รายงานการวิจัย

การควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความจากโทรศัพท์เคลื่อนที่

Electrical Device Controlled With SMS Message of Mobile Phone

สัญญา ผาสุข
วิสุทธิ์ พงศ์พฤกษ์ธาตุ

มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย วิทยาเขตภาคใต้
พ.ศ. 2549

บทคัดย่อ

งานวิจัยนี้นำเสนอการควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่ ด้วยวิธีการใช้โทรศัพท์เคลื่อนที่ของผู้ส่งสร้างข้อความให้เป็นรหัสควบคุมแล้วส่งไปยังหมายเลขโทรศัพท์เคลื่อนที่เครื่องรับแล้วไมโครคอนโทรลเลอร์จะนำข้อความ SMS ที่ได้รับ ไปถอดรหัสเพื่อควบคุมการทำงานของอุปกรณ์ไฟฟ้าให้ทำงานหรือหยุดทำงานตามความต้องการของผู้ส่งข้อความ SMS ซึ่งต้นแบบของงานวิจัยนี้สามารถนำไปควบคุมอุปกรณ์ไฟฟ้าได้เป็นจำนวน 8 ช่อง

Abstract

This research is represents the remote control of Electrical Device with SMS Message from Mobile Phone. The SMS code of sender's mobile phone was sent to receiver 's number of mobile phone. This SMS code then was decode by microcontroller to control remote devices . The prototype have 8 channels for control electrical device.

กิตติกรรมประกาศ

รายงานการวิจัยฉบับนี้ได้ดำเนินการมาจนสำเร็จลุล่วงนั้น เกิดจากความร่วมมือของคณะวิจัย แม้ว่าจะมีเวลาในการทำวิจัยกันน้อย แต่ก็สามารถจัดสรรเวลาจึงสามารถช่วยกันกระทั่งสำเร็จลุล่วง และทั้งนี้ได้รับการสนับสนุนจากสาขาวิชาวิศวกรรมไฟฟ้ากำลัง ที่ให้การสนับสนุนทั้งสถานที่และเครื่องมือในการทำวิจัย และงานวิจัยนี้มีขึ้นมาได้เนื่องจากการสนับสนุนทุนอุดหนุนการทำวิจัยจากมหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย วิทยาเขตภาคใต้

สัญญา ผาสุข
วิสุทธิ์ พงศ์พฤกษ์ธาตุ

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของการวิจัย

การควบคุมอุปกรณ์ไฟฟ้าระยะไกลให้ทำงานและหยุดทำงานตามความต้องการ ยังเป็นสิ่งที่มีความจำเป็นมากในงานควบคุม เช่น การปิด เปิด ของเครื่องทวนสัญญาณวิทยุที่ติดตั้งไว้บนภูเขา หรือการปิด เปิด กังหันตีน้ำในนากุ้ง ซึ่ง โดยปกติจะต้องใช้คนประจำที่ชุดควบคุมทำให้ แทนที่จะสั่งการควบคุมจากที่พักก็ต้องเดินทางมาปิดหรือเปิดกังหันตีน้ำ และนอกจากจะใช้ควบคุมอุปกรณ์ดังตัวอย่างแล้วยังสามารถนำไปใช้ควบคุมอุปกรณ์ไฟฟ้าอย่างอื่น ๆได้อีก จากที่ผ่านมามีการสร้างชุดควบคุมอุปกรณ์ไฟฟ้าผ่านคู่สายโทรศัพท์ของผู้ให้บริการต่าง ๆ เช่น องค์การโทรศัพท์ แต่จะไม่สะดวกที่ต้องทำการขอเลขหมายและการเดินสายติดตั้ง และหากเปลี่ยนสถานที่การควบคุมอุปกรณ์ไฟฟ้าก็ยังต้องมีความยุ่งยากและค่าใช่จ่ายเพิ่ม

ปัจจุบันโทรศัพท์เคลื่อนที่เข้ามามีบทบาทในการสื่อสารมากขึ้น การที่จะหาโทรศัพท์เคลื่อนที่มาใช้งานเป็นเรื่องไม่ยากอีกต่อไปและสะดวกกับการนำไปใช้งานบริเวณใดก็ได้ที่สามารถรับสัญญาณจากผู้ให้บริการ ซึ่งนอกจากโทรศัพท์เคลื่อนที่จะสามารถใช้พูดคุยแล้วยังสามารถส่งข้อความสั้น ๆ (Short Message Service :SMS) ถึงกันได้ด้วย ดังนั้นข้อความ SMS ที่ส่งไป เราสามารถที่กำหนดเป็นรหัสใช้ควบคุมอุปกรณ์ไฟฟ้าให้ทำงานหรือหยุดทำงานได้หลายอุปกรณ์ตามความต้องการ เพียงแต่ผู้ควบคุมส่งข้อความ SMS จากเครื่องโทรศัพท์เคลื่อนที่จากเครื่องใดก็ได้มาที่เครื่องตัวรับ ก็สามารถควบคุมให้อุปกรณ์ไฟฟ้าตัวนั้นปิดหรือเปิดได้ ทั้งนี้จำนวนของอุปกรณ์ไฟฟ้าที่จะควบคุมขึ้นอยู่กับการออกแบบ

1.2 วัตถุประสงค์

1.2.1 เพื่อให้ได้เครื่องต้นแบบของเครื่องควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่

1.2.2 ได้องค์ความรู้เกี่ยวกับการควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่

1.3 ขอบเขตงานวิจัย

1.3.1 ใช้โทรศัพท์เคลื่อนที่รุ่นที่สามารถควบคุมด้วยคำสั่ง AT Command

1.3.2 สามารถควบคุมอุปกรณ์ไฟฟ้าให้ปิด หรือ เปิด ได้ถึง 8 เครื่อง

1.3.4 ควบคุมการทำงานโดยใช้ไมโครคอนโทรลเลอร์

13.5 สามารถใช้ได้กับอุปกรณ์ไฟฟ้าทุกชนิดที่ต้องการทำงานแบบ ปิด เปิดโดยการควบคุมระยะไกล

1.4 ประโยชน์ที่คาดว่าจะได้รับและหน่วยงานที่นำผลการวิจัยไปใช้ประโยชน์

1.4.1 ได้เครื่องต้นแบบของเครื่องควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่

1.4.2 สามารถนำไปควบคุมอุปกรณ์ไฟฟ้าระยะไกลตามความต้องการ

1.4.3 สามารถนำไปสร้างเป็นผลิตภัณฑ์ที่เสนอให้กับบุคคลภายนอก

1.4.4 ได้องค์ความรู้เกี่ยวกับการสื่อสารระยะไกลด้ายโทรศัพท์เคลื่อนที่

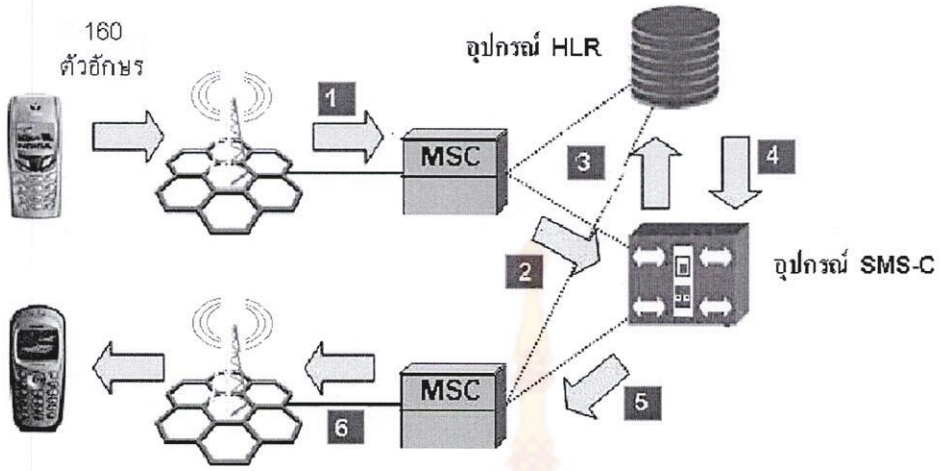
1.4.5 สามารถนำองค์ความรู้ที่ได้ไปประยุกต์ใช้กับงานวิจัยที่ต้องใช้องค์ความรู้นี้เป็นพื้นฐานต่อไป

บทที่ 2
ทฤษฎี

2.1 กระบวนการส่งข้อความ SMS

SMS (Short Message Service) เป็นบริการพื้นฐานประเภทหนึ่งที่ถือกำเนิดขึ้นมาพร้อมกับระบบโทรศัพท์เคลื่อนที่ยุค 2 G ไม่ว่าจะเป็นมาตรฐาน GSM หรือ CDMA โดยมีวัตถุประสงค์เพื่อให้เป็นการรับส่งข้อความสั้น ๆ ระหว่างผู้ใช้โทรศัพท์เคลื่อนที่ด้วยกัน การรับส่งข้อความ SMS สามารถทำได้ทุกเมื่อตราบใดที่เครื่องลูกข่ายโทรศัพท์เคลื่อนที่ยังเปิดเครื่องทำงานอยู่ โดยไม่จำเป็นว่าเครื่องลูกข่ายกำลังถูกใช้งานโทรศัพท์อยู่หรือไม่ ทั้งนี้เพราะการรับส่งข้อความ SMS จะกระทำผ่านช่องสื่อสารควบคุม (Control Channel) ระหว่างเครื่องลูกข่ายกับสถานีฐานโทรศัพท์เคลื่อนที่ ซึ่งเป็นช่องสื่อสารขนาดแบนด์วิดธ์เล็ก ๆ เป็นสาเหตุที่ทำให้ต้องมีการจำกัดขนาดของข้อความ SMS แต่ละชุดไว้ไม่ให้ใหญ่เกินกว่า 160 ตัวอักษร เนื่องจากหากข้อความ SMS มีขนาดใหญ่มากเกินไปจะทำให้เกิดผลกระทบต่อความหนาแน่นของข้อมูลที่มีการรับส่งผ่านช่องสื่อสารควบคุม ซึ่งอาจมีผลต่อการให้บริการเชื่อมต่อวงจรยกเลิกการเชื่อมต่อหรือการย้ายข้ามเซลล์ได้ ทำให้คุณภาพในการให้บริการของเครือข่ายโทรศัพท์เคลื่อนที่นั้น ๆ ด้อยประสิทธิภาพลง

โครงสร้างการเชื่อมต่อเครือข่ายโทรศัพท์เคลื่อนที่กับอุปกรณ์ SMS-C หรือ Short Message Service Center ซึ่งทำหน้าที่คล้ายกับที่ทำการไปรษณีย์สำหรับ รับและส่งต่อ (Store and Forward) ข้อความ SMS จากผู้ใช้บริการรายหนึ่งไปสู่อีกรายหนึ่ง เป็นไปดังแสดงในภาพที่ 2-1 การเชื่อมต่อระหว่างชุมสายโทรศัพท์เคลื่อนที่ (Mobile Service Switching Center หรือ MSC) กับอุปกรณ์ SMS-C จะทำโดยใช้ระบบสัญญาณแบบ CCS7 (Common Channel Signaling System No.7) ซึ่งหมายความว่าไม่จำเป็นต้องมีการสร้างวงจรสื่อสารในลักษณะของการให้บริการสนทนาระหว่างบุคคลโดยทั่วไป อุปกรณ์ SMS-C ยังมีการติดต่อสื่อสารกับอุปกรณ์ HLR (Home Location Register) เพื่อทำหน้าที่ตรวจสอบหาตำแหน่งที่อยู่ของเลขหมายโทรศัพท์เคลื่อนที่ปลายทางว่ากำลังอยู่ในพื้นที่ให้บริการของ MSC ใด ในกรณีที่ผู้ใช้บริการเลขหมายปลายทางปิดเครื่องหรืออยู่นอกพื้นที่ให้บริการ ข้อความ SMS ก็จะถูกเก็บไว้ชั่วคราวภายในอุปกรณ์ SMS-C จนกว่าอุปกรณ์ HLR จะตรวจพบว่าผู้ใช้บริการนั้น ๆ เปิดเครื่องหรือกลับเข้าสู่พื้นที่บริการอีกครั้งหนึ่ง ก็จำทำการส่งสัญญาณแจ้งมายัง SMS-C บอกตำแหน่งที่อยู่ในระดับของ MSC ให้อุปกรณ์ SMS-C ทราบเพื่อทำการส่งข้อความ SMS นั้นไปยังเลขหมายปลายทาง



ภาพที่ 2-1 โครงสร้างการเชื่อมต่อเครือข่าย GSM กับอุปกรณ์ SMS-C

กระบวนการรับส่งข้อความ SMS ในลักษณะข้างต้น ซึ่งนิยมเรียกกันว่า "Point-to-Point SMS" สามารถนำมาอธิบายเป็นขั้นตอนย่อย ๆ ได้ดังนี้

