ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS

(FM Broadcast Status Reporting System via Mobile SMS)

โดย

นางสาวสุชญา	ชะมังพล	รหัสนักศึกษา	B5103713
นางสาวมัลลิกา	แสงจันทร์	รหัสนักศึกษา	B5120291
นางสาวอัจฉริยา	คงบรรทัด	รหัสนักศึกษา	B5128044

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 429499 โครงงานวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2554

ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS

(FM Broadcast Status Reporting System via Mobile SMS)

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ ทองทา)
กรรมการ/อาจารย์ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร.พีระพงษ์ อุฑารสกุล)
กรรมการ

(อาจารย์ ดร. สมศักดิ์ วาณิชอนันต์ชัย)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม รายวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2554

โครงงาน	ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS (FM Broadcast Status Reporting System via Mobile SMS)
จัดทำโดย	นางสาวสุชญา ชะมังพล นางสาวมัลลิกา แสงจันทร์ นางสาวอัจฉรียา คงบรรทัด
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ ทองทา
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่	3/2554

บทคัดย่อ

เนื่องจากสถานีวิทยุกระจายเสียงต้องมีการกระจายเสียงอย่างต่อเนื่องตลอดเวลา จึงอาจมีความผิดปกติอันเกิดจากเครื่องส่งวิทยุกระจายเสียงมีค่าพารามิเตอร์ต่าง ๆ ที่เกินขีดจำกัด ทำให้ต้องมีผู้ดูแลเครื่องส่งวิทยุกระจายเสียงคอยดูแลเครื่องส่งวิทยุกระจายเสียงที่บริเวณหน้าเครื่องด้วย คณะผู้จัดทำโครงงานจึงนำปัญหาดังกล่าวมาคิดและพัฒนาเป็นระบบรายงานสถานะเครื่องส่งวิทยุผ่านระบบ SMS นี้ขึ้น เพื่อให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถทราบถึงสถานะการทำงานและค่าพารามิเตอร์ต่าง ๆ ของเครื่องส่งวิทยุกระจายเสียงได้ โดยไม่จำเป็นต้องเดินทางไปที่สถานีวิทยุกระจายเสียงด้วยตนเอง ระบบรายงานสถานะเครื่องส่งวิทยุผ่านระบบ SMS ได้นำการหลักทำงานของสองอุปกรณ์หลัก คือ บอร์ด ไมโครคอนโทรลเลอร์ และ GSM Module มาประกอบกันทำให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถส่งข้อความ SMS ผ่านทางโทรศัพท์เคลื่อนที่เพื่อสอบถามข้อมูลหรือสั่งการให้เครื่องส่งวิทยุกระจายเสียงทำงานได้ด้วยระบบ GSM ซึ่งเป็นระบบสื่อสารที่โทรศัพท์เคลื่อนที่ทั่วไปสามารถรองรับได้ดี ระบบรายงานสถานะเครื่องส่งวิทยุผ่านระบบ SMS สามารถส่งข้อความ SMS รายงานข้อมูลต่าง ๆ ของเครื่องส่งวิทยุกระจายเสียงได้แก่ Date , Time , Status , Ratio , Forward , Reflect , Temp. และ Temp. Room ซึ่งผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะได้รับข้อความ SMS แสดงข้อมูลเหล่านี้ได้ใน 3 กรณี ได้แก่ 1.ขณะต้องการทราบข้อมูล 2.ตามรอบเวลาที่ตั้งไว้ 3.ขณะที่เครื่องส่งวิทยุเกิดความผิดปกติ

กิตติกรรมประกาศ

จากการที่คณะผู้จัดทำโครงงานได้รับมอบหมายให้ทำโครงงานเรื่อง ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านระบบ SMS (FM Broadcast Status Reporting System via Mobile SMS) ส่งผลให้คณะผู้จัดทำโครงงานได้รับความรู้และประสบการณ์ต่าง ๆ เกี่ยวกับการเขียนโปรแกรมด้วยโปรแกรม Keil uVision3 การใช้งานบอร์ดไมโครคอนโทรเลอร์ CP-JR ARM7 USB-LPC2148 และการใช้งาน GSM Module เป็นอย่างมาก บัดนี้โครงงานดังกล่าวพร้อมทั้งรายงานได้สำเร็จลงแล้ว ทั้งนี้ด้วยความร่วมมือและสนับสนุนจากบุคคลต่างๆ ดังนี้

1. ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ ทองทา (อาจารย์ที่ปรึกษาโครงงาน)
2. นายปัญญา หันตุลา (นักศึกษาบัณฑิตศึกษาสาขาวิชาวิศวกรรมโทรคมนาคม)

คณะผู้จัดทำโครงงานใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูลและเป็นที่ปรึกษาในการทำโครงงานและรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับพื้นฐานการใช้งานโปรแกรม ซึ่งขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

นางสาวสุชญา ชะมังพล
นางสาวมัลลิกา แสงจันทร์
นางสาวอัจฉรียา คงบรรทัด

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมา

ปัจจุบันมีการใช้งานเครื่องส่งวิทยุกระจายเสียงกันอย่างแพร่หลาย ซึ่งปกติแล้วผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องประจำอยู่บริเวณหน้าเครื่องตลอดเวลา เราจึงมีแนวคิดในการสร้างระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS เพื่อทำให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถทราบสถานะหรือควบคุมการทำงานของเครื่องส่งวิทยุกระจายเสียงได้อย่างสะดวกผ่านทางโทรศัพท์เคลื่อนที่ แม้ว่าผู้ดูแลเครื่องส่งวิทยุกระจายเสียงไม่ได้ประจำอยู่ที่หน้าเครื่อง

ระบบทำงานโดยใช้ 1) บอร์ดไมโครคอนโทรลเลอร์ รุ่น CP-JR ARM7 USB-LPC2148 ควบคุมการทำงานโดย In-Circuit Download โปรแกรมเข้าหน่วยความจำภายในผ่านทาง Port RS232 2) GSM Module สามารถทำงานได้เช่นเดียวกับโทรศัพท์เคลื่อนที่ คือ การโทรออก การรับสายเรียกเข้า และการส่งข้อความ SMS โดยการใช้ Sim Card จากเครือข่ายผู้ให้บริการที่สามารถรับสัญญาณโทรศัพท์ในท้องถิ่นนั้น ๆ ซึ่งเมื่ออุปกรณ์ทั้งสองทำงานร่วมกัน มีผลทำให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถส่งข้อความ SMS ผ่านทางโทรศัพท์เคลื่อนที่เพื่อสื่อสารกับระบบ ส่วนระบบสามารถส่งข้อความ SMS ตอบกลับผ่านทาง GSM Module เพื่อแสดงข้อมูลต่าง ๆ ได้ตามที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการ ข้อมูลที่แสดงได้แก่ Date , Time , Status , Ratio , Forward , Reflect , Temp. และ Temp. Room ระบบทำงานเสมือนกับการสื่อสารระหว่างโทรศัพท์เคลื่อนที่สองเครื่อง โดยโทรศัพท์เครื่องแรกควบคุมโดยผู้ดูแลเครื่องส่งวิทยุกระจายเสียง ส่วนโทรศัพท์เครื่องที่สองควบคุมโดยระบบ

นอกจากนี้ ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงยังสามารถตั้งค่าการรับข้อความ SMS แสดงข้อมูลได้ตามรอบเวลา เช่น รับข้อมูลทุก ๆ หนึ่งชั่วโมง โดยระบบจะทำการส่งข้อความ SMS ไปยังโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงตามรอบเวลาดังกล่าว และหากเครื่องส่งวิทยุกระจายเสียงเกิดความผิดปกติ เช่น อุณหภูมิเครื่องส่งวิทยุกระจายเสียงสูงกว่าที่กำหนด ระบบจะทำการส่งข้อความ SMS เตือนไปยังโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงทันที ไม่ว่าผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะอยู่สถานที่ใดก็ตาม

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาการทำงานของ GSM Module

1.2.2 เพื่อศึกษาโปรแกรมควบคุมและการประยุกต์ใช้งาน Microcontroller Broad ARM7

1.2.3 เพื่อเพิ่มประสิทธิภาพการใช้งานของเครื่องส่งวิทยุกระจายเสียง

1.2.4 เพื่ออำนวยความสะดวกต่อผู้ดูแลเครื่องส่งวิทยุกระจายเสียง

1.2.5 เพื่อนำความรู้ที่ได้จากการศึกษาภาคทฤษฎีของวิชาต่าง ๆ ที่ได้ศึกษามาปฏิบัติและประยุกต์ใช้เพื่อสร้างชิ้นงานขึ้นมาและสามารถนำไปใช้งานได้จริง

1.3 ขอบเขตของการทำงาน

1.3.1 รับค่าที่ตรวจสอบ ค่ากำลังส่ง , ค่ากำลังสะท้อนกลับ และค่าอุณหภูมิของเครื่องส่งวิทยุกระจายเสียง มาส่งผ่านทางระบบ SMS เป็นรายวันได้

1.3.2 ใช้อุปกรณ์ GSM Module (wave com) รุ่น Fastrack Supreme 20 ในการควบคุมการทำงาน

1.3.3 ในกรณีที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการทราบค่าที่ตรวจสอบ สามารถส่งข้อความ SMS สอบถามระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง เพื่อให้ระบบส่งข้อความ SMS ตอบกลับแสดงข้อมูล

1.3.4 จัดทำรูปเล่มรายงาน

1.4 ขั้นตอนการทำงาน

1.4.1 ปรึกษาอาจารย์ที่ปรึกษาโครงงานเกี่ยวกับขอบเขตของโครงงานที่จะทำ

1.4.2 ศึกษาการใช้งานบอร์ดไมโครคอนโทรลเลอร์ CP-JR ARM7 USB-LPC 2148 และ GSM Module (wave com) รุ่น Fastrack Supreme 20

1.4.3 จัดหาและสั่งซื้ออุปกรณ์ที่เกี่ยวข้อง

1.4.4 ศึกษาและฝึกการเขียนโปรแกรมไมโครคอนโทรลเลอร์ เพื่อควบคุมสั่งการให้อุปกรณ์ GSM Module (wave com) รุ่น Fastrack Supreme 20 ทำงานตามวัตถุประสงค์

1.4.5 ทดลองใช้งานและปรับปรุงแก้ไขสิ่งที่ผิดพลาด

1.5 ผลประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้เรียนรู้หลักการทำงานของ GSM Module

1.5.2 ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ (Microcontroller)

1.5.3 ได้เรียนการทำงานของวงจรอิเล็กทรอนิกส์

1.5.4 ได้เรียนรู้การทำงานเป็นกลุ่มและได้นำความรู้ไปประยุกต์ใช้ในชีวิตประจำวัน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ระบบโทรศัพท์ GSM

2.1.1 ระบบ GSM

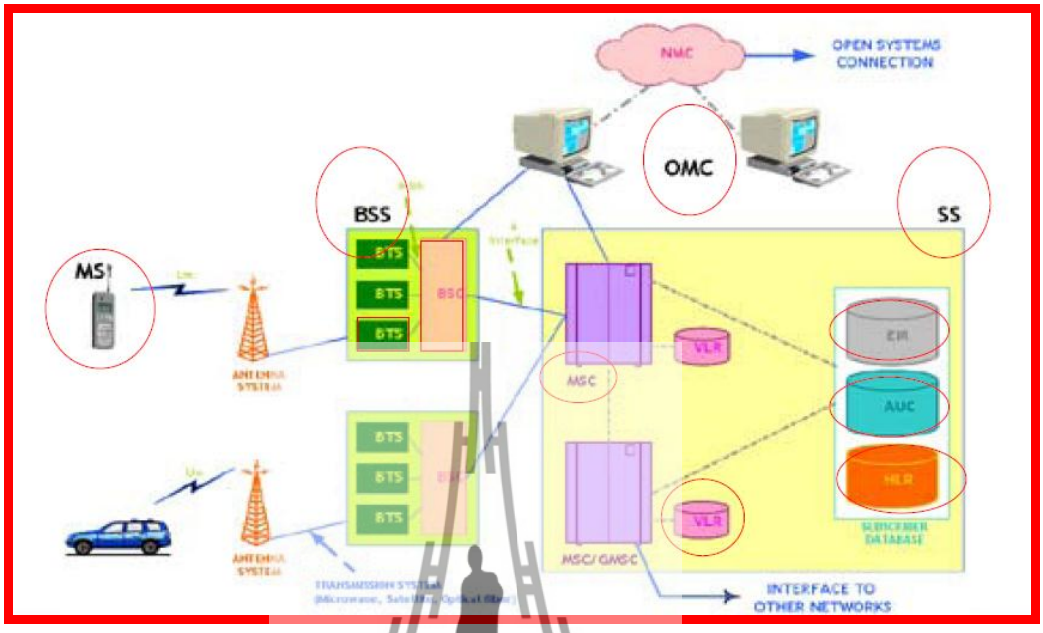
GSM ย่อมาจาก Global System for Mobile Communications เป็นเทคนิคที่ใช้รองรับการใช้งานที่มีความต้องการใช้สูง เน้นลักษณะการรับค่าเป็นสัญญาณดิจิตอล โดยใช้ระบบการแบ่งเวลาที่เรียกว่า TDMA-Time Division Multiple Access

GSM ใช้เทคโนโลยีดิจิตอลสำหรับช่องสัญญาณควบคุมและสัญญาณเสียงแบบ TDMA ซึ่งแตกต่างจากเทคโนโลยีโทรศัพท์มือถือก่อนหน้านั้น จึงถือว่าเป็นโทรศัพท์มือถือในยุคที่สอง หรือ 2G มีพัฒนาการมาจากโทรศัพท์แบบเซลลูลาร์ จนกลายมาเป็น GSM ในปี 1990 ที่มีความเสถียรมากที่สุด

การพัฒนาอย่างแพร่หลายของ GSM เป็นประโยชน์ต่อผู้บริโภคที่สามารถใช้งานได้อย่างสะดวกสบายมากขึ้น และนอกจากนี้ยังเป็นประโยชน์ต่อผู้ควบคุมระบบเนตเวิร์ค ให้มีตัวเลือกในการใช้งานมากขึ้น เนื่องจากมีผู้จัดทำอย่างแพร่หลาย GSM เริ่มต้นด้วยทางเลือกใหม่ ซึ่งมีราคาที่ถูกเป็นอีกหนึ่งทางเลือกในการติดต่อสื่อสาร นั่นก็คือ Short Message Service (SMS) หรือเรียกอีกอย่างว่า เท็กเมสเสสจิ้ง ซึ่งโทรศัพท์มือถือทั่วไปสามารถรองรับได้อย่างดี

รูปที่ 2.1 โครงข่ายระบบเซลลูล่าร์

2.1.2 โครงสร้างของระบบ GSM



รูปที่ 2.2 ส่วนประกอบของระบบ GSM

1) SWICHING SYSTEM (SS) ประกอบไปด้วย 5 ส่วน ดังนี้

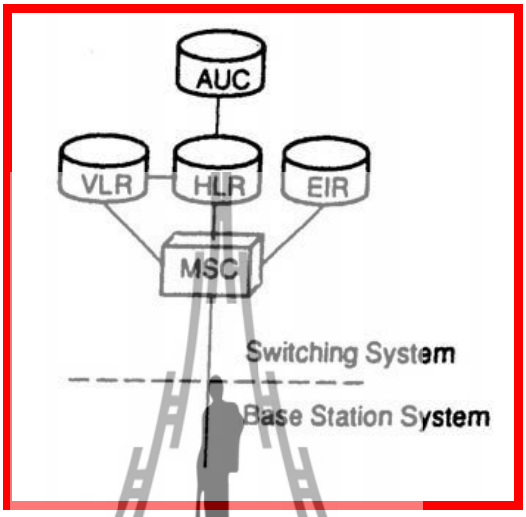
1.1) MOBILE SERVICE SWITCHING CENTER (MSC) คือ ชุมสายของระบบโทรศัพท์เคลื่อนที่ GSM มีหน้าที่ควบคุมระบบและคิดเงินค่าใช้บริการ

1.2) HOME LOCATION REGISTER (HLR) เป็น DATABASE ใช้เก็บข้อมูลเกี่ยวกับชื่อและที่อยู่ของเจ้าของโทรศัพท์เพื่อใช้ในการเก็บเงินค่าบริการ นอกจากนี้ยังเก็บข้อมูล ตำแหน่งโทรศัพท์เคลื่อนที่อยู่ใน MSC ไหน การติดตั้ง HLR อาจจะติดตั้งอยู่รวมกับ MSC หรือแยกกันก็ได้

1.3) VISITOR LOCATION REGISTER (VLR) เป็น DATABASE ที่เก็บข้อมูลโทรศัพท์เคลื่อนที่ที่เข้ามาอยู่ในชุมสายนี้ ในการติดตั้ง VLR ส่วนใหญ่จะติดตั้งรวมกับ MSC เสมอ

1.4) AUTHENTICATION CENTER (AUC) ใช้สำหรับเก็บ AUTHENTICATION เพื่อตรวจสอบว่าผู้ใช้โทรศัพท์ได้ลงทะเบียนอย่างถูกต้องหรือไม่ และ ENCRYPTION PARAMETER ซึ่งใช้สำหรับระบบการป้องกันการดักฟัง

1.5) EQUIPMENT INDETITY REGISTER (EIR) เป็น DATABASE ที่เก็บข้อมูล IDENTITY ของเครื่องโทรศัพท์เคลื่อนที่ เพื่อป้องกันไม่ให้โทรศัพท์เคลื่อนที่ที่ไม่ลงทะเบียน หรือได้มาอย่างไม่ถูกต้องตามกฎหมาย เข้ามาใช้งานในระบบได้ การติดตั้ง ELR ส่วนใหญ่จะอยู่รวมกับ AUC



