รูปที่ 3.5 แสดงการต่อระหว่าง Microcontrollerกับ keypad

3.1.6 Fingerprint module



รูปที่ 3.6 แสดงการต่อระหว่าง Microcontrollerกับ Module

ขั้นตอนการทำงานของส่วนบอร์ดหลักโดยคร่าวจะเป็นดังนี้ ในสภาวะปกติเครื่องจะ Stand by รอให้ผู้ใช้เอานิ้วทาบที่โมดูลแสกนลายนิ้วมือ เมื่อผู้ใช้ทาบนิ้วลงบนโมดูล โมดูลก็จะทำการสแกนลายนิ้วมือของผู้ใช้ หากข้อมูลตรงกันกับลายนิ้วมือที่มีการลงทะเบียนไว้แล้ว ก็จะทำการบันทึกข้อมูลของผู้ใช้ลงใน EEPROM

ในส่วนการทำงานในโหมดการลงทะเบียนและการบันทึกข้อมูลการเข้าใช้งานนั้นจะต้องทำการติดต่อกับ PC ผ่าน RS 232 โดยจะติดต่อกับโปรแกรมบน PC ที่เขียนขึ้นด้วยโปรแกรม PHP

ในส่วนของการเก็บข้อมูลนั้นหน่วยประมวลผลกลางจะทำการจัดเก็บข้อมูลไว้ใน EEPROM 24LC04B ซึ่งมีขนาดหน่วยความจำ 64 kByte จำนวน 1 ตัว และสามารถรองรับผู้ใช้งานได้ถึง 3999 คน โดยการจัดเก็บข้อมูลเข้าใช้งานนั้นจะจัดเก็บวันเวลาที่เข้าใช้โดยอ้างอิงจาก ไอซีเรียลไทม์คล็อก DS1307

3.2 การออกแบบระบบการรับส่งข้อมูล

Protocol รับส่ง

กำหนด U 1 แทน controller Register

กำหนด U 2 แทน Fingerprint

1. ขั้นตอนการลงทะเบียนให้กับรหัสนิ้ว

กดปุ่ม "REG" เพื่อ สู่ การลงทะเบียนเลขประจำตัวผู้ใช้ 4 หลัก

จากนั้น จะเข้าสู่โหมดบันทึกลายนิ้วมือ โดยเราต้องใส่เลขประจำตัว 4 หลักเช่น ID 1234 จากนั้น กด "EN" เพื่อทำการลงทะเบียน

ขั้นตอนที่ 1

ID = DEC ----- > 1234 || HEX -------> 04 D2

FE = BYTE START, FD = BYTE STOP

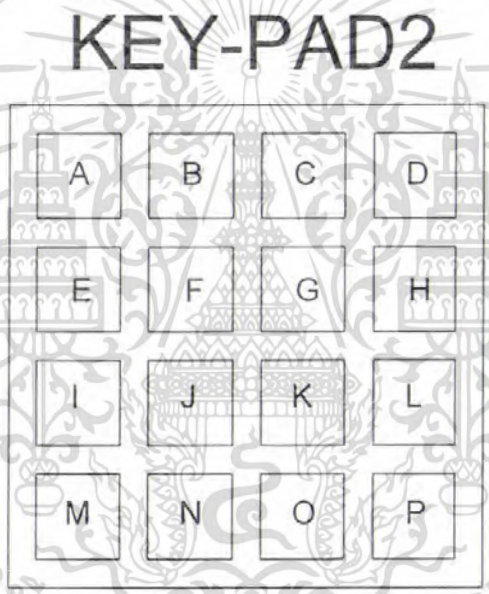
BYTE 7 = BYTE 3 (xor) BYTE 4 (xor) BYTE 5 (xor) BYTE 6

U 1 send command ————————> FE 00 02 04 D2 03 D7 FD

ขั้นตอนที่ 2

เมื่อ U 2 ได้ รับ command จาก U 1

จากนั้นทำการ เลือก รหัส นิ้ว จาก U 2



U 2 send command ————————> FE 00 02 04 D2 03 D7 FD

ซึ่งข้อมูลไม่เปลี่ยนแปลง

ขั้นตอนที่ 3

เมื่อ U 1 ได้ รับ command จาก U 2 จะทำการเปลี่ยนข้อมูลใน BYTE 3 ให้เป็น **04** ซึ่งข้อมูลในไบต์นี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลง

BYTE 7 = BYTE 3 (xor) BYTE 4 (xor) BYTE 5 (xor) BYTE 6

U1 send command ————————> FE 00 04 04 D2 03 D1 FD

ขั้นตอนที่ 4

เมื่อ U2 ได้ รับ command จาก U1

U2 send command ————> FE 00 44 00 00 00 44 FD

ซึ่งข้อมูลไม่เปลี่ยนแปลง

ขั้นตอนที่ 5

เมื่อ U1 ได้ รับ command จาก U2 จะทำการเปลี่ยนข้อมูลใน BYTE 3 ให้เป็น **03** ซึ่งข้อมูลในไบต์นี้จะเป็นข้อมูลคงที่ไม่มีการเปลี่ยนแปลง

BYTE 7 = BYTE 3 (xor) BYTE 4 (xor) BYTE 5 (xor) BYTE 6

U1 send command ————> FE 00 03 04 D2 03 D6 FD

ขั้นตอนที่ 6

เมื่อ U2 ได้ รับ command จาก U1 จะส่งมาว่าเสร็จสมบูรณ์แล้ว

U2 send command ————> FE 00 43 00 00 00 43 FD

ซึ่งข้อมูลไม่เปลี่ยนแปลง

2. ขั้นตอนการตรวจสอบการสแกนนิ้ว

ขั้นตอนที่ 1

กด KEY "SET" ที่ Key pad1

U1 send command ————> FE 00 12 00 00 00 12 FD

ขั้นตอนที่ 2

เมื่อ U2 ได้ รับ command จาก U1

แบ่งเป็น 2 กรณี

- ถ้าไม่มีการกดรหัสนิ้วที่ KEY PAD 2

  U2 send command ————> FE 00 12 00 00 00 12 FD

- ถ้ามีการกดรหัสนิ้วที่ KEY PAD 2

  ตัวอย่างเช่นถ้า รหัสนิ้ว นั้น เป็น ID 1234

  U2 send command ————> FE 00 12 04 D2 12 D6 FD

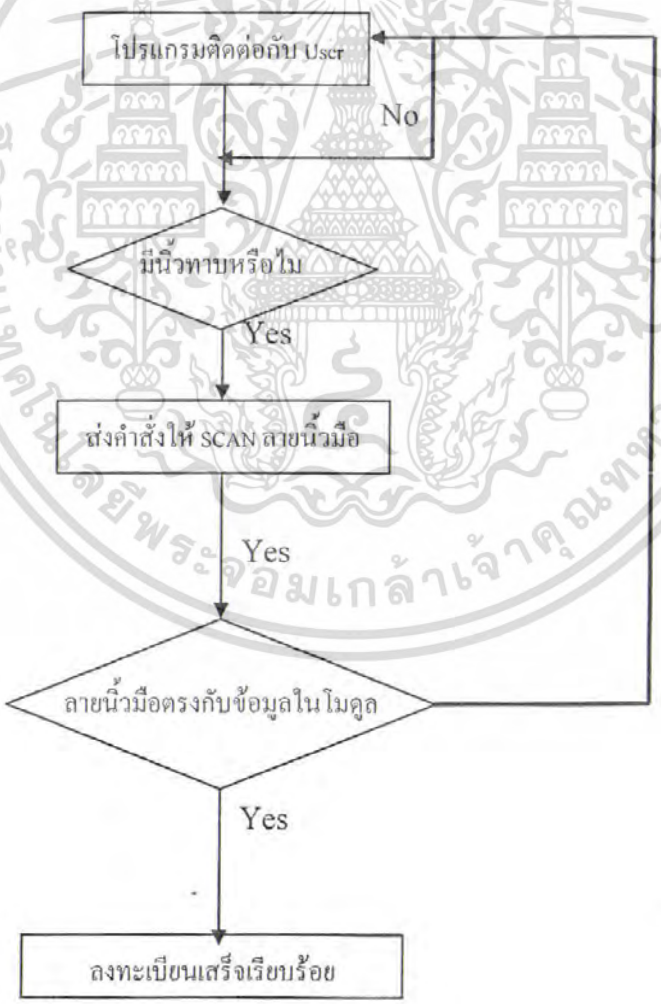
3. ขั้นตอนในการลบ ID ออกจาก U2

ขั้นตอนที่ 1

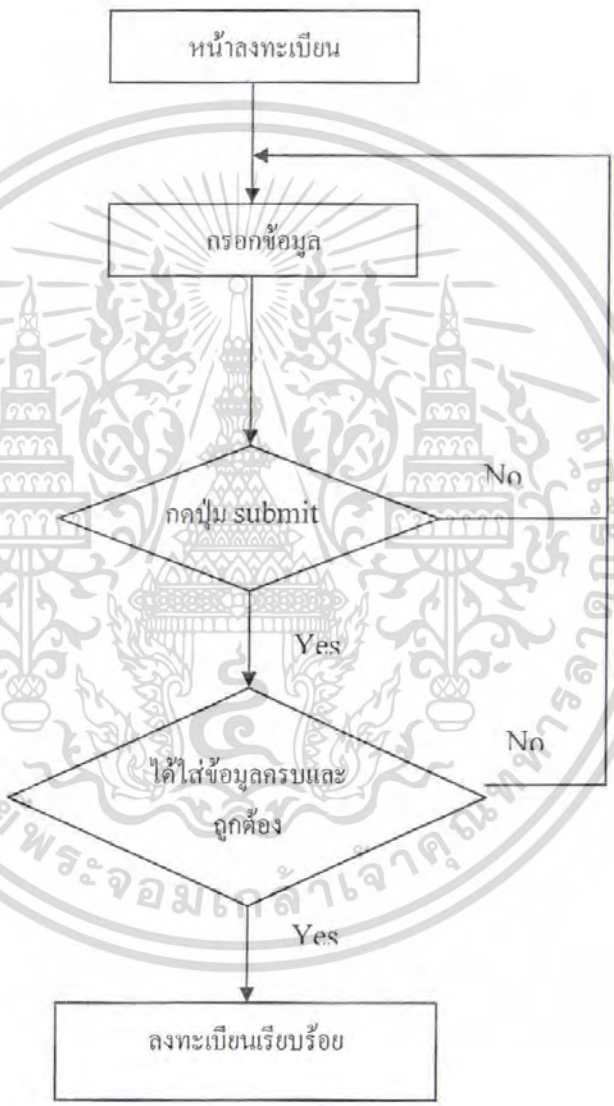
กด KEY "DEL" ที่ Key pad1 เข้า สู่ MODE DEL จากนั้น KEY ID ที่ จะลบ 4 หลัก