1. ผู้ใช้บริการโทรศัพท์เคลื่อนที่ต้นทางทำการพิมพ์ข้อความ SMS ความยาวไม่เกิน 160 ตัวอักษร (จำกัดโดยเครื่องลูกข่ายโทรศัพท์เคลื่อนที่) แล้วทำการส่งไปยังเลขหมายโทรศัพท์เคลื่อนที่ปลายทาง ทั้งนี้ภายในเครื่องลูกข่ายจำเป็นต้องมีการตั้งค่าหมายเลขของอุปกรณ์ SMS-C ซึ่งโดยทั่วไปมักทำเพียงครั้งเดียวตั้งแต่ซื้อเครื่องมาใช้ใหม่ ๆ ข้อความ SMS ที่ถูกส่งออกจากเครื่องลูกข่ายจะถูกส่งผ่านช่องสื่อสารควบคุมของเครือข่ายโทรศัพท์เคลื่อนที่จากสถานีฐานไปสู่อุปกรณ์ MSC ซึ่งจะทราบว่าสัญญาณควบคุมที่ได้รับมาเป็นข้อความ SMS ที่จะต้องถูกส่งต่อไปยังอุปกรณ์ SMS-C ตัวใด ซึ่งในกรณีของการเปิดให้บริการรับส่งข้อความ SMS ข้ามเครือข่ายกันระหว่างบริษัทผู้ให้บริการหลาย ๆราย อุปกรณ์ MSC ก็จะต้องทราบว่าอุปกรณ์ SMS-C ทุกตัว มีเลขหมายกำกับเป็นเลขหมายใดบ้าง

2. ข้อความ SMS ที่อยู่ในรูปของสัญญาณควบคุมแบบ CCS7 ถูกส่งต่อไปยังอุปกรณ์ SMS-C ปลายทาง ซึ่งข้อความ SMS จะถูกนำไปเข้าคิวไว้ภายในอุปกรณ์ SMS-C เพื่อการรอส่งไปยังเลขหมายปลายทาง ในขั้นตอนนี้จะเห็นได้ว่าอุปกรณ์ SMS-C ทำหน้าที่เหมือนกับที่ทำการไปรษณีย์ ในลักษณะของการให้บริการส่งข้อความที่เข้ามาในคิวก่อน (First Come First Service) จุดนี้เองที่ทำให้ในบางครั้งการส่งข้อความ SMS อาจกินเวลานาน กว่าจะถูกลำเลียงต่อไปยังหมายเลขปลายทาง เนื่องจากมีจำนวนข้อความ SMS เข้าคิวรออยู่เป็นจำนวนมาก สังเกตุได้ง่าย ๆ ในช่วงเทศกาลต่าง ๆ ที่ผู้ให้บริการโทรศัพท์เคลื่อนที่นิยมส่งข้อความ SMS กันเป็นจำนวนมาก

3. เมื่อถึงคิวการส่งข้อความ SMS นั้น ๆ แล้ว อุปกรณ์ SMS-C จะส่งสัญญาณผ่านเครือข่าย CCS7 ไปยังอุปกรณ์ HLR ซึ่งเป็นฐานข้อมูลหลัก ที่หน้าที่เก็บรวบรวมข้อมูลที่สำคัญเกี่ยวกับเลขหมายโทรศัพท์เคลื่อนที่ทั้งหมดภายในเครือข่าย เพื่อสอบถามถึงตำแหน่งที่อยู่ของเลขหมายโทรศัพท์เคลื่อนที่ปลายทาง

4. อุปกรณ์ HLR จะตอบกลับไปยัง SMS-C ถึงรหัสตำแหน่งที่อยู่ของเลขหมายปลายทางที่ต้องการ ซึ่งในขั้นตอนนี้หากเลขหมายดังกล่าวปิดเครื่องหรืออยู่นอกพื้นที่ให้บริการ HLR ก็จะทำการแจ้งกลับไปยังอุปกรณ์ SMS-C เพื่อให้เก็บข้อความ SMS นั้นไว้ในฐานข้อมูลชั่วคราวพร้อมกับทดรหัสไว้ภายในตัว HLR เองเพื่อเตือนให้มีการส่งข้อความยืนยันกลับไปยัง SMS-C ทันทีที่ผู้ใช้บริการรายดังกล่าวเปิดเครื่องหรือกลับเข้ามาอยู่ในพื้นที่ให้บริการแล้ว ทั้งผู้ใช้บริการและผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่จะกำหนดนโยบายว่า ยินยอมให้มีการเก็บข้อความ SMS รอไว้ภายในอุปกรณ์ SMS-C ได้นานเท่าไรก่อนที่ข้อความนั้นจะถูกลบออกจากระบบในกรณีที่หมายเลขปลายทางปิดเครื่องไปเป็นเวลานานหลาย ๆ วัน

5. ในกรณีที่อุปกรณ์ SMS-C ทราบถึงตำแหน่งที่อยู่ของเลขหมายปลายทางแล้วก็จะทำการสร้างสัญญาณควบคุมซึ่งบรรจุข้อความ SMS นั้นผ่านไปบนเครือข่าย CCS7 สู่ชุมสายโทรศัพท์เคลื่อนที่ปลายทางทั้งนี้ไม่สำคัญว่าอุปกรณ์ SMS-C จะเชื่อมต่อโดยตรงกับชุมสายโทรศัพท์เคลื่อนที่ปลายทางหรือไม่ เพราะมาตรฐานเครือข่ายสัญญาณ CCS7 จะทำหน้าที่นำข้อความ SS7 จากอุปกรณ์ SMS-C เคลื่อนที่ผ่านเครือข่ายชุมสายโทรศัพท์เคลื่อนที่ไปยังอุปกรณ์ MSC ปลายทางเอง

6. ชุมสายโทรศัพท์เคลื่อนที่ทำการรับข้อความ SMS นั้นแล้วลำเลียงส่งผ่านต่อไปยังสถานีฐานที่ให้บริการแก่เลขหมายปลายทางนั้น เป็นอันเสร็จสิ้นกระบวนการรับส่งข้อความแบบ SMS

จะเห็นว่าการสื่อสารแบบ SMS นั้น แม้จะมีข้อดีในแง่ของการรับฝากข้อความไว้กับระบบเครือข่ายโดยไม่ต้องกังวลว่าเลขหมายปลายทางจะเปิดเครื่องอยู่หรือไม่ อีกทั้งยังสามารถรับส่งข้อความได้ทั้ง ๆที่กำลังใช้เครื่องโทรศัพท์อยู่ก็ตาม

2.2 หลักการรับส่งข้อความ SMS

องค์การ ETSI (European Telecommunications Standards Institute) เป็นองค์กรอิสระที่ไม่แสวงหาผลกำไรทำหน้าที่กำหนดมาตรฐานทางด้านโทรคมนาคมได้กำหนดมาตรฐานการส่งข้อความ SMS ไว้ในคู่มือ GSM 03.40 และ GSM 03.38 ซึ่งสามารถส่งได้สูงถึง 160 ตัวอักษร โดยแต่ละตัวอักษรใช้รหัสขนาด 7 บิต ที่กำหนดไว้ในตารางที่ 3 นอกจากนั้นยังมีการใช้ตัวอักษร

ชนิดอื่น ๆ เช่น ขนาด 8 บิต หรือ 16 บิต ซึ่งมีวัตถุประสงค์เพื่อการใช้งานที่แตกต่างกันออกไป ซึ่งในบทนี้จะพูดถึงเฉพาะแบบ 7 บิต

โหมดของการรับส่งข้อมูล

การรับส่งข้อมูล SMS มีอยู่ด้วยกัน 2 โหมด คือ เท็กซ์โหมด (Text Mode) และพีดียูโหมด PDU (Protocol Description Unit Mode) การส่งข้อความในเท็กซ์โหมดนั้นจะเป็นการนำข้อความที่ต้องการส่งมาเข้ารหัสก่อน แล้วค่อยส่งข้อมูลในพีดียูโหมดอีกที อย่างไรก็ตามเครื่องโทรศัพท์เคลื่อนที่บางรุ่นอาจไม่สนับสนุนการใช้งานในเท็กซ์โหมด ซึ่งการเข้ารหัส (ส่งข้อความ) และถอดรหัส (รับข้อความ) สำหรับในเท็กซ์โหมดนี้มีหลายแบบด้วยกันเช่น "PCCP437W, "PCDN" , "8859-1","IRA" และ "GSM"

การเชื่อมต่อกับเครื่องโทรศัพท์เคลื่อนที่เพื่อรับส่งข้อความสามารถเลือกใช้ได้ทั้ง 2 โหมด แต่จะเห็นได้ว่าการเลือกใช้เท็กซ์โหมดจะมีข้อจำกัดเนื่องจากโทรศัพท์เคลื่อนที่บางรุ่นอาจจะไม่สนับสนุนและยังถูกจำกัดวิธีการเข้าและถอดรหัส ซึ่งมีเพียงไม่กี่แบบตามที่กล่าวมาข้างต้น ซึ่งในบางกรณีอาจไม่สะดวกนักแต่ถ้าเลือกพีดียูโหมดจะสามารถเลือกหรือสร้างการเข้ารหัสและถอดรหัสได้ทุกรูปแบบตามต้องการโดยไม่มีข้อจำกัด

การรับข้อความ SMS ในพีดียู โหมด

ถ้าหากเราเชื่อมต่อกับโทรศัพท์เคลื่อนที่แล้วทำการสั่งอ่านข้อความ SMS ที่อยู่ใน Inbox โดยใช้คำสั่ง AT+CMGR ข้อมูลที่ได้รับจะอยู่ในรูปของสตริงที่ประกอบไปด้วยข้อมูลของผู้ส่ง, ข้อมูล SMS Service Center (SMSC), Time Stamp และอื่น ๆ ที่จำเป็นและตามด้วยส่วนของข้อความซึ่งจะอยู่ท้ายสุดของสตริง ซึ่งตัวอย่างข้อมูลสตริงต่อไปนี้รับได้จาก SIEMENS รุ่น C45 ซึ่งข้อความที่ส่งมาคือ "10101010" จากมือถืออีกเครื่องหนึ่ง โดยข้อมูลสตริงจะอยู่ในรูปของตัวเลขฐาน 16 และฐาน 10 (ในบางส่วน) โดยจะเรียกตัวเลขแต่ละคู่ว่า Octet ซึ่งมีรายละเอียดดังตารางที่ 2-1

ตัวอย่างข้อมูลสตริง

0691666113013004 0A91665124841700005070207182308208315 80C1683C560

ตารางที่ 2-1 ส่วนประกอบของข้อมูลสตริงจากข้อความ SMS

กลุ่มตัวเลข 8 บิต (Octet)	รายละเอียด
06	ความยาวของ SMSC Information 6 Octets (bytes)
91	รูปแบบของเลขหมาย SMSC 91 หมายถึง เลขหมายแบบสากล (international format)

66 61 13 01 30	เลขหมาย SMSC (แบบ decimal semi-octets) ซึ่งจะเป็นเลขฐาน 10 สลับ Nibble ในกรณีนี้เลขหมายจริงของ Cervice Center คือ “+6616311003”
04	Firt octer of this SMS-DELIVER message
0A	ความยาวของเลขหมายผู้ส่ง (0A hex = 10 ตัว
91	รูปแบบของเลขหมายผู้ส่ง 91 หมายถึง เลขหมายแบบสากล (international format)
66 51 24 84 17	เลขหมายผู้ส่ง (แบบ decimal semi-octets) เป็นเลขฐาน 10 สลับ nibble หมายผู้ส่งที่แท้จริงคือ “+6615424871”
00	TP-PID.(Protocal identifier) ในกรณีนี้คือ 00
00	TP-DCS (Data coding scheme)00 คือเข้ารหัสข้อความแบบ 7 bits Default Alphabet
50 70 20 71 82 30 82	TP-SCTS. ข้อมูล Time Stamp (แบบ Decimal semi-octets) สลับ nibble
08	TP-UDL.User data length จำนวนตัวอักษรของข้อความที่ส่งในที่นี้คือ 8 ตัว
31580C1683C560	TP-UD. ข้อความ “10101010” ที่เข้ารหัสแล้วจากตัวอักษรแบบ 7 bits เป็นข้อมูล byteขนาด 8 bits