รูปที่ 2.3 แสดงภาพ Switching System

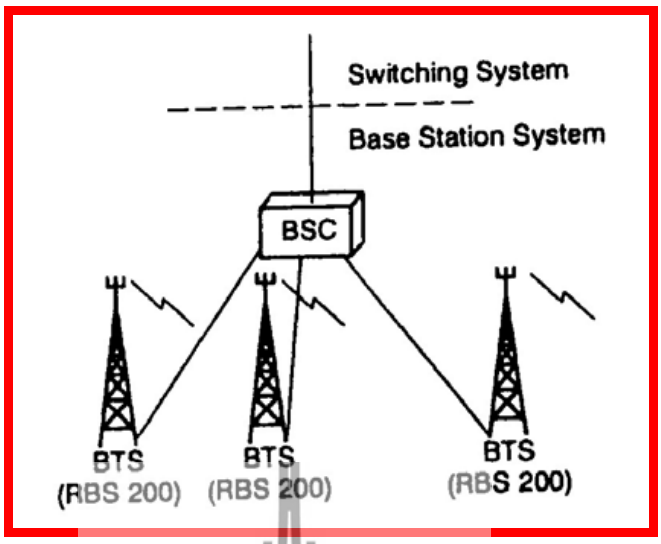
2) BASE STATION SYSTEM (BSS)

2.1) BASE STATION CONTROLLER (BSC)

2.2) BASE TATION TRANSCEIVER (BTS)

BASE STATION CONTROLLER (BSC) คือชุมสายหนึ่งที่ทำหน้าที่ควบคุมเกี่ยวกับคลื่นวิทยุในระบบ เช่น ควบคุมการ HANDOVER จัดการเกี่ยวกับช่องสัญญาณวิทยุต่างๆ และเก็บข้อมูลเกี่ยวกับ CELL นอกจากนี่ยังควบคุมกำลังส่งสัญญาณของสถานีฐานและโทรศัพท์เคลื่อนที่

ในการทำงานนั้น MSC แต่ละ MSC จะควบคุม BSC 1 BSC หรือมากกว่า และในแต่ละ BSC จะควบคุม BTS หลาย ๆ BTS



รูปที่ 2.4 แสดงภาพของ Base Station System

3) OPERATION & SUPPORT SYSTEM (OSS)

มีหน้าที่ในการควบคุมและรายงานสถานะภาพของอุปกรณ์ต่าง ๆ ในระบบ เช่น ตรวจเช็คว่ามีความผิดพลาดอะไรที่เกิดขึ้น และความรุนแรงมากแค่ไหน จากนั้นจะส่ง ALARM ไปยัง OMC (OPERATION AND MAINTENANCE CENTER)

โทรศัพท์เคลื่อนที่ (MS) สำหรับโทรศัพท์เคลื่อนที่ของระบบ DIGITAL ประกอบไปด้วย 2 ส่วน คือ

1. อุปกรณ์โทรศัพท์ ซึ่งจะเก็บ IDENTITY CODE ที่เรียกว่า INTERNATIONL MOBILE EQUIPMENT IDENTITY (IMEI)
2. SUBSCRIBER IDENTITY MODULE (SIM CARD) ซึ่งมี 2 แบบ คือ
   2.1 ISO SMART CARD
   2.2 IC PLUG IN

ซึ่งจะใช้เก็บข้อมูลประจำตัวผู้ใช้มีลักษณะเป็น IDENTITY CODE ที่เรียกว่า INTERNATIONAL MOBILE SUBSCRIBER IDENTITY (IMSI) ทั้งอุปกรณ์โทรศัพท์และ SIM CARD จะแยกออกจากกัน นั่นคือเมื่อผู้ใช้โทรศัพท์ใส่ SIM CARD ในเครื่องโทรศัพท์ของผู้อื่น โทรศัพท์เคลื่อนที่จะเหมือนโทรศัพท์ของเราทุกประการ และถ้าโทรศัพท์และ SIM CARD ถูกขโมย เราสามารถที่จะไม่ให้ขโมยใช้เครื่องของเราได้ โดยการ BAR ทั้งเครื่องโทรศัพท์และ SIM CARD ได้ที่ EIR และ HLR ตามลำดับ

2.1.3 ข้อดีของระบบ GSM

1. มีการส่งข้อมูลได้อย่างรวดเร็ว
2. สัญญาณครอบคลุมพื้นที่ได้หมด ทำให้สามารถเข้าถึงได้จากทุกหนทุกแห่ง
3. ลักษณะของมาตรฐานของสัญญาณสามารถปรับเปลี่ยนได้ หรือการปรับเปลี่ยนสัญญาณ (Adaptive Signal) ทำให้ระบบมีความหลากหลายไม่เจาะจงอยู่แบบใดแบบหนึ่ง
4. มีการใช้เครือข่ายอื่นประกอบ เช่น อินเทอร์เน็ต อินทราเน็ต หรือเครือข่ายเฉพาะกิจอื่น ๆ ทำให้ระบบมีความยืดหยุ่นและขยายตัวได้มากโดยเฉพาะเครือข่าย internet ที่เชื่อมโยงกันทั่วโลก
5. โครงสร้างเครือข่ายมีพื้นฐานอย่างชัดเจน มีการวางเครือข่ายหลัก (Backbone) ของชุมสาย การวางสถานีฐาน การเชื่อมต่อระหว่างสถานี จะครอบคลุมพื้นที่และมีการเชื่อมโยงเครือข่ายเข้าด้วยกัน

2.1.4 ข้อเสียและปัญหาสำคัญของระบบ GSM

การพัฒนาของการสื่อสารไร้สายและระบบติดตามตัวระบบ GSM ยังไม่เป็นที่ใช้งานในวงกว้าง ทั้งนี้เพราะมีอุปสรรคและปัญหาที่สำคัญ ซึ่งเป็นปัญหาหลัก 4 ประการ คือ

1. ระบบไร้สายใช้อัตราการรับส่งข้อมูลได้ต่ำ
2. ค่าบริการค่อนข้างแพง
3. โมเด็มรับส่งแบบคลื่นวิทยุ ใช้กำลังงานไฟฟ้าสูง
4. ระบบเชื่อมต่อกับผู้ใช้ที่ใช้กับระบบติดตามตัวยังไม่ดี ไม่เหมาะกับการใช้งานขณะเคลื่อนที่

ปัญหาเหล่านี้เป็นปัญหาที่ระบบไร้สายในยุคที่ 3 ต้องแก้ไขให้ได้ให้หมด โดยเฉพาะระบบโทรศัพท์เคลื่อนที่ที่ต้องเพิ่มอัตราการรับส่งข้อมูลให้ได้มาก เพื่อจะส่งรูปภาพหรือภาพเคลื่อนไหวได้ และจะต้องมีอัตราค่าใช้บริการที่ถูกลง รวมทั้งเครื่องที่ใช้ต้องใช้กำลังงานต่ำหรือกินไฟฟ้าน้อย

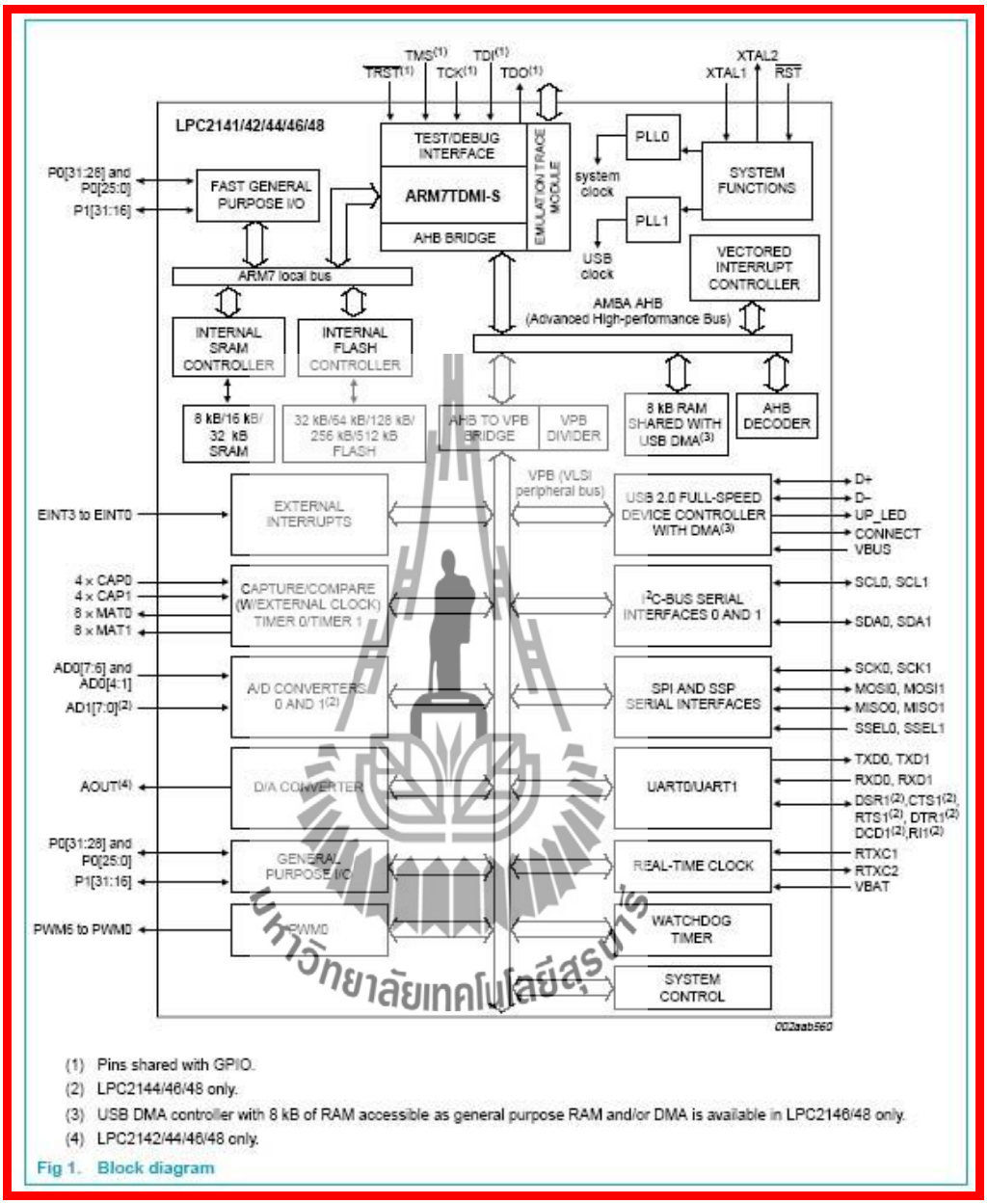
2.2 ไมโครคอนโทรลเลอร์ ARM7

2.2.1 ไมโครคอนโทรลเลอร์ ARM7 Philips LCP2148

ความสามารถของไมโครคอนโทรลเลอร์ Philips LCP2148 มีดังนี้

- ไมโครคอนโทรลเลอร์ขนาด 16/32 บิต ARM7TDMI-S มี LQFP 64 ขา
- หน่วยความจำ Static RAM มีขนาด 40kB
- หน่วยความจำ Flash Program Memory มีขนาด 512kB อยู่ภายในชิปที่สามารถ ลบ/เขียนซ้ำได้ถึง 10000 ครั้ง โปรแกรมชิปสามารถได้ทันทีผ่าน In-System Programming (ISP) และ In-Application Programming (IAP) โดยใช้ซอฟต์แวร์ boot-loader ที่อยู่ภายในชิปตัวควบคุมอุปกรณ์ USB 2.0 Full-speed โดยมี ARM สำหรับ endpoint ขนาด 8 kB สำหรับการติดต่อแบบ DMAวงจรแปลงแอนะลอกเป็นดิจิตอลความละเอียด 10 บิตจำนวน 2 ชุด ที่รับอินพุตได้ถึง 14 อินพุต โดยมีเวลาในการแปลงค่าต่ำถึง 2.44 µsวงจรแปลงดิจิตอลเป็นแอนะลอกความละเอียด 10 บิต 1 ตัววงจรไทเมอร์ขนาด 32 บิต 2 ชุด(มี 4 capture และ 4 compare channel)PWM (Pulse width modulation) 6 เอาต์พุทโมดูลนาฬิกาเวลาจริง (Real Time Clock) ที่สามารถต่อกับคริสตอลความถี่ 32 kHz และแบตเตอรี่ภายนอกได้ และวอชดอกซ์ (Watchdog)
- วงจรสื่อสารอนุกรม UART(16C550) จำนวน 2 ชุดวงจรสื่อสารอนุกรม I^2C ความเร็วสูง(400Kbits/s)วงจรสื่อสารอนุกรม SPI และ SSP มีวงจร Phase lock loop ภายในเพื่อคูณค่าให้สัญญาณนาฬิกาภายในทำงานที่ความถี่สูงสุด 60 MHz Vectored Interrupt Controller ที่สามารถกำหนดลำดับความสำคัญ และกำหนดแอดเดรสของเวกเตอร์ได้ใช้กับแหล่งจ่ายไฟชุดเดียวขนาด 3.0 V และ 3.6V (3.3±10%)
- มี I/O pin อเนกประสงค์ที่สามารถใช้กับระดับแรงดัน 5 V ได้สูงสุด 45 ขาโดยสามารถจัดเป็นขาอินเตอร์รัปต์จากภายนอกได้สูงสุด 21 ขามีโหมดประหยัดพลังงาน 2 โหมดได้แก่ Idle และ Power-down

Block Diagram LPC2148



รูปที่ 2.5 Block Diagram LPC2148

จากรูปที่ 2.5 เป็นไมโครโปรเซสเซอร์ ARM7TDMI-S ซึ่งเป็นปัจจัยหลัก ด้านซ้ายมือเป็นส่วนของ AMR7 Local Bus ที่ใช้ในการติดต่อกับหน่วยความจำแบบ Flash ที่ใช้เก็บโปรแกรม และหน่วยความจำ SRAM ที่ใช้เก็บข้อมูล ส่วนที่ใช้ในการติดต่อกับหน่วยความจำภายนอกมีการ

ติดต่อผ่านบัส AMBA AHB (Advanced High-performance Bus) ซึ่งใน LPC2148 ไม่สามารถต่อกับหน่วยความจำภายนอกได้

ในการติดต่อกับอุปกรณ์ที่เกี่ยวข้อง เช่น GPIO, I^2C, SPI, UART ฯลฯ จะติดต่อผ่านบัส VPB (VLSI peripheral BUS) ซึ่ง VPB บัสต่อกับ AHB บัส ผ่าน AHB to VPB Bridge โดยสามารถปรับลดค่าความถี่ของ VPB บัสให้ทำงานช้ากว่าความถี่ของซีพียูได้เพื่อให้ทำงานร่วมกับอุปกรณ์เสริมต่าง ๆ ที่มีความเร็วต่ำกว่าได้

การจัดหน่วยความจำของ LPC2148

เนื่องจาก ARM7 เป็นซีพียูขนาด 32 บิต ที่มีขาแอดเดรสต่อกับหน่วยความจำ จำนวน 32 ขา ทำให้สามารถอ้างถึงหน่วยความจำได้ 4 GB (2^{32} = 4 GB) อุปกรณ์หลักของ ARM7TDMI จะมีสถาปัตยกรรมแบบ Von Neumann ที่ใช้บัสขนาด 32 บิต ชุดเดียวกันสำหรับตัวคำสั่งของโปรแกรม และข้อมูล โดยมีแค่คำสั่ง load, store, swap เท่านั้น ที่ใช้การเรียกข้อมูลที่เก็บในหน่วยความจำ การติดต่อกับคอมพิวเตอร์อินพุต หรือ เอาต์พุตก็จะใช้คำสั่งเดียวกันกับการใช้คำสั่งจัดการเกี่ยวกับหน่วยความจำในไมโครคอนโทรลเลอร์ LPC2148 ได้จัดสรรหน่วยความจำดังรูปที่ 2.6 เมื่อซีพียูถูกรีเซตจะเริ่มต้นทำงานที่หน่วยความจำ 0x000 0000 จากหน่วยความจำทั้งหมด 4 GB ได้แบ่งออกเป็น 4 ส่วน ดังนี้

- 1 GB แรก (แอดเดรส 0x0000 0000 – 0x3FFF FFFF) จัดเป็นส่วนของหน่วยความสำหรับเก็บโปรแกรม คือ หน่วยความจำ Flash memory ขนาด 512 kB ซึ่งมีแอดเดรส 0x0000 0000 – 0x0007 FFFF
- หน่วยความจำในช่วง 1 GB - 2 GB (แอดเดรส 0x4000 0000 – 0x7FFF FFFF) จัดเป็นส่วนของหน่วยความจำ ARM จะเป็น SRAM ขนาด 40 kB โดยแบ่งออกเป็น 2 ส่วน
    - ส่วนที่ 1 คือ 32 kB แรกอยู่ที่แอดเดรส 0x4000 0000 – 0x4000 7FFFF
    - ส่วนที่ 2 คือ 8 kB เป็นหน่วยความจำใช้สำหรับ USB โดยมีแอดเดรส 0x7FD0 0000- 0x7FD0 1FFF ในกรณีที่ไม่ใช้การติดต่อกับ USB สามารถนำ หน่วยความจำ ARM ส่วนนี้มาใช้เป็น ARM สำหรับงานทั่วไปได้ หน่วยความจำที่ใกล้ 2 GB จะเป็นส่วนของ Boot Block ขนาด 12 kB ซึ่งเป็นโปรแกรมที่ทำงานเพื่อเขียนโปรแกรมลงในหน่วยความจำ Flash

- หน่วยความจำ 2 GB - 3 GB (แอดเดรส 0x8000 0000 – 0xDFFF FFFF) สงวนไว้สำหรับต่อกับหน่วยความจำภายนอก ซึ่งไม่ได้ใช้งาน
- หน่วยความจำ 2 GB - 3 GB จะเป็นพื้นที่สำหรับติดต่อกับอุปกรณ์อื่น ๆ ที่อยู่ในชิป โดยแบ่งเป็น อุปกรณ์ที่ต่อกับ VPB บัส จะติดต่อกับหน่วยความจำช่วง 0xE000 00000 – 0xEFFF FFFF ถ้าเป็นอุปกรณ์ที่ติดต่อกับผ่านทาง AHB จะมีช่วงแอดเดรส 0xF000 0000 – 0xFFFF FFFF ไม่สามารถติดต่อกับหน่วยความจำภายนอก จึงไม่ได้ใช้หน่วยความจำส่วนนี้