ตัวอย่างเช่น ลบ ID 1234

U1 send command ————> FE 00 20 04 D2 00 21 FD

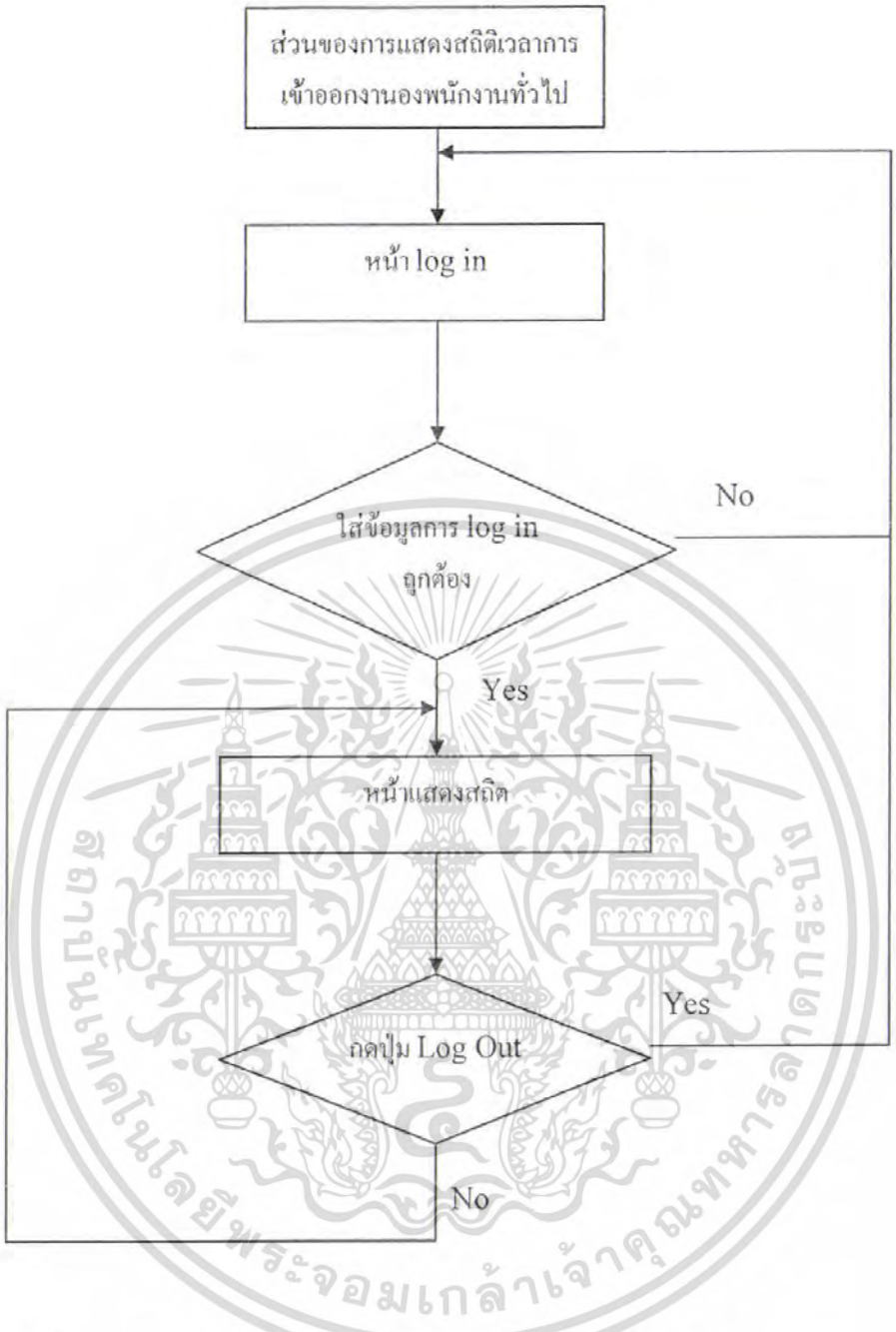


รูปที่ 3.7 แสดงแผนผังโปรแกรมติดต่อกับ User

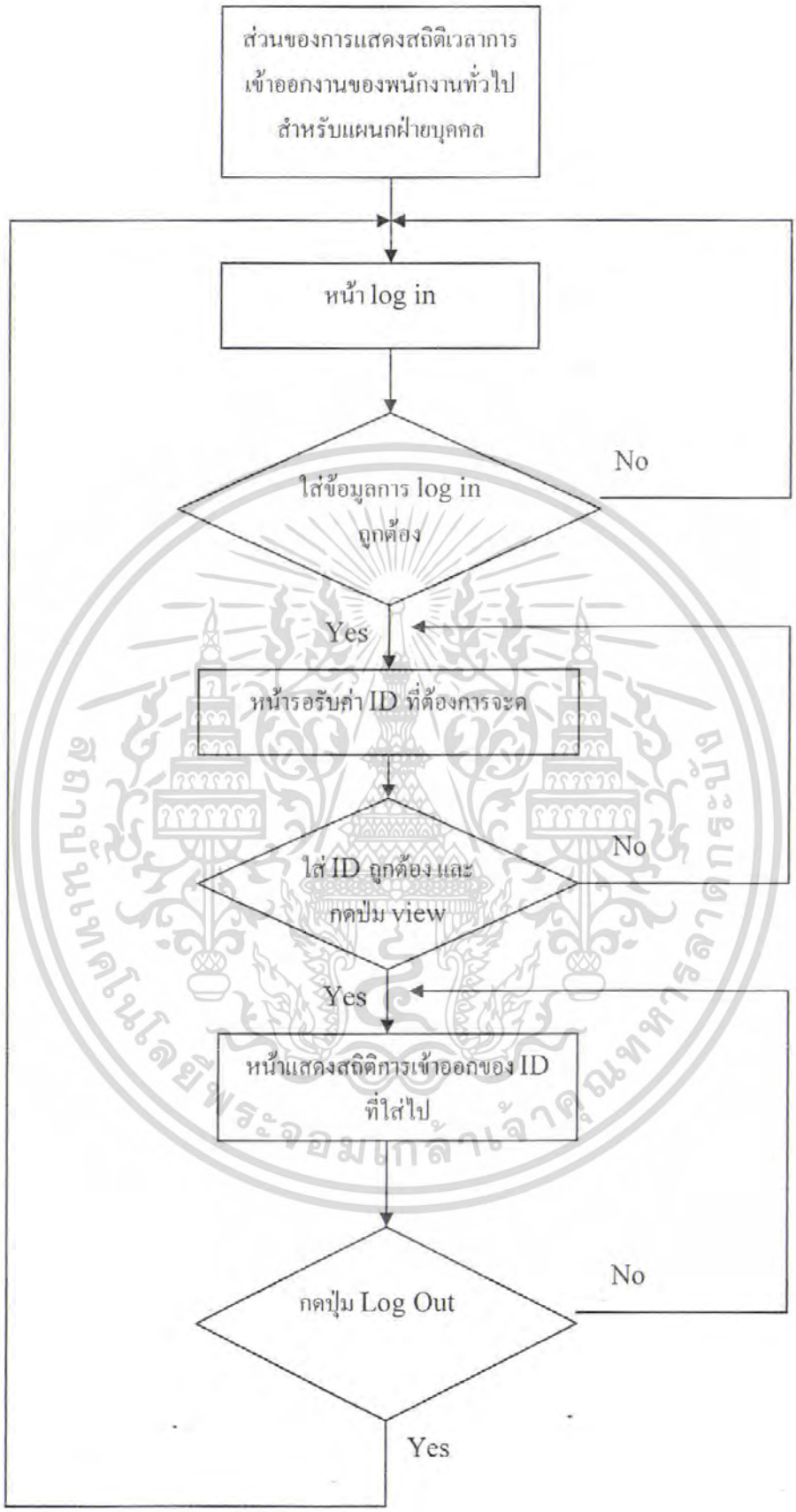


รูปที่ 3.8 แสดงแผนผังการลงทะเบียนผ่าน web service

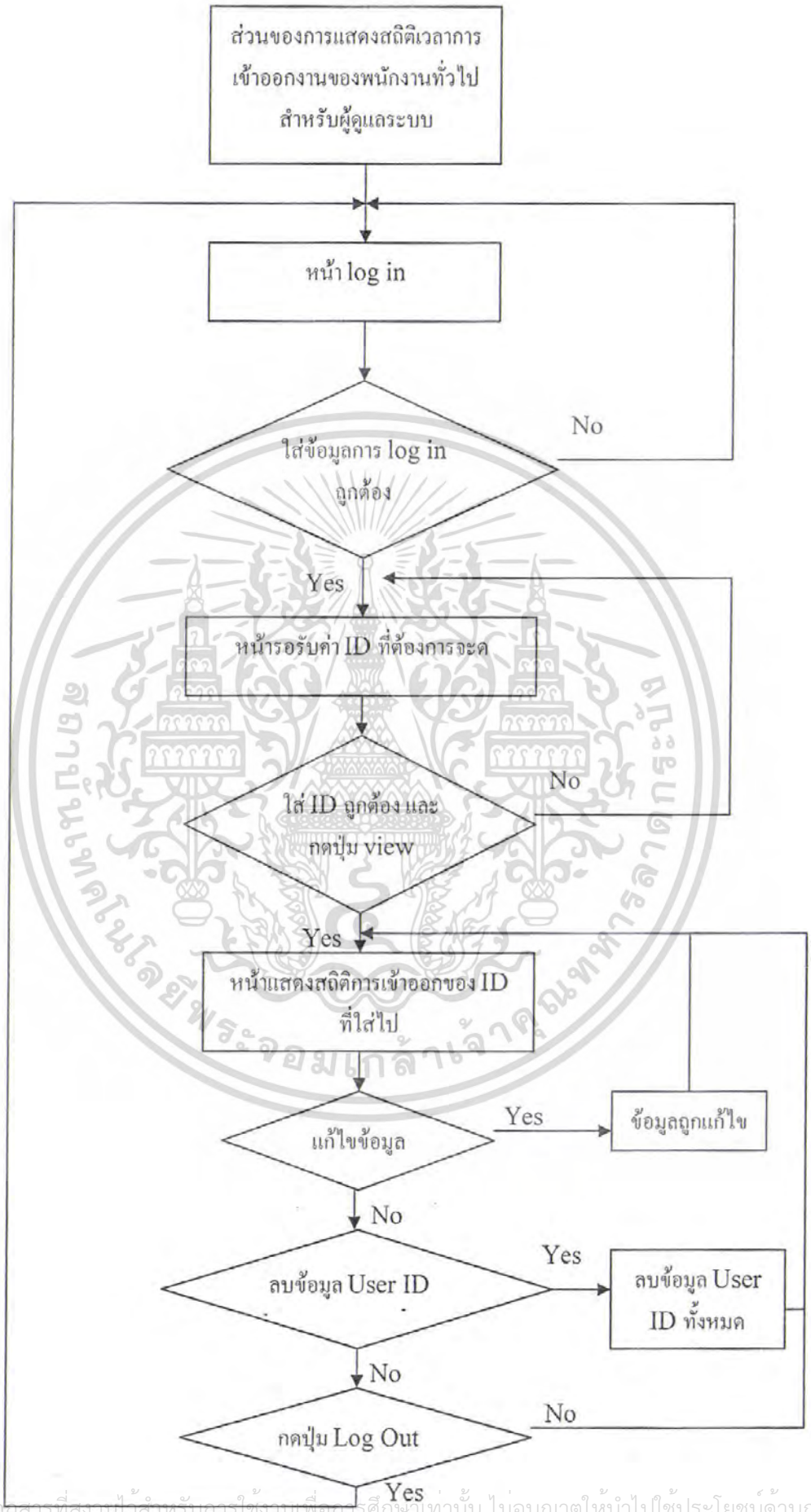
รูปที่ 3.9 แสดงแผนผังการ Upload ข้อมูล ผ่าน web service



รูปที่ 3.10 แสดงแผนผังการแสดงสถิติสำหรับพนักงานทั่วไปบน web service



รูปที่ 3.11 แสดงแผนผังการแสดงสถิติสำหรับแผนกฝ่ายบุคคลบน web service



รูปที่ 3.12 แสดงแผนผังการแสดงสถิติสำหรับผู้ดูแลระบบบน web service

บทที่ 4
การทดลอง และผลการทดลอง

การทดลองได้ทำการทดลองโดยแยกเป็นส่วนของ วงจรบอร์ดหลัก และส่วนของภาคแสดงผลบนคอมพิวเตอร์

การทดลอง และ ผลการทดลอง

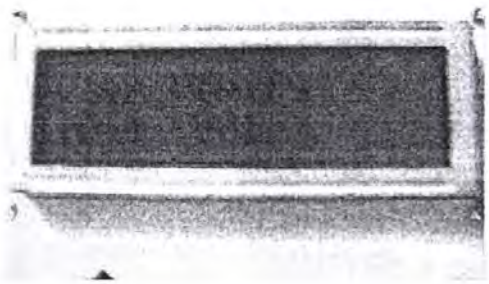
การทดลองได้ทำการทดลองโดยแบ่งเป็นส่วนต่างๆดังนี้ วงจรบอร์ดหลัก การติดต่อกับโมดูลสแกนลายนิ้วมือ ละส่วนของโปรแกรมบนคอมพิวเตอร์

4.1 การทดลองของส่วนวงจรบอร์ดหลัก

ในการทดลองของวงจรบอร์ดหลักนั้น ได้ทำการทดลองโดยการต่อวงจรและเขียนโปรแกรมควบคุม Microcontroller ให้ทำการติดต่อกับอุปกรณ์ต่างๆคือ IC DS1307 หน่วยความจำ EEPROM จอ LCD และKey-Pad โมดูลสแกนลายนิ้วมือ และทำการทดสอบการติดต่อสื่อสารกับคอมผ่าน RS-232 โดยมีผลการทดลองดังนี้