ข้อมูลทั้งหมดในตารางเป็นเลขฐาน 16 ขนาด 8 บิต ยกเว้นหมายเลข Service center , เลขหมายที่ส่ง , Time Stamp จะเป็นเลขฐานสิบขนาด 8 บิต สลับเป็นหลักเป็นคู่ ๆ (สลับ Nibble) ในส่วนของข้อมูลที่เป็นข้อความนั้นเป็นเลขฐาน 16 ขนาด 8 บิตเช่นกัน โดยข้อมูลนี้จะใช้แสดงข้อความที่ประกอบไปด้วยตัวอักษรขนาด 7 บิต ซึ่งผ่านการแปลง (เข้ารหัส) ข้อมูลจากตัวอักษรขนาด 7 บิตให้เป็นเลขตัวอักษรขนาด 8 บิตมาแล้ว ในส่วนของข้อมูลที่เป็นเลขฐานสิบเช่นเลขหมายที่ส่ง ตัวเลขในแต่ละคู่ (1 Byte) จะถูกสลับหลักกันเช่น เลขหมายจริง “+66 015424871” จะถูกสลับในแต่ละคู่เป็น “66 51 24 84 17” (66 คือรหัสประเทศ ส่วนเลขหมวดของหมายเลขของมือถือจะถูกตัดเลข 0 ออก เช่น 01 จะเหลือแค่ 1 แล้วจึงนำตัวเลขทั้งหมดมาต่อกันแล้วสลับคู่) ในส่วนของเวลาก็เช่นเดียวกัน ข้อมูล “50 70 20 71 82 30 82” ซึ่งมีรูปแบบเป็น “YY/N/DD/HH:MM:SS:ss) หมายถึงข้อความนี้ส่งเมื่อ “05/07/20 17:28:03:28”

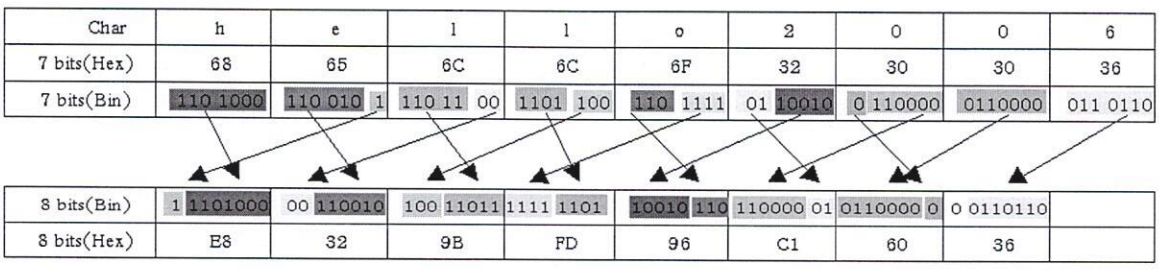
2.3 การแปลงรหัสตัวอักษรชนิด 7 บิตเป็นข้อมูล 8 บิต

มาตรฐานการรับส่งข้อความ SMS นั้นได้ถูกกำหนดให้ รับส่งได้สูงถึง 160 ตัวอักษร (มาตรฐาน ETSI GSM 03.40 และ GSM 03.38) โดยแต่ละตัวอักษรของข้อความจะเป็นข้อมูลขนาด 7 บิต ที่จะถูกเข้ารหัสให้กลายเป็นข้อมูลขนาด 8 บิต ดังตัวอย่างข้อความ "hello2006" จะถูกแปลงเป็นข้อมูลขนาด 8 บิต ตามมาตรฐาน ETSI GSM 03.38 ดังแสดงในตารางที่ 2-2

ตารางที่ 2-2 แสดงตัวอย่างข้อมูลตัวเลขขนาด 7 บิต แทนตัวอักษร

ตัวอักษร	ข้อมูลขนาด 7 บิต (Bin)	ข้อมูลขนาด 7 บิต (Hex)
h	110 1000	68
e	110 0101	65
l	110 1100	6C
l	110 1100	6C
o	110 1111	6F
2	011 0010	32
0	011 0000	30
0	011 0000	30
6	011 0110	36

โดยข้อมูลขนาด 8 บิตตัวแรกได้จากการนำบิตด้าน LSB ของตัวอักษรถัดไปมาเรียงต่อด้าน HSB ให้ครบ 8 บิต ก็จะได้ข้อมูลขนาด 8 บิตของตัวอักษรตัวแรก สำหรับตัวอักษรตัวที่ 2 ของ 7 บิต ก็จะเหลือ 6 บิต ก็ให้นำบิตด้าน LSB ของตัวอักษรถัดไปคือตัวที่ 3 มาเรียงต่อให้ครบ 8 บิต ก็จะได้ข้อมูลขนาด 8 บิตของตัวอักษรตัวที่ 2 ซึ่งต้องกระทำอย่างนี้จนครบทุกตัวอักษร โดยตัวอักษรตัวสุดท้ายถ้าข้อมูลหมดจะกำหนดให้บิตที่เหลือมีค่าเท่ากับ 0 ดังแสดงตัวอย่างในภาพที่ 2-2



ภาพที่ 2-2 แสดงการแปลงข้อความอักษรขนาด 7 บิตเป็นข้อมูลสตริงอักษรขนาด 8 บิต

หลังจากแปลงข้อความ“hello2006” จากรหัสอักษร 7 บิต 68 65 6C 6C 6F 32 30 30 36 เป็นข้อมูล 8 บิตจะได้ข้อมูลเป็นเลขฐานสิบหกใหม่เป็น E8 32 9B FD 96 C1 60 36 ซึ่งเห็นได้ว่าสามารถลดจำนวนข้อมูลจากจำนวน 9 ไบต์ เหลือ 8 ไบต์ ทำให้ใช้หน่วยความจำในการส่งข้อความน้อยลงแต่จำนวนตัวอักษรเพิ่มขึ้น

2.4 การแปลงข้อมูล 8 บิต เป็นรหัสตัวอักษรชนิด 7 บิต

เมื่อเครื่องโทรศัพท์มือถือรับข้อความสั้นซึ่งเป็นข้อมูลที่ถูกเข้ารหัสเป็นข้อมูลขนาด 8 บิตแล้ว ทำให้ข้อความที่ได้มานั้นหากแปลงเป็นรหัสของตัวอักษรเลยจะทำให้ไม่สามารถรู้ข้อความแท้จริงได้เนื่องจากรหัสของตัวอักษรจะเปลี่ยนไป จึงจำเป็นต้องทำการแปลงรหัสจากข้อมูลขนาด 8 บิตให้กลายเป็นรหัสของตัวอักษรขนาด 7 บิต เสียก่อน โดยวิธีย้อนกลับการแปลงรหัสตัวอักษร 7 บิต เป็นข้อมูลขนาด 8 บิต ซึ่งต้องเลื่อนบิตข้อมูลขนาด 8 บิต แล้วแปลงเป็นรหัสตัวอักษรขนาด 7 บิต ดังแสดงในตัวอย่าง เพื่อแปลงเป็นข้อความให้ผู้รับเข้าใจ

ตัวอย่างข้อมูลสตริงอักษรขนาด 8 บิต

E8 32 9B FD 96 C1 60 36

แปลงเป็นข้อความอักษรขนาด 7 บิตซึ่งแสดงข้อความ “hello2006” ดังแสดงในภาพที่ 2-3

8 bits(Hex)	E8	32	9B	FD	96	C1	60	36	
8 bits(Bin)	1 1101000	00 110010	100 11011	1111 1101	10010 110	110000 01	0110000 0	0011 0110	
7 bits(Bin)	110 1000	110 010 1	110 11 00	1101 100	110 1111	01 10010	0 110000	0110000	011 0110
7 bits(Hex)	68	65	6C	6C	6F	32	30	30	36
Char	h	e	l	l	o	2	0	0	6

ภาพที่ 2-3 แสดงการแปลงข้อมูลสตริงอักษรขนาด 8 บิต เป็นข้อความอักษรขนาด 7 บิต

2.5 คำสั่ง AT Command กับโทรศัพท์เคลื่อนที่

การสื่อสารระหว่างคอมพิวเตอร์กับอุปกรณ์สื่อสารต่าง ๆ เช่น โมเด็มหรืออุปกรณ์ DTE (Data Terminal Equipment) นั้นสามารถใช้ชุดคำสั่งที่เป็นมาตรฐานที่เรียกว่า AT Command ในการติดต่อเพื่อโต้ตอบ ตั้งค่าหรือสั่งงานอุปกรณ์เหล่านั้น ให้ทำงานตามที่ต้องการโดยใช้ชุดคำสั่งพื้นฐานที่ถูกกำหนดไว้ใน Hayes AT command ซึ่งบริษัท Hayes เป็นผู้คิดค้นชุดคำสั่งนี้

เพื่อใช้กับโมเด็มของตนและต่อมาได้กลายเป็นมาตรฐานสำหรับผู้ผลิตโมเด็มรายอื่น ๆ โดยอาจจะมีชุดคำสั่งขยาย (Extended AT command) เพื่อใช้เป็นการเฉพาะสำหรับผู้ผลิตรายนั้น ๆ ก็ได้

การติดต่อกับโทรศัพท์เคลื่อนที่ก็เช่นเดียวกันเราสามารถใช้ชุดคำสั่งที่กำหนดไว้ใน GSM AT command ซึ่งมีคำสั่งเพิ่มเติมที่เหมาะสมสำหรับการใช้งานและควบคุมโทรศัพท์เคลื่อนที่ ไม่ว่าจะเป็นคำสั่งหมุนหมายเลขโทรศัพท์ไปยังเลขหมายหมายทาง หรือการรับโทรศัพท์ และการส่งข้อความสั้น SMS ก็สามารถทำได้โดยใช้คำสั่ง AT Command ไปควบคุมการทำงานของโทรศัพท์เคลื่อนที่ได้ ซึ่งการควบคุมโทรศัพท์เคลื่อนที่ด้วยคำสั่ง AT Command นั้นจะควบคุมผ่านทาง พอร์ตอนุกรม (Serial Port) ของคอมพิวเตอร์ หรือของไมโครคอนโทรลเลอร์ ไปยังเทอร์มินอล (Terminal) ของโทรศัพท์เคลื่อนที่แต่ละรุ่นที่รองรับคำสั่ง AT Command ซึ่งจำเป็นต้องมีสาย Data Link เชื่อมต่อระหว่างคอมพิวเตอร์กับโทรศัพท์เคลื่อนที่ โดยใช้โปรแกรมเทอร์มินอลต่าง ๆ เช่น Hyper Terminal ของ Windows ติดต่อควบคุมให้โทรศัพท์เคลื่อนที่ทำงานตามต้องการ

ตัวอย่างคำสั่ง GSM AT Command

atd018289492 // สั่งให้ต่อโทรศัพท์ไปยังหมายเลขนี้ (บางเครื่องอาจต้องใช้ ; ต่อท้าย)

BUSY // สายไม่ว่าง (ถ้าอีกฝั่งรับสายจะตอบ CONNECT)

ath // สั่งวางสาย

ok // ตกลง

คำสั่งเกี่ยวกับ SMS

at+csms=0 // เช็คว่าสนับสนุนคำสั่งเกี่ยวกับ SMS หรือไม่

+CSMS: 1,1,1 //

OK // สนับสนุน

AT+CMGF : คำสั่งเลือกรูปแบบของข้อความ

AT+CMGF = <[mode]> *รูปแบบการใช้งาน*

โดยหาก mode มีค่าเป็น

0 หมายถึงรูปแบบข้อความเป็นแบบโหมดพีดียู (PDU Mode)

1 หมายถึงรูปแบบข้อความเป็นแบบโหมดเท็กซ์ (Text Mode)

AT+CMGL : คำสั่งแสดงข้อความที่เก็บอยู่ในหน่วยความจำ

AT+ CMGL=<Stat> *รูปแบบการใช้งาน*

โดยหาก Stat มีค่าเป็น

0 หมายถึง แสดงข้อความที่ยังไม่ได้อ่าน

1 หมายถึง แสดงข้อความที่ได้อ่านไปแล้ว

2 หมายถึง แสดงข้อความที่ยังไม่ได้ส่ง

3 หมายถึง แสดงข้อความที่ได้ส่งไปแล้วซึ่งยังเก็บอยู่ในหน่วยความจำ

4 แสดงข้อความทั้งหมดในหน่วยความจำ

AT+CMGR : เป็นคำสั่งอ่านข้อความสั้นจากหน่วยความจำ

AT+CMGR = <index> *รูปแบบการใช้งาน*

index หมายถึง ตำแหน่งของข้อความที่เก็บอยู่ในหน่วยความจำ

เช่น

AT+cmgr=1 // อ่านข้อความที่ 1 ใน inbox

โทรศัพท์มือถือจะตอบกลับมายังตัวควบคุมเป็น

+CMGR: 1,27

0691668111808804 0A91662950268000004040121171938 20AE8301C9E4787

E1F03C

OK // อ่านข้อความเรียบร้อย (เป็นข้อความยังไม่ถอดรหัส)

ซึ่งข้อมูลที่โทรศัพท์เคลื่อนที่ตอบกลับมาหลังจากใช้คำสั่ง AT+CMGR ต้องนำไปทำการถอดรหัสให้เป็นข้อความ ดังที่ได้กล่าวไปแล้วข้างต้นเพื่อนำไปใช้ในการควบคุมอุปกรณ์ไฟฟ้าต่อไป