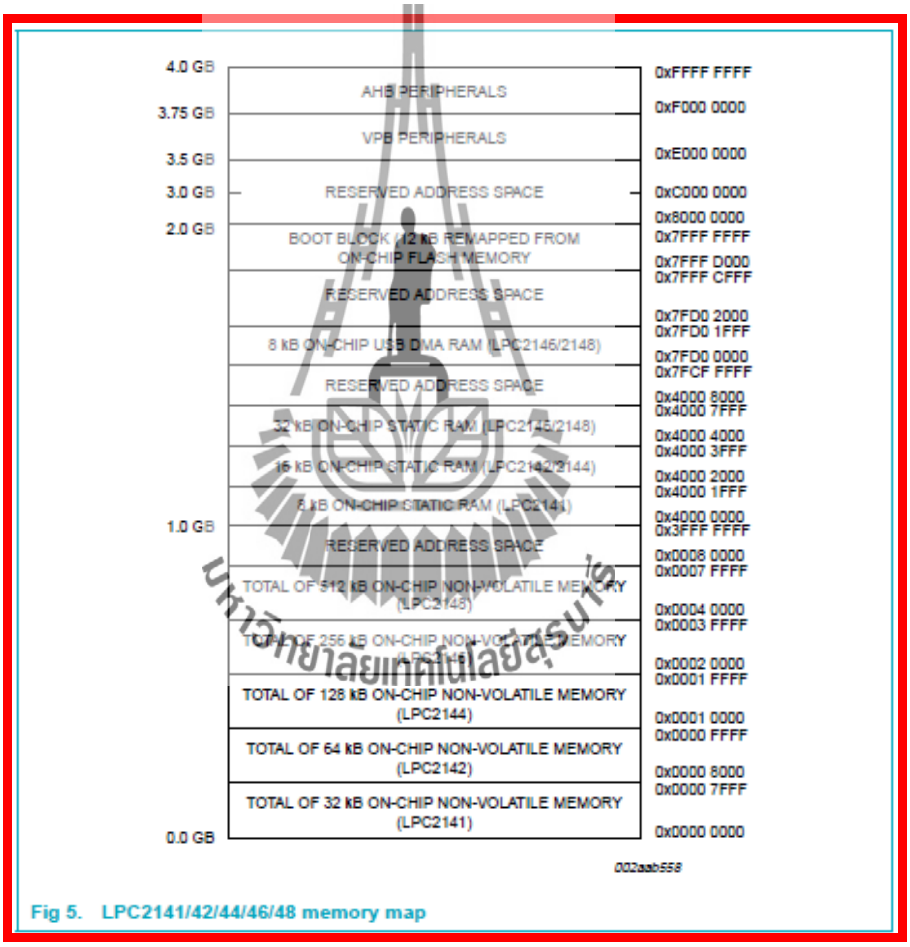
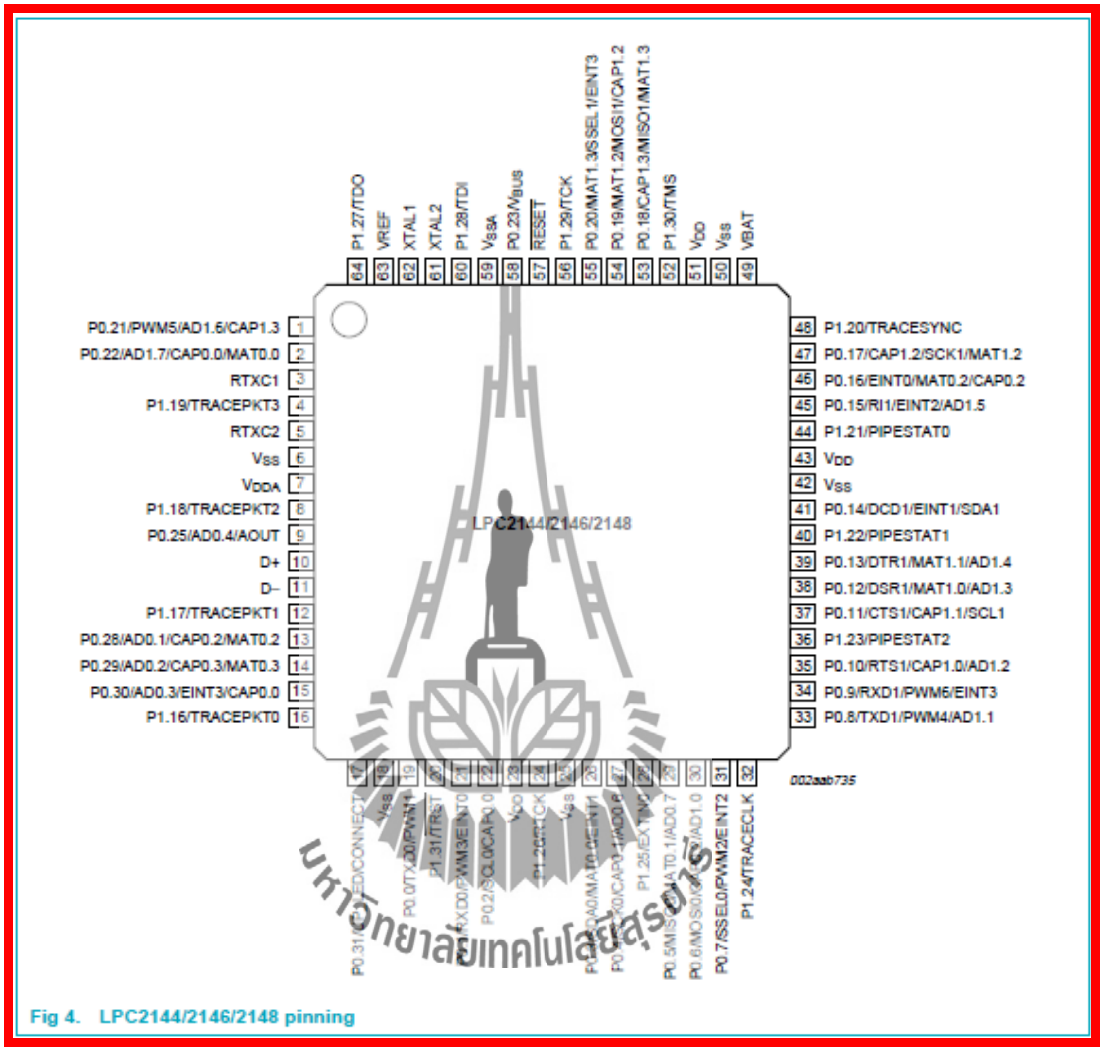


Fig 5. LPC2141/42/44/46/48 memory map

รูปที่ 2.6 Memory map LPC2148

การจัดขาของ LPC2148

มีจำนวนขาทั้งหมด 64 ขา โดยมีการจัดขาแสดงดังรูปที่ 2.7 ดังนี้



รูปที่ 2.7 LPC2148 pinning

หลังจากการรีเซตขาพอร์ตทั้งหมดจะถูกกำหนดให้ทำหน้าที่เป็นอินพุตขาแต่ละขา จะมีการทำงาน เช่นขาที่ 14 สามารถทำหน้าที่ได้ 4 หน้าที่ คือ

- เป็น P0.29/AD0.2/CAP0.3/MAT0.3
- ถ้าเป็นอินพุตเอาต์พุตจะเรียกว่า General Purpose Input Output : GPIO ก็คือขา P0.23

- ถ้าใช้วงจรแปลงแอนะลอกเป็นดิจิตอลขานี้คือ AD0.2 เป็นขาอินพุตสำหรับสัญญาณแอนะลอก ADC0 อินพุตสอง
- ถ้าใช้งาน Time0 ขานี้จะเป็น CAP0.3 คือ Capture input for timer 0, Channel3 หรือ MAT0.3 ซึ่งเป็น Match output for timer0, Channel3

2.2.2 ฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการทดลอง

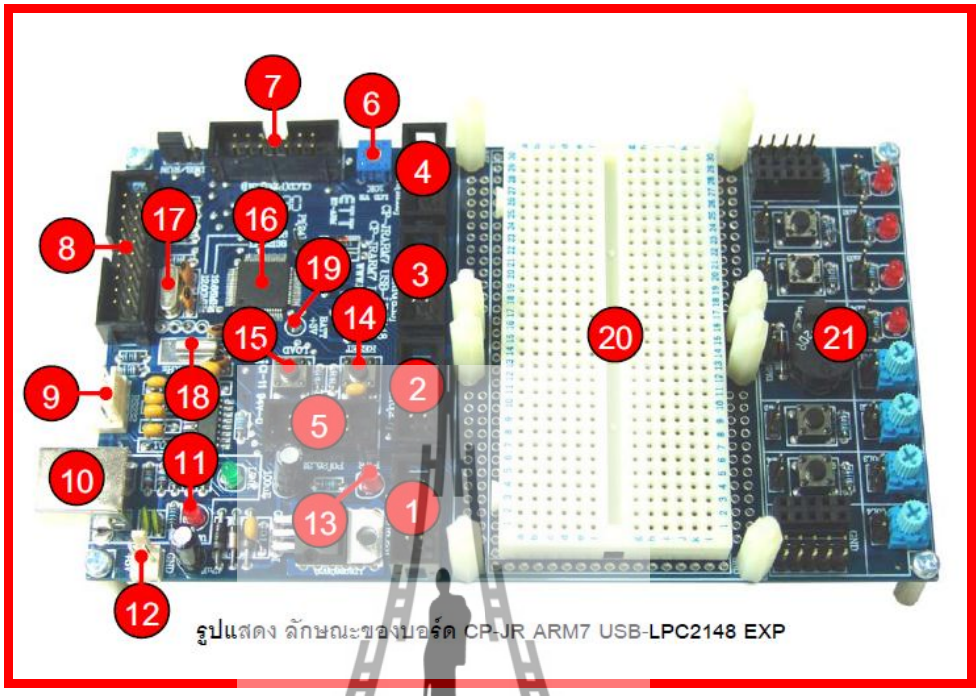
ฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการเขียนไมโครคอนโทรลเลอร์ Philips LPC 2148 เป็นไมโครคอนโทรลเลอร์ตระกูล ARM7TDMI-S ฮาร์ดแวร์ที่ใช้ประกอบด้วย แผงวงจร CP-JR ARM7 USB-LPC2148 EXP บริษัท อีทีที จำกัด ส่วนของซอฟต์แวร์คอมไพเลอร์เพื่อแปลภาษาซีเป็นภาษาเครื่องของ LPC2148 จะใช้ชุดพัฒนา RealView Microcontroller Development Kit Version 3.02a โดยใช้คอมไพเลอร์ CARM ของบริษัท Keill เป็นรุ่น Evaluation ที่อนุญาตให้ดาวน์โหลดมาใช้ทดสอบโปรแกรมฟรี โดยมีข้อจำกัดว่าโปรแกรมที่แปลเป็นภาษาเครื่องแล้วจะมีขนาดไม่เกิน 16 kB หลังจากคอมไพล์เป็นภาษาเครื่องจะทำการลงโปรแกรมชิปไมโครคอนโทรลเลอร์ด้วยโปรแกรม LPC2000 Flash Utility ของบริษัท Philips ซึ่งอนุญาตให้ใช้งานได้ฟรี

คุณสมบัติของบอร์ดไมโครคอนโทรเลอร์ CP-JR ARM7 USB-LPC2148

1. ใช้ MCU ตระกูล ARM7TDMI-S เบอร์ LPC2148 ของ Philips ซึ่งเป็น MCU ขนาด 16/32-Bit
2. ใช้ Crystal 12.00MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 60MHz เมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง
3. รองรับการโปรแกรมแบบ In-System Programming (ISP) และ In-Application Programming (IAP) ผ่านทาง On-Chip Boot-Loader Software ทางพอร์ต UART0 (RS232)
4. Power Supply ใช้แรงดันไฟฟ้า +5VDC โดยใช้ขั้วต่อแบบ CPA-2PIN จากภายนอก หรือใช้พลังงานจาก USB Port ได้ (ในกรณีใช้กระแสไม่เกิน 500 mA)
5. ภายใน MCU มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 512kB , Static RAM ขนาด 40 kB
6. มีวงจร USB มาตรฐาน 2.0 แบบ Full Speed ภายในตัว (USB Function มี 32 End Point)

7. จำนวน GPIO สูงสุดถึง 47 I/O Pins สามารถเชื่อมต่อกับระบบ I/O ที่เป็นสัญญาณ 5 V ได้ ซึ่งขาสัญญาณ GPIO จะมีการใช้งานร่วมกันของ Function อื่นๆอีกดังนี้วงจรสื่อสารอนุกรมแบบ SPI จำนวน 2 ช่อง และ วงจรสื่อสารอนุกรมแบบ I2C จำนวน 2 ช่องวงจร ADC ขนาด 10 Bit จำนวน 14 ชุด และ วงจร DAC ขนาด 10 Bit จำนวน 1 ชุดวงจร UART แบบ Full-Duplex จำนวน 2 ช่อง คือ UART-0 มาตรฐาน 4 Pin ETT เป็นสัญญาณระดับ RS232 Level และ UART-1 เป็นสัญญาณระดับ TTL LevelTimer 32-bit จำนวน 2 ช่อง (4 Input Capture / 4 Output Compare), 6-Channel PWMOutput, Watchdog Timer และ Real Time Clock
8. มีวงจรเชื่อมต่อกับ Character LCD โดยใช้วงจรการเชื่อมต่อแบบ 4 บิต จาก GPIO1[25..31] พร้อมวงจรปรับความสว่างหน้าจอ
9. มีวงจรเชื่อมต่อกับ JTAG ARM ขนาด 20 Pin มาตรฐาน เพื่อทำการ Debug แบบ Real Time ได้
10. มีวงจรทดลองขั้นพื้นฐานสำหรับสนับสนุนการใช้งานและทดลองเรียนรู้ ขั้นพื้นฐานอย่างครบถ้วน จัดเตรียมไว้ภายในบอร์ด (ติดตั้งไว้เฉพาะรุ่น CP-JR ARM7 USB-LPC2148 EXP) ซึ่งได้แก่
    - LED Output แบบ Sink Current สำหรับแสดงสถานะของ Output จำนวน 4 ชุด
    - Push Button Switch แบบ Active Logic “0” สำหรับทดสอบ Input Logic จำนวน 4 ชุด
    - Volume ปรับค่าแรงดัน 0-3.3 V สำหรับทดสอบการทำงานของ ADC จำนวน 4 ชุด
    - ชุดกำเนิดสัญญาณเสียง Mini Speaker สำหรับทดสอบการเสียงแบบต่างๆ จำนวน 1 ชุด
    - แผงต่อวงจร Project Board รุ่น AD-100 ขนาด 360 จุด สำหรับเป็นพื้นที่ต่อทดลองวงจรขนาดเล็ก ๆ เพื่อใช้งานร่วมกับ CPU ได้อย่างอิสระ
    - จุดต่อแหล่งจ่ายไฟ +3.3V และ GND สำหรับเชื่อมต่อไปยังวงจรภายนอกอื่น ๆ
11. ทนอุณหภูมิใช้งานระหว่าง -40 to +85°C

โครงสร้างบอร์ดไมโครคอนโทรเลอร์ CP-JR ARM7 USB-LPC2148



รูปที่ 2.8 ลักษณะโครงสร้างของบอร์ด CP-JR ARM7 USB-LPC2148 / EXP

หมายเลข 1 คือ ขั้วต่อ Port1[16..23] จำนวน 8 บิต

หมายเลข 2 คือ ขั้วต่อ Port0[2..7] จำนวน 6 บิต

หมายเลข 3 คือ ขั้วต่อ Port0[8..15] จำนวน 8 บิต

หมายเลข 4 คือ ขั้วต่อ Port0[16..23] จำนวน 8 บิต

หมายเลข 5 คือ ขั้วต่อ Port0[25..31] จำนวน 7 บิต

หมายเลข 6 คือ ตัวต้านทานสำหรับปรับค่าความสว่าง (Contrast) ของหน้าจอ LCD

หมายเลข 7 คือ ขั้วต่อ Character LCD โดยใช้สัญญาณ Port1[25..31] ในการเชื่อมต่อ

หมายเลข 8 คือ ขั้วต่อ JTAG โดยใช้สัญญาณ Port1[26..31] และ Reset ของ CPU

หมายเลข 9 คือ ขั้วต่อ RS232 สำหรับใช้งาน และ Download Hex File ให้ CPU

หมายเลข 10 คือ ขั้วต่อ USB สำหรับเชื่อมต่อกับ USB Hub รุ่น 2.0

หมายเลข 11 คือ LED แสดงสถานะ Power จาก USB และ สถานของการเชื่อมต่อกับ USB

หมายเลข 12 คือ ขั้วต่อ Power ขนาด +5VDC และ GND เพื่อจ่ายให้กับบอร์ด

หมายเลข 13 คือ LED แสดงสถานะของแหล่งจ่ายไฟ Power ของบอร์ด

หมายเลข 14 คือ Switch RESET สำหรับสั่ง Reset การทำงานของ CPU

หมายเลข 15 คือ Switch LOAD ใช้ร่วมกับ Switch RESET เพื่อ Download Hex ให้ CPU

หมายเลข 16 คือ CPU เบอร์ LPC2148 ของ Philips ซึ่งเป็น CPU ประจำบอร์ด

หมายเลข 17 คือ Crystal 12.00 MHz สำหรับป้อนให้เป็นสัญญาณนาฬิกาของ LPC2148

หมายเลข 18 คือ Crystal 32.768 KHz สำหรับ Real Time Clock (RTC) ในตัวของ LPC2148

หมายเลข 19 คือ จุดเชื่อมต่อ ลังถ่าน Battery ขนาด +3V (อยู่ด้านใต้บอร์ด) สำหรับต่อให้กับRTC เพื่อเก็บรักษาค่าเวลาของ RTC ในขณะที่ไม่ได้จ่ายไฟเลี้ยงให้กับบอร์ด

หมายเลข 20 คือ แผง Project Board รุ่น AD-100 ขนาด 360 จุด สำหรับต่อวงจร (มีติดตั้งไว้เฉพาะในรุ่น CP-JR ARM7 USB-LPC2148 EXP)

หมายเลข 21 คือ ส่วนของวงจร I/O พื้นฐาน สำหรับใช้ทดสอบการทำงานของ Function ต่างๆของ CPU (มีติดตั้งไว้เฉพาะในรุ่น CP-JR ARM7 USB-LPC2148 EXP) โดยมีรายละเอียดดังนี้คือ