รูปที่ 4.1 การอ่านค่าจาก DS1307 และแสดงผลบนจอ LCD

รูปที่ 4.2 การติดต่อกับ โมดูลสแกนลายนิ้วมือเมื่อมีนิ้วมาทาบ



รูปที่ 4.3 แสดง การรับค่า User ID จากโมดูลสแกนลายนิ้วมือและแสดงผ่าน LCD

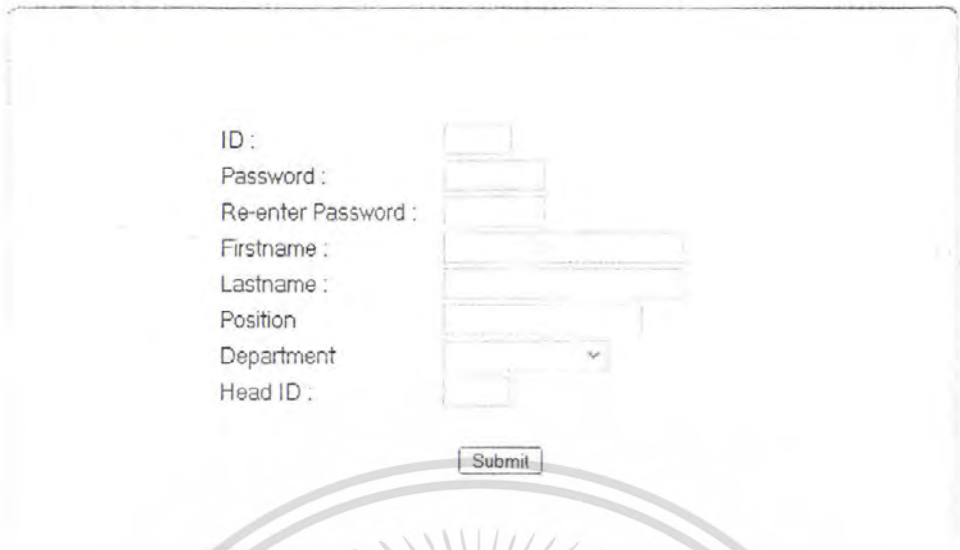
4.2 การทดลองในส่วนของภาคแสดงผลบนคอมพิวเตอร์

4.2.1 Web Service

ในส่วนของ Web Service ได้แบ่งออกเป็น 5 ส่วนย่อยด้วยกัน ดังนี้

- 4.2.1.1 ส่วนของการลงทะเบียน (demo : http://www.lemonsplitz.com/project/register)
- 4.2.1.2 ส่วนของการ upload ข้อมูล
  (demo : http://www.lemonsplitz.com/project/upload)
- 4.2.1.3 ส่วนของหน้าการแสดงสถิติสำหรับพนักงานทั่วไป
  (demo: http://www.lemonsplitz.com/report id : 0001 password : 0001)
- 4.2.1.4 ส่วนของหน้าการแสดงสถิติสำหรับแผนกฝ่ายบุคคล
  (demo : http://www.lemonsplitz.com/view password : view)
- 4.2.1.5 ส่วนของหน้าการแสดงสถิติสำหรับผู้ดูแลระบบ
  (demo : http://www.lemonsplitz.com/admin password : admin)

4.2.2.1 ส่วนของการลงทะเบียน เป็นส่วนของการลงทะเบียนพนักงานใหม่เข้าสู่ระบบ



รูปที่ 4.4 รูปแสดงหน้าของการลงทะเบียน

4.2.2.2 ส่วนของการ upload ข้อมูล เป็นส่วนของการนำข้อมูลเวลาการเข้าออกงานของพนักงานที่ได้เก็บอยู่ในเครื่องแสกนลายนิ้วมือ ส่งไปยังฐานข้อมูลบน เครือข่ายผ่านทาง Serial Port RS232 และทำการลบข้อมูลในเครื่องแสกนลายนิ้วมืออัตโนมัติ

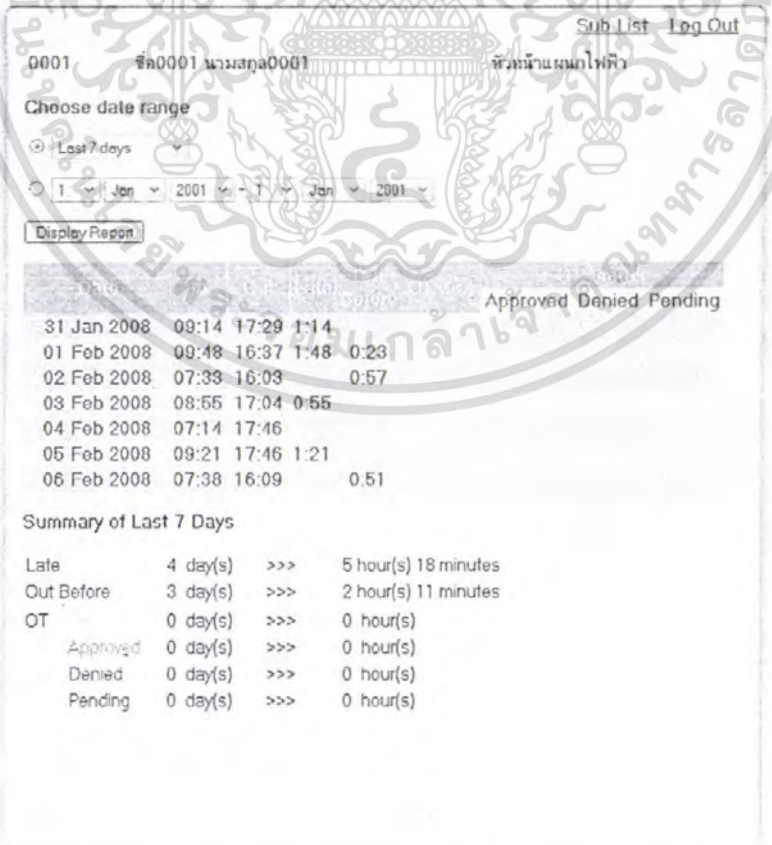
รูปที่ 4.5 รูปแสดงหน้าของการอัพโหลด(Upload)ข้อมูล

4.2.2.3 ส่วนของหน้าการแสดงสถิติสำหรับพนักงาน เป็นส่วนสำหรับพนักงานทั่วไปที่สามารถเข้าไปดูสถิติเวลาการเข้าออกงานของตัวเองได้ ทั้งยังดูในส่วนของ OT สถานะของ OT ว่า หัวหน้าแผนกได้อนุมัติการทำ OT ของวันที่ทำ OT หรือยัง และในกรณีของหัวหน้าแผนกนอกจากจะดูสถิติเวลา

การเข้าออกของตัวเองได้แล้ว ยังสามารถที่จะเข้าไปดูสถิติของลูกน้องได้ และมีระบบจัดการการอนุมัติการทำ OT ของลูกน้องสำหรับหัวหน้าแผนก

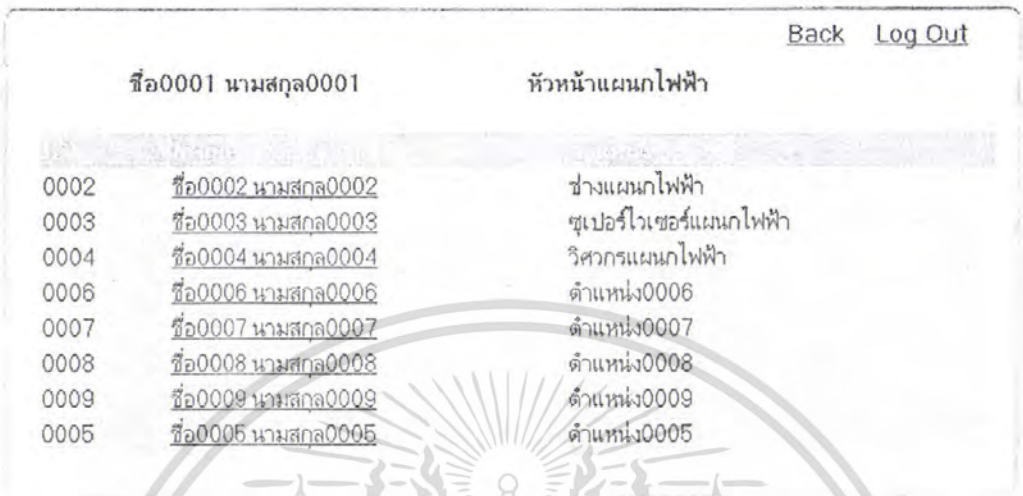
รูปที่ 4.6 แสดงหน้า Log in เพื่อเข้าระบบ

เมื่อ Log in เข้าสู่ระบบ ก็จะเข้าสู่สถิติเวลาการเข้าออกงานของ ID ที่ได้ทำการ log in เข้าไป



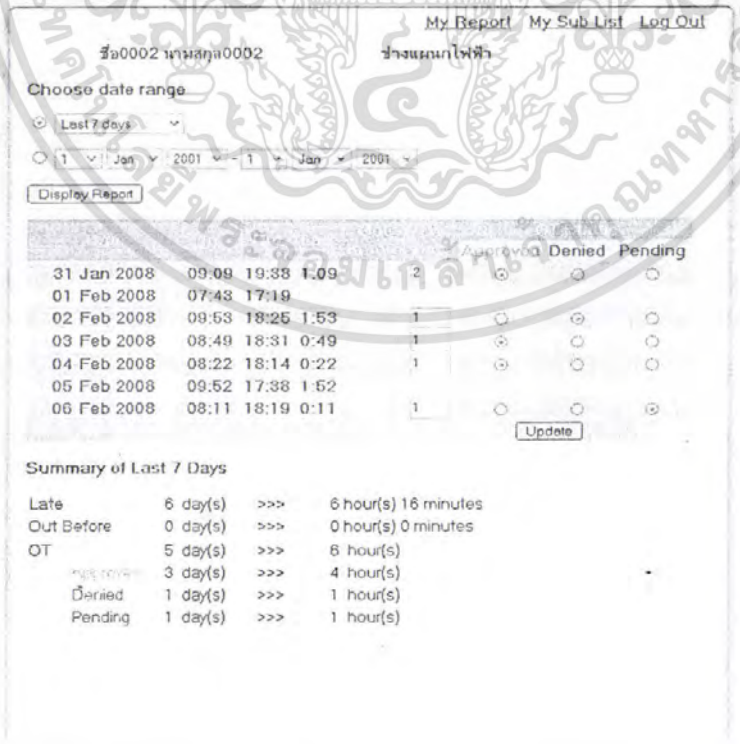
รูปที่ 4.7 แสดงหน้าสถิติเวลาการเข้าออกของพนักงาน ID นั้นๆ

ตัวอย่าง : http://www.lemonsplitz.com/project/report ID : 0001 Password : 0001
ซึ่งเป็น ID ของหัวหน้าแผนกไฟฟ้า และเมื่อคลิกที่ Link "Sub List" ที่มุมบนขวามือ ก็จะไปหน้า list รายชื่อของลูกน้องในสังกัด ดังรูปที่ 3.5



รูปที่ 4.8 แสดงหน้าของรายชื่อลูกน้องของหน้าหน้าแผนก ID 0001 ตามตัวอย่าง

และเมื่อคลิกไปที่ link ชื่อของพนักงาน ก็จะเข้าไปที่หน้าสถิติเวลาการเข้าออกของพนักงานผู้นั้น



รูปที่ 4.9 แสดงหน้าสถิติเวลาเข้าออกของลูกน้องที่คลิกเข้ามาดูจากรายชื่อทั้งหมดของหัวหน้าแผนก

จะเห็นว่าในส่วนของ OT Status จะมีให้เลือกว่าเมื่อพนักงานคนนั้นมีการทำงานล่วงเวลาในวันนั้นได้ทำ OT จริง และจะอนุมัติหรือไม่

Approved – อนุมัติ

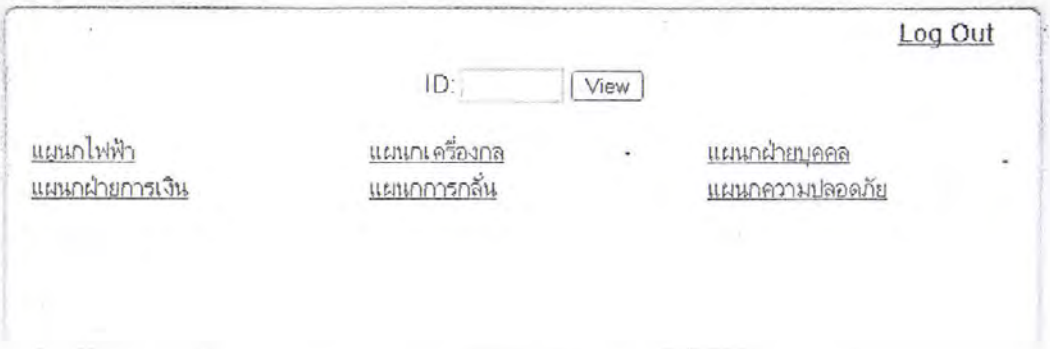
Denied – ไม่อนุมัติ

Pending – รอการอนุมัติ

4.2.2.4 **ส่วนของหน้าการแสดงสถิติสำหรับแผนกฝ่ายบุคคล** สำหรับในส่วนนี้เป็นส่วนสำหรับแผนกฝ่ายบุคคลในการเข้าไปดูสถิติเวลาการเข้าออกของพนักงานแต่ละคน เนื่องจากแผนกฝ่ายบุคคลนี้เป็นฝ่ายที่ต้องทำการสรุปผลมา ลา สาย ขาด การทำ OT ของพนักงาน ในส่วนนี้จึงทำเป็นระบบเพื่อให้ฝ่ายบุคคลใช้งาน