ทดสอบการเชื่อมต่อกับมือถือ

ก่อนอื่นเราต้องทดสอบการเชื่อมต่อกับมือถือด้วยสาย Data Link เพื่อให้มั่นใจว่ามือถือของเราสามารถทำงานได้ถูกต้องและรองรับการส่ง SMS โดยทำการต่อตามรูปที่ 1 โดยนำสาย Data Link ที่ใช้ได้กับมือถือของเรามาต่อปลายข้างหนึ่งที่มักจะเป็นหัวต่อแบบ DB 9 ต่อเข้ากับคอมพิวเตอร์ที่ Port Com 1 หรือ Com 2 ก็ได้ถ้ามีแล้วต่ออีกปลายหนึ่งของสายซึ่งจะเป็นหัวต่อที่เหมาะสมกับมือถือแต่ละรุ่นเสียบเข้ากับตัวมือถือ

เมื่อต่อสายเรียบร้อยดีแล้วและตรวจให้แน่ใจว่าทั้งคอมพิวเตอร์และมือถือได้เปิดเครื่องไว้แล้วทางฝั่งคอมพิวเตอร์ให้เรียกใช้โปรแกรม Hyper terminal บน Windows โดยเลือกการเชื่อมต่อเป็น Direct to COM 1 หรือ COM 2 ซึ่งก็แล้วแต่ว่าเราต่อสายเอาไว้ทาง Port ใด หลังจากนั้นให้เลือกความเร็วเป็น 9600 bps และเลือก Flow control เป็น Hardware

เมื่อเข้าสู่โปรแกรมแล้วทดลองคอมพิมพ์ at แล้ว กด Enter ถ้าการเชื่อมต่อถูกต้องมันจะตอบ OK กลับมา หลังจากนั้น ทดลองพิมพ์คำสั่ง at+csms=0 แล้ว กด Enter ถ้าหากมันตอบข้อมูลและ OK กลับมา แสดงว่ามือถือเครื่องนี้พร้อมจะใช้งานสำหรับโครงงานนี้แล้ว

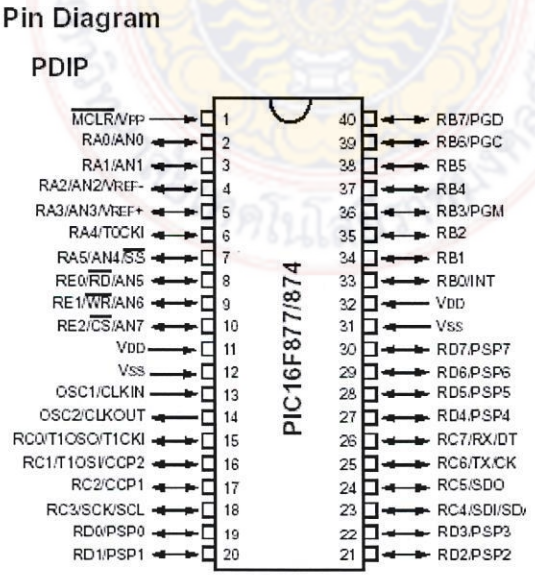
บทที่ 3
ฮาร์ดแวร์

งานวิจัยนี้ได้ใช้ไมโครคอนโทรลเลอร์ PIC16F877 เป็นตัวประมวลผลและควบคุมการทำงานของอุปกรณ์ไฟฟ้าตามรหัสคำสั่งจากข้อความ SMS ซึ่งถูกส่งมาจากโทรศัพท์เคลื่อนที่อีกเครื่องหนึ่ง ซึ่งฮาร์ดแวร์และวงจรที่เกี่ยวข้องมีดังต่อไปนี้

3.1 คุณสมบัติและหน่วยความจำของไมโครคอนโทรลเลอร์ PIC16F877

ไมโครคอนโทรลเลอร์ตระกูล PIC มีสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard architecture) กล่าวคือ มีการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกัน โดยมีบัสสำหรับติดต่อแยกกันด้วยจะเห็นได้ว่าซีพียูภายในไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมด้วยบัสแอดเดรส 13 บิต และบัสข้อมูลหน่วยความจำ 14 บิต ในขณะที่บัสสำหรับติดต่อหน่วยความจำข้อมูลและรีจิสเตอร์ภายในเป็นแบบ 8 บิตทั้งบัสแอดเดรสและบัสข้อมูล

นอกจากการจัดสถาปัตยกรรมแบบนี้แล้ว การกระทำคำสั่งของไมโครคอนโทรลเลอร์ PIC ยังใช้กระบวนการที่เรียกว่า ไปป์ไลน์ (Pipeline) ทำให้สามารถเฟตช์คำสั่งถัดไป ในขณะที่กำลังเอ็กซิคิวต์คำสั่งปัจจุบัน ส่งผลให้ความเร็วในการทำงานของไมโครคอนโทรลเลอร์เพิ่มมากขึ้น นั่นจึงเป็นที่มาของความสามารถกระทำคำสั่ง 1 คำสั่งภายในสัญญาณนาฬิกา 1 ลูก



ภาพที่ 3-1 แสดงชื่อและตำแหน่งขาของ PIC16F877

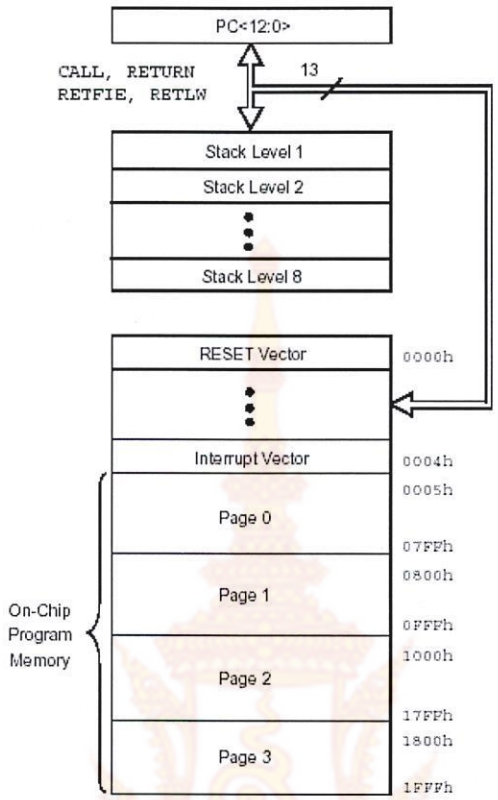
คุณสมบัติของ PIC16F877

- ซีพียูเป็นแบบ RISC (Reduce Instruction Set Computer) มีคำสั่งใช้งานเพียง 35 คำสั่ง
- สามารถกระทำคำสั่งโดยใช้สัญญาณนาฬิกาเพียง 1 ลูก ยกเว้นคำสั่งกระโดด
- ความถี่สัญญาณนาฬิกาตั้งแต่ไฟตรงถึง 20 MHz
- หน่วยความจำโปรแกรม 8 กิโลเวิร์ด
- หน่วยความจำข้อมูลแรม 368 ไบต์
- หน่วยความจำข้อมูลอีอีพรอม 256 ไบต์
- ตอบสนองแหล่งกำเนิดสัญญาณอินเตอร์รัพต์สูงสุดถึง 15 แหล่ง
- มีวงจรวอตช์ด็อกไทเมอร์ (WDT) ที่มีวงจรออสซิลเลเตอร์ในตัว
- ไฟเลี้ยง +2 ถึง +5.5 โวลท์
- กระแสซิงก์และซอร์สของพอร์ต 25 mA
- มีวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 10 บิต จำนวน 8 ช่อง
- มีวงจรเชื่อมต่ออุปกรณ์ทั้ง PSI และ I2C
- มีวงจรสื่อสารข้อมูลอนุกรม (USART)

การจัดสรรหน่วยความจำและรีจิสเตอร์ควบคุมของ PIC16F877

หน่วยความจำโปรแกรม (Program Memory) เป็นส่วนที่มีความสำคัญมาเพราะเป็นที่เก็บข้อมูลคำสั่งทั้งหมดซึ่งใช้ในการกำหนดให้ไมโครคอนโทรลเลอร์ทำงาน หน่วยความจำของ PIC16F877 เป็นแบบแฟลช (Flash memory) ทำให้สามารถลบและเขียนใหม่ได้นับแสนครั้ง ซึ่งหน่วยความจำโปรแกรมหลังจากทำการเขียนในขั้นตอนของการโปรแกรมแล้ว ก็จะเป็นหน่วยความจำที่สามารถอ่านได้อย่างเดียว

PIC16F877 มีโปรแกรมเคาน์เตอร์ (PC) ขนาด 13 บิต เพื่อกำหนดการเข้าถึงหน่วยความจำโปรแกรม เนื่องจากไมโครคอนโทรลเลอร์มีขนาดหน่วยความจำโปรแกรมค่อนข้างใหญ่ (8K x 14 บิต) จึงต้องมีการจัดสรรเป็นเพจ (Page) หรือเป็นหน้า โดยแต่ละเพจจะมีขนาด 2 กิโลเวิร์ด ทั้งนี้เนื่องจากชุดคำสั่งเกี่ยวกับการกระโดดของไมโครคอนโทรลเลอร์ตระกูล PIC สามารถอ้างถึงตำแหน่งของหน่วยความจำได้สูงสุด 2,048 ตำแหน่ง



ภาพที่ 3-2 การจัดสรรหน่วยความจำ PIC16F877

จากภาพที่ 3-2 การจัดสรรหน่วยความจำ PIC16F877 ซึ่งมีขนาด 8 กิโลเวิร์ด ได้มีการสงวนแอดเดรส 0x0000 และ 0x0004 ไว้ และได้ทำการแบ่งหน่วยความจำโปรแกรมออกเป็น 4 เพจ ดังนี้

เพจ 0 มีแอดเดรสในช่วง 0x0000 – 0x07FF (สงวนแอดเดรส 0x0000 – 0x0004)

เพจ 1 มีแอดเดรสในช่วง 0x0800 – 0x0FFF

เพจ 2 มีแอดเดรสในช่วง 0x1000 – 0x17FF

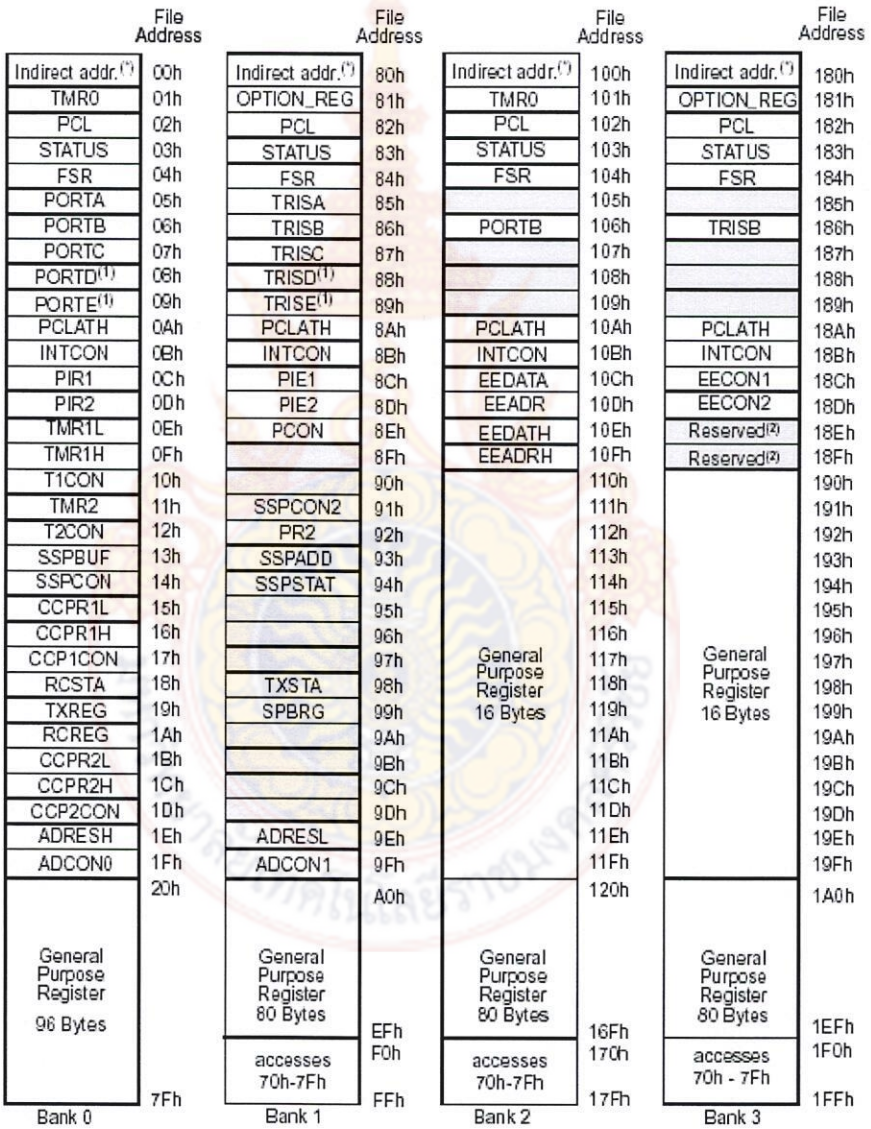
เพจ 3 มีแอดเดรสในช่วง 0x1800 – 0x1FFF

นอกจากนั้นใน PIC16F877 ยังมีพื้นที่หน่วยความจำพิเศษสำหรับเก็บค่าของโปรแกรมเคาน์เตอร์ชั่วคราวขนาด 13 บิต เรียกว่า สแต็ก (Stack) ซึ่งมีบทบาทมากในการกระโดดไปทำงายังโปรแกรมย่อยของ PIC16F877 โดยเมื่อกระทำคำสั่งให้กระโดดไปทำงานยังโปรแกรมย่อย ซีพียูจะทำการเก็บค่าโปรแกรมเคาน์เตอร์หรือ PC ในขณะนั้นไว้ในสแต็ก จากนั้นจึงกระโดดไปทำงานยังโปรแกรมย่อย เมื่อทำงานเรียบร้อยแล้ว ซีพียูจะไปอ่านค่า PC จากสแต็กกลับมา