- LED สำหรับแสดงผลการทำงานของ Output แบบ Sink Current มีทั้งหมด 4 ชุด
- Push Button Switch สำหรับกำเนิด Logic เพื่อทดสอบการทำงาน Input มีทั้งหมด 4 ชุด
- Volume สำหรับปรับค่าแรงดัน 0-3.3V เพื่อใช้ทดสอบการทำงานของ A/D มีทั้งหมด 4 ชุด
- Mini Speaker สำหรับใช้กำเนิดเสียง เช่น Beep จำนวน 1 ชุด
- จุดต่อแหล่งจ่ายไฟ +3.3V และ GND

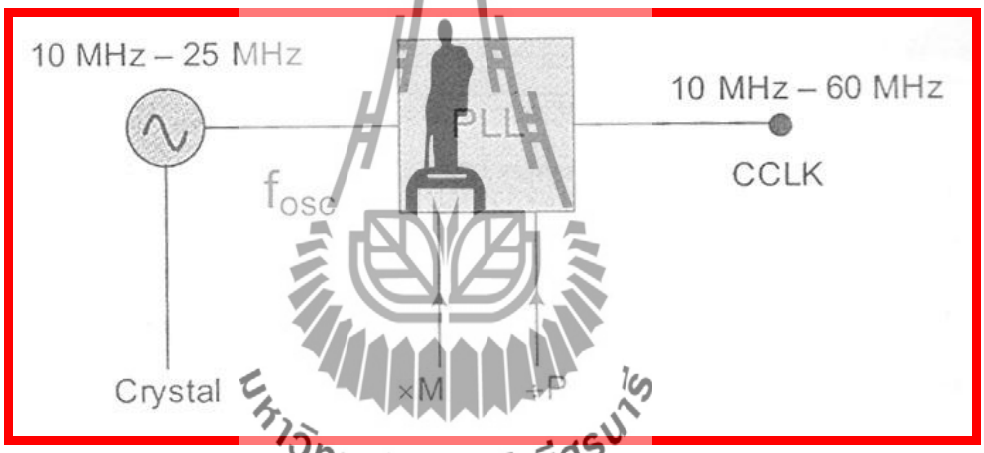
2.2.3 การกำหนดค่าเริ่มต้นให้กับไมโครคอนโทรลเลอร์ ARM7

การกำหนดค่าเริ่มต้นให้ก่อนโดยค่าที่ต้องการกำหนด คือ ส่วนของหน่วยความจำ RAM ที่ใช้เป็น stack pointer การกำหนดค่าของ stack pointer ต้องเขียนคำสั่งเป็นโปรแกรมภาษาแอสแซมบลี ภายในของไมโครคอนโทรลเลอร์ ARM มีวงจร Phase Lock Loop : PLL สำหรับคูณค่าความถี่ของสัญญาณนาฬิกาจากภายนอก เพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานที่ความถี่สูง ๆ ได้โดยการกำหนดค่าตัวคูณความถี่ สำหรับ LPC2148 ภายในจะมีวงจร PLL อยู่สองวงจรโดยตั้งชื่อเป็น PLL0 และ PLL1 โดย PLL0 ใช้คูณค่าความถี่ของสัญญาณนาฬิกาภายนอกให้กับซีพียู ARM 7 และ PLL1 ที่ทำหน้าที่คูณวามถี่ให้กับวงจรส่วนของ USB โดยต้องคูณค่าให้ได้ความถี่ 48 MHz หลังจากที่ไมโครคอนโทรลเลอร์ทำงานที่ความถี่สูงแล้ว อุปกรณ์ประกอบ (Peripheral) ต่าง ๆ ที่อยู่ภายนอกไมโครคอนโทรลเลอร์ เช่น UART , I^2C ฯลฯ จะทำงานไม่ทัน ดังนั้นภายในไมโครคอนโทรลเลอร์ ARM7 จะมีวงจรหารความถี่เพื่อลดความถี่ที่ป้อนให้อุปกรณ์ต่างๆหน่วยความจำแบบ Flash ภายในของ LPC2148 มีการทำงานช้ากว่าตัวซีพียู

ดังนั้นภายในของ LPC2148 จะมีวงจร Memory Accelerator Module (MAM) เพื่อทำหน้าที่เร่งความเร็วของการติดต่อกับหน่วยความจำแฟลช(Flash) ภายในชิป

วงจร Phase Lock Loop : PLL

วงจรคริสตัลออสซิลเลเตอร์ภายในของไมโครคอนโทรลเลอร์ตระกูล LPC2000 ใช้ได้กับคริสตัล ค่าความถี่ 1 MHz – 30MHz เอาต์พุตของวงจรเรียกว่า f_{osc} ส่วนความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์จะมีชื่อเรียกว่า CCLK โดยปกติถ้าไม่ได้เปิดใช้วงจร PLL ไมโครคอนโทรลเลอร์ ARM7 จะนำความถี่ f_{osc} ไปใช้กับไมโครคอนโทรลเลอร์ ARM7 เช่นเดียวกัน (กรณีนี้ค่าของ f_{osc} และ CCLK จะมีค่าเท่ากัน)



รูปที่ 2.9 ผังวงจร PLL ภายในไมโครคอนโทรลเลอร์ ARM7

วงจร PLL จะรับค่าความถี่สัญญาณนาฬิกาจากคริสตัลออสซิเลเตอร์ที่ถูกควบคุมค่าโดยคริสตัลภายนอก แล้วนำมาคูณด้วยค่าคงที่ M ให้เป็นความถี่ 10 MHz – 60 MHz โดยใช้วงจร Current Controlled Oscillator (CCO) ทำหน้าที่คูณความถี่ ค่าตัวคูณ M จะมีค่าตั้งแต่ 1 ถึง 32 วงจร CCO ทำงานในช่วงความถี่ 156 MHz - 320MHz ดังนั้นภายในลูปของ CCO จะต้องมีวงจรหารค่าอีกหนึ่งตัวเพื่อให้เอาต์พุตของ PLL สร้างความถี่ให้ได้ค่าตามที่ต้องการ ค่าตัวหาร คือ P กำหนดค่าได้เป็น 2,4,8 หรือ 16 เมื่อไมโครคอนโทรลเลอร์ถูกรีเซต วงจร PLL จะถูกปิดการทำงาน ซึ่งต้องให้ซอฟต์แวร์สั่งเปิดการทำงานของโปรแกรม ตัวโปรแกรมจะต้องกำหนดค่าตัวคูณ M ตัวหาร P และกระตุ้นการทำงานของ PLL และรอให้ PLL ล็อกความถี่ได้ก่อน จึงจะสั่งต่อ

ให้ PLL เป็นสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ ARM7 ค่าเวลาในการเซต PLL เป็น 10 μs เอาต์พุตของ PLL คำนวณได้ดังนี้

$$CCLK = M * f_{osc} \quad \text{หรือ} \quad CCLK = fcco/(2 * P)$$

ค่าความถี่ CCO คำนวณได้ดังนี้

$$fcco = CCLK * 2 * P \quad \text{หรือ} \quad fcco = f_{osc} * 2 * M * P$$

ค่า P เมื่อนำไปคูณแล้ว fcco จะมีเงื่อนไขดังนี้

$$156MHz < fcco < 320MHz$$

แผงวงจร CP-JR ARM7 USB-LPC2148 EXPใช้คริสตัลความถี่ 12.00 MHz ดังนั้น f_{osc} = 12 MHz ต้องการคูณให้ได้ความถี่สูงสุดไม่เกิน 60 MHz จะได้ตัวคูณ M = 5 ดังนั้น CCLK = 5*12.00 = 60.00 MHz

ในการกำหนดค่า PLL1 เพื่อกำเนิดสัญญาณนาฬิกาสำหรับวงจร USB จะต้องคำนวณค่า M ที่นำไปคูณกับความถี่ของสัญญาณนาฬิกาให้ได้ความถี่ 48 MHz จาก แผงวงจร CP-JR ARM7 USB-LPC2148 EXP จะคำนวณได้ M = 48 MHz / f_{osc} = 48 MHz / 12 MHz =4

การหาค่าของ P ให้ใช้ตามค่าที่ได้จากกรณีของ PLL0 คือให้ค่า P = 2 ได้ fcco =48 MHz * 2 *2 = 192 MHz ซึ่งตรงตามเงื่อนไขในการกำหนดค่าการทำงานของวงจร PLL จะสั่งผ่านทางรีจิสเตอร์ ที่เกี่ยวข้องกับ PLL ดังแสดงในตารางที่ 2.2.1 (ภาคผนวก)

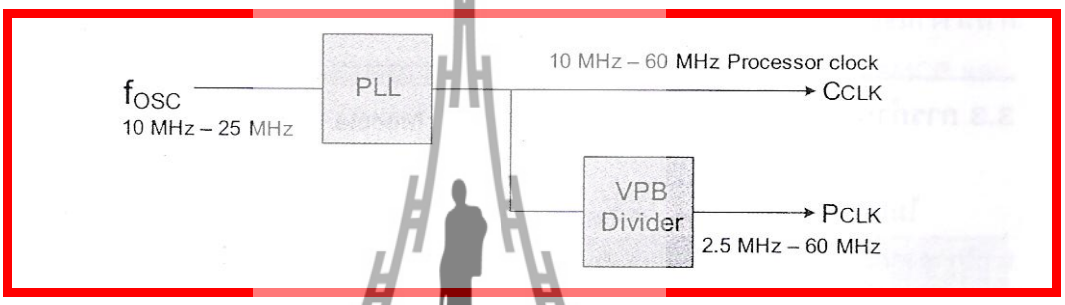
การกำหนดค่าของ P, M กำหนดที่รีจิสเตอร์ PLLCFG ซึ่งแสดงรายละเอียดแต่ละบิตของรีจิสเตอร์ PLLCFG ได้ดังตาราง 2.2.2 (ภาคผนวก)

เมื่อกำหนดค่าตัวหาร P และตัวคูณ M แล้วคำสั่งให้วงจร PLL ทำงานและต่อ PLL เป็นสัญญาณนาฬิกาหลักได้ที่รีจิสเตอร์ PLLCON ซึ่งมีค่าดังตารางที่ 2.2.3 (ภาคผนวก) ค่าแต่ละบิตของรีจิสเตอร์ PLLSTAT เพื่อดูสถานะของ PLL แสดงได้ในตารางที่ 2.2.4 (ภาคผนวก)

เราสามารถกำหนดโหมดการทำงานของวงจร PLL ได้จากบิต PLLC และ PLLE ของรีจิสเตอร์ PLLCON ได้ดังตารางที่ 2.2.5 (ภาคผนวก)

การกำหนดค่าความถี่ให้กับ VLSI Peripheral Bus

เมื่อกำหนดให้กับวงจร PLL ภายในไมโครคอนโทรลเลอร์ ARM7 ทำงานสร้างความถี่สูง และต่อให้เป็นสัญญาณนาฬิกาของโปรเซลเซอร์แล้ว อุปกรณ์ประกอบต่างๆ เช่น UART, GPIO, I^2C ที่ต่อกับบัส VPB จะทำงานไม่ทัน ดังนั้นภายในไมโครคอนโทรลเลอร์ ARM7 จะมีวงจรหารลดความถี่ของ CCLK ให้เป็นความถี่ของ VPB Bus ที่เรียกว่า PCLK โดยบล็อกไดอะแกรมภายในแสดงดังรูปที่ 2.10



รูปที่ 2.10 Block Diagram วงจรกำเนิดสัญญาณภายใน ARM7

รีจิสเตอร์ที่เกี่ยวข้องคือ VPBDIV ที่อยู่แอดเดรส 0xE10FC100 โดยสามารถอ่านและเขียนได้

ค่าของรีจิสเตอร์ VPBDIV จะมีค่า 2 บิตสุดท้าย โดยกำหนดได้ดังตารางที่ 2.2.6 และตารางที่ 2.2.7 (ภาคผนวก)

การกำหนดค่าให้กับ Memory Accelerator Module (MAM)

การกำหนดค่าการทำงาน MAM มีรีจิสเตอร์ที่เกี่ยวข้องสองตัว คือ MAMCR, MAMTIM ดังแสดงในตารางที่ 2.2.8 (ภาคผนวก)

การกำหนดค่าโหมดการทำงานของ MAM ผ่านทาง MAMCR ได้ดังตาราง 2.2.9 (ภาคผนวก)

การกำหนดค่ารีจิสเตอร์ MAMTIM กำหนดได้ดังตารางที่ 2.2.10 (ภาคผนวก)

ในการกำหนดค่าเริ่มต้นการทำงานของ MAM เริ่มต้นด้วยการหยุดทำงานของ MAM ด้วยการโหลดค่า 0 ให้กับ MAMCR แล้วจึงกำหนดค่าของ MAMTIM ตามที่ต้องการแล้วจึงเปิดการทำงานของ MAM ด้วยการเขียนค่า 1 หรือ 2 ให้กับ MAMCR

ถ้าซีพียูมีความถี่ CCLK น้อยกว่า 20 MHz ให้ใช้ MAMTIM = 1*CCLK ถ้า CCLK มีค่าระหว่าง 20 –40 MHz ให้กำหนด MAMTIM = 2*CCLK ถ้าความถี่มากกว่า 40 MHz ให้กำหนดค่า MAMTIM = 3*CCLK

2.2.4. การต่อ GPIO เป็นเอาต์พุต

ภายในไมโครคอนโทรลเลอร์ มีพอร์ตอเนกประสงค์ ขนาด 32 บิต ให้ใช้งาน 2 พอร์ต คือ Port 0 , Port1 โดยให้ Port0 มีขาต่อใช้งานได้ 29 ขา คือ P0.0 – P0.31 โดยไม่มีขา P0.24, P0.26, P0.27 ให้ใช้งาน และขา P0.31 ทำหน้าที่เป็นเอาต์พุตได้อย่างเดียว ส่วน Port1 จะมีต่อให้ใช้งานแค่ 16 ขา คือ P1.16-P1.31 โดยแต่ละขาของพอร์ต P0 และ P1 มีหน้าที่การทำงานได้หลายประเภท ซึ่งต้องกำหนดรีจิสเตอร์ PINSEL

การกำหนดหน้าที่การทำงานของพอร์ต

หลังจากเกิดการรีเซตไมโครคอนโทรลเลอร์ ARM7 วงจรภายในสั่งรีเซตค่ารีจิสเตอร์ PINSEL ทุกตัวกำหนดค่าให้ Port0 , Port1 ทุกขาเป็น GPIO และให้ทุกขาเป็นอินพุต

ขาแต่ละขาของ Port0,Port1 มีหน้าที่การทำงานได้หลายหน้าที่ โดยกำหนดการทำงานของ Port ที่รีจิสเตอร์ PINSEL0, PINSEL1 และกำหนดหน้าที่การทำงานของ Port1 ที่รีจิสเตอร์ PINSEL2 โดยรีจิสเตอร์แต่ละตัวที่มีแอดเดรสแสดงในตารางที่ 2.2.11 (ภาคผนวก)

ตัวอย่างการกำหนดค่าขาให้ ขา P0.16-P0.31 มีการทำงานเป็น GPIO ทำโดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL 1 ดังนี้

```
PINSEL1 = 0x00000000;  //set P0.16-P0.31  to  GPIO  Function
```

ค่าแต่ละบิตของริจิสเตอร์ PINSEL2 ที่ใช้กำหนดหน้าที่การทำงาน Port P1 แสดงได้ในตารางที่ 2.2.14 (ภาคผนวก) ตัวอย่างการกำหนดค่าให้ขา P1.16-P1.31 มีการทำงานเป็น GPIO ทำการเขียนค่า 0 ทุกบิตให้กับ PINSEL1

```
PINSEL2 = 0x000000000;    //set P1.16-P1.31  to GPIO Function
```

การกำหนดค่าควบคุมการทำงานพอร์ตอเนกประสงค์ (GPIO)

หลังจากที่กำหนดค่าให้พอร์ตแต่ละตัวมีการทำงานเป็น GPIO แล้ว ควบคุมการทำงานของ GPIO จะสั่งงานผ่านทางรีจีสเตอร์ 4 ตัวได้แก่ IOPN , IOSET, IOCKR, IODIRโดยที่รีจิสเตอร์แต่ละตัวจะมีแอดเดรสแสดงในตารางที่ 2.2.15 (ภาคผนวก)

ในไมโครคอนโทรลเลอร์ LPC2148 มีพอร์ตอเนกประสงค์ที่ให้ทำงานได้เร็วขึ้นจะเรียกว่าเป็น Fast GPIO โดยให้กำหนดรีจิสเตอร์เพิ่มเติมขึ้นอีก แต่การทำงานที่รวดเร็วขึ้นจะต้องเขียนโปรแกรมเป็นภาษาแอสแซมบลีเท่านั้น

การสั่งงานของ GPIOเริ่มต้นด้วยการกำหนดทิศทางของพอร์ตก่อนว่าจะให้เป็นอินพุตหรือเอาท์พุต โดยกำหนดค่ารีสเตอร์ IODIR โดยค่าบิตใดเป็น 0 คือให้ขาของบิตนั้นเป็นอินพุต ถ้าให้ค่าเป็น 1 ขาของบิตนั้นจะเป็นเอาต์พุต

ตัวอย่างเช่น ต้องการให้ขา P0.22, P0.20, P0.19, และ P0.16 เป็นเอาท์พุต โดยขาของ P0 ที่เหลือเป็นอินพุต จะต้องกำหนดค่าให้กับรีจิสเตอร์ IODIR0 แต่ละบิตดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ค่า	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1
	0			0			5			9						

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0			0			0			0						

เขียนเป็นโปรแกรมภาษาซีได้ดังนี้

IODIR 0 = 0X00590000;

หลังจากการกำหนดให้พอร์ตเป็นเอาต์พุตแล้ว ถ้าต้องการสั่งให้พอร์ตที่เป็นเอาต์พุตนี้มีค่าเป็น 1 ต้องสั่งที่รีสเตอร์ IOSET จากตัวอย่างต้องสั่งให้ขา P0.16 และ P.19 เป็น 1 จะต้องกำหนดให้ค่า รีจิสเตอร์ IOSET0 ดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	0			0			0			9						

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0				0				0				0			

เขียนเป็นโปรแกรมภาษาซีได้ดังนี้

IOSET 0 = 0X00090000;