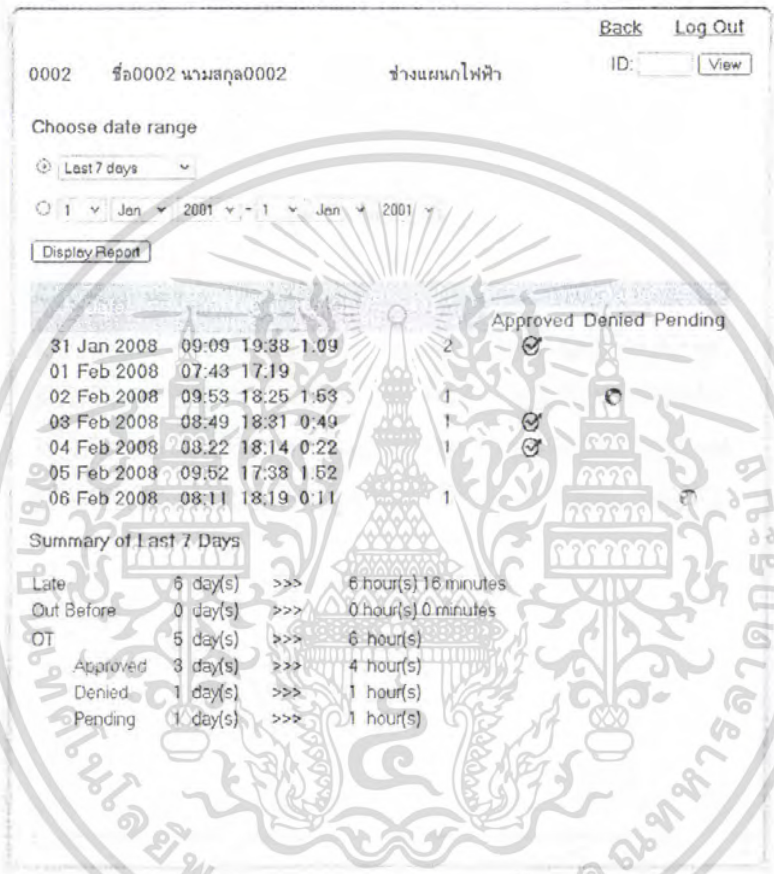
รูปที่ 4.10 แสดงหน้า log in สำหรับแผนกฝ่ายบุคคล

จากตัวอย่างเมื่อได้ log in เข้าสู่ระบบแล้ว จะไปหน้าเว็บดังรูปที่ 4.10



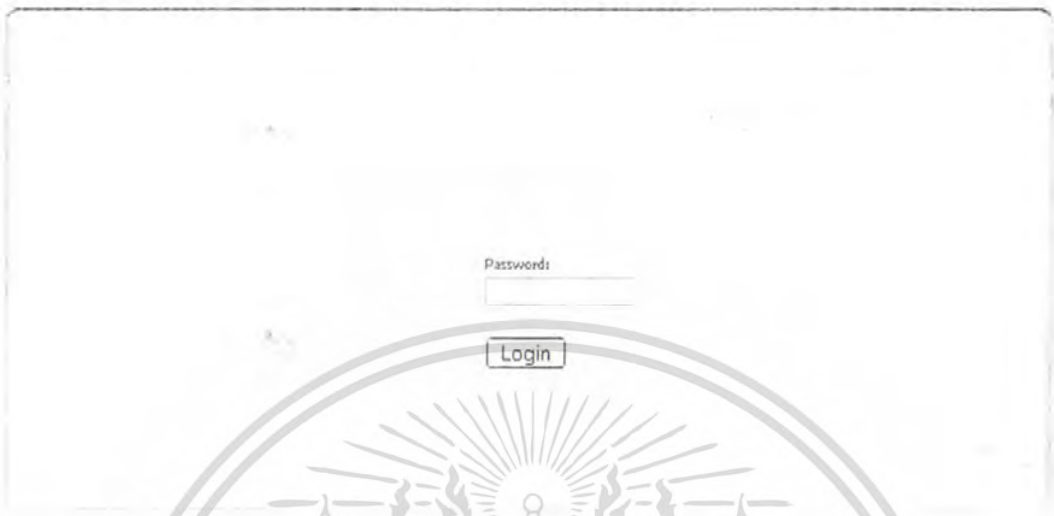
รูปที่ 4.11 แสดงหน้าเมื่อ log in เข้าสู่ระบบ

จากหน้านี้จะเห็นช่องใส่ ID อยู่ด้านบน ก็ให้ใส่ ID ที่ต้องการจะดู แล้วคลิกที่ปุ่ม view ก็จะเข้าไปที่หน้าสถิติเวลาเข้าออกของพนักงานคนคนนั้น ดังรูป 3.8 เมื่อใส่ ID 0002 ลงไปในช่องและคลิกที่ปุ่ม view จะไปหน้าสถิติเวลาเข้าออกงานของพนักงาน ID 0002 ดังรูป 4.12



รูปที่ 4.12 แสดงหน้าสถิติของ ID ที่เลือก

4.2.2.5 ส่วนของหน้าการแสดงสถิติสำหรับผู้ดูแลระบบ สำหรับในส่วนนี้เป็นส่วนของผู้ดูแลระบบ ซึ่งสามารถที่จะแก้ไขข้อมูลต่างๆได้ทั้งหมด



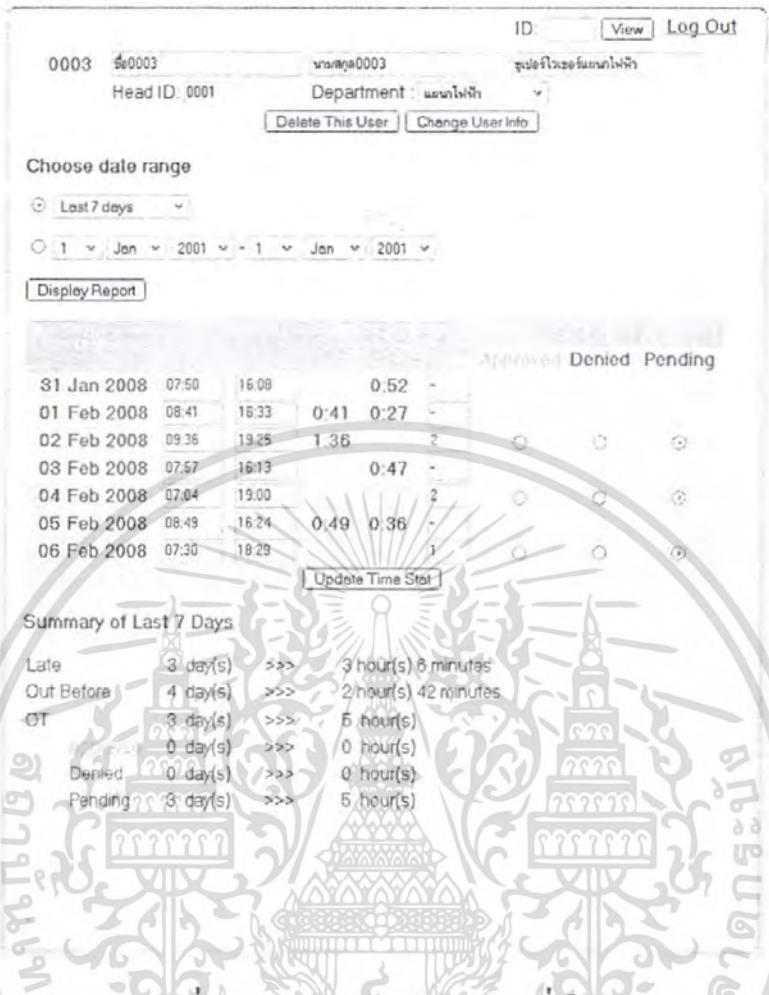
รูปที่ 4.13 แสดงหน้า log in สำหรับผู้ดูแลระบบ

ตัวอย่าง : http://www.lemonsplitz.com/project/admin Password : admin

เมื่อ log in เข้าสู่ระบบแล้ว จะเป็นดังรูป 4.11

รูปที่ 4.14 แสดงหน้าเมื่อ log in เข้าสู่ระบบ

จากหน้านี้จะเห็นช่องใส่ ID อยู่ด้านบน ก็ให้ใส่ ID ที่ต้องการจะดู แล้วคลิกที่ปุ่ม view ก็จะเข้าไปที่หน้าสถิติเวลาเข้าออกของพนักงานคนคนนั้น ดังรูป 4.11 เมื่อใส่ ID 0003 ลงไปในช่องและคลิกที่ปุ่ม view จะไปหน้าสถิติเวลาเข้าออกงานของพนักงาน ID 0002 ดังรูป 4.12



รูปที่ 4.15 แสดงหน้าสถิติของ ID ที่เลือก

จากตัวอย่างจะเห็นว่า ในหน้านี้สามารถแก้ไขทุกอย่างได้ในกรณีที่เกิดข้อผิดพลาดต่างๆเกิดขึ้น รวมถึงการลบประวัติทั้งหมดของพนักงานคนนั้นเมื่อพนักงานคนนั้นได้ออกไปแล้วได้

บทที่ 5
บทสรุป วิจารณ์ ปัญหาที่พบ และการพัฒนา

บทสรุป

จากการศึกษาและทดลองลงมือปฏิบัติโครงงานระบบควบคุมการเข้าออกด้วยลายนิ้วมือ (Access Control by Finger Print) ในภาคเรียนที่ 2 ปีการศึกษา 2549 พบว่า เกิดอุปสรรคในการทำงานคือ โมดูลสแกนลายนิ้วมือรุ่น MRB 200 ขัดข้องทำให้การทำเนินงานหยุดชะงักไประยะนึง ทั้งนี้ผู้จัดทำโครงงานพยายามติดต่อกับบริษัทผู้ขาย เพื่อซ่อมหรือเปลี่ยนอุปกรณ์ แต่ทั้งนี้พบว่าโมดูลรุ่นที่นำมาพัฒนาไม่มีการผลิตแล้ว ทางอาจารย์ที่ปรึกษาจึงเห็นควรว่าให้ทางผู้จัดทำ ใช้ Microcontroller ตระกูล MCS-51 ขึ้นมารับ-ส่ง โปรโตคอลตามโมดูลเสมือนจริง

ทั้งนี้จากการปฏิบัติงานจริงในโครงงานนี้ทำให้นักศึกษาได้ทราบถึงปัญหาต่างๆที่เกิดขึ้นจากการลงมือปฏิบัติ รวมทั้งเป็นการฝึกการแก้ปัญหา ซึ่งจะทำให้สามารถนำไปประยุกต์ใช้กับชีวิตจริงและหน้าที่การงานต่อไป

ปัญหาที่พบ

เครื่องสแกนเกิดปัญหาบ่อยทำให้เกิดความล่าช้าในการทำงาน การจัดทำโครงงานนี้ต้องใช้ความรู้ในการเขียนโปรแกรมหลายภาษาทำให้เกิดความล่าช้าในการศึกษาและทดลอง

แนวทางการพัฒนาต่อ

ในอนาคตสำหรับการพัฒนาโครงงานควรซื้อโมดูลสแกนลายนิ้วมือรุ่นใหม่ และสามารถหาอะไหล่เปลี่ยนซ่อมได้

บรรณานุกรม

[1] ประจิน พลังสันติกุล. **เรียนรู้และใช้งาน CCS C คอมไพล์เลอร์ เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร.** พิมพ์ครั้งที่1. กรุงเทพนคร : อินโนเวตีฟ เอ็กเพอริเมนต์.2547.

[2] นิรุธ อำนวยศิลป์. **คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง.** กรุงเทพนคร : ซัคเซส มีเดีย.

ภาคผนวก ก.