แล้วทำงานตามกระบวนการในโปรแกรมหลักต่อไป สำหรับสแต็กใน PIC16F877 มีขนาด 13 บิต สามารถเก็บค่าของ PC ได้ 8 ระดับ

การจัดสรรหน่วยความจำข้อมูล

PIC16F877 มีหน่วยความจำข้อมูลแรมสำหรับใช้งานทั่วไป 368 ไบต์ และมีรีจิสเตอร์ไฟล์ 8 บิต 57 ตัว แต่ละแบงก์มีขนาดสูงสุด 128 ไบต์ แต่การใช้งานจริงในแต่ละแบงก์ต่างกัน โดยในแต่ละแบงก์มีการจัดสรรพื้นที่ดังนี้



ภาพที่ 3-3 การจัดสรรหน่วยความจำข้อมูลแรม (RAM) ของ PIC16F877

แบงก์ 0 มีช่วงแอดเดรส 0x00 – 0x7F

แอดเดรส 0x00 – 0x1F เป็นพื้นที่ของรีจิสเตอร์ไฟล์

แอดเดรส 0x20 – 0x7F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 96 ไบต์

แบงก์ 1 มีช่วงแอดเดรส 0x80 – 0xFF

แอดเดรส 0x80 – 0x9F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0xA0 – 0xEF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

แอดเดรส 0xF0 – 0xFF บรรจุข้อมูลเหมือนกับในแอดเดรส 0x70 – 0x7F ในแบงก์ 0 เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70 – 0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงก์

แบงก์ 2 มีช่วงแอดเดรส 0x100 – 0x17F

แอดเดรส 0x100 – 0x10F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0x110 – 0x11F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 ไบต์

แอดเดรส 0x120 – 0x16F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

แอดเดรส 0x170 – 0x17F บรรจุข้อมูลเหมือนกับแอดเดรส 0x70 – 0x7F ในแบงก์ 0 เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70 – 0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงก์

แบงก์ 3 มีช่วงแอดเดรส 0x180 – 0x1FF

แอดเดรส 0x180 – 0x18F เป็นพื้นที่ของรีจิสเตอร์ไฟล์ แต่มีบางแอดเดรสไม่ใช้งาน

แอดเดรส 0x190 – 0x19F เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 16 ไบต์

แอดเดรส 0x1A0 – 0x1EF เป็นพื้นที่ของหน่วยความจำข้อมูลสำหรับใช้งานทั่วไป 80 ไบต์

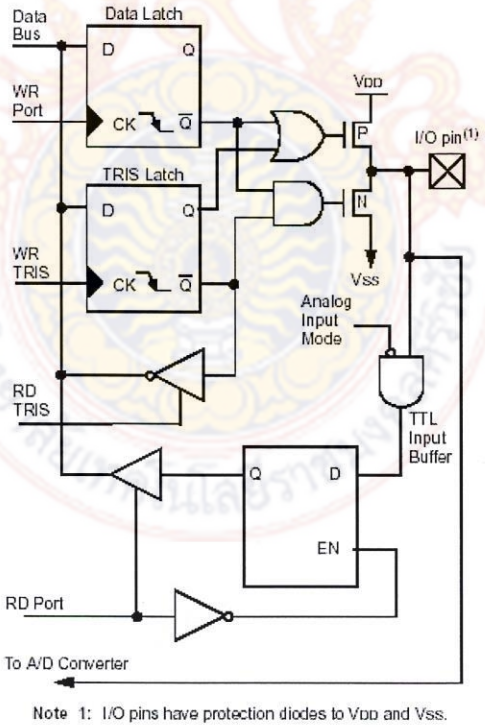
แอดเดรส 0x1F0 – 0x1FF บรรจุข้อมูลเหมือนกับแอดเดรส 0x70 – 0x7F ในแบงก์ 0 เพื่อช่วยให้สามารถใช้ข้อมูลจากแอดเดรส 0x70 – 0x7F ได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแบงก์

3.2 พอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ PIC16F877

การติดต่อสื่อสารกับอุปกรณ์ภายนอกของของไมโครคอนโทรลเลอร์นั้นจะติดต่อผ่านทางขาอินพุตและเอาท์พุต ต่างๆ ของไมโครคอนโทรลเลอร์ ซึ่ง มีหลักๆด้วยกัน 4 พอร์ต ซึ่งในแต่ละพอร์ตมีคุณสมบัติพื้นฐานดังต่อไปนี้

พอร์ต A

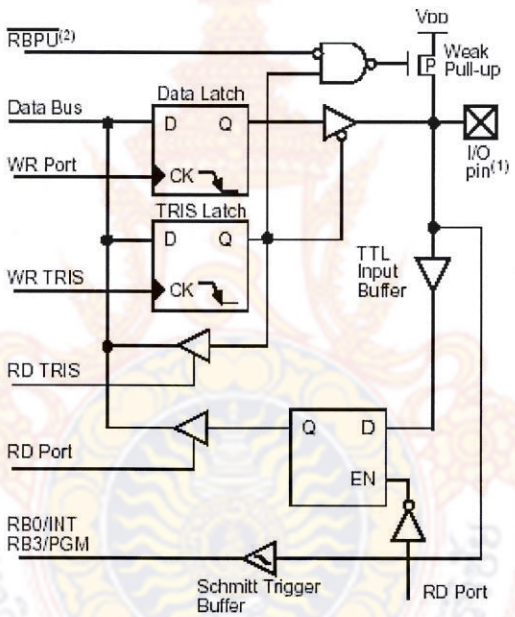
มีทั้งสิ้น 6 ช่อง หรือ 6 บิต กำหนดชื่อขาเป็น RA0 – RA5 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORTA มีแอดเดรสอยู่ที่ 0x05 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต แต่ใช้งานจริงเพียง 6 บิต ที่เหลือ 2 บิต ต้องกำหนดให้เป็น 0 ส่วนการกำหนดทิศทางของพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISA ซึ่งมีแอดเดรสอยู่ที่ 0x85 (แบงก์ 1) มีขนาด 8 บิต และใช้งานจริงเพียง 6 บิตเช่นกัน 2 บิตบน คือบิต 6 และบิต 7 ต้องกำหนดเป็น 0 บิต 0 ของ TRISA ใช้กำหนดทิศทางของพอร์ต RA0 ไล่เลียงลำดับจนถึง บิต 5 ของ TRISA ใช้กำหนดทิศทางของพอร์ต RA5 หากต้องการกำหนดให้ขาพอร์ตในบิตใดเป็นอินพุต ต้องกำหนดเป็น 1 ไปยังบิตนั้น และหากต้องการกำหนดให้ขาพอร์ตใดเป็นเอาต์พุตให้กำหนดบิตนั้นใน TRISA ให้เป็น 0



ภาพที่ 3-4 โครงสร้างขาอินพุตและเอาท์พุตของพอร์ต A

พอร์ต B

พอร์ต B มีจำนวน 8 บิตกำหนดชื่อขาเป็น RB0 RB7 รีจิสเตอร์ที่ใช้ในการเก็บข้อมูลคือ PORTB มีแอดเดรสอยู่ที่ 0x06 (แบงก์ 0) และ 0x106 (แบงก์ 2) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตกระทำผ่านรีจิสเตอร์ TRISB ซึ่งมีแอดเดรสอยู่ที่ 0x86 (แบงก์ 1) และ 0x186 (แบงก์ 3) มีขนาด 8 บิตเช่นเดียวกับพอร์ต A บิต 0 ของ TRIB ใช้กำหนดทิศทางของขา RB0 ไล่ไปจนถึง บิต 7 ของ TRIB ใช้กำหนดทิศทางของขา RB7 หากต้องการให้ขาพอร์ตในบิตใดเป็นอินพุต ต้องเขียนข้อมูล " 1 " ไปยังบิตนั้น ในทางตรงกันข้ามหากต้องการกำหนดให้บิตใดเป็นขาเอาท์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น

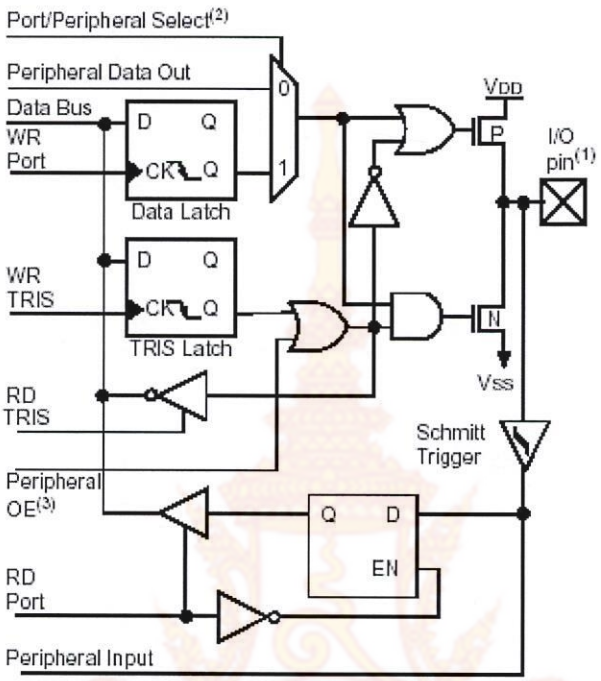


ภาพที่ 3-5 โครงสร้างขาอินพุตและเอาท์พุตของพอร์ต B

พอร์ต C

พอร์ต C มีจำนวน 8 บิต กำหนดชื่อขาเป็น RC0 – RC7 รีจิสเตอร์ที่ใช้เก็บข้อมูลคือ PORTC มีแอดเดรสอยู่ที่ 0x07 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตนี้จะกระทำผ่านรีจิสเตอร์ TRISC มีแอดเดรสอยู่ที่ 0x87 (แบงก์ 1) มีขนาด 8 บิต เช่นเดียวกับพอร์ต A และ B บิต 0 ของ TRIC ใช้กำหนดทิศทางของขา RC0 ไล่ไปจนถึง บิต 7 ของ TRIC ใช้กำหนดทิศทางของขา RC7 หากต้องการให้ขาพอร์ตในบิตใดเป็นอินพุต ต้องเขียน

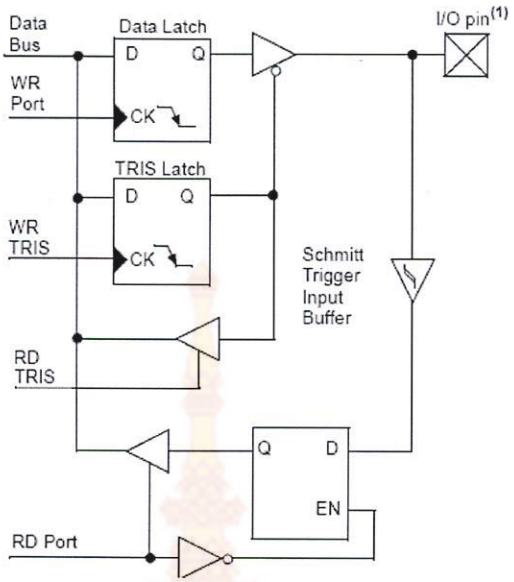
ข้อมูล " 1 " ไปยังบิตนั้น ในทางตรงกันข้ามหากต้องการกำหนดให้บิตใดเป็นขาเอาท์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น