การเขียนค่าเป็น 0 ให้รีจิสเตอร์ IOSET จะไม่มีผลต่อขาของพอร์ตที่ตรงกับบิตนั้น ขาของพอร์ตจะมีค่าคงเดิม ถ้าต้องการส่งให้พอร์ตที่เป็นเอาต์พุตนี้มีค่าเป็น 0 ต้องสั่งที่รีจิสเตอร์ IOCLR ด้วยจากตัวอย่าง ต้องการสั่งให้ขา P0.22 และ P0.20 เป็น 0 จะต้องกำหนดค่าให้กับรีจิสเตอร์ IOCLR0 ดังนี้

บิตที่	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ค่า	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
	0				0				5				0			

บิตที่	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ค่า	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0				0				0				0			

เขียนโดยใช้โปรแกรมภาษาซีได้ดังนี้

IOCLR = 0X00500000;

ข้อควรระวัง ในการสั่งให้ขาพอร์ตมีค่าเป็น 0 นี้ไม่ได้สั่งให้เขียนค่า 0 ให้กับรีจิสเตอร์ IOCLR ต้องเขียนค่าเป็น 1 เท่านั้น

ในการเขียนค่าให้พอร์ต จะต้องแยกข้อมูล ถ้าบิตใดเป็น 0 จะต้องเขียนสั่งที่รีจิสเตอร์ IOCLR ถ้าบิตใดเป็น 1 จะต้องเขียนสั่งที่รีจิสเตอร์ IOSET

2.2.5 อินเตอร์รัปต์ไมโครคอนโทรลเลอร์ ARM7

ในไมโครคอนโทรลเลอร์ ARM7 มีโมดูล Vectored Interrupt Controller (VIC) เป็นตัวควบคุมกลไกลของการอินเตอร์รัปต์ โดยสามารถรับอินเตอร์รัปต์จากอุปกรณ์ทั้งภายในและภายนอกไมโครคอนโทรลเลอร์ ARM7 ได้ทั้งหมด 32 อินพุตตามที่แสดงในตารางที่ 2.2.16 (ภาคผนวก) VIC จะนำอินพุตจากการอินเตอร์รัปต์ทั้ง 32 แหล่งมาจัดแบ่งตามประเภทได้เป็น 3 ประเภท ทำให้สามารถปรับลำดับความสำคัญของอินเตอร์รัปต์จากอุปกรณ์ประกอบต่างๆได้ตามต้องการ คือ

- Fast Interrupt request (FIQ) จะลำดับความสำคัญสูงสุดโดยรีบตอบสนองเร็วสุด
- Vectored IRQs จะมีการลำดับความสำคัญอยู่กึ่งกลาง โดยสามารถนำอินเตอร์รัปต์แค่ 16 บิต หรือ 32 แหล่งมาจัดให้เป็น Vectored IRQs ได้ 16 ตัว โดย Slot 0 จะมีความสำคัญสูงสุด Slot 15 มีความสำคัญต่ำสุด
- Non-vectored IRQ มีความสำคัญต่ำสุด

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์มีทั้งหมด 43 ตัว ดังแสดงในตารางที่ 2.2.17 (ภาคผนวก) โดยในหัวข้อนี้จะเป็นการเลือกเฉพาะรีจิสเตอร์ที่เกี่ยวกับการอินเตอร์รัปต์ในแบบ Vector IRQ คือรีจิสเตอร์ VICVectAddr0-15 , VICVectCntl0-15 และ VICIntEanble

รีจิสเตอร์ VICVectAddr0-15 เป็นรีจิสเตอร์ที่เก็บค่าแอดเดรสของโปรแกรมที่ตอบสนองต่ออินเตอร์รัปต์โดย VICVectAddr0 จะเป็นอินเตอร์รัปต์ Slot 0 ที่มีความสำคัญสูงสุดตามลำดับไปจนถึง VICVectAddr15 จะเป็นอินเตอร์รัปต์ Slot 15 ที่มีความสำคัญต่ำสุด

VICVectCntl0 - 15 ใช้ในการควบคุม คือ Slot หมายเลขที่ตรงกับ VICVectCntl นี้จะรีบอินเตอร์รัปต์จากแหล่งใด จากทั้งหมด 32 แหล่ง ตามตารางที่ 2.2.17(ภาคผนวก) โดยกำหนดค่าในบิตที่ 4:0 โดยบิตที่ 5 ถ้าเป็น 1 หมายถึง อนุญาตให้อินเตอร์รัปต์จากอุปกรณ์ที่กำหนดใน Slot นี้

VICInEnable ใช้เปิดอินเตอร์รัปต์จากแหล่งต่างๆ ทั้ง 32 แหล่งตามตารางที่ 2.2.17 (ภาคผนวก) โดยบิตใดมีค่าเป็น 1 หมายถึงอนุญาตให้อินเตอร์รัปต์ได้ ถ้าบิตใดเป็น 0 ไม่อนุญาตให้อุปกรณ์นั้นๆ อินเตอร์รัปต์

การอินเตอร์รัปต์จากอินพุตภายนอกของไมโครคอนโทรลเลอร์ ARM7

ในไมโครคอนโทรลเลอร์ LPC2148 จะรับอินเตอร์รัปต์จากภายนอกได้สูงสุด 4 แหล่ง คือ EINT0, ENIT1, ENIT2 และ ENIT3 โดยคิดเป็นอินพุตได้ทั้งหมด 9 ตัว

- EINT0 อยู่ที่ขา P0.1 และ P0.16
- EINT1 อยู่ที่ขา P0.3 และ P0.14
- EINT2 อยู่ที่ขา P0.7 และ P0.15
- EINT3 อยู่ที่ขา P0.9, P0.20 และ P0.30

โดยสามารถนำอินพุตจากการอินเตอร์รัปต์ทั้งสี่ตัวนี้ มาใช้ในการกระตุ้นให้ไมโครคอนโทรลเลอร์ออกจากการทำงานในโหมด Power Down ได้

รีจิสเตอร์ที่เกี่ยวข้องกับอินเตอร์รัปต์จากภายนอก

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์จากภายนอกมีทั้งหมด 4 ตัว คือ EXTINT , EXTWAKE, EXXTMODE และ EXTPOLAR ดังแสดงในตารางที่ 2.2.18 (ภาคผนวก)

เมื่อเกิดการอินเตอร์รัปต์จากภายนอกเกิดขึ้น จะทำให้ค่าบิตบางบิตของรีจิสเตอร์ EXTINT มีค่าเป็น 1 และส่งต่อการอินเตอร์รัปต์ไปยัง VIC เพื่อสร้างอินเตอร์รัปต์ขัดการทำงานของไมโครคอนโทรลเลอร์ เมื่อโปรแกรมตอบสนองต่อการอินเตอร์รัปต์ ทำงานเสร็จแล้วจะต้องเขียนค่า 1 ให้กับบิตเหล่านี้เพื่อหยุดการอินเตอร์รัปต์ ค่าแต่ละบิตของรีจิสเตอร์ EXTINT แสดงในตารางที่ 2.2.19 (ภาคผนวก)

ในการกำหนดว่าสัญญาณที่อินเตอร์รัปต์ทำงานที่ระดับสัญญาณหรือที่ขาขอบของสัญญาณกำหนดได้ที่รีจิสเตอร์ EXTMODE ซึ่งมีค่าดังตารางที่ 2.2.20 (ภาคผนวก)

หลังจากที่กำหนดโหมดการทำงานของสัญญาณที่ใช้อินเตอร์รัปต์ว่าเป็นระดับของสัญญาณหรือขอบสัญญาณแล้วที่รีจิสเตอร์ EXTMODE แล้วต้องกำหนดว่าสัญญาณที่อินเตอร์รัปต์นี้ทำงานที่ระดับลอจิก 1 หรือ 0 หรือทำงานที่ขอบขาขึ้นหรือขาลงของสัญญาณ โดยกำหนดที่รีจิสเตอร์ EXTPOLAR ซึ่งแสดงรายละเอียดดังตารางที่ 2.2.21 (ภาคผนวก)

ถ้าต้องการให้สัญญาณที่ขา EINT0-EINT3 สามารถกระตุ้นให้ไมโครคอนโทรลเลอร์ ARM7 ออกจากการทำงานในโหมด Power Down จะต้องสั่งที่รีจิสเตอร์ EXTWAKE ซึ่งมีค่าบิตของรีจิสเตอร์แสดงในตารางที่ 2.2.22 (ภาคผนวก)

2.2.6 การกำหนดค่าเริ่มต้นสำหรับ UART0

UART0 มีขาสัญญาณสองขา คือ

1. ขาสัญญาณ TxD0 ไว้สำหรับส่งข้อมูลอยู่ที่ขา
2. ขาสัญญาณ RxD0 ไว้สำหรับรับข้อมูลอยู่ที่ขา P0.1

การใช้งาน UART0 เริ่มต้นด้วยการเขียนค่ายังรีจิสเตอร์ PINSEL0 เพื่อกำหนดให้ขา P0.0 และ P0.1 มีการทำงานเป็น UART0 โดยต้องกำหนดบิตที่ 3-0 ของ PINSEL0 ให้มีค่าเป็น 1010 ซึ่งเขียนเป็นคำสั่งได้ดังนี้

PINSEL0 = 0x00000005;

รีจิสเตอร์ที่เกี่ยวข้องกับ UART0 มีทั้งหมด10 ตัวโดยแต่ละตัวมีขนาด 8 บิต ดังแสดงในตารางที่ 2.2.23 (ภาคผนวก)

หลังจากที่กำหนดให้ขา P0.0 และ P0.1 ทำงานเป็น UART0 แล้วถัดมาเป็นการกำหนดรูปแบบการติดต่อเช่น ติดต่อแบบ 8 ใช้การตรวจสอบบิตพลาดแบบใด เช่น even parity จำนวนของ Stop bit โดยการกำหนดค่าผ่านทางรีจิสเตอร์ UART Line Control Register : LCR : ซึ่งมีรายละเอียดของรีจิสเตอร์ดังแสดงในตารางที่ 2.2.24 (ภาคผนวก)

ในรีจิสเตอร์ LCR มีบิตที่เรียกว่า DLAB (Divisor Latch Access bit) ถ้าต้องการปรับค่าของวงจรกำเนิด Broad Rate ต้องเซตบิตนี้ให้เป็น 1 ค่าของ Baud rate generator เป็นค่าของตัวหารขนาด 16 บิต เพื่อนำไปใช้หารค่าของ PCLK เพื่อให้ได้ความถี่สูงกว่าความเร็ว Broad Rate 16 เท่า ทำให้สมการคำตัวหารเป็นดังนี้

Divisor = PCLK/(16*Baud)

แผงวงจร CP-JR ARM7 USB-LPC2148 EXP ใช้คริสตัลความถี่ 12.000 MHz สั่ง PLL คูณ 5 จะได้ CCLK = 60.0 MHz และกำหนดให้ VPBDIV = 2 จะได้ PCLK = 30.0 MHz ด้วย ถ้าต้องการอัตราบอดที่ 9600 bps ตัวหารจะมีค่าเท่ากับ

Divisor = 30,000,000/(60*9600) = 195.3125 ปัดเศษลง

Divisor = 195 หรือ 0xC3 นำค่าที่ได้ไปคำนวณหาอัตราบอดจะได้

Baud = PCLK/(16*Divisor) = 30,000,000/(16*195) = 9615 bps

ซึ่งผิดพลาดไป 0.156% สามารถให้งานได้ เนื่องมาตรฐานของการสื่อสารแบบอนุกรมสามารถรับอัตราบอดที่ผิดพลาดได้ถึง 5%

นำค่าหารที่ได้ไปเก็บลงในรีจิสเตอร์ขนาด 8 บิตสองตัว คือ Divisor Latch MAB(DLM) และ Divisor Latch LSB (DLL) ในขณะที่เขียนค่าเก็บในรีจิสเตอร์ DLM และ

DLL ค่าบิต DLAB ต้องมีค่าเป็น 1 เมื่อเขียนเสร็จแล้วต้องรีเซตค่าบิตนี้ให้กลับเป็น 0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
U0DLM = 0x00;
U0DLL = 0xC3;
U0LCR = 0x7F;
```

ในการเรียกใช้ฟังก์ชัน uart0_init() จะต้องส่งค่าอัตราบอดที่ต้องการให้ฟังก์ชัน ตัวอย่างเช่นต้องการอัตราบอดที่ 9600 bps จะต้องเรียกใช้ฟังก์ชันดังนี้