โปรแกรมควบคุมการติดต่อกับโมดูล fingerprint ด้วยภาษาซี

Main control

```
#include <reg52.h>
#include <stdio.h>

#include <lcd.h>                    // open file lcd.h //
#include <serial_command.c>         // open file serial_command.c //
#include <i2c.c>                    // open file i2c.c //
#include <DS1307.c>                 // open file DS1307.c //
#include <scan_key.c>               // open file scan matrix switch. //


// --- delay (ms) --- //
void dmsec(unsigned int count)
{
   unsigned char i;
   while (count)
         {
      for (i=1;i<=228;i++);
      count--;
   }
}
//////////////////////////


void send_comport_ascii()
{
        unsigned char digit_Hour[2],digit_Min[2],digit_Date[2],digit_Mon[2],digit_Year[2];
        unsigned char ID_ASCII[4];

        Rd_DS1307();
```

```
digit_Hour[0] = hex_to_deh(Hour)+48;
digit_Hour[1] = hex_to_del(Hour)+48;

digit_Min[0]  = hex_to_deh(Min)+48;
digit_Min[1]  = hex_to_del(Min)+48;

digit_Date[0] = hex_to_deh(Date)+48;
digit_Date[1] = hex_to_del(Date)+48;

digit_Mon[0]  = hex_to_deh(Mon)+48;
digit_Mon[1]  = hex_to_del(Mon)+48;

digit_Year[0] = hex_to_deh(Year)+48;
digit_Year[1] = hex_to_del(Year)+48;


ID_ASCII[0] =  digit_ID[0]+48;
ID_ASCII[1] =  digit_ID[1]+48;
ID_ASCII[2] =  digit_ID[2]+48;
ID_ASCII[3] =  digit_ID[3]+48;


send_char('S');                 delay1(100);          // byte 1  start.  //
send_char(ID_ASCII[0]);         delay1(100);          // byte 2  ID[0].  //
send_char(ID_ASCII[1]);         delay1(100);          // byte 3  ID[1].  //
send_char(ID_ASCII[2]);         delay1(100);          // byte 4  ID[2].  //
send_char(ID_ASCII[3]);         delay1(100);          // byte 5  ID[3].  //
send_char('#');                 delay1(100);          // byte 6  END.    //
send_char(digit_Hour[0]);       delay1(100);          // byte 7  hour[0]. //
send_char(digit_Hour[1]);       delay1(100);          // byte 8  hour[1]. //
send_char(digit_Min[0]);delay1(100);                  // byte 9  min[0].  //
send_char(digit_Min[1]);delay1(100);                  // byte 10 min[1].  //
```

```
        send_char(digit_Date[0]);       delay1(100);        // byte 11 date[0]. //
        send_char(digit_Date[1]);       delay1(100);        // byte 12 date[1]. //
        send_char(digit_Mon[0]);        delay1(100);        // byte 13 mon[0]. //
        send_char(digit_Mon[1]);        delay1(100);        // byte 14 mon[1]. //
        send_char(digit_Year[0]);       delay1(100);        // byte 15 year[0]. //
        send_char(digit_Year[1]);       delay1(100);        // byte 16 year[1]. //
        send_char('E');                 delay1(100);        // byte 17 END. //

}

// ---------- START MAIN PROGRAMS ---------- //
void main(void)
{
//      unsigned char begin;
//      unsigned char count;
        unsigned char y;
        init_interr_serial();       // intialtion rs232. //
        lcd_init();                 // initialion lcd. //
            // --- set CD2052. --- //
                CA = 1;
            CB = 0;
            // --- use x1,y1; --- //
/*
        // -------- Set time -------- //
            S_Sec  = 0x00;          // write sec. //
            S_Min  = 0x00;          // write min. //
            S_Hour = 0x17;          // write hour. //
            S_Day  = 0x02;          // write day. //
            S_Date = 0x22;          // write date. //
            S_Mon  = 0x01;          // write month. //
            S_Year = 0x08;          // write year. //
            Wr_DS1307();            // write chip ds1307. //
```

```
  // ---------------------------- //
*/

        lcd_command(0x00); lcd_puts(" TEST SINE ");
        lcd_command(0XC0);   lcd_puts("  CHON HON  ");
        dmsec(2000);                              // delay 5 milisec. //
while(1)
{
// -------------------- mode check fingerprint ------------------ //
        if(mode==0)
        {
                enter = 0;
                ES = 0;                                  // Disble Interrupt Coz,Not Work. //
                lcd_command(0x01);              // clear display //
                Rd_DS1307();                    // Read chip ds1307      //
                lcd_command(0x00);              lcd_puts("Time ");
                timetodisplay();          // show time. //
                while(mode==0)
                {
                  Rd_DS1307();                  // Read chip ds1307      //
                  timetodisplay();  // show time. //
// ------------------------ scan key pad. ---------------------------- //
                  C2 = 1;
                  C2 = 0;
                  if(C2==0)
                  {
                        delay1(100);
                        if( R3 == 0 )       // ENTER mode check finger print. //
                  {
                                        LED = 0;
                                        mode = 3;
                  }
```

```
                while(R3==0){delay1(100);}
                    }
                    C2 = 1;
                    C3 = 1;
                C3 = 0;
                if(C3 == 0)
                {
                        delay1(100);
                  if((R0==0)||(R2==0))
                  {
                    if( R0 == 0 )       // ENTER mode clear id. //
                    {
                        //      begin = 0; count = 0;
                                mode=2;
                    }
                              if( R2 == 0 )       // ENTER mode registor. //
                    {
                        //      begin = 0; count = 0;
                                mode=1;
                    }
                    while((R0==0)||(R2==0)){delay1(100);}
                  }
                }
                    C3 = 1;
// ---------------------------- end scan key pad ----------------------- //
                }
        }
// ------------------------- End mode check printf ------------------------ //
// ------------------------- MODE registor -------------------------------- //
if(mode==1)
{
    ES = 0;                         // Disble Interrupt Coz,Not Work. //
```

```
        lcd_command(0x01); dmsec(100);                  // clear display //
        lcd_command(0x00); lcd_puts("Mode registor.");
        lcd_command(0xC0); lcd_puts("ID = ");
while(mode==1)
{
        if(enter==0)
        {
                ID = 0;
                upper_id  = 0;
                lower_id  = 0;
                position = 6;
                while(enter==0)
                {
                  key_id();
                }
                if(position==7)
                {
                        ID = get_id[0];
                }
                if(position==8)
                {
                        ID = (get_id[0]*10)+(get_id[1]);
                }
                if(position==9)
                {
                        ID = (get_id[0]*100)+(get_id[1]*10)+(get_id[2]);
                }
                if(position==10)
                {
                        ID = (get_id[0]*1000)+(get_id[1]*100)+(get_id[2]*10)+(get_id[3]);

                }
```

```
}
if(enter==1)
{
        swap_ID();                                  // change id is upper id,lower id. //
        lcd_command(0x01); dmsec(100);              // clear display  //
        lcd_command(0x00); lcd_puts("Put finger print");
        while(enter==1)
        {
            // function send command U1 to U2 //
            // command 1 FE 00 02 xx1 xx2 03 byte_control FD //
            // xx1 = upper id. //
            // xx2 = lower id. //
            // byte_control = byte3 (xor) byte4 (xor) byte5 (xor) byte6. //
            // send protocal ------ > FE 00 02 xx xx 03 xx FD //
            send_command(reg,upper_id,lower_id,reg2);
            recv_command();          delay1(100);
            //      recv protocal ------ > FE 00 02 xx xx 03 xx FD   //
            if( (recv_serial[2]==reg) && (recv_serial[5]==reg2) )
            {
                lcd_command(0xC0); lcd_puts("step 1 pass");
                // send protocal ----- > FE 00 04 xx xx 03 xx FD //
                send_command(reg3,upper_id,lower_id,reg2);
                recv_command();          delay1(100);
            }
            // recv protocal  ------ > FE 00 44 00 00 00 44 FD  //
            if( (recv_serial[2]==reg4) && (recv_serial[6]==reg4) )
            {
                lcd_command(0xC0); lcd_puts("step 2 pass"); dmsec(800);
                // send protocal ----- > FE 00 03 xx xx 03 xx FD
                send_command(reg2,upper_id,lower_id,reg2);
                recv_command();          delay1(100);
```

```
                        // ------------------------------------------------------ //

                        }
                        // recv protocal  ------ > FE 00 43 00 00 00 43 FD  //
                        if( (recv_serial[2]==reg5) && (recv_serial[6]==reg5) )
                        {
                                lcd_command(0xC0); lcd_puts("REGISTOR OK.");
                                dmsec(2000);
                                enter = 0; mode=0;
                        }
                }
        }
}
}
// ------------------------ End mode registor ------------------------ //
// ------------------------ mode clear id ------------------------ //
if(mode==2)
{
        ES = 0;                                 // Disble Interrupt Coz,Not Work. //
        lcd_command(0x01); dmsec(100);                  // clear display  /
        lcd_command(0x00); lcd_puts("Mode Delete ID.");
        lcd_command(0xC0); lcd_puts("ID = ");
        while(mode==2)
        {
                if(enter==0)
                {
                        ID = 0;
                        upper_id  = 0;
                        lower_id  = 0;
                        position = 6;
                        while(enter==0)
                        {
```

```
            key_id();
        }
        if(position==7)
        {
                ID = get_id[0];
        }
        if(position==8)
        {
                ID = (get_id[0]*10)+(get_id[1]);
        }
        if(position==9)
        {
                ID = (get_id[0]*100)+(get_id[1]*10)+(get_id[2]);
        }
        if(position==10)
        {
                ID =
(get_id[0]*1000)+(get_id[1]*100)+(get_id[2]*10)+(get_id[3]);
        }
    }
    if(enter==1)
    {
        swap_ID();                          // change id is upper id.lower id. //
        // ---------- delete all ---------- //
        if( ( upper_id == 0x00 ) && ( lower_id == 0x00 ) )
        {
                // send protocal ------ > FE 00 21 xx xx 00 xx FD //
                send_command(del2,upper_id,lower_id,delay);
                // ---------- //
        }
        else
        {
```

```
                    // ---------- delete ID ---------- //
                    // send protocal ------ > FE 00 20 xx xx 00 xx FD //
                    send_command(del1,upper_id,lower_id,delay);
                    // ------------------------------------------- //
                }
                recv_command();
                //      recv protocal ------ > FE 00 02 xx xx 03 xx FD    //
                if( (recv_serial[2]==del1) || (recv_serial[2]==del2) )
                {
                    lcd_command(0xC0); lcd_puts("DELETE OK.");
                }
                dmsec(2000);
                mode = 0;
                // enter=0;
            }
        }
}
// ---------- End mode clear id ---------- //
// ---------- MODE CHECK ID ---------- //
if(mode==3)
{
    lcd_command(0x01); dmsec(100);          // clear display //
    lcd_command(0x00); lcd_puts("Mode CHECK ID.");
//  ES = 1;         // Enable Interrupt Coz.Not Work. //
    send_command(check,delay,delay,delay);  // FD 00 12 00 00 00 12 FE. //
    while(mode==3)
    {
        recv_command();
        // ---------- check ID ---------- //
        if(recv_serial[2]!=recv_serial[6])
        {
```

```
                CA = 0;
                lcd_command(0xC0); lcd_puts("HAVE FINGERPRINT"); dmsec(800);
                if((upper_id==0x00)&&(lower_id==0x00))
                {
                        lcd_command(0xC0); lcd_puts("NONE ID       ");
                }
                else
                {
                        hex_to_dec(upper_id,lower_id);
                        lcd_command(0xC0); lcd_puts("ID =          ");
                        lcd_gotoxy(6,2);
                        lcd_data(data_code[digit_ID[0]]);
                        lcd_data(data_code[digit_ID[1]]);
                        lcd_data(data_code[digit_ID[2]]);
                        lcd_data(data_code[digit_ID[3]]);
                }
                send_comport_ascii();
                for(y=0;y<8;y++)
                {
                        recv_serial[y] = 0;
                }
                CA = 1;
        }
        dmsec(1000);
        mode = 0;
    }
}
}
}
// ---------------------------------------------------------- //
```