ภาพที่ 3-6 โครงสร้างขาอินพุตและเอาท์พุตของพอร์ต C

พอร์ต D

พอร์ต D มีจำนวน 8 บิต กำหนดชื่อขาเป็น RD0 – RD7 รีจิสเตอร์ที่ใช้เก็บข้อมูลคือ PORTD มีแอดเดรสอยู่ที่ 0x08 (แบงก์ 0) เป็นรีจิสเตอร์ขนาด 8 บิต ส่วนการกำหนดทิศทางของพอร์ตนี้จะกระทำผ่านรีจิสเตอร์ TRISD มีแอดเดรสอยู่ที่ 0x88 (แบงก์ 1) มีขนาด 8 บิต เช่นเดียวกับพอร์ต A และ B บิต 0 ของ TRIC ใช้กำหนดทิศทางของขา RC0 ไล่ไปจนถึง บิต 7 ของ TRIC ใช้กำหนดทิศทางของขา RC7 หากต้องการให้ขาพอร์ตในบิตใดเป็นอินพุต ต้องเขียนข้อมูล " 1 " ไปยังบิตนั้น ในทางตรงกันข้ามหากต้องการกำหนดให้บิตใดเป็นขาเอาท์พุตให้เขียนข้อมูล "0" ไปยังบิตนั้น



ภาพที่ 3-7 โครงสร้างขาอินพุตและเอาท์พุตของพอร์ต D

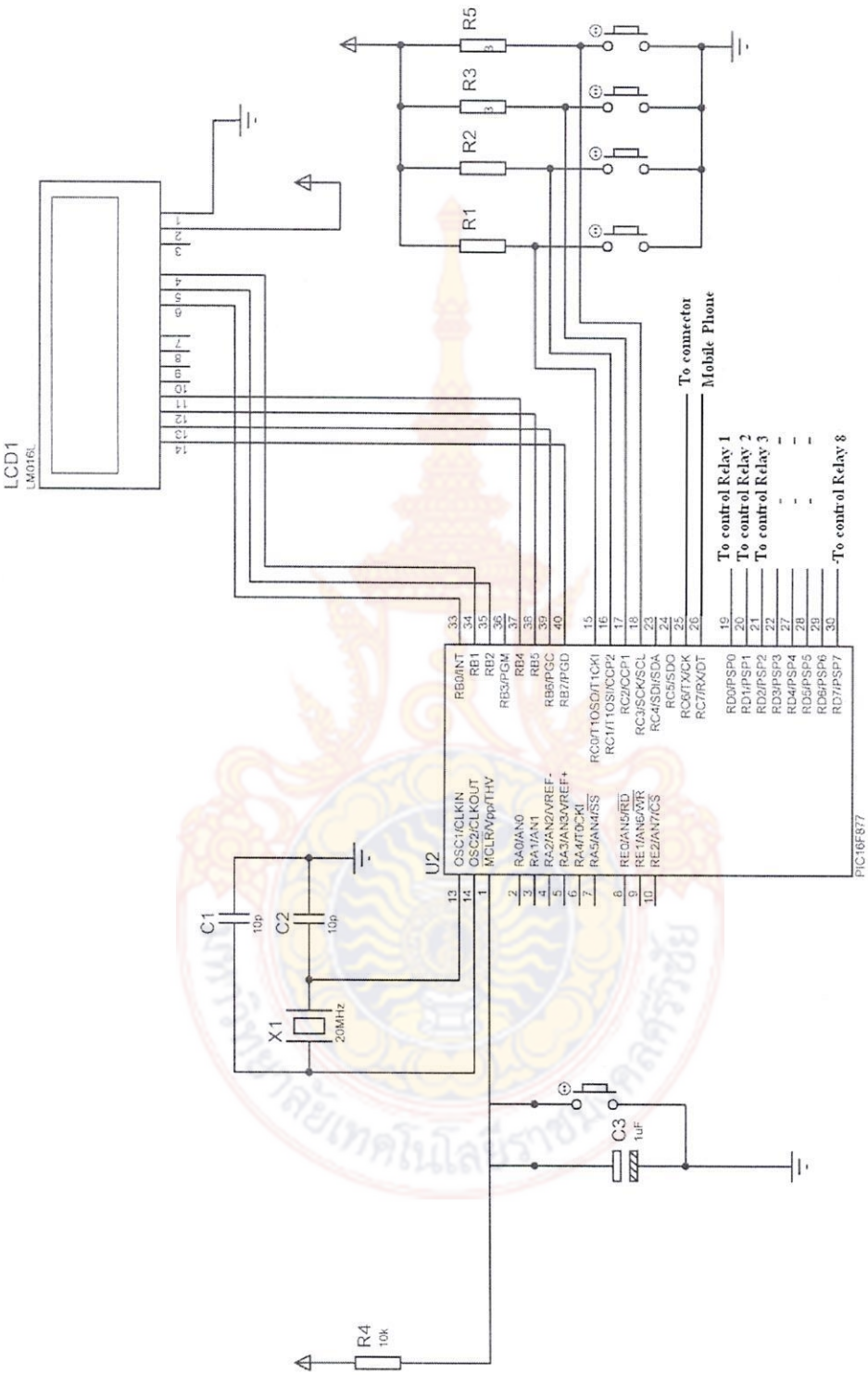
3.3 วงจรและการกำหนดพอร์ตอินพุตเอาท์พุต

การทำงานของเครื่องควบคุมอุปกรณ์ไฟฟ้าด้วยข้อความ SMS นั้น มีการรับค่าจากอินพุตซึ่งเป็นคีย์แพ็ด รับรหัสข้อความ SMS แล้วแปลงเป็นรหัสควบคุม แสดงข้อความการควบคุมด้วยจอแสดงผล LCD และสั่งงานให้รีเลย์แต่ละตัวทำงานนั้น จะทำการติดต่อผ่านทางพอร์ตอินพุตเอาท์พุต ซึ่งแต่ละพอร์ตได้กำหนดให้ทำงานดังต่อไปนี้

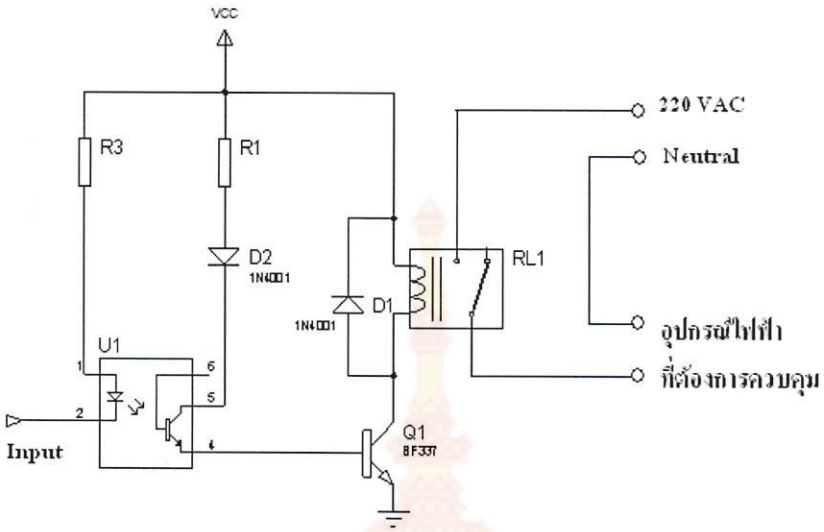
พอร์ต B ขา B0 ถึง B7 ใช้ควบคุมการทำงานของรีเลย์ในการจ่ายไฟให้กับอุปกรณ์ไฟฟ้า และใช้เป็นตัวแสดงผลการทำงานของอุปกรณ์ไฟฟ้าแต่ละตัว

พอร์ต C ขา C0 ถึง C3 ใช้เป็นอินพุตของคีย์แพ็ด (Key Pad) เพื่อใช้กำหนดค่าของตัวเครื่องและใช้ควบคุมให้เครื่องใช้ไฟฟ้าที่ต่ออยู่ของแต่ละช่องทำงานหรือหยุดทำงาน ส่วนขา C6 และ C7 ใช้เป็นพอร์ตอนุกรมสำหรับติดต่อกับคอนเน็กเตอร์ของโทรศัพท์เคลื่อนที่

พอร์ต D ขา D0 ถึง D7 ใช้ในการควบคุมจอแสดงผล LCD เพื่อสื่อให้ผู้ใช้ทราบว่าอุปกรณ์ไฟฟ้าตัวใดทำงานอยู่บ้างซึ่งนอกจากควบคุมการทำงานของอุปกรณ์ไฟฟ้าด้วยข้อความจากโทรศัพท์เคลื่อนที่แล้วยังสามารถกำหนดให้ทำงานและหยุดทำงานโดยการกดสวิชต์ที่ตัวเครื่องได้อีกด้วยดังแสดงในภาพที่ 3-8 โดยสัญญาณเอาท์พุตแต่ละขาจากพอร์ต D จะไปควบคุมทรานซิสเตอร์เพื่อขับรีเลย์ให้ทำงานเพื่อควบคุมอุปกรณ์ไฟฟ้าให้ทำงานหรือหยุดทำงานดังแสดงดังในภาพที่ 3-9

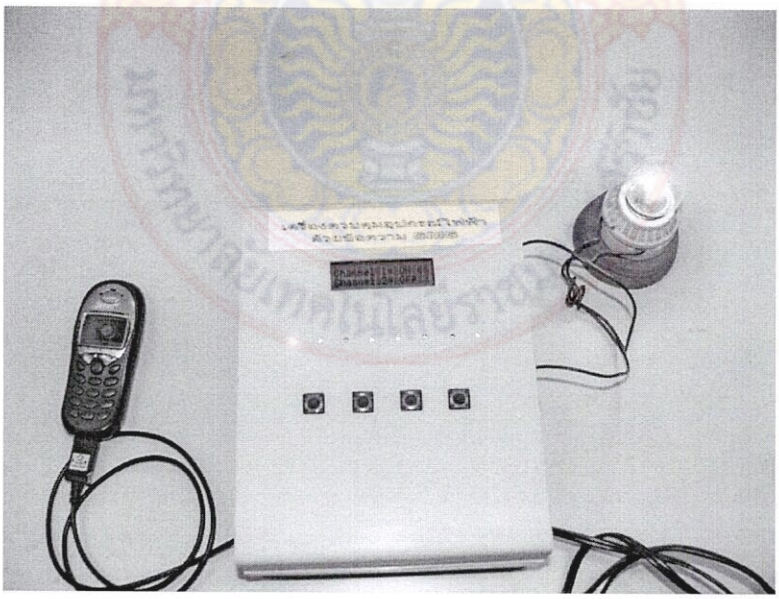


ภาพที่ 3-8 แสดงวงจรการทำงานของไมโครคอนโทรลเลอร์ร่วมกับ
อุปกรณ์อินพุตเอาท์พุต



ภาพที่ 3-9 แสดงวงจรควบคุมอุปกรณ์ไฟฟ้าของแต่ละช่อง

หลังจากออกแบบวงจรแล้ว นำอุปกรณ์ต่าง ๆ มาประกอบเป็นวงจรแล้วบรรจุรวมลงกล่อง และทำการทดสอบการทำงาน



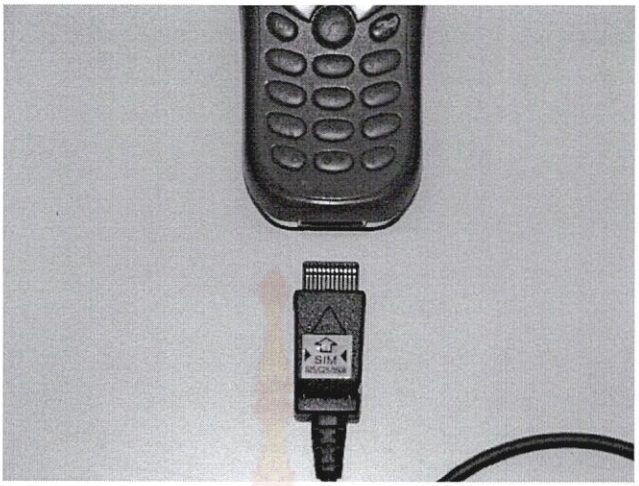
ภาพที่ 3-10 แสดงเครื่องควบคุมอุปกรณ์ไฟฟ้าด้วยข้อความ SMS

3.4 พอร์ตควบคุมการทำงานของโทรศัพท์เคลื่อนที่

การอ่านข้อความ SMS จากโทรศัพท์เคลื่อนที่นั้น จะให้ไมโครคอนโทรลเลอร์ส่งคำสั่ง "AT+CMGR = ตำแหน่งข้อความ" ทางพอร์ตอนุกรม (Serial Port) ของไมโครคอนโทรลเลอร์กับคอนเน็กเตอร์ของโทรศัพท์เคลื่อนที่ ซึ่งอยู่ที่ท้ายของโทรศัพท์ โดยในงานวิจัยนี้จะใช้โทรศัพท์เคลื่อนที่ยี่ห้อ SIEMENS รุ่น C45 ซึ่งสามารถนำใช้ได้กับรุ่น C35, M35 และ S35 ได้อีกด้วย ซึ่งคอนเน็กเตอร์ของโทรศัพท์เคลื่อนที่รุ่นนี้จะมีขาจำนวน 12 ขา โดยแต่ละขาจะมีหน้าที่และการใช้งานแตกต่างกันไปดังแสดงในตารางที่ 3-1