```
uart0_init(9600);
```

เมื่อกำหนดการทำงานให้กับ UART แล้ว จะสามารถรับส่งค่าผ่านพอร์ตอนุกรมได้ในการส่งข้อมูลต้องเขียนข้อมูลไปยังรีจิสเตอร์ Transmit Holding Register (THR) ถ้าต้องการอ่านข้อมูลที่รับพอร์ตอนุกรมต้องอ่านค่าจากรีจิสเตอร์ Receiver Buffer Register (RBR) จากตารางที่ 2.2.24 (ภาคผนวก) จะพบว่าค่าแอดเดรสของรีจิสเตอร์ทั้งสอง มีค่าอยู่ที่ตำแหน่งเดียวกัน แสดงว่าการเขียนค่าให้กับ THR เป็นการเขียนค่าลงในบัฟเฟอร์แบบ FIFO ของ UART0 การอ่านค่าจากรีจิสเตอร์ RBR เป็นการอ่านค่าจากบัฟเฟอร์แบบ FIFO

ก่อนที่จะอ่านหรือเขียนค่าลงในรีจิสเตอร์ THR หรือ RBR จะต้องอ่านค่าสถานะของ UART ก่อนว่ามีการผิดพลาดหรือไม่ โดยอ่านค่าสถานะที่รีจิสเตอร์ Line Status Register (LSB) ก่อน โดยค่าประจำบิตของรีจิสเตอร์แสดงได้ในตารางที่ 2.2.25 (ภาคผนวก)

ก่อนที่จะเขียนข้อมูลให้ UART ต้องตรวจสอบบิตที่บิต Transmitter Empty (TEMT) ของรีจิสเตอร์ LSR ก่อนว่าบัฟเฟอร์สำหรับส่งว่างหรือไม่ ถ้าว่างจะได้ค่าเป็น 1 จึงส่งข้อมูลได้ ก่อนที่จะอ่านข้อมูลจาก UART ต้องตรวจสอบบิต Receiver Data Ready (RDR) ก่อน ถ้ามีข้อมูลพร้อมแล้วบิตนี้จะมีค่าเป็น 1 จึงอ่านค่าจาก UART ได้

สามารถเขียนเป็นฟังก์ชัน putchar(); สำหรับเขียนข้อมูลจำนวน 1 ไบต์ให้กับ UART และเขียนเป็นฟังก์ชัน getchar() สำหรับอ่านค่าจาก UART ได้

สำหรับการเขียนข้อมูลออกพอร์ตอนุกรม ในโปรแกรม Keil uVision3 ได้จัดเตรียมฟังก์ชัน printf(); ไว้ให้แล้ว และถ้าต้องการรับข้อมูลจากพอร์ตอนุกรม สามารถใช้ฟังก์ชัน scanf(); โดยต้องสั่ง #include<stdio.h>

2.3 GSM Module

2.3.1 GSM Module (wave com)

GSM Module (wave com) เป็นชุดเรียนรู้และพัฒนาระบบการสื่อสารไร้สาย โดยใช้ โมดูลGSM/GPRS รุ่น Fastrack Supreme 20 ของ “Wavecome Ltd.” เป็นอุปกรณ์หลัก ซึ่ง Fastrack Supreme 20 เป็นโมดูลสื่อสารระบบ GSM/GPRS ขนาดเล็ก รองรับระบบสื่อสาร GSM ความถี่ 900/1800/1900MHz โดยสั่งงานผ่านทางพอร์ตสื่อสารอนุกรม RS232 ด้วยชุดคำสั่ง AT Command สามารถประยุกต์ใช้งานได้มากมายหลายรูปแบบ ไม่ว่าจะเป็นการรับส่งสัญญาณแบบ Voice , SMS , Data , FAX และยังรวมถึงการสื่อสารด้วย Protocol TCP/IP ด้วย ซึ่งตามปกติแล้ว ถึงแม้ว่าโมดูล Fastrack Supreme 20 จะมีวงจรและ Firmware บรรจุไว้ภายในตัวเป็นที่เรียบร้อยแล้วก็ยังไม่สามารถนำไปใช้งานได้โดยตรงทันที เนื่องจากในการใช้งานจริงๆนั้น ผู้ใช้งานเองจำเป็นต้องออกแบบวงจรรอบนอกที่จำเป็นมาเชื่อมต่อกับขาสัญญาณของตัวโมดูลอีกในบางส่วน ไม่ว่าจะเป็นวงจรภาค Power Supply,วงจรเชื่อมต่อกับ Sim Card รวมไปถึงวงจร Line Driver ของ RS232 เป็นต้น ดังนั้นจึงได้จัดสร้างบอร์ดสำหรับเป็นตัวกลางในการเชื่อมต่อระหว่างโมดูล Fastrack Supreme 20 กับอุปกรณ์ภายนอกเพื่อให้ผู้ใช้งานสามารถนำโมดูล GSM ของ Fastrack Supreme 20 ไปทำการทดลองและศึกษาเรียนรู้การสั่งงานต่างๆได้โดยสะดวก ก่อนที่จะนำเอาโมดูลตัวนี้ไปออกแบบดัดแปลงและประยุกต์ใช้งานในด้านต่างๆได้ต่อไปในอนาคต ซึ่งถึงแม้ว่าวงจรการเชื่อมต่อทั้งหมดที่ทาง wavecome ได้จัดทำขึ้นมานี้จะยังไม่สามารถรองรับการใช้งานทรัพยากรต่างๆที่มีอยู่ภายในโมดูลได้ครบถ้วนทั้งหมดก็ตามที แต่ในส่วนของการใช้งานโมดูลในส่วนที่เป็นความสามารถหลักๆที่จำเป็นนั้น มีไว้รองรับอย่างครบถ้วนเพียงพอแล้ว

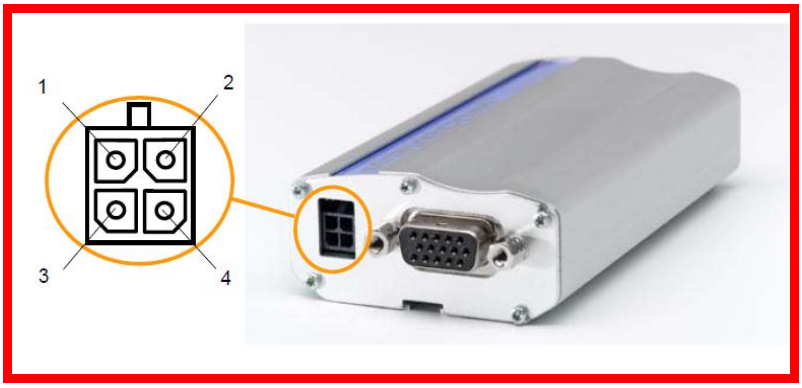
อย่างไรก็ตามถ้าผู้ใช้งาน ต้องการพัฒนา Application ที่สูงขึ้นไป ก็สามารถประยุกต์ดัดแปลงหรือทำการเชื่อมต่ออุปกรณ์เพิ่มเติมให้กับบอร์ดได้โดยง่าย ทั้งนี้ก็เพราะว่าขาสัญญาณต่างๆจากโมดูล ในส่วนที่ยังไม่ได้ทำการออกแบบวงจรเตรียมไว้ให้ภายในบอร์ด

2.3.2 ส่วนประกอบของโมดูล

รูปที่ 2.11 Connetor ของ GSM Module

รูปที่ 2.12 ช่องใส่ SIM ของ GSM Module

Power supply connector



รูปที่ 2.13 Power supply connector

ตารางที่ 2.1 Power supply connector pin description

Pin #	Signal	I/O	I/O type	Description	Reset State	Comment
1	V+BATTERY	I	Power supply	Battery voltage input: - 5.5 V Min. - 13.2 V Typ. - 32 V Max.		High current
2	GND		Power supply	Ground		
3	GPIO21	I/O	2.8 V	General Purpose Input/output	Undefined	Not mux
4	GPIO25	I/O	2.8 V	General Purpose Input/output	Z	Multiplex with INT1

2.3.3 การใช้งาน AT Command เพื่อสั่งงานโมดูล Fastrack Supreme 20

โมดูล GSM/GPRS รุ่น Fastrack Supreme 20 ถูกออกแบบให้ทำหน้าที่เหมือน Modem โดยจะใช้การติดต่อสั่งงานและสื่อสารกับโมดูล ผ่านทางพอร์ตสื่อสาร RS232 รองรับ Baudrate ตั้งแต่ 9600 BPS โดยใช้ชุดคำสั่งแบบ AT Command ซึ่งจะมีรูปแบบการใช้งานเหมือนกับ Modem มาตรฐานทั่วๆไปเพียงแต่จะมีการเพิ่มเติม Option และคำสั่งพิเศษอื่นๆเพิ่มเติมขึ้นมาอีก เพื่อให้เหมาะสมและสอดคล้องกับความสามารถในการทำงานของโมดูลได้อย่างครบถ้วน โดยรูปแบบของคำสั่งต่างๆที่เป็น AT Command นั้น จะเริ่มต้นคำสั่งด้วยรหัส ASCII ของตัวอักษร 2 ตัว คือ “A” และ “T” ซึ่งจะใช้ ตัวอักษรแบบพิมพ์เล็ก หรือ พิมพ์ใหญ่ก็ได้ มีความหมายเหมือนกัน จากนั้นก็จะตามด้วยรหัสคำสั่งและ Option ต่างๆของคำสั่ง(ถ้ามี) โดยทุกๆคำสั่งจะต้องจบด้วยรหัส Enter หรือ 0DH (13) เสมอ เช่นคำสั่ง รีเซ็ตจะใช้รูปแบบคำสั่งเป็น “ATZ” หรือ “atz” ก็สามารถใช้งานได้ถูกต้องเหมือนกัน โดยรูปแบบคำสั่งทั้งหมดจะแบ่งออกเป็น 4 แบบด้วยกัน คือ

ตารางที่ 2.2 แสดงรูปแบบการใช้งาน AT Command (เมื่อ <x> คือ รหัสคำสั่ง)

การใช้งาน	รูปแบบคำสั่ง	รายละเอียด
ทดสอบคำสั่ง	AT+<x>=?	รูปแบบการใช้คำสั่งแบบนี้ จะใช้สำหรับสั่งอ่านค่ารูปแบบและพารามิเตอร์ต่างๆของคำสั่ง โดยถ้าคำสั่งนั้นมีอยู่จริง โมดูลจะตอบรับด้วยการพิมพ์ค่าของพารามิเตอร์ต่างๆของคำสั่งที่มีอยู่ทั้งหมดให้ทราบ
อ่านค่าพารามิเตอร์	AT+<x>?	รูปแบบการใช้คำสั่งแบบนี้ จะใช้สำหรับสั่งอ่านค่าพารามิเตอร์ที่กำหนดไว้แล้วของคำสั่งนั้นๆ โดยโมดูลจะตอบรับด้วยการพิมพ์ค่าพารามิเตอร์ปัจจุบันที่กำหนดไว้แล้วให้ทราบ
กำหนดค่าการทำงาน	AT+<x>=<…>	รูปแบบการใช้คำสั่งแบบนี้ จะใช้สำหรับสั่งเขียนหรือกำหนดค่าพารามิเตอร์ให้กับคำสั่ง เช่น การกำหนดค่า Baudrate
สั่งให้ทำงาน	AT+<x>	รูปแบบการใช้คำสั่งแบบนี้ จะใช้สำหรับสั่งงานให้โมดูลปฏิบัติตามคำสั่งที่ต้องการ เช่น การสั่งรีเซ็ต (ATZ)

การทดสอบการสั่งงานโมดูล

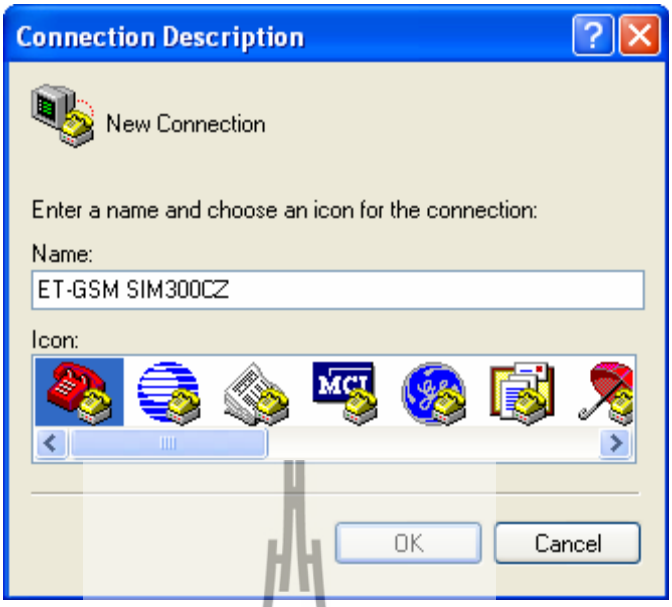
ดังได้ทราบแล้วว่าในการสั่งงานโมดูล Fastrack Supreme 20 นั้น จะใช้วิธีการส่งค่ารหัสคำสั่งในรูปแบบของ AT Command ผ่านทางพอร์ตสื่อสารอนุกรมไปให้กับโมดูล ซึ่งตามปกติจะต้องเขียนโปรแกรมเพื่อส่งรหัสคำสั่งต่าง ๆ ไปให้กับโมดูลเอง ขึ้นอยู่กับว่าจะใช้อุปกรณ์ใดเป็นตัวควบคุมการทำงานของโมดูล ซึ่งไม่ได้จำกัดว่าเป็นอุปกรณ์แบบใด อาจจะเป็นคอมพิวเตอร์ PC หรือ ไมโครคอนโทรลเลอร์ตระกูลใด ๆ ก็ได้ ขอให้มีพอร์ตสื่อสารอนุกรม RS232 อยู่ก็สามารถนำมาเชื่อมต่อเพื่อสั่งงานโมดูล Fastrack Supreme 20 ได้แล้ว ส่วนที่ว่าจะเขียนโปรแกรมอย่างไรและจะใช้ภาษาใดในการเขียนนั้น ขึ้นอยู่กับผู้พัฒนาโปรแกรมว่า มีความถนัดอย่างไรและมีพื้นฐานอะไรอยู่บ้าง ซึ่งหลักสำคัญก็คือ ผู้พัฒนาต้องหาคำตอบให้ได้ว่า การจะเขียนโปรแกรมสั่งงานอุปกรณ์ทำการ ส่ง และ รับข้อมูล ด้วยพอร์ตสื่อสารอนุกรม RS232 นั้นจะต้องทำอย่างไร ซึ่งจะไม่ขอกล่าวถึงในที่นี้ด้วย สำหรับในการศึกษาเบื้องต้นนั้น ยังไม่จำเป็นต้องใช้วิธีการเขียนโปรแกรมก็ได้ แต่สามารถใช้โปรแกรมสำเร็จรูปจำพวก Serial Terminal ต่าง ๆ ของคอมพิวเตอร์เป็นตัว

ทดสอบการทำงานเพื่อทำความ เข้าใจกับรูปแบบคำสั่งและผลของการทำงานต่างๆให้เข้าใจเสียก่อน ตัวอย่างเช่น ถ้าต้องการจะสั่งให้โมดูล Fastrack Supreme 20 โทรออกไปยังโทรศัพท์มือถือหมายเลข 0811234567 นั้น ในอันดับแรกจะต้องศึกษารูปแบบการทำงานของคำสั่งให้เข้าใจเสียก่อน จนสามารถเข้าใจแล้วว่าจะต้องใช้คำสั่ง “ATD0811234567;” เพื่อสั่งให้โทรออก จากนั้นจึงค่อยปรับเปลี่ยนไปเป็นการเขียนโปรแกรมในภายหลัง ซึ่งผู้ใช้ก็จะต้องไปศึกษาหาคำตอบต่อไปอีกว่าการที่จะเขียนโปรแกรมเพื่อสั่งให้อุปกรณ์ส่งค่ารหัส “ATD0811234567;” ออกไปทางพอร์ตสื่อสารอนุกรมนั้นต้องทำอย่างไรบ้าง ซึ่งในที่นี้จะขอแนะนำให้ใช้โปรแกรม Hyper Terminal ของ Windows เป็นเครื่องมือในการทดลองในเบื้องต้นไปก่อน โดย Hyper Terminal เป็นโปรแกรม Terminal สำเร็จรูป ซึ่งแถมมาพร้อมกับระบบปฏิบัติการ Windows อยู่แล้ว โดยความสามารถของโปรแกรมตัวนี้จะมีอยู่มากมายหลายส่วน ซึ่งในที่นี้เราจะใช้ประโยชน์เฉพาะในส่วนของการทำหน้าที่เป็น Serial Terminal ในText Mode เท่านั้น โดยหลังจากสั่ง Run โปรแกรมแล้ว ข้อมูลใดๆที่รับได้จากสัญญาณด้านรับ (RXD) ของพอร์ตสื่อสารอนุกรม ในย่านที่เป็นรหัส ASCII Code (20H..FFH) จะถูกนำมาแปลงเป็นตัวอักษรและแสดงผลที่หน้าจอของโปรแกรมให้เห็นทันที ส่วนรหัสของข้อมูลที่มีค่าต่ำกว่า 20H (00H-1FH) จะไม่ถูกนำมาแสดงผล แต่จะถือว่าเป็นคำสั่ง เช่น เมื่อได้รับ รหัส 0DH โปรแกรม Hyper Terminal จะถือว่าเป็นคำสั่งให้เลื่อน Cursor ของการแสดงผลไว้ในตำแหน่งเริ่มต้นของบรรทัด หรือเมื่อได้รับรหัส 0AH ก็จะทำการเลื่อน Cursor ของการแสดงผลให้ขึ้นบรรทัดใหม่แทนดังนี้เป็นต้น และ ในทางตรงกันข้าม เมื่อเราทำการกดคีย์ใด ๆ โปรแกรมก็จะแปลค่าการกดคีย์นั้นให้เป็นรหัส ASCII ของตัวอักษรของตำแหน่งคีย์นั้นๆส่งออกไป ยังขา TXD ของพอร์ตสื่อสารอนุกรมโดยอัตโนมัติ

โดยการใช้งานโปรแกรม สามารถทำได้ดังนี้

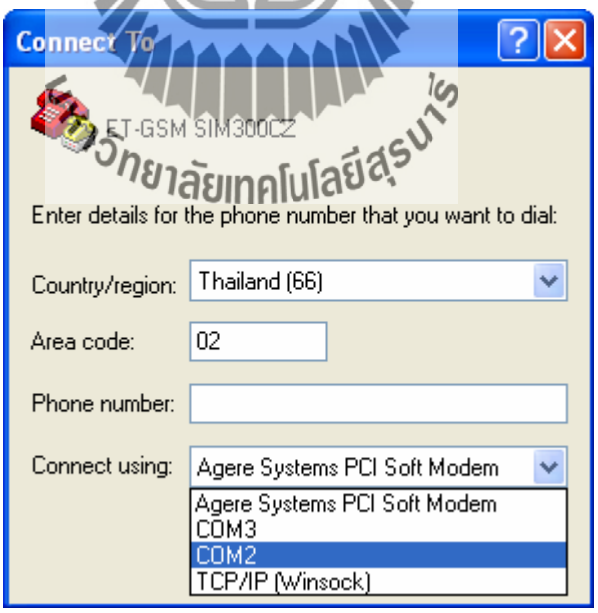
Start → Programs → Accessories → Communication → Hyper Terminal

ในขั้นตอนแรกจะปรากฏหน้าต่าง Connection Description ขึ้นมา ให้คลิกเมาส์เลือกรูปแบบของICON และกำหนดชื่อของการเชื่อมต่อตามต้องการ แล้วเลือก “OK” ดังตัวอย่าง



รูปที่ 2.14 หน้าต่างเริ่มต้นการทำงานของโปรแกรม Hyper terminal

ในขั้นตอนนี้ให้คลิกเมาส์ที่ Connect using แล้วเลื่อนเมาส์ไปยังหมายเลข Comport ที่ใช้ในการเชื่อมต่อกับบอร์ดตามจริง แล้วเลือก “OK” ดังตัวอย่าง

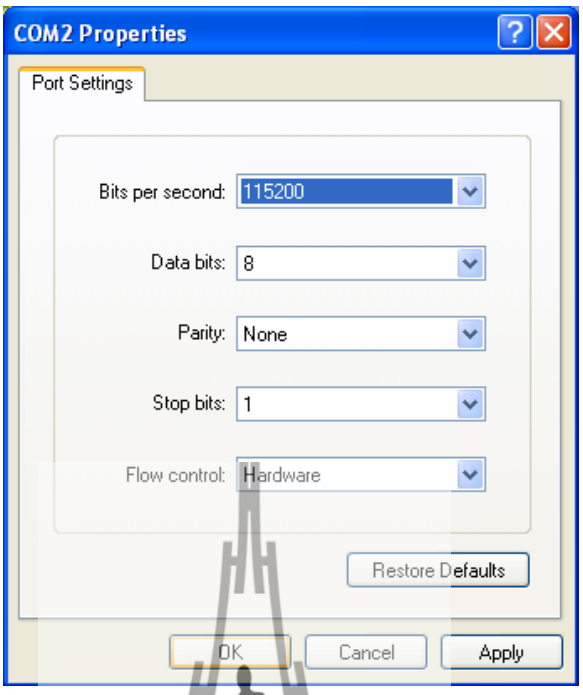


รูปที่ 2.15 หน้าต่างเริ่มต้นการทำงานของโปรแกรม Hyper terminal

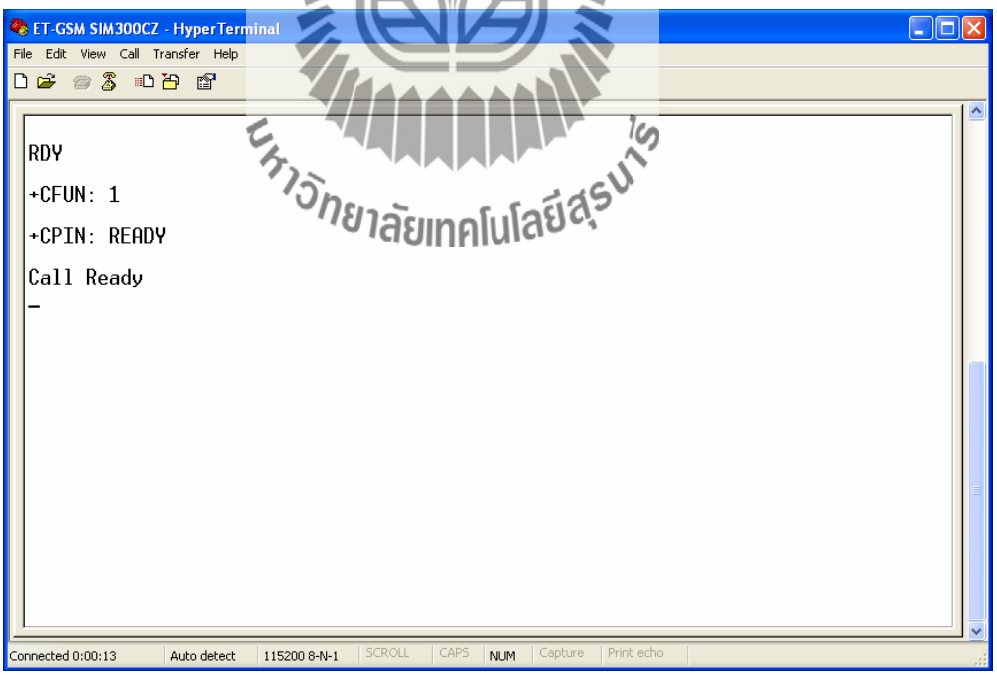
รูปที่ 2.16 การเลือก Com Port ใช้งานโปรแกรม Hyper Terminal

ในขั้นตอนนี้ให้เลือกค่า Baud rate ให้ตรงและสอดคล้องกับที่กำหนดให้กับโมดูลไว้ หรือในกรณีที่กำหนดค่า Baud rate ของโมดูลเป็น Auto-Baud rate ไว้ก็สามารถกำหนดค่าใดๆ ที่โมดูลสามารถรองรับได้ระหว่าง 300 , 1200 , 2400 , 4800 , 9600 , 14400 , 19200 , 28800 , 38400 , 57600 , 115200

ส่วน Data ให้เลือกเป็น 8 Bit, Parity =None, Stop bits=1, Flow Control = Hardware แล้วเลือก “OK” ดังตัวอย่าง



รูปที่ 2.17 ทำการตั้งค่าที่ COM2 ให้เป็นค่าเริ่มต้น



รูปที่ 2.18 แสดงลักษณะของหน้าจอ Hyper Terminal เมื่อโมดูลพร้อมทำงาน

2.3.4 การกำหนด Flow Control

โมดูล Fastrack Supreme 20 สามารถกำหนด Flow Control หรือ รูปแบบการตรวจสอบความพร้อมในการสื่อสารและรับส่งข้อมูลได้ด้วย ซึ่ง Flow Control จะมีความสำคัญเป็นอย่างมาก เนื่องจากการประมวลผลของอุปกรณ์ต่างๆจะมีความช้าเร็วที่แตกต่างกัน เมื่อมีการรับส่งข้อมูลที่มีจำนวนข้อมูลมากๆแบบต่อเนื่องนั้น ถ้าฝ่ายรับไม่พร้อมรับข้อมูลแต่ฝ่ายส่งยังคงส่งข้อมูลออกไป ก็จะทำให้ข้อมูลสูญหายและเกิดความผิดพลาดขึ้นได้ โดย Fastrack Supreme 20 เองรองรับการตรวจสอบความพร้อมหรือ Flow Control ได้ 2 แบบ คือ

- **Software Flow Control (XON/XOFF Flow Control)** เป็นการตรวจสอบความพร้อมด้วย Software โดยจะใช้รหัส XOF(13H) เป็นตัวสั่งหยุดการส่งข้อมูลจากฝ่ายส่งและใช้รหัส XON(11H) เพื่อบอกหรืออนุญาตให้ฝ่ายส่งเริ่มต้นส่งข้อมูลลำดับต่อไปมายังโมดูลได้ โดยการใช้ Flow Control แบบนี้เหมาะกับการเชื่อมต่อกับอุปกรณ์ที่ไม่มีสัญญาณตรวจสอบความพร้อม เช่น ไมโครคอนโทรลเลอร์หรืออุปกรณ์ที่ใช้การต่อสายสัญญาณเพียง 3 เส้น (RXD,TXD และ GND)
- **Hardware Flow Control (RTS/CTS Flow Control)** เป็นการตรวจสอบความพร้อมด้วยสัญญาณทางฮาร์ดแวร์ โดยใช้การ Active(“LOW”) สัญญาณ CTS เพื่อบอกให้ฝ่ายส่งหยุดการส่งข้อมูลเมื่อโมดูลไม่พร้อมรับข้อมูล และในทางกลับกันก่อนการส่งข้อมูลกลับออกไปมันจะตรวจสอบสถานะของ RTS ว่า Active อยู่หรือไม่ ถ้า Active(“LOW”) แสดงว่าฝ่ายรับยังไม่พร้อมรับมันจะหยุดรอจนกว่า RTS จะเป็น “HIGH”

2.3.5 การ Setup และตรวจสอบค่า Configuration

ตามปกติแล้วการทำงานของโมดูล Fastrack Supreme 20 นั้นจะสามารถกำหนดรูปแบบการทำงานได้มากมายหลายลักษณะ เช่น เงื่อนไขในการติดต่อสื่อสารกับโมดูล ผู้ใช้สามารถเปลี่ยนแปลงค่าต่างๆได้มากมาย ไม่ว่าจะเป็นค่า Baud rate หรือรูปแบบของการ Handshake ต่างๆ ที่จะใช้ในการสื่อสาร เป็นต้น ดังนั้นจึงจำเป็นต้องมีการกำหนดรูปแบบการทำงานของโมดูลให้ตรงกับความต้องการ ซึ่งตามปกติแล้ว เงื่อนไขต่างๆเหล่านี้จะมีค่าที่แน่นอนอยู่ค่าหนึ่งเสมอหลังการรีเซ็ต หรือ Power ON โดยโมดูลจะกำหนดค่าเงื่อนไขต่างๆให้กับตัวมันเองในตอนเริ่มต้นการทำงานด้วยค่าที่กำหนดไว้ใน Configuration ที่ถูกบันทึกไว้แล้ว แต่อย่างไรก็ตามผู้ใช้สามารถสั่งเปลี่ยนแปลงแก้ไขค่า Configuration ต่างๆได้เองตามต้องการ ซึ่งวิธีการกำหนดเงื่อนไขการทำงานให้กับโมดูลนั้นสามารถทำได้ 2 แบบ

- **การกำหนดค่าแบบถาวร** จะเป็นการสั่งบันทึกค่าเงื่อนไขการทำงานต่างๆของโมดูล ตามรูปแบบที่เรากำหนดไว้ในหน่วยความจำถาวรภายในตัวโมดูล โดยใช้คำสั่ง **“AT&W”** ซึ่งหลังจากโมดูลเริ่มต้นทำงานใหม่ หรือ หลังการรีเซ็ตโมดูลแต่ละครั้ง ค่าการทำงานต่างๆของโมดูลจะถูกกำหนดเงื่อนไขตามที่เรากำหนดไว้แล้วเสมอ
- **การกำหนดค่าแบบชั่วคราว** เป็นการใช้คำสั่ง AT Command ต่างๆ เพื่อกำหนดเงื่อนไขการทำงาน ให้กับโมดูล แต่ไม่มีการสั่งบันทึกค่า Configuration ด้วยคำสั่ง **“AT&W”** ซึ่งการทำงานของโมดูลก็จะปรับเปลี่ยนไปตามการสั่งงานในขณะนั้นๆ แต่เมื่อสั่งรีเซ็ตการทำงานของโมดูล หรือ มีการPower ON ใหม่คุณสมบัติการทำงานของโมดูลจะถูกเปลี่ยนกลับเป็นค่าเดิมอีก โดยเราสามารถใช้คำสั่ง AT Command ในการสั่ง ตรวจสอบ และ บันทึกค่า Configuration ต่างๆ ให้กับโมดูล Fastrack Supreme 20 ได้ดังนี้
  - ใช้คำสั่ง **“AT&V”** เพื่อสั่งให้โมดูลแสดงค่า Configuration ปัจจุบันให้ทราบ
  - ใช้คำสั่ง **“AT&F”** เพื่อสั่งกำหนดค่า Configuration ทั้งหมดให้กลับเป็นค่ามาตรฐาน
  - ใช้คำสั่ง **“AT&W”** เพื่อสั่งบันทึกค่า Configuration ด้วยค่าที่เรากำหนดไว้ในขณะนั้นๆ

ค่า Configuration ที่แนะนำ

- AT+CMGF=1(SMS Message = Text Mode)
- ATE=1 (Echo Mode ON)
- AT+CSCLK=0(Disable Sleep Mode)

2.3.6 การโทรออก การรับสาย การยกเลิกการโทร และการรับ/ส่ง SMS

การโทรออก การรับสาย การยกเลิกการโทร

- ใช้คำสั่ง **“ATD”** เพื่อสั่งโทรออก โดยรูปแบบการใช้คำสั่งให้ตามด้วยเบอร์ปลายทาง
- ใช้คำสั่ง **“ATDL”** เพื่อสั่งโทรออกด้วยหมายเลขโทรออกครั้งสุดท้าย
- ใช้คำสั่ง **“ATA”** เพื่อรับสายเรียกเข้า โดยเมื่อมีสายเรียกเข้าจะมีเสียงเรียกเข้าที่ Buzzer ให้เราทราบ ถ้าต้องการรับสายให้ใช้คำสั่ง “ATA” เพื่อรับสายได้ทันที

ซึ่งหลังจากสั่งรับสายแล้วผู้ใช้จะสามารถพูดคุยกับปลายสายได้ทันที โดยใช้ Handset หรือชุด ปากพูดหูฟังของโทรศัพท์บ้าน

- ใช้คำสั่ง **"ATH"** เพื่อสั่งวางสาย หรือยกเลิกการโทรออก

ตัวอย่างการโทรออก ซึ่งเป็นการสื่อสารด้วย Voice จะต้องปิดท้ายคำสั่งด้วยเครื่องหมายเซมิโคลอน (;) และจบคำสั่งด้วย Enter เช่นถ้าต้องการโทรออกไปยังเบอร์ 0811234567 จะเป็นดังนี้