LCD function

```
#define LCDPORT P2
```

```
#define lcd_delay 400

sbit RS=LCDPORT^0;
sbit RW=LCDPORT^1;
sbit E =LCDPORT^2;

bit  status=0;

void delay(unsigned int j)
{
        unsigned int i;
        for(i=0;i<j;i++);
}


void lcd_init_write(unsigned char a)
{
        RS=0;
        RW=0;
        LCDPORT=a;
        E=1;
        delay(lcd_delay);
        E=0;
}

void lcd_command(unsigned char a)
{
        unsigned char temp;
        if(status==1)
        {
                status=0;
```

```
            goto next123321;
        }
        RS=0;
        next123321:
        RW=0;
        temp=a;
        temp&=0xf0;
        LCDPORT&=0x0f;
        LCDPORT|=temp;
        E=1;
        delay(lcd_delay);
        E=0;
        temp=a<<4;
        temp&=0xf0;
        LCDPORT&=0x0f;
        LCDPORT|=temp;
        E=1;
        delay(lcd_delay);
        E=0;
}

void lcd_data(unsigned char a)
{
        status=1;
        RS=1;
        lcd_command(a);
}



void lcd_init(void)
{
```

```
        delay(lcd_delay);
        lcd_init_write(0x30);    delay(lcd_delay);
        lcd_init_write(0x30);    delay(lcd_delay);
        lcd_init_write(0x30);    delay(lcd_delay);
        lcd_init_write(0x20);    delay(lcd_delay);
        lcd_command(0x28);              delay(lcd_delay);                    //
LCD 4 bit mode. //
        lcd_command(0x0c);              delay(lcd_delay);                    //
clear clearse. //
        lcd_command(0x06);              delay(lcd_delay);                    //
Entry Mode. //
        lcd_command(0x02);              delay(lcd_delay);                    //
Return Home. //
}

void lcd_puts(char *aaa)
{
        unsigned int i=0;
        for(i=0;aaa[i]!=0;i++)
                lcd_data(aaa[i]);
}

void lcd_gotoxy(unsigned char x,unsigned char y)
{
        unsigned char address;

        if(y!=1)
        address = 0xC0;
        else
        address = 0x00;
        address+=x-1;
        lcd_command(0x80|address);
```

```
        lcd_command(0x0c);      // clear clearse. //

serial command
#define start     0xFE              // BYTE 1 START. //
#define stop      0xFD              // BYTE 7 STOP. //
#define reg       0x02              // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg3      0x04              // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg4      0x44              // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg5    0x43                // BYTE 3 BYTE CONTROL REGISTOR. //
#define check     0x12              // BYTE 3 BYTE CONTROL FIND FINGER. //
#define del1      0x20              // BYTE 3 BYTE CONTROL DEL ID. //
#define del2      0x21              // BYTE 3 BYTE CONTROL DEL ID. //
#define del3      0x22              // BYTE 3 BYTE CONTROL DEL ID. /
#define delay   0x00                // BYTE DELAY MODULE. //
#define reg2    0x03                // BYTE 5 BYTE CONTROL REGISTOR. //

// ---- bit control IC CD4052 ---- //
// ------------------------------- //
// | bit B | bit A | port out |  //
// |  0  |  0  |  X0,Y0  |  //
// |  0  |  1  |  X1,Y1  |  //
// |  1  |  0  |  X2,Y2  |  //
// |  1  |  1  |  X3,Y3  |  //
// ------------------------------- //
sbit CA = P1^7;             // control bit A. //
sbit CB = P1^6;             // control bit B. //
sbit LED = P3^2;


void init_interr_serial(void)
{
        // XTAL 11.0592 MHZ
        // 12Machines/Cycle
```

```
        TMOD = 0x20;
        SCON = 0x52;
        PCON = 0x80;
        TH1  = 0xFA;                    //BaudRate 9600
             IE = 0x90;
        TR1  = 1;
        REN  = 1;
        TI   = 1;
        RI   = 0;
             ES   = 0;                  //Disable Interrupt Coz,Not
Work
}

// --------- delay key sw ---------- //
void delay1(unsigned int time)
{
    while(time)
    time--;
}
// ------------------------------- //

// ------ function recv char ------ //
char recv_char()
{
  char c;
  while(!RI)
        delay1(1);
  c=SBUF;
  RI=0;
  return(c);
}
//////////////////////////////////////
```

```
// ------- function send cahr ------ //
void send_char(char c)
{
   TI=0;
   SBUF=c;
   while (!TI)
     delay1(1);
}
////////////////////////////////////////

unsigned char upper_id;                    // upper id. //
unsigned char lower_id;                    // lower id. //
unsigned int ID;
void swap_ID()
{
        upper_id = ID>>8;
        lower_id = ID;
}

unsigned char digit_ID[4];

void hex_to_dec(unsigned int up_id,unsigned int low_id)
{
        unsigned int recv_id;
        recv_id = (up_id<<8)+low_id;
        digit_ID[0] = recv_id/1000;
        digit_ID[1] = (recv_id%1000)/100;
        digit_ID[2]= ((recv_id%1000)%100)/10;
        digit_ID[3] = ((recv_id%1000)%100)%10;
}
```

```
void send_command(unsigned char byte3,unsigned char byte4,unsigned char byte5, unsigned
char byte6)
{
        unsigned char byte7;
        byte7 = byte3^byte4^byte5^byte6;
        delay1(100);
        send_char(start);               delay1(100);    // BYTE 0  BYTE START. //
        send_char(delay);               delay1(100);    // BYTE 1  BYTE DELAY. //
        send_char(byte3);               delay1(100);    // BYTE 2. BYTE . //
        send_char(byte4);               delay1(100);    // BYTE 3  BYTE UPPER ID. //
        send_char(byte5);               delay1(100);    // BYTE 4. BYTE LOWER ID. //
        send_char(byte6);               delay1(100);    // BYTE 5. BYTE //
        send_char(byte7);               delay1(100);    // BYTE 6. BYTE CONTROL. //
        send_char(stop);        delay1(100);    // BYTE 7 BYTE STOP. //
        delay1(100);

}

unsigned char recv_serial[8];
unsigned char mode=0;

void recv_command()
{
        unsigned char x;
        recv_serial[0] = recv_char();
        if(recv_serial[0]==start)
        {
                for(x=1;x<8;x++)
                {
                        recv_serial[x]   = recv_char();
                        upper_id         = recv_serial[3];
                        lower_id         = recv_serial[4];
```

```
                }
        }
}

void serial_isr() interrupt 4
{
  unsigned char x;
  if(RI) // if rx occer
  {
        RI = 0;
        recv_serial[0] = SBUF;
        if(recv_serial[0]==start)
        {
                for(x=1;x<8;x++)
                {
                        recv_serial[x]   = recv_char();
                        upper_id         = recv_serial[3];
                        lower_id         = recv_serial[4];
                }
        }
  }

}
I2C
sbit SDA =   P1^4;
sbit SCL =   P1^5;

void    I2c_Wait(void)
{ unsigned  char  i;
        for (i=0;i<60;i++) { }
}
void  I2c_Clk(void)
```

```
{       SCL     =       1;
        I2c_Wait();
        SCL     =       0;
        I2c_Wait();
}

void I2c_Ack(void)
{       SDA     = 0;
        I2c_Clk();
}

void I2c_Nack(void)
{       SDA     = 1;
        I2c_Clk();
}

void I2c_Start(void)
{       SCL     =       0;
        SDA     =       1;
        SCL     =       1;
        I2c_Wait();
        SDA     =       0;
        I2c_Wait();
}
void I2c_Stop(void)
{       SCL     =       0;
        SDA     =       0;
        SCL     =       1;
        I2c_Wait();
        SDA     =       1;
        I2c_Wait();
}
```

```
void  I2c_Write(unsigned char Data )
{ unsigned char  i;
 bit      out;
          SCL   =        0;
    for (i=0;i<8;i++)
         {
          out   =        Data & 0x80;
          Data  =        Data << 1;
          SDA   =        out;
          I2c_Clk();
         }
     SDA    =            1;
     I2c_Clk();
}


unsigned  char  I2c_Read(void)
{ unsigned char  Data,i;
 bit      out;
          SCL   =        0;
          SDA   =        1;
          I2c_Wait();
          for (i=0;i<8;i++)
     {
                SCL   =        1;
                I2c_Wait();
                out   =        SDA;
                Data  =        Data << 1;
                Data  =        Data | out;
                SCL   =        0;
```

```
                I2c_Wait();
            }
        return(Data);
}
DS1307
unsigned char    S_Sec,S_Min,S_Hour,S_Day,S_Date,S_Mon,S_Year;
unsigned char    Sec,Min,Hour,Day,Date,Mon,Year;
void  Wr_DS1307(void)
{
        I2c_Start();
        I2c_Write(0xD0);
        I2c_Write(0x00);
        I2c_Write(S_Sec);
        I2c_Write(S_Min);
        I2c_Write(S_Hour);
        I2c_Write(S_Day);
        I2c_Write(S_Date);
        I2c_Write(S_Mon);
        I2c_Write(S_Year);
        I2c_Stop();
}
void  Rd_DS1307(void)
{
        I2c_Start();
        I2c_Write(0xD0);
        I2c_Write(0x00);
        I2c_Start();
        I2c_Write(0xD1);
        Sec    =          I2c_Read(); I2c_Ack();
        Min    = I2c_Read(); I2c_Ack();
        Hour   = I2c_Read(); I2c_Ack();
        Day    = I2c_Read(); I2c_Ack();
```

```
        Date    = I2c_Read(); I2c_Ack();
        Mon     = I2c_Read(); I2c_Ack();
        Year    = I2c_Read(); I2c_Nack();
        I2c_Stop();
}

unsigned char hex_to_deh(unsigned char x)
{
        x=x&0xf0;
        return(x>>4);
}

unsigned char hex_to_del(unsigned char x)
{
        x=x&0x0f;
        return(x);
}

unsigned char buf[4];
unsigned char data_code[] = { 0x30,0x31,0x32,0x33,0x34,
                                                0x35,0x36,0x37,0x38,0x39}; // --
ascii code num "0-9" --//
void timetodisplay()
{
                buf[0] = data_code[hex_to_del(Min)];
                buf[1] = data_code[hex_to_deh(Min)];
                buf[2] = data_code[hex_to_del(Hour)];
                buf[3] = data_code[hex_to_deh(Hour)];
                lcd_gotoxy(6,1);  // line 1. //
                lcd_data(buf[3]);
                lcd_data(buf[2]);
                lcd_puts(":");
```

```
                lcd_data(buf[1]);
                lcd_data(buf[0]);
}
Keypad
sbit R0 = P0^0;
sbit R1 = P0^1;
sbit R2 = P0^2;
sbit R3 = P0^3;

sbit C0 = P0^4;
sbit C1 = P0^5;
sbit C2 = P0^6;
sbit C3 = P0^7;

unsigned char position=6;
unsigned char get_id[4];
unsigned char enter = 0;
void key_id()
{
// ---------------------- scan row 1 ---------------------------------- //
    R0 = 1;
    R0 = 0;
    if(R0 == 0)
    {
      delay1(500);
      if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
      {
        if( C0 == 0 )        // ENTER 1. //
        {
                                if(position==10)
                                {
                                        position = 10;
```

```
                }
                else
                {
                        get_id[position-6] = 1;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
                        lcd_puts("1");
                        position++;
                }
}
if( C1 ==0 )        // ENTER 4. //
{
                if(position==10)
                {
                        position = 10;
                }
                else
                {
                        get_id[position-6] = 4;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
                        lcd_puts("4");
                        position++;
                }
}
if( C2 == 0 )       // ENTER 7. //
{
                if(position==10)
                {
                        position = 10;
                }
                else
                {
                        get_id[position-6] = 7;
```

```
                    lcd_gotoxy(position,2);   // write_lcd line 2. //
                    lcd_puts("7");
                    position++;
                }
    }
    if( C3 == 0 )       // ENTER clear ID. //
    {
                    lcd_gotoxy(6,2);// write_lcd line 2. //
                    lcd_puts("    ");
                    position = 6;
    }
    while(C0==0||C1==0||C2==0||C3==0){}
  }
}
// ------------------------- END ROW 1 ------------------------------ //
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
// ------------------------- scan row 2 ------------------------------ //
R0 = 1;
R1 = 1;
R1 = 0;
if(R1 == 0)
{
  delay1(500);
  if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
  {
    if( C0 == 0 )       // ENTER 2. //
    {
                    if(position==10)
                    {
                        position = 10;
                    }
```

```
                else
                {
                        get_id[position-6] = 2;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
                        lcd_puts("2");
                        position++;
                }
}
if( C1 ==0 )        // ENTER 5. //
{
                if(position==10)
                {
                        position = 10;
                }
                else
                {
                        get_id[position-6] = 5;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
                        lcd_puts("5");
                        position++;
                }
}
if( C2 == 0 )       // ENTER 8. //
{
                if(position==10)
                {
                        position = 10;
                }
                else
                {
                        get_id[position-6] = 8;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
```

```
                    lcd_puts("8");
                    position++;
                }
    }
    if( C3 == 0 )       // ENTER 0. //
    {
                    if(position==10)
                    {
                        position = 10;
                    }
                    else
                    {
                        get_id[position-6] = 0;
                        lcd_gotoxy(position,2);   // write lcd line 2. //
                        lcd_puts("0");
                        position++;
                    }
    }
    while(C0==0||C1==0||C2==0||C3==0){}
  }
}
// ---------------------------- END ROW 2 ---------------------------------- //
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
    // ------------------------ scan row 3 ---------------------------------- //
    R1 = 1;
    R2 = 1;
    R2 = 0;
    if(R2 == 0)
    {
      delay1(500);
      if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
```

```
{
 if( C0 == 0 )        // ENTER 3. //
 {
                        if(position==10)
                        {
                                position = 10;
                        }
                        else
                        {
                                get_id[position-6] = 3;
                                lcd_gotoxy(position,2);   // write_lcd line 2. //
                                lcd_puts("3");
                                position++;
                        }
}
if( C1 ==0 )        // ENTER 6. //
{
                        if(position==10)
                        {
                                position = 10;
                        }
                        else
                        {
                                get_id[position-6] = 6;
                                lcd_gotoxy(position,2);   // write_lcd line 2. //
                                lcd_puts("6");
                                position++;
                        }
}
if( C2 == 0 )        // ENTER 9. //
{
                        if(position==10)
```

```
                {
                        position = 10;
                }
                else
                {
                        get_id[position-6] = 9;
                        lcd_gotoxy(position,2);   // write_lcd line 2. //
                        lcd_puts("9");
                        position++;
                }
        }
        if( C3 == 0 )       // ENTER HELP. //
        {

        }
        while(C0==0||C1==0||C2==0||C3==0){}
      }
    }
    // ---------------------------- END ROW 3 ------------------------------ //
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
    // ------------------------ scan row 4 ------------------------------------ //
    R2 = 1;
    R3 = 1;
    R3 = 0;
    if(R3 == 0)
    {
      delay1(500);
      if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
      {
        if( C0 == 0 )        // ENTER up. //
        {
```

```
        }
        if( C1 ==0 )        // ENTER down. //
        {

        }
        if( C2 == 0 )       // ENTER 2nd. //
        {

        }
                            // ENTER sw registor. // (U1 send command)
        if( C3 == 0 )
        {
                            //      ID =
(get_id[0]*1000)+(get_id[1]*100)+(get_id[2]*10)+(get_id[3]);
                                enter = 1;
        }
        while(C0==0||C1==0||C2==0||C3==0){}
      }
    }
    // ------------------------- END ROW 4 ------------------------------ //
    R3 = 1;
////////////////////////////////////////////////////////////////////////////////
}
```