ตารางที่ 3-1 แสดงตำแหน่งขาและหน้าที่การทำงานของคอนเน็กเตอร์ของโทรศัพท์เคลื่อนที่รุ่น C35, M35, S35 และ C45

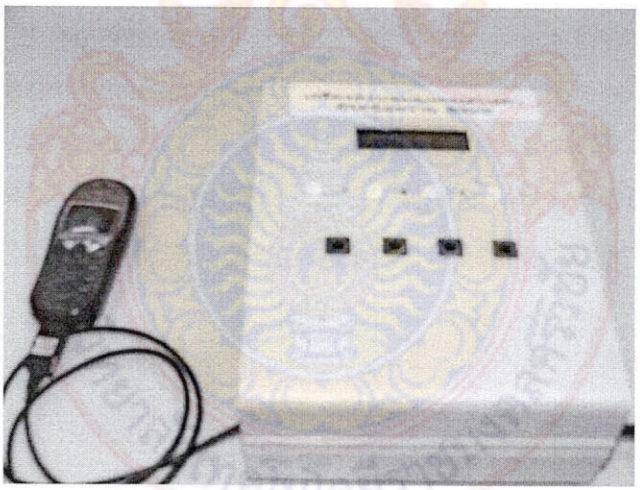
Pin	Name	Function	In/Out
1	GND	Ground	
2	SELF-SERVICE	Recognition/Control Battery charger	In/Out
3	LOAD	Charging Voltage	In
4	BATTERY	Battery	Out
5	DATA OUT	Data sent	Out
6	DATA IN	Data received	In
7	Z_CLK	Recognition/Control accessories	
8	Z_DATA	Recognition/Control accessories	
9	MICG	Ground for microphone	In
10	MIC	Microphone input	
11	AUD	Loudspeaker	Out
12	AUDG	Ground for external speaker	



ภาพประกอบที่ 3-11 แสดงคอนเน็กเตอร์ตัวเมียของโทรศัพท์เคลื่อนที่กับคอนเน็กเตอร์ตัวผู้ที่ใช้ต่อกับไมโครคอนโทรลเลอร์

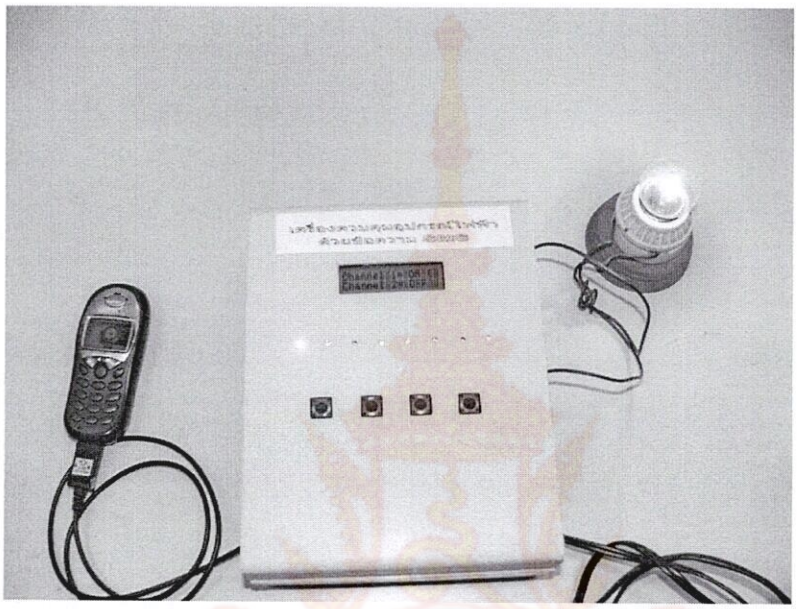
บทที่ 4
การทดลองและการทดสอบ

หลังจากได้ออกแบบวงจรอิเล็กทรอนิกส์แล้ว ได้ทำการเขียนโปรแกรมและแก้ไขปรับปรุงการทำงานของไมโครคอนโทรลเลอร์ให้สามารถติดต่อและสั่งงานโทรศัพท์เคลื่อนที่ด้วย AT Command เพื่ออ่านข้อความ SMS จากโทรศัพท์เคลื่อนที่แล้วมาทำการควบคุมการทำงานของอุปกรณ์ไฟฟ้าที่ต่ออยู่แต่ละช่อง ซึ่งมีจำนวน 8 ช่อง นั่นหมายความว่าสามารถนำไปควบคุมการทำงานของอุปกรณ์ไฟฟ้าให้ทำงานหรือหยุดทำงานได้ 8 เครื่องอิสระต่อกัน โดยในการทดสอบต้องใช้โทรศัพท์เคลื่อนที่อีกเครื่องเป็นตัวส่งข้อความ SMS โดยกำหนดให้ใช้ตัวเลขเป็นตัวกำหนดการทำงานของอุปกรณ์ไฟฟ้าแต่ละตัว เลข 1 หมายถึงให้อุปกรณ์ไฟฟ้าทำงาน เลข 0 หมายถึง อุปกรณ์ไฟฟ้าหยุดทำงาน ซึ่งตำแหน่งตัวเลขจากซ้ายไปขวาคือตำแหน่งของอุปกรณ์ไฟฟ้าแต่ละช่อง เช่นส่งข้อความ “10101010” หมายถึง ให้เครื่องใช้ไฟฟ้าช่องที่ 1 ทำงาน ช่องที่2 หยุดทำงาน ช่องที่ 3 ทำงาน ช่องที่ 4 หยุดทำงาน สลับกันไปจนถึงช่องที่ 8 ซึ่งจากการทดสอบเครื่องใช้ไฟฟ้าแต่ละช่องทำงานและหยุดทำงานตามรหัสข้อความที่ส่งมา ดังแสดงในภาพที่ 4-1



ภาพที่ 4-1 แสดงการทำงานของเครื่องควบคุมอุปกรณ์ไฟฟ้าด้วยข้อความ SMS หลังจากได้ส่งข้อความ "10101010"

ทำการทดสอบส่งรหัสควบคุมอีกครั้งด้วยรหัส 10000000 ซึ่งเป็นการกำหนดให้อุปกรณ์ไฟฟ้าช่องที่ 1 ทำงานส่วนช่องอื่นหยุดทำงานทั้งหมด ซึ่งหลังจากส่งข้อความจากเครื่องส่งข้อความจะมาถึงเครื่องรับในเวลาต่อมา และไมโครคอนโทรลเลอร์สามารถควบคุมให้อุปกรณ์ไฟฟ้าทำงานได้ดังรหัสที่ส่งมาดังแสดงในภาพที่ 4-2



ภาพที่ 4-2 แสดงการทำงานของเครื่องควบคุมอุปกรณ์ไฟฟ้าด้วยข้อความ SMS หลังจากได้ส่งข้อความ "10000000"

ทำการทดสอบส่งรหัสควบคุมอีกครั้งด้วยรหัส 01000000 ซึ่งเป็นการกำหนดให้อุปกรณ์ไฟฟ้าช่องที่ 1 ทำงานส่วนช่องอื่นหยุดทำงานทั้งหมด ซึ่งหลังจากส่งข้อความจากเครื่องส่งข้อความจะมาถึงเครื่องรับในเวลาต่อมา และไมโครคอนโทรลเลอร์สามารถควบคุมให้อุปกรณ์ไฟฟ้าทำงานได้ดังรหัสที่ส่งมาจากโทรศัพท์เคลื่อนที่อีกเครื่องหนึ่ง และได้ทำการทดสอบต่อไปจนครบทุกช่อง ปรากฏว่าเครื่องควบคุมอุปกรณ์ไฟฟ้าด้วยข้อความ SMS สามารถควบคุมอุปกรณ์ไฟฟ้าได้ตามรหัสคำสั่งจากผู้ส่งทุกครั้ง

บทที่ 5
สรุปผล

จากการทำงานวิจัยที่ผ่านมาสามารถสร้างต้นแบบเครื่องควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่ได้เป็นผลสำเร็จ ซึ่งสามารถควบคุมอุปกรณ์ไฟฟ้าได้ตามต้องการด้วยวิธีการส่งข้อความ SMS ที่เป็นรหัสควบคุมจากเครื่องโทรศัพท์เคลื่อนที่ของผู้ต้องการควบคุมไปยังหมายเลขโทรศัพท์เคลื่อนที่ของเครื่องตัวรับซึ่งต่ออินเตอร์เฟสอยู่กับเครื่องควบคุมอุปกรณ์ไฟฟ้าและเครื่องควบคุมก็จะนำข้อความที่ได้รับมาทำการถอดรหัสควบคุมและนำไปควบคุมอุปกรณ์ไฟฟ้าแต่ละตัวได้ ซึ่งในงานวิจัยนี้ได้ใช้โทรศัพท์เคลื่อนที่ยี่ห้อ SIEMENS รุ่น C45 ซึ่งเป็นรุ่นที่มีราคาถูกและมีใช้อยู่เป็นส่วนใหญ่ หากจะซื้อเครื่องมือสองก็สามารถหาได้ง่าย

นอกจากจะสามารถควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่ได้แล้ว ยังสามารควบคุมให้เครื่องใช้ไฟฟ้าแต่ละตัวทำงานหรือหยุดทำงานโดยการเซ็ตคำสั่งที่ตัวเครื่องควบคุมได้อีกด้วย ซึ่งทำให้ผู้ที่อยู่ใกล้เครื่องควบคุมนี้สามารถควบคุมเครื่องใช้ไฟฟ้าแต่ละตัวโดยตรงได้

บรรณานุกรม

ประจิน พลังสันติกุล. เรียนรู้และใช้งาน CCS C คอมไพเลอร์เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์ PIC . กรุงเทพฯ : อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด,2521

ณัฏฐพล วงศ์สุนทรชัย, ชัยวัฒน์ ลิ้มพรจิตรวิไล. ปฏิบัติการไมโครคอนโทรลเลอร์ PIC16F87X. กรุงเทพฯ : อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด,2521

ร.ท. เดชา ล้อเจริญกิจ, รน.ธนันท์ รังสีพรหม, พรเทพ เลิศบัวรักษ์. "โมบายล์อินเตอร์เฟสสำหรับมือถือซีเมนส์ C35/M35/S35" กรุงเทพฯ : ซีเอ็ดยูเคชั่น(จำกัดมหาชน) ฉบับที่ 237,2545

ไพโรจน์ ไววานิชกิจ. "Mobile Data Application ธุรกิจแห่งอนาคต ตอนสีสันแห่งเทคโนโลยีจาก SMS สู่ MMS ตอนที่ 1" กรุงเทพฯ : ซีเอ็ดยูเคชั่น(จำกัดมหาชน) ฉบับที่ 241,2545

Serasidis Vasilis. SMSRemoteControl,
http://www.serasidis.gr/circuits/smscontroller/smscontroller.htm