```
ATD0811234567;<Ent>
OK
```

ในกรณีที่สั่งโทรออกแล้วไม่มีการรับสาย หรือ สายไม่ว่างโมดูลจะรายงานผลด้วยข้อความ"BUSY" ดังตัวอย่าง

```
ATD0812505187;<Ent>
OK
BUSY
```

ตัวอย่างการรับสายเรียกเข้า เมื่อมีสายเรียกเข้าโมดูล Fastrack Supreme 20 จะมีข้อความ "RING" และสร้าง เสียงเรียกเข้าเป็นจังหวะที่ Buzzer ให้ทราบ ถ้าผู้ใช้ต้องการรับสาย ให้ใช้คำสั่ง "ATA" เพื่อสั่งรับสาย หรือใช้คำสั่ง "ATH" เพื่อวางหูหรือยกเลิกไม่รับสาย ดังตัวอย่าง


```
RING
ATA<Ent>
OK
```

ตัวอย่างการรับข้อความ SMS

ตามปกติแล้วโมดูล Fastrack Supreme 20 จะสามารถกำหนดโหมดการทำงานของข้อความหรือ SMS ได้ 2โหมด คือ PDU Mode และ Text Mode โดย PDU Mode การรับและแสดงผลการทำงานของคำสั่งจะเป็น รูปแบบของรหัสตัวเลขแบบ Binary Code ส่วน Text Mode การรับและแสดงผลการทำงานของคำสั่งจะเป็น ข้อความ ซึ่งจะง่ายต่อการแปลความหมายและทำความเข้าใจมากกว่า PDU Mode ซึ่งในการทดสอบ จะขอแสดงให้เห็นด้วย Text Mode

- ใช้คำสั่ง “AT+CMGF=1” เพื่อกำหนดรูปแบบของข้อความเป็น Text Mode ซึ่งเมื่อมีการส่งข้อความ SMS มายังโมดูล จะมีข้อความแจ้งให้ทราบ เช่น +CMTI: “SM”,3 ซึ่งหมายความว่า มีข้อความส่งเข้าและเก็บไว้ในหน่วยความจำลำดับที่ 3
- ใช้คำสั่ง “AT+CMGR” เพื่อสั่งอ่านข้อความ เช่นถ้าต้องการอ่านข้อความลำดับที่3 ก็ให้ใช้คำสั่งเป็น “AT+CMGR=3”
- ใช้คำสั่ง “AT+CMGL” เพื่อสั่งแสดงข้อความทั้งหมดที่เก็บไว้ในหน่วยความจำ โดยสามารถเลือกประเภทของข้อความได้ เช่น ข้อความใหม่ ข้อความทั้งหมด
- ใช้คำสั่ง “AT+CMGD” เพื่อสั่งลบข้อความออกจากหน่วยความจำ เช่น ถ้าต้องการสั่งลบข้อความลำดับที่3 ก็ให้ใช้คำสั่งเป็น “AT+CMGD=3”

```
+CMTI: "SM",3
AT+CMGR=3<Ent>
+CMGR: "REC UNREAD","+668125O5187",,"07/11/19,13:29:25+28"
Hello 12345
OK
```

ถ้ามีการสั่งอ่านข้อความเดิมซ้ำใหม่สถานะของข้อความจะเปลี่ยนเป็น “REC READ” แทน เพื่อแสดงให้ทราบว่าข้อความนี้ถูกอ่านไปแล้วดังตัวอย่าง


```
AT+CMGR=3<Ent>
+CMGR: "REC READ","+66812505187",,"07/11/19,13:29:25+28"
Hello 12345
OK
```

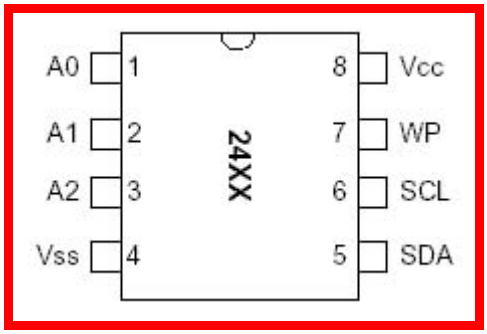
ตัวอย่างการส่งข้อความ SMS

ในการส่งข้อความ SMS นั้นจะใช้คำสั่ง “AT+CMGS” ในการสั่งงาน โดยในกรณีที่ใช้ Text Mode นั้นให้ใช้รูปแบบคำสั่งเป็น “AT+CMGS=”+เบอร์ผู้รับ” โดยเบอร์ของผู้รับต้องใส่รหัสประเทศนำหน้าแทนศูนย์ด้วยเสมอ ซึ่งในกรณีที่เป็นประเทศไทยจะใช้รหัสประเทศเป็น “66” ดังนั้นถ้าต้องการส่งข้อความ SMS ให้กับเบอร์ที่ใช้งานอยู่ในประเทศไทย เช่น 081-1234567 ก็จะต้องกำหนดหมายเลขของเบอร์ผู้รับปลายทางเป็น 6681-1234567 แทน ซึ่งในกรณีนี้จะได้รหัสเบอร์ผู้รับข้อความเป็น “+66811234567” ซึ่งเมื่อโมดูล Fastrack Supreme 20 ได้รับคำสั่ง “AT+CMGS” เรียบร้อยแล้วมันจะตอบรับด้วยการส่งเครื่องหมาย “>” กลับมาบอก ซึ่งหลังจากนี้เป็นต้นไปผู้ใช้ก็สามารถจะทำการพิมพ์ข้อความต่างๆที่ต้องการจะส่งให้กับโมดูลได้ทันที โดยให้ปิดท้ายข้อความด้วยรหัส “Ctrl+Z” ตามด้วย “Enter” เช่นถ้าต้องการส่งข้อความ SMS ให้กับหมายเลข 0811234567 ด้วยข้อความ “Hello Test SMS” จะเป็นดังนี้