Finger control

```
#include <reg52.h>
#include <stdio.h>
#include <serial_command.c>              // open file serial_command.c //
#include <i2c.c>
#include <EEPROM.c>
#include <scan_key.c>                    // open file scan matrix switch. //
```

```
sbit LED1 = P3^5;


// ---------- START MAIN PROGRAMS ---------- //
void main(void)
{
//      unsigned char check_id;
//      unsigned char check_finger;
//      unsigned char mode;
        // mode = 1 check finger. //
        // mode = 2 check id. //
//      unsigned int count;
        unsigned int x;
        unsigned char z,y;
        unsigned char id_delete[2];
        unsigned char id_check[2];
        unsigned char check_pass;
        unsigned char delete_pass;
        init_interr_serial();       // intialtion rs232. //


  // -------- chec EEPROM ---------- //
        if(RD_EEPROM(0)!=0)
        {
                LED = 0 ;
                for(x=0;x<=200;x++)
                  WR_EEPROM(x,0); dmsec(500);
                LED = 1;
        }
        // ---------------------------------- //


        // --- adder in EEPROM 24LC04B ---- //
        // adder 1  memory upper  id code 1. //
```

// adder 21 memory ulower id code 1. //

// adder 2 memory upper id code 2. //

// adder 22 memory ulower id code 2. //

// adder 3 memory upper id code 3. //

// adder 23 memory ulower id code 3. //

// adder 4 memory upper id code 4. //

// adder 24 memory ulower id code 4. //

// adder 5 memory upper id code 5. //

// adder 25 memory ulower id code 5. //

// adder 6 memory upper id code 6. //

// adder 26 memory ulower id code 6. //

// adder 7 memory upper id code 7. //

// adder 27 memory ulower id code 7. //

// adder 8 memory upper id code 8. //

// adder 28 memory ulower id code 8. //

// adder 9 memory upper id code 9. //

// adder 29 memory ulower id code 9. //

// adder 10 memory upper id code 10. //

// adder 30 memory ulower id code 10. //

// adder 11 memory upper id code 11. //

// adder 31 memory ulower id code 11. //

```
// adder 12 memory upper  id code 12. //
// adder 32 memory ulower id code 12. //

// adder 13 memory upper  id code 13. //
// adder 33 memory ulower id code 13. //

// adder 14 memory upper  id code 14. //
// adder 34 memory ulower id code 14. //

// adder 15 memory upper  id code 15. //
// adder 35 memory ulower id code 15. //

// adder 16 memory upper  id code 16. //
// adder 36 memory ulower id code 16. //

// adder 40 memory mode check_key. //
// adder 41 memory mode enter_key. //
// adder 42 memory mode clear_key. //

// adder 45 memory ID delete upper. //
// adder 46 memory ID check upper. //


while(1)
{
// ---------------------------- MODE RECV COMMAND ---------------------------------- //
        if(mode==0)
        {
                while(mode==0)
                {
                        recv_command();
                        // recv protocal ------> FE 00 12 00 00 00 12 FD . //
```

```
// ---------- check finger print ---------- //
if((recv_serial[2]==check) && (recv_serial[6]==check))
{
        mode = 1;                    // mode send id in code finger. //
}
// recv protocal -----> FE 00 02 xx xx 03 xx FD .//
// ---------- registor ID ---------- //
if((recv_serial[2]==reg)&&(recv_serial[5]==reg2))
{
        mode = 2;                              // mode registor. //
}
// recv protocal -----> FE 00 20 xx xx 00 xx FD . //
// ---------- delete ID ---------- //
if(recv_serial[2]==del1)
{
        mode = 3;
        upper_id      = recv_serial[3];
        lower_id      = recv_serial[4];
}
// recv protocal -----> FE 00 21 00 00 00 21 FD . //
// ---------- delete ID ----------//
if((recv_serial[2]==del2) && (recv_serial[6]==del2))
{
        LED = 0;
        mode = 4;
}

    }
  }
// ---------- //
// ---------- ENTER MODE SEND ID IN CODE FINGER PRINT ---------- //
  if(mode==1)
```

```
        {
                LED = 0;
                key = 0;
                while(mode==1)
                {
                        scan_fingerprint();
                        if(key!=0)
                        {
                                key = RD_EEPROM(41);
        dmsec(50);
                                upper_id = RD_EEPROM(key);                      dmsec(50);
                                lower_id = RD_EEPROM(key+20);                   dmsec(50);
                                send_command(check,upper_id,lower_id,check); // send ID in
code finger print. //
                                key = 0; mode = 0;
                        }
                }
                LED = 1;
        }
// ------------------------------------------------------------------------ //
// ------------------------- ENTER MODE REGISTOR ---------------------------- //
if(mode==2)
{
        LED1= 0;
        key=0;
        while(mode==2)
        {
                if(key==0)
                {
                        scan_fingerprint();
                }
                else
```

```
{
        key = RD_EEPROM(40);  dmsec(50);    // (Addr,Data)
        // send protocal ------ > FE 00 02 xx xx 03 xx FD . //

send_command(recv_serial[2],recv_serial[3],recv_serial[4],recv_serial[5]);
        recv_command();          delay1(50);
        // ------------ check module ------------- //
        for(z=1;z<=16;z++)
        {
            id_check[0] = RD_EEPROM(z); dmsec(50);
            if(id_check[0] == upper_id)
            {
                WR_EEPROM(46,z);     dmsec(50);
                check_pass = 1;
                z = 20;
            }
        }
        for(z=21;z<=36;z++)
        {
            id_check[1] = RD_EEPROM(z); dmsec(50);
            if(id_check[1] == lower_id)
            {
                if(check_pass==1)
                {
                    y = RD_EEPROM(46);  dmsec(50);
                    WR_EEPROM(z,0);          dmsec(50);
                    WR_EEPROM(y,0);          dmsec(50);
                    WR_EEPROM(45,0);    dmsec(50);
                }
                x = 40;
            }
            check_pass = 0;
```

```
                    }
                    WR_EEPROM(key,upper_id);            dmsec(50);      //
(Addr,Data)
                    WR_EEPROM(key+20,lower_id);                 dmsec(50);      //
(Addr,Data)
                    // recv protocal ------ > FE 00 04 xx xx 03 xx FD
                    if((recv_serial[2]==reg3)&&(recv_serial[5]==reg2))
                    {
                        /// send protocal ------ > FE 00 44 00 00 00 44 FD . //
                        send_command(reg4,delay,delay,delay);
                        recv_command();         delay1(50);
                        // ------------------------------------------------- //
                    }
                    // recv protocal ------ > FE 00 03 xx xx 03 xx FD
                    if((recv_serial[2]==reg2)&&(recv_serial[5]==reg2))
                    {
                        LED = 0;
                        /// send protocal ------ > FE 00 43 00 00 00 43 FD  . //
                        send_command(reg5,delay,delay,delay);
                        mode = 0;                                   // END MODE. //
                    }
                }
        }
        LED1= 1;
        key=0;
}
// ------------------------------------------------------------------------------- //
// ---------------------------- ENTER MODE DELETE ID ---------------------------- //
if(mode==3)
{
                for(x=1;x<=16;x++)
                {
```

```
        id_delete[0] = RD_EEPROM(x); dmsec(50);
        if(id_delete[0] == upper_id)
        {
                WR_EEPROM(45,x);      dmsec(50);
                delete_pass = 1;
                x = 20;
        }
}
for(x=21;x<=36;x++)
{
        id_delete[1] = RD_EEPROM(x); dmsec(50);
        if(id_delete[1] == lower_id)
        {
                if(delete_pass==1)
                {
                        y = RD_EEPROM(45);   dmsec(50);
                        WR_EEPROM(x,0);              dmsec(50);
                        WR_EEPROM(y,0);              dmsec(50);
                        WR_EEPROM(45,0);     dmsec(50);
                        delete_pass = 0;
                }
                /// send protocal ------ > FE 00 20 00 00 00 20 FD  . //
                send_command(del1,delay,delay,delay);
                x = 40; mode = 0;
        }
}
if(x==36)
{
        /// send protocal ------ > FE 00 22 00 00 00 22 FD  . //
        send_command(del3,delay,delay,delay);
        delete_pass = 0;
        mode = 0;
```

```
                }
                LED = 1;
}
// ------------------------------------------------------------------------------------ //
if(mode==4)
{
        for(x=1;x<=36;x++)
        {
                WR_EEPROM(x,0); dmsec(50);
        }
        /// send protocal ------ > FE 00 21 00 00 00 21 FD  . //
        send_command(del2,delay,delay,delay);
        delete_pass = 0;
        mode = 0;
        LED = 1;
}
}
}
// ------------------------------------------------------------ //
EEPROM
void  WR_EEPROM (unsigned int  Addr,unsigned char Data)
{
        I2c_Start();
        I2c_Write(0xA0);
        I2c_Write(Addr);//change 0x00FF>>0x00
        I2c_Write(Data);
        I2c_Stop();
}


unsigned char  RD_EEPROM (unsigned int  Addr)
{
```

```
        unsigned char temp;
        I2c_Start();
        I2c_Write(0xA0);
        I2c_Write(Addr);//change 0x00FF>>0x00
        I2c_Start();
        I2c_Write(0xA1);
        temp = I2c_Read();
        I2c_Nack();
        I2c_Stop();
        return(temp);
keypad
sbit R0 = P1^0;
sbit R1 = P1^1;
sbit R2 = P1^2;
sbit R3 = P1^3;

sbit C0 = P1^4;
sbit C1 = P1^5;
sbit C2 = P1^6;
sbit C3 = P1^7;

sbit LED = P3^4;

// --- delay (ms) --- //
void dmsec(unsigned int count)
{
  unsigned char i;
  while (count)
        {
    for (i=1;i<=228;i++);
    count--;
  }
```

```
}
//////////////////////

unsigned char mode;
// mode = 1 check finger. //
// mode = 2 check id. //
//unsigned char check_code;
//unsigned char enter_key;
unsigned char key;

void scan_fingerprint()
{
// ------------------------ scan row 1 ---------------------------------- //
    R0 = 1;
    R0 = 0;
    if(R0 == 0)
    {
      delay1(500);
      if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
      {
        if( C0 == 0 )        // code A. //
        {
                              key = 1;
                              if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                              if(mode==2) {WR_EEPROM(40,key); dmsec(50);}


        }
        if( C1 ==0 )          // code E. //
        {
                              key = 5;
                              if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                              if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
```

```
    }
    if( C2 == 0 )                // code I. //
    {
                                 key = 9;
                                 if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                 if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
    }
    if( C3 == 0 )                // code M. //
    {
                                 key = 13;
                                 if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                 if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
    }
    while(C0==0||C1==0||C2==0||C3==0){delay1(500);}
  }
}
// ---------- END ROW 1 ---------- //
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
  // ---------- scan row 2 ---------- //
  R0 = 1;
  R1 = 0;
  if(R1 == 0)
  {
    delay1(500);
    if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
    {
      if( C0 == 0 )      // code B. //
      {
                                 key = 2;
                                 if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                 if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
```

```
    }
    if( C1 ==0 )        // code F. //
    {
                        key = 6;
                        if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                        if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
    }
    if( C2 == 0 )       // code J. //
    {
                        key = 10;
                        if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                        if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
    }
    if( C3 == 0 )       // code N. //
    {
                        key = 14;
                        if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                        if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
    }
    while(C0==0||C1==0||C2==0||C3==0){delay1(500);}
  }
}
// ------------------------- END ROW 2 ------------------------- //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// ------------------------- scan row 3 ------------------------- //
R1 = 1;
R2 = 0;
if(R2 == 0)
{
  delay1(500);
  if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
```

```
    {
      if( C0 == 0 )                 // code C. //
      {
                                    key = 3;
                                    if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                    if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
      }
      if( C1 ==0 )                  // code G. //
      {
                                    key = 7;
                                    if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                    if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
      }
      if( C2 == 0 )                 // code K. //
                  {
                                    key = 11;
                                    if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                    if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
      }
      if( C3 == 0 )                 // code O. //
      {
                                    key = 15;
                                    if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                                    if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
      }
      while(C0==0||C1==0||C2==0||C3==0){delay1(500);}
    }
  }
  // ---------------------------- END ROW 3 ---------------------------------- //
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
  // ---------------------------- scan row 4 ---------------------------------- //
```

```
R2 = 1;
R3 = 0;
if(R3 == 0)
{
 delay1(500);
 if((C0 == 0)||(C1 == 0)||(C2 == 0)||(C3==0))
 {
  if( C0 == 0 )        // code D. //
  {
                       key = 4;
                       if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                       if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
  }
  if( C1 == 0 )        // code H. //
  {
                       key = 8;
                       if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                       if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
  }
  if( C2 == 0 )        // code L. //
  {
                       key = 12;
                       if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                       if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
  }
  if( C3 == 0 )        // code P. //
  {
                       key = 16;
                       if(mode==1) {WR_EEPROM(41,key); dmsec(50);}
                       if(mode==2) {WR_EEPROM(40,key); dmsec(50);}
  }
  while(C0==0||C1==0||C2==0||C3==0){delay1(500);}
```

```
    }
  }
  // ---------------------------- END ROW 4 ---------------------------------- //
  R3 = 1;
////////////////////////////////////////////////////////////////////////////////////////////////////
}
I2C
sbit SCL = P3^2;
sbit SDA = P3^3;

void    I2c_Wait(void)
{
        unsigned char i;
          for (i=0;i<0x0f;i++) { }
}
void I2c_Clk(void)
{       SCL     =       1;
        I2c_Wait();
        SCL     =       0;
        I2c_Wait();
}

/*
void I2c_Ack(void)
{       SDA     = 0;
        I2c_Clk();
}
*/
void I2c_Nack(void)
{       SDA     = 1;
        I2c_Clk();
}
```

```
void I2c_Start(void)
{       SCL     =       0;
        SDA     =       1;
        SCL     =       1;
        I2c_Wait();
        SDA     =       0;
        I2c_Wait();
}
void I2c_Stop(void)
{       SCL     =       0;
        SDA     =       0;
        SCL     =       1;
        I2c_Wait();
        SDA     =       1;
        I2c_Wait();
}


bit I2c_Write(unsigned char Data )
{
                unsigned char i;
                bit     out;
                SCL     =       0;
        for (i=0;i<8;i++)
                {
                  out   =       Data & 0x80;
                  Data  =       Data << 1;
                  SDA   =       out;
                  I2c_Clk();
                }
      SDA     =         1;
```

```
    I2c_Clk();
            return(out);
}

unsigned char I2c_Read(void)
{
    unsigned char   Data,i;
    bit     out;
    SCL     =       0;
    SDA     =       1;
    I2c_Wait();
    for (i=0;i<8;i++)
    {
        SCL     =       1;
        I2c_Wait();
        out     =       SDA;
        Data    =       Data << 1;
        Data    =       Data | out;
        SCL     =       0;
        I2c_Wait();
    }
  return(Data);
}
```