ภาคผนวก

โปรแกรมภาษาซีของเครื่องควบคุมอุปกรณ์ไฟฟ้าระยะไกลด้วยข้อความ SMS จากโทรศัพท์เคลื่อนที่

```
#include <16F877.h>          // Standard Header file for the PIC16F628 device
#define  TxD       PIN_C6   // Define Transmitted Data
#define  RxD       PIN_C7   // Define Received Data
#define  CLOCK_SP   20000000   // Clock Speed(Hz)
#fuses   HS                    // Oscillator mode HS
#fuses   NOLVP, NOWDT        // No Low Voltage Program, No Watchdog timer
#fuses   NOPROTECT           // Code no protection
#use delay (clock=CLOCK_SP)   // Use built-in function: delay_ms() & delay_us()
#use rs232(baud=19200, xmit=PIN_C6,rcv=PIN_C7,ERRORS) // Use serial I/O port (RS232)
connect computer
//#use rs232(baud=19200, xmit=PIN_C4,rcv=PIN_C5,stream = com2) // Use serial I/O port
(RS232) connect controller
#define LED1    PIN_B0
//#use fast_io(B)            // programming of the direction register.
#define use_portb_lcd
#include "lcd.C"
//#include "input.C"
/*****************************************************************************
 *    Constant
*****************************************************************************/
#define STRING_SIZE  17
#define msg0  " Wellcome To "
#define msg1  "SMS Controller"
#define SetNum  " Set Phone NUM"
#define YesNo  "   Yes /  NO"
#define OnOff  " Set ON OFF  "
#define On   "ON "
#define Off  "OFF "
#define Exit  "Exit Set ON/Off "
#define msg5  " Enter  "
```

```
#define msg6  "Hello World AST."
#define chx   "Channel x=    "
char buffer[40];
unsigned char  *ptr_buffer,time,index,set,recieve,point_first;
//int i,b;
unsigned char menu,choose,OutPort,ch,key;
/*****************************************************************************
 *    Function prototype
 *****************************************************************************/
 void LCD_Command(int cm);
 void LCD_ShiftLeft(void);
 void LCD_ShiftRight(void);
 void LCD_MoveRight(char p);
 void LCD_MoveLeft(char p);
 void strcpy(char *s1,char *s2);
 void LCD_String(char *s, int dly);
 void LCD_Show(void);
 void LedControl(unsigned int led);
/*********************************************************************
 *          Interrupt Function
 *********************************************************************/

#int_rda
 Void rs232_isr()
 {
  digit = getc();
  rx_int = false;
 }
```

```
/*************************************************************************
 *            LCD Function
 *************************************************************************/
/*************************************************************************
*   Function LCD_Command
*   Description  :  LCD Command
*   Parameters   :  nothing
*   Returned     :  nothing
*************************************************************************/
void LCD_Command(int cm)
{
  lcd_send_byte(0,cm);
}
/*************************************************************************
*   Function strcpy
*   Description  :  string copy
*   Parameters   :  *s1, *s2
*   Returned     :  nothing
*************************************************************************/
void LCD_strcpy(char *s1,char *s2)
{
 while(*s1++=*s2++)  ;
 }
 *************************************************************************
*   Function LCD_string
*   Description  :  LCD string*s , dly
*   Returned     :  nothing
*************************************************************************/
```

```
void LCD_String(char *s, int dly)
 {
  while(*s !=0)
  {
    lcd_putc(*s++);
    delay_ms(dly);
  }
 }
/*****************************************************************************
*    Function LCD_Show
*    Description  :  LCD Show
*    Parameters   :  nothing
*    Returned     :  nothing
*****************************************************************************/
 void LCD_Show(void)
 {
  int i,j;
  char str[17];
  LCD_Command(0x01);  // clear LCD
  LCD_Command(0x80);
  strcpy(str,msg0);
  LCD_String(str,0);
  LCD_Command(0xC0);
  strcpy(str,msg1);
  LCD_String(str,0);
//  delay_ms(100);
 }
```

```
/*****************************************************************************
 * Function : Convert ASCII to HEX
 * Parameter: num
 * Returned Heximal
 *****************************************************************************/
unsigned char Ascii2Hex(unsigned char num)
 {
  if((num>=0x30)&&(num<=0x39))
    return(num-0x30);
  else
   if((num>=0x41)&&(num<=0x45))
     return(num-0x37);
 }
unsigned char  decode_massage(unsigned char num,unsigned char *ptr_massage)
  {
  unsigned char data,buff,i,data_buff,buffer[8],out_portb,display_buff[30];
  char str[17];
  for(i=0;i<(num-0x30);i++)
      {
    buffer[i] = (Ascii2Hex(*(ptr_massage+i*2))<<4)|(Ascii2Hex(*(ptr_massage+i*2+1)));
   }
  buff = 0x00;
  out_portb=00;
  for(i=0;i<num-0x30;i++)
   {
       data_buff = buffer[i];
       data = ((data_buff<<(i%8))|buff) & 0x7F;
       buff = data_buff>>(7-(i%8));
       display_buff[i] = data;    // store data ASCII to buffer
```

```
        out_portb = (((data & 0x01)<<(7-i)) | out_portb);  // out port for control
    }
        output_d(out_portb);
        return(out_portb);
  }
 unsigned char scankey(void)
{
  unsigned char key;                          // check keypress if Not keypress will out
Function
  // P0  = 0xff;
   key = (input_C() & 0xff)|0xf0;             // scand input
   key = (~key)&0xff;
   return(key);
}
void SetPhoneNumber(void)
{
   char str[17];
   LCD_Command(0x01);  // clear LCD
   //line1
   LCD_Command(0x80);
   strcpy(str,SetNum);
   LCD_String(str,0);
   //line 2
   LCD_Command(0xC0);
   strcpy(str,YesNo);
   LCD_String(str,0);
}
void SetOnOff(void)
{
   char str[17];
```

```
  LCD_Command(0x01);   // clear LCD
    //line1
  LCD_Command(0x80);
  strcpy(str,OnOff);
  LCD_String(str,0);
    //line2
  LCD_Command(0xC0);
  strcpy(str,YesNo);
  LCD_String(str,0);
}
void ExitOnOff(void)
{
  char str[17];
  LCD_Command(0x01);   // clear LCD
    //line1
  LCD_Command(0x80);
  strcpy(str,Exit);
  LCD_String(str,0);
    //line2
  LCD_Command(0xC0);
  strcpy(str,YesNo);
  LCD_String(str,0);
}
void EnterKey(void)
{
 ch=Key;
}
```

```
int1 DetectOnOff(char point)
{
  int1 a;
  a = (OutPort>>(point-1)&0x01);
  return(a);
}
void ShowOnOff1(char input1)
{
char str[17];
char lin1,lin2;
   lin1=input1;
   if(lin1>7) lin1=7;
   lin2=lin1+1;
   LCD_Command(0x01);   // clear LCD
   LCD_Command(0x80);
   strcpy(str,Chx);
   if(DetectOnOff(lin1)==0) strcpy(str[11],Off);
   else strcpy(str[11],On);
   str[8] = 0x30+lin1;
   str[14] = 0x3c;
   LCD_String(str,0);
    //line2
   LCD_Command(0xC0);
   strcpy(str,Chx);
   if(DetectOnOff(lin2)==0) strcpy(str[11],Off);
   else strcpy(str[11],On);
   str[8] = 0x30+lin2;
   //str[15] = 0x10;
   LCD_String(str,0);
```

```
}
void ShowOnOff2(char input2)
{
char str[17];
char line1,line2;
   line1=input2-1;
   line2=input2;
   LCD_Command(0x01);   // clear LCD
   LCD_Command(0x80);
   strcpy(str,Chx);
   if(DetectOnOff(line1)==0) strcpy(str[11],Off);
   else strcpy(str[11],On);
   str[8] = 0x30+line1;
 //  str[15] = 0x10;
   LCD_String(str,0);
    //line2
   LCD_Command(0xC0);
   strcpy(str,Chx);
   if(DetectOnOff(line2)==0) strcpy(str[11],Off);
   else strcpy(str[11],On);
   str[8] = 0x30+line2;
   str[14] = 0x3c;
   LCD_String(str,0);
}
void SetOnOffFunction(void)
{
char str[17];
int1 bit;
char line1,line2,index,key,line,select,press,type1,OutFunction;
```

```
 index =1;
 line  =1;
 line1= index;
 line2= index+1;
 LCD_Command(0x01);   // clear LCD
   //line1
 LCD_Command(0x80);
 strcpy(str,Chx);
 if(DetectOnOff(line1)==0) strcpy(str[11],Off);
 else strcpy(str[11],On);
 str[8] = 0x31;
 if(line==1) str[14] = 0x3c;      // cursor
 LCD_String(str,0);
   //line2
 LCD_Command(0xC0);
 strcpy(str,Chx);
 if(DetectOnOff(line2)==0) strcpy(str[11],Off);
  else strcpy(str[11],On);
 str[8] = 0x32;
 LCD_String(str,0);
 type1=0;
// select = scankey();
output_D(OutPort);
OutFunction =0;
while(OutFunction != 1)
{
 press = scankey();
 select = type1 + press;
 switch(select)
  {
```

```
case 1:                          // up key
      index = index-1;
      if(index<1) index=1;
      showOnOff1(index);
      delay_ms(200);
      line = 1;
      break;
case 2:                          //down key
      index = index+1;
      if(index>8) index=8;
      showOnOff2(index);
      delay_ms(200);
      line =2;
      break;
case 4: ExitOnOff();
       type1=0x40;
        delay_ms(200);
      break;
case 8:  //EnterKey();
      if(DetectOnOff(index)==0) bit_set(OutPort,index-1);
      else bit_clear(OutPort,index-1);
      output_D(OutPort);
      if(line==1)ShowOnOff1(index);
      else ShowOnOff2(index);
      delay_ms(200);
      break;
case 0x44: showOnOff1(index);
      type1=0;
       delay_ms(200);
       break;
```

```
    case 0x48:
            OutFunction =1;
            delay_ms(200);
           break;
  defaual: delay_ms(300);
}
}
}
/*
#int_rda
 Void rs232_isr()
 {
  unsigned char f,b;

   if(recieve == 1)
     {
      f  = getchar();
     // if(time>0) fputc(f);

      if((f=='0')&&(set==0))
        {
         point_first = index;   // find first '0'
         set = 1;
         }
        if(index<80)
          write_bank(1,index,f);
        else if(index<160)
         {
          b= index%80;
          write_bank(2,b,f);
```

```
        }
        else if(index<240)
        {
         b= index%80;
         write_bank(3,b,f);
         }
       index++;
     }
}
*/
void read_data_mobile(void)
 {
 unsigned char a,j,data,num_char,fonts,status;
 unsigned char b,num,d,f,set,point_first,count_enter;
 num = 0;
 index = 0;
 point_first =0;
 set =0;
 d= 0;
 recieve = 1;
puts("at+cmgr=10");
count_enter=0;
  do{
      f  = getchar();
    // if(time>0) fputc(f);
        if(index<80)
        write_bank(1,index,f);
       else if(index<160)
        {
        b= index%80;
```

```
            write_bank(2,b,f);
            }
           else if(index<240)
            {
            b= index%80;
            write_bank(3,b,f);
            }
        index++;
      }
while(f != 'K');
     delay_ms(1000);
     for(a=0;a<index;a++)
      {
        if(a<80)
        {
         f = read_bank(1,a);
         putc(f);
        }
        else if(a<160)
         {
         b = a%80;
          f = read_bank(2,b);
          putc(f);
         }
         else if(a<240)
         {
          b = a%160;
          f = read_bank(3,b);
          putc(f);
         }
```

```
        if(f ==0x0a) count_enter++;
        if((count_enter == 3)&&(set==0))
          {
           point_first = a+1;   // find first '0'
           set = 1;
          }
         }
       putc(13);
for(a=point_first;a<index;a++)
     {
       if(a<80)
       putc(read_bank(1,a));
       else if(a<160)
        {
        b = a%80;
         putc(read_bank(2,b));
        }
        else if(a<240)
        {
        b = a%160;
          putc(read_bank(3,b));
        }
       }
       putc(13);
    puts("cata sms is =>");
     num =0;
   for(a=point_first+50;a<index;a++)
    {
      if(a<80)
```

```
        {
         buffer[num] = read_bank(1,a);
        }
       else if(a<160)
        {
         b = a%80;
         buffer[num] = read_bank(2,b);
        }
        else if(a<240)
        {
        b = a%160;
        buffer[num] = read_bank(3,b);
        }
          putc(buffer[num]);
          num++;
      }
     putc(13);  // enter
/*       // copy data to buffer
     num =0;
   for(a=point_first+50;a<index;a++)
    {
      if(a<80)
       buffer[num] = read_bank(1,a);
      else if(a<160)
       {
       b = a%80;
       buffer[num] = read_bank(2,b);
       }
       else if(a<240)
       {
```

```
        b = a%160;
          buffer[num] = read_bank(3,b);
        }
         num++;
      } // end for
          */
      recieve = 0;
 }
/*************************************************************************
 * Function   LED Control
 * Description :  On/off  LED port A
*************************************************************************/
void main(void)
{
 lcd_init();   // keyboard init
 time = 0;
 //enable_interrupts(GLOBAL);
 // enable_interrupts(INT_RDA);
 set_tris_b(0x00);         // set trisb output
 set_tris_C(0xfF);
 LCD_Show();
 //TestPortD(2);
 choose = 0;
 ch = 0;
 OutPort = 0x00;   // set output device on off
 while(true)
 {
 menu = scankey();
 choose = ch+menu;
 output_D(OutPort);   //  output to port D
```

```
if(choose != 0) {
 switch(choose)
  {
    case 1: SetPhoneNumber();
          key = 0x10;
           break;
    case 2: SetOnOff();
          key = 0x20;
          break;
    case 4: //SetOnOffFunction();
          LCD_Show();
          key = 0;
          break;
    case 8: EnterKey();
          delay_ms(200);
          break;
   case 0x10: read_data_mobile();
           OutPort = decode_massage(0x38,buffer);
           LCD_Show();
            ch=0;
            break;
   case 0x20: SetOnOffFunction();
            LCD_Show();
            ch=0;
            break;
  defaual: delay_ms(100);
  } //switch
}else delay_ms(100);
 }
}
```