```
AT+CMGS="+66811234567"<Ent>
> Hello Test SMS<Ctrl+Z><Ent>
+CMGS: 6
OK
```

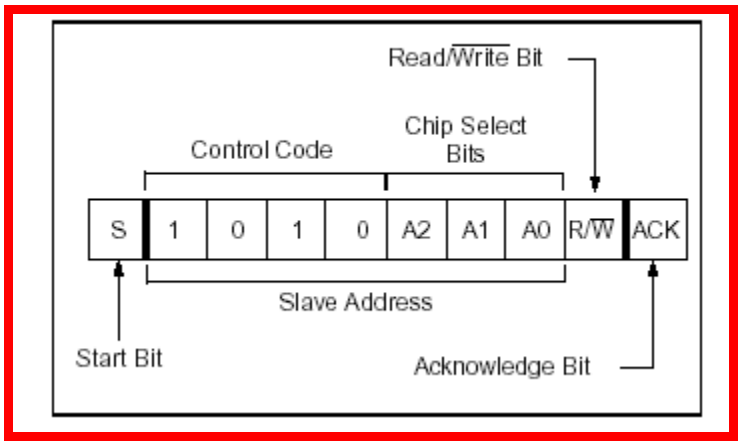

2.4 EEPROM

2.4.1 I²C EEPROM24LCXX

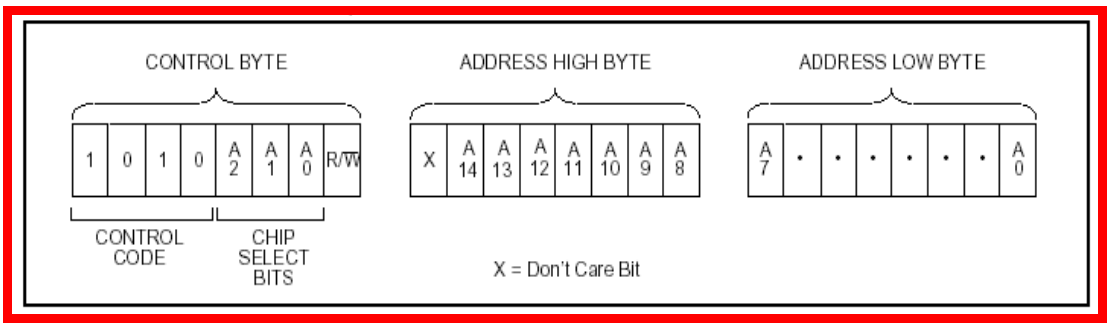


รูปที่ 2.19 แสดงตำแหน่งของขา EEPROM

- **A0,A1,A2** เป็นขาสัญญาณแอดเดรส Input ใช้สำหรับกำหนดตำแหน่งการทำงานของ EPROM แต่ละตัวที่จะเชื่อมต่อกันภายในบัส ซึ่งขาสัญญาณแอดเดรสนี้แต่ละตัวอาจมีไม่เท่ากันบางตัวอาจมี 3 ขา บางตัวอาจมีเพียง 1 หรือ 2 ขา บางตัวอาจไม่มีเลย โดยถ้าตัวใดไม่มีการออกแบบให้กำหนดค่าแอดเดรสจากทางฮาร์ดแวร์ได้ ขาสัญญาณเหล่านี้จะถูกปล่อยว่าง (NC) ไว้
- **VSS** เป็นขาสัญญาณ อ้างอิง หรือ GND
- **SDA** เป็นขาข้อมูล แบบ 2 ทิศทาง ของ I^2C ใช้สำหรับรับส่งข้อมูลระหว่าง EEPROM และไมโครคอนโทรลเลอร์ โดยจะทำหน้าที่เป็น Input ในการรับข้อมูลจากไมโครคอนโทรลเลอร์ที่จะส่งให้กับ EEPROM และในทางกลับกันก็จะทำหน้าที่เป็น Output สำหรับส่งข้อมูลจาก EEPROM ให้กับไมโครคอนโทรลเลอร์
- **SCL** เป็นขาสัญญาณนาฬิกา Input ของ I^2C ใช้สำหรับควบคุมการรับส่งหรืออ่านเขียนข้อมูลระหว่างไมโครคอนโทรลเลอร์และ EEPROM
- **WP** เป็นขาสัญญาณ Write Protect โดยมีสภาวะเป็น Input ทำหน้าที่ป้องกันการเขียนข้อมูลให้กับ EEPROM โดยถ้าขานี้มีสภาวะเป็น “0” จะสามารถสั่งเขียนข้อมูลให้กับ EEPROM ได้ แต่ถ้าขานี้มีสภาวะเป็น “1” จะไม่สามารถสั่งเขียนข้อมูลให้กับ EEPROM ได้
- **VDD** เป็นขาสัญญาณไฟเลี้ยงวงจรของ EEPROM 24LC256 ขนาดของจำนวนความจุในการเก็บข้อมูล เป็น 256Kbit ข้อมูลที่เก็บได้ครั้งละ 8 Bit ดังนั้นจะเก็บข้อมูลได้ เป็น (32K *8) สามารถอ้างอิง Address ได้ เป็น 32 * 1024 = 32768 นั่นก็จะมีค่าเท่ากับ 215 (A0 – A14)



รูปที่ 2.20 Control Byte Format



รูปที่ 2.21 Address Sequence Bit Assignments

2.4.2 ขั้นตอนการเขียนข้อมูลลงใน 24LCXX

1. ส่งสภาวะเริ่มต้น (Start Condition) ไปยังบัส เพื่อเริ่มต้นการสื่อสาร
2. ส่งรหัส Control Byte ของ EEPROMสำหรับการเขียน ซึ่งก็คือ “10101110”
3. ส่งค่าตำแหน่งแอดเดรสไบท์สูงที่ต้องการเขียนข้อมูลไปให้ EEPROM
4. ส่งค่าตำแหน่งแอดเดรสไบท์ต่ำที่ต้องการเขียนข้อมูลไปให้ EEPROM
5. ส่งค่าข้อมูลที่ต้องการเขียนไปยัง EEPROM ตำแหน่งแอดเดรสที่ระบุในข้อ 3 และ4
6. ส่งค่าข้อมูลไบท์ถัดไปที่ต้องการเขียน ไปยัง EEPROM จนกว่าจะครบ Page หรือส่งค่าสภาวะสิ้นสุด (Stop Condition) เพื่อจบการสื่อสารถ้าต้องการเขียนเพียง 1 ไบท์ (Byte Write)
7. ส่งสภาวะสิ้นสุด(Stop Condition) ไปยังบัสเพื่อจบการสื่อสาร

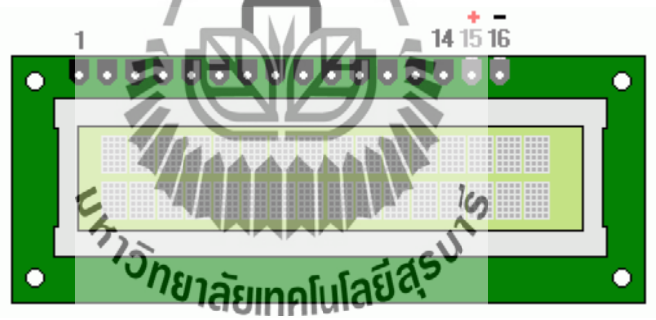
2.4.3 ขั้นตอนการอ่านข้อมูลลงใน 24LCXX

1. ส่งสภาวะเริ่มต้น (Start Condition) ไปยังบัส เพื่อเริ่มต้นการสื่อสาร
2. ส่งรหัส Control Byte ของ EEPROMสำหรับการเขียน ซึ่งก็คือ “10101110”
3. ส่งค่าตำแหน่งแอดเดรสไบท์สูงที่ต้องการเริ่มต้นการอ่านข้อมูลไปให้ EEPROM
4. ส่งค่าตำแหน่งแอดเดรสไบท์ต่ำที่ต้องการเริ่มต้นการอ่านข้อมูลไปให้ EEPROM
5. ส่งรหัส Control Byte ของ EEPROMสำหรับการอ่าน ซึ่งก็คือ “10101111”
6. อ่านข้อมูลจากหน่วยความจำจากแอดเดรสที่ระบุในข้อ 1.3 และ 1.4 จำนวน 1ไบท์
7. ส่งสภาวะสิ้นสุด(Stop Condition) ไปยังบัสเพื่อจบการสื่อสาร

2.5 LCD 16x2

หน้าจอ LCD (Liquid Crystal Display) เป็นโมดูลแสดงผลอิเล็กทรอนิกส์และหาที่หลากหลายของการใช้งาน จอแสดงผล LCD 16x2 เป็นโมดูลพื้นฐานมากและมีการใช้บ่อยมากในอุปกรณ์ต่างๆ และวงจร

16x2 หมายความว่ามันสามารถแสดงผล 16 ตัวอักษรต่อบรรทัดและมี 2 สายดังกล่าว ในจอแอลซีดีนี้อักขระแต่ละตัวจะถูกแสดงในเมทริกซ์ 5x7 พิกเซล จอแอลซีดีนี้มีสองจิคือคำสั่งและข้อมูลลงทะเบียนคำสั่งเก็บคำแนะนำการใช้คำสั่งที่กำหนดให้จอแอลซีดี คำ สั่งคือคำสั่งที่กำหนดให้จอแอลซีดีที่จะทำงานที่กำหนดไว้ล่วงหน้าเช่นการ เริ่มต้นมันล้างหน้าจอของการตั้งค่าตำแหน่งเคอร์เซอร์, การควบคุมการแสดงผลและอื่น ๆ จัดเก็บข้อมูลการลงทะเบียนข้อมูลที่จะปรากฏบนจอแอลซีดี ข้อมูลเป็นค่า ASCII ของตัวละครที่จะปรากฏบนจอแอลซีดี



รูปที่ 2.22 แสดงภาพจำลองของ LCD 16x2

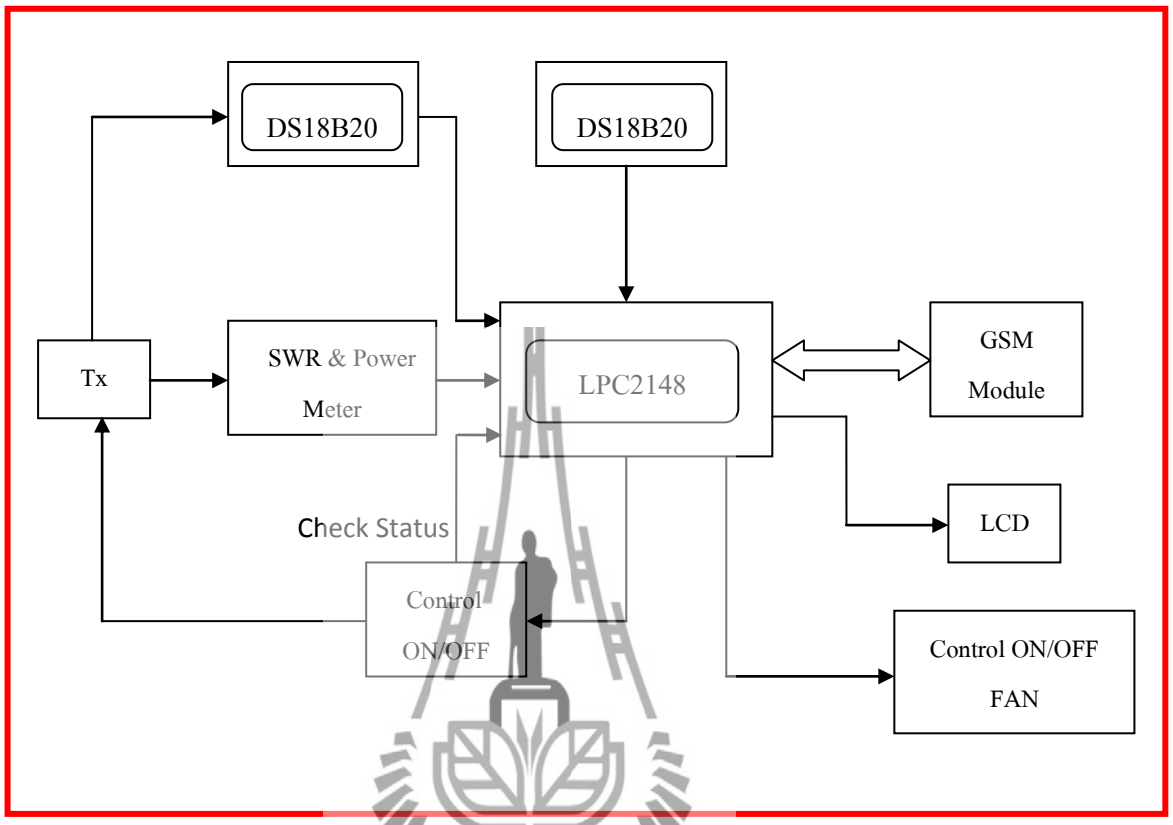
บทที่ 3
การออกแบบโครงงาน

โครงงานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านเครือข่ายโทรศัพท์เคลื่อนที่ ได้มีการออกแบบเพื่อการบอกสถานะของเครื่องส่งวิทยุกระจายเสียงในรูปแบบ Short Messaging Service (SMS) ซึ่งโครงงานประกอบด้วย GSM Module และ บอร์ดไมโครคอนโทรเลอร์ ARM7 โดยบอร์ดไมโครคอนโทรเลอร์จะนำข้อมูลต่าง ๆ ที่ได้รับจากเครื่องส่งวิทยุกระจายเสียง ไปรายงานผลในรูปแบบ SMS ผ่านทาง GSM Module และนอกจากนี้บอร์ดไมโครคอนโทรเลอร์ยังสามารถประมวลผลเพื่อหาค่าผิดปกติ ซึ่งคำนวณจากค่าอ้างอิงเพื่อรายงานผล โดยโครงงานนี้ประกอบด้วยด้วยส่วนของการทำงานต่าง ๆ ดังนี้

1. การรับข้อมูลจากเครื่องส่งวิทยุกระจายเสียง
2. การส่งข้อมูลระหว่างไมโครคอนโทรเลอร์ กับ GSM Module
3. การแสดงผล

ซึ่งในแต่ละส่วนจะมีไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงานหลักเพื่อให้สามารถทำงานได้ตามต้องการ ดังนั้นเพื่อให้การทำงานในแต่ละส่วนเป็นไปอย่างสมบูรณ์จึงได้ทำการออกแบบส่วนการทำงานในแต่ละส่วนดังนี้

3.1 การออกแบบฮาร์ดแวร์



รูปที่ 3.1 แผนภาพการออกแบบฮาร์ดแวร์

3.2 การออกแบบซอฟแวร์

บอร์ดไมโครคอนโทรเลอร์ ARM7 จะเป็นตัวควบคุมการทำงานหลัก โดยจะต้องมีซอฟต์แวร์ที่ใช้ในการเขียนคำสั่งควบคุม ดังนั้นในโครงงานนี้ ผู้จัดทำเลือกใช้ภาษาซีโดยได้ใช้โปรแกรม Keil uVision3 ในการเขียนคำสั่งควบคุม เนื่องจากภาษาซีเป็นภาษาโครงสร้างง่ายต่อการทำความเข้าใจและสามารถปรับปรุงพัฒนาต่อได้ง่าย นอกจากนั้นภาษาซียังเป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์ (ไมโครคอนโทรเลอร์) มีความยืดหยุ่นในการใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย

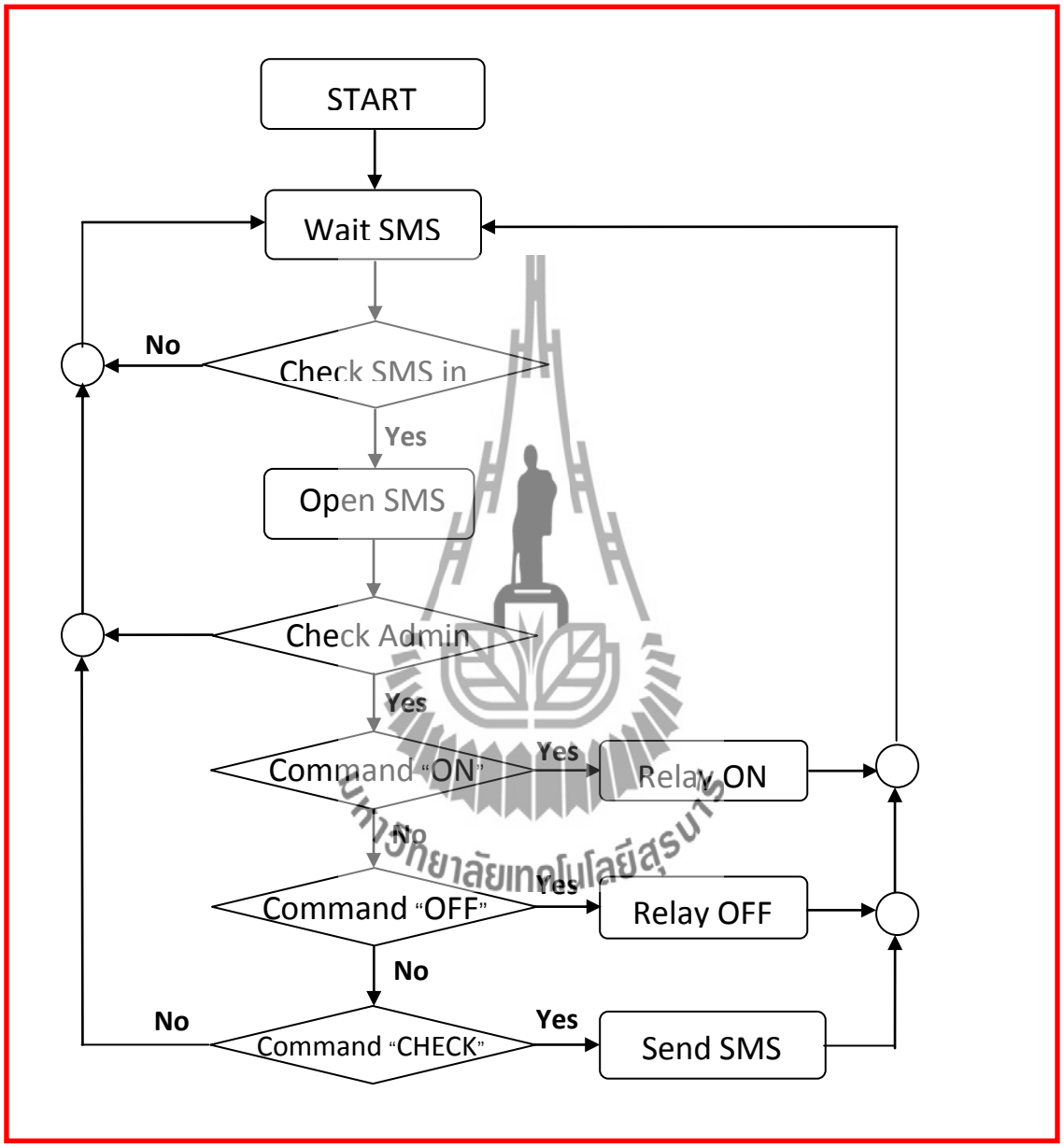
3.3 การทำงานของโปรแกรม

เนื่องจากการส่งข้อความ SMS โดย GSM Module ใช้หลักการเช่นเดียวกับการใช้โทรศัพท์เคลื่อนที่ คือ จะต้องมีข้อมูลและส่งข้อมูลผ่านทางเครือข่ายโทรศัพท์เคลื่อนที่ ซึ่งข้อมูลถูกส่งจากเครื่องส่งวิทยุกระจายเสียงมายังบอร์ดไมโครคอนโทรลเลอร์เพื่อประมวลผล ก่อนที่ข้อมูลเหล่านั้นจะถูกส่งผ่าน GSM Module เพื่อไปแสดงผลที่โทรศัพท์เคลื่อนที่ในรูปแบบข้อความ SMS

ในการออกแบบโครงงานนี้เพื่อตอบสนองความต้องการในการได้รับข้อมูลในจังหวะเวลาต่าง ๆ จึงได้แบ่งการทำงานออกเป็น 3 กรณีดังนี้

1. **กรณีร้องขอ (Requesting)** คือ ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงทำการส่งข้อความ SMS ไปยังบอร์ดไมโครคอนโทรลเลอร์ เพื่อต้องการสั่งการเครื่องส่งวิทยุกระจายเสียงให้ เปิด/ปิด โดยการพิมพ์คำว่า “ON.” เมื่อต้องการเปิดเครื่อง หรือพิมพ์คำว่า “OFF.” เมื่อต้องการปิดเครื่อง นอกจากนี้หากต้องการทราบสถานะเครื่องส่งวิทยุกระจายเสียงสามารถพิมพ์คำว่า “CHECK.” จะได้รับข้อความ SMS ตอบกลับแสดงข้อมูลที่ต้องการ
2. **กรณีตามรอบเวลา (Scheduling)** คือ ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการทราบสถานะเครื่องส่งวิทยุกระจายเสียงอย่างสม่ำเสมอ โดยมีการกำหนดเวลาเป็นคาบ เช่น ต้องการทราบข้อมูลทุก ๆ 1 ชั่วโมง ดังนั้นในหนึ่งวัน ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะได้รับข้อความ SMS ที่แสดงข้อมูลเป็นจำนวน 24 ข้อความ ข้อความที่ถัดกันมามีระยะเวลาห่างกัน 1 ชั่วโมง
3. **กรณีเตือน (Warning)** คือ ข้อมูลหรือค่าต่าง ๆ ของเครื่องส่งวิทยุเกิดความผิดปกติไปจากค่าที่กำหนด เช่น อุณหภูมิเกิน 30°C ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะได้รับข้อความ SMS แจ้งเตือนในทันที

แผนผังการทำงาน



รูปที่ 3.2 แผนผังการทำงานของระบบ

3.4 อธิบายการทำงานของโครงงาน

เริ่มต้นการทำงานโดยโปรแกรมจะรอรับข้อความ SMS เมื่อมีข้อความ SMS เข้ามา โปรแกรมจะหยุดรับข้อความอื่น จากนั้นโปรแกรมจะทำการเปิดข้อความ SMS นั้นเพื่อดึงข้อมูล หมายเลขโทรศัพท์และคำสั่ง (ON./OFF./CHECK.) ออกมา หลังจากนั้นเข้าสู่ขั้นตอนการ ตรวจสอบหมายเลขโทรศัพท์ว่าใช่หมายเลขโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงหรือไม่ ถ้าไม่ใช่หมายเลขโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