Serial

```
#define start    0xFE          // BYTE 1 START. //
#define stop     0xFD          // BYTE 7 STOP. //
#define reg      0x02          // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg3     0x04          // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg4     0x44          // BYTE 3 BYTE CONTROL REGISTOR. //
#define reg5   0x43            // BYTE 3 BYTE CONTROL REGISTOR. //
#define check    0x12          // BYTE 3 BYTE CONTROL FIND FINGER. //
#define del1     0x20          // BYTE 3 BYTE CONTROL DEL ID. //
```

```
#define del2      0x21                  // BYTE 3 BYTE CONTROL DEL ID. //
#define del3      0x22                  // BYTE 3 BYTE CONTROL DEL ID. //
#define delay   0x00                    // BYTE DELAY MODULE. //
#define reg2    0x03                    // BYTE 5 BYTE CONTROL REGISTOR. //



void init_interr_serial(void)
{
        // XTAL 11.0592 MHZ
        // 12Machines/Cycle
          TMOD = 0x20;
          SCON = 0x52;
          PCON = 0x80;
          TH1  = 0xFA;                          //BaudRate 9600
             IE = 0x90;
          TR1  = 1;
          REN  = 1;
          TI   = 1;
          RI   = 0;
               ES   = 0;                        //Disable Interrupt Coz,Not
Work
}

// --------- delay key sw ---------- //
void delay1(unsigned int time)
{
    while(time)
    time--;
}
// ------------------------------- //
```

```
// ------ function recv char ------ //
char recv_char()
{
   char c;
   while(!RI)
         delay1(1);
   c=SBUF;
   RI=0;
   return(c);
}
////////////////////////////////////////
// ------- function send cahr ------ //
void send_char(char c)
{
   TI=0;
   SBUF=c;
   while (!TI)
     delay1(1);
}
////////////////////////////////////////

unsigned char upper_id;                    // upper id. //
unsigned char lower_id;                    // lower id. //

void send_command(unsigned char byte3,unsigned char byte4,unsigned char byte5, unsigned
char byte6)
{
        unsigned char byte7;
        byte7 = byte3^byte4^byte5^byte6;
        delay1(100);
        send_char(start);          delay1(100);    // BYTE 0  BYTE START. //
        send_char(delay);          delay1(100);    // BYTE 1  BYTE DELAY. //
```

```
        send_char(byte3);               delay1(100);    // BYTE 2. BYTE . //
        send_char(byte4);               delay1(100);    // BYTE 3  BYTE UPPER ID. //
        send_char(byte5);               delay1(100);    // BYTE 4. BYTE LOWER ID. //
        send_char(byte6);               delay1(100);    // BYTE 5. BYTE //
        send_char(byte7);               delay1(100);    // BYTE 6. BYTE CONTROL. //
        send_char(stop);        delay1(100);    // BYTE 7 BYTE STOP. //
        delay1(100);

}

unsigned char recv_serial[8];

void recv_command()
{
        unsigned char x;
        recv_serial[0] = recv_char();
        if(recv_serial[0]==start)
        {
                for(x=1;x<8;x++)
                {
                        recv_serial[x]  = recv_char();
                        upper_id        = recv_serial[3];
                        lower_id        = recv_serial[4];
                }
        }
}
```

ภาคผนวก ข.

เอกสารคู่มือประกอบการใช้ Module Fingerprint MRB200

MRB200 FINGERPRINT MODULE

FOR G6103 DEVELOPER'S MANUAL

V1.0.0.2

Editor by Zack

2004-10-17

MRB 200 fingerprint module for G6102 developer's manual

2004-10 - 1 -

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 2 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 3 -

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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 4 -

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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 5 -

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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 6 -

☐ 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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 7 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 9 -

☐ 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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 10 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 11 -

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;

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 12 -

☐ 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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 13 -

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.

2004-10

- 14 -

☐ 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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 15 -

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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 16 -

☐ 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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 17 -

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)

0 0x00000~0x00108 0x020000~0x020108

…… …… Program data ……

124 0x0F800~0x0F908

125 0x0FA00

.....

Const

It msut let the program 'const_move' to finish

the copy task located in 0x20100. 185 0x17200

Limited by

const_move

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.

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 18 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 19 -

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)

0 0x200000~0x2007FF 0x020000~0x02FFF

…… …… Program data ……

7 0x203800~0x203FFF 0x027000~0x027FFF

8 0x205000

……

Const data

It must let the program 'const_move' to finish

copying task located in 0x20100h. 13 0x207800

Limited by

const_move in the

program

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 20 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 21 -

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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 22 -

#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

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 23 -

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
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 24 -
#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();
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 25 -
} 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.l_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
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 26 -
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;
}
}
}
//
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 27 -
/*************************************************************************
**********************/
//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;
//
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 28 -
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)
{
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 29 -
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)
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 30 -
{
//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)
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 31 -
{
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);
//
```

MRB 200 fingerprint module for G6102 developer's manual

2004-10

```
- 32 -
}
//
/*****************************************************************************/
/*****************************************************************************/
//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,
MRB 200 fingerprint module for G6102 developer's manual
2004-10
- 33 -
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;
}
}
```

MRB 200 fingerprint module for G6102 developer's manual

2004-10

- 34 -

```
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(CMD) Answer(ACK)

Command Name Content

Form at

HE

AD

CH CODE

P1 P2 P3 CHK EN

D Return Value

Format

HEAD+CH+CODE+

XX+XX+AP+CHK+E

ND

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 P1P2 is the user's number

FE CH 00 P1 P2 00 CH

K

FD

AP=User power FE+CH+40+00+00+A

P+CHK+FD

CMD_VERIFY_DB

Compare the

fingerprint collected

with the one in the

module. P1P2 is the user's number

FE CH 01 P1 P2 00 CH

K

FD AP=Return processing

result.

FE+CH+41+00+00+A

P+CHK+FD

CMD_REG_START_DB

Collect the

fingerprint for the

first time P1P2 is the user's number and P3 is the user's power

FE CH 02 P1 P2 P3 CH

K

FD AP= Return

processing result.

FE+CH+42+00+00+A

P+CHK+FD

CMD_REG_SECOND_DB

Collect the

fingerprint for the

second time. P1P2 is the user's number and P3 is the user's power

FE CH 04 P1 P2 P3 CH

K

FD AP= Return

processing result.

FE+CH+44+00+00+A

P+CHK+FD

CMD_REG_END_DB

Collect the

fingerprint for the

third time P1P2 is the user's number and P3 is the user's power

FE CH 03 P1 P2 P3 CH

K

FD AP= Return

processing result.

FE+CH+43+00+00+A

P+CHK+FD

CMD_REG_DELETE_DB Delete the appointed

user. P1P2 is the user's number.

FE CH 20 P1 P2 00 CH

K

FD AP= Return

processing result.

FE+CH+60+00+00+A

P+CHK+FD

CMD_REG_ALLDEL_DB Delete all of the

users P3 is the user's power

FE CH 21 00 00 P3 CH

K

FD AP= Return

processing result.

FE+CH+61+00+00+A

P+CHK+FD

CMD_GET_USER_NUMBER_DB

Show the user's

number and his

power who has

registered.

P1P2 is the beginning user's record number. P3 is the fixed

return number, witch should be less than 30.

FE CH 10 P1 P2 P3 CH

K

FD Return the user's

number P1+P2 and his

power AP.

FE+CH+50+00+XX+

AP+CHK+FD

FE+P11+P21+AP1+o o

o

+P1p3+P2p3+APp3+CH

K+FD

P1+P2 is the user's

number Ap is user's

power

The length form P11 to
APp3 is XX

CMD_GET_VALUE

Collect the
fingerprint image and
get fingerprint
eigenvalue.

FE CH 26 00 00 00 CH
K

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 CH
K

FD Return the compare
result AP and user's
number P1+P2

FE+CH+52+P1+P2+A
P+CHK+FD

CMD_FROM_VALUE_DB

Send the eigenvalue
and save it to the
fingerprint library.

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.

FE CH 31 XX XX 00 CHK FD

Return processing

result.

FE+CH+71+P1+P2+A

P+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. P1P2 is the user's number.

FE CH 22 P1 P2 00 CH

K

FD Return processing

result and user's

number

FE+CH+62+00+XX+

AP+CHK+FD

FE+XX

(eigenvalue)+CHK+F

D

CMD_FROM_VERIFY_DB

Send fingerprint

eigenvalue to the

module and compare

it with the appointed

one in the library.

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.

FE CH 33 XX XX 00 CH

K

FD

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+00+00+00+ (XXXX-3) fingerprint eigenvalue

+CHK+FD

XXXX is the length of the transported characters.

FE CH 35 XX XX 00 CH

K

FD

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+00+00+00+ (XXXX-3) fingerprint eigenvalue

+CHK+FD

XXXX is the length of the transported characters

FE CH 34 XX XX 00 CH

K

FD

Return processing

result AP and the

user's number P1+P2

FE+CH+74+P1+P2+A

P+CHK+FD

CMD_GET_IMAGE

Collect the

fingerprint image and

get its outline.

FE CH 27 00 00 00 CH

K

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.

P3 : New transport speed

FE CH 0F 00 00 P3 CH

K

FD AP=Return the

original speed.

FE+CH+4F+00+00+A

P+CHK+FD

CMD_GET_VERSION Query the module's

version information.

FE CH 2A 00 00 00 CH

K

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 CH

K

FD Return processing

result AP.

FE+CH+6D+00+XX+

AP+CHK+FD

FE+XX

(eigenvalue)+CHK+F

D

CMD_TEST_COMM

Test if the

communication

works well.

FE CH 2C 00 00 00 CH

K

FD Return processing

result AP.

FE+CH+6C+00+00+

AP+CHK+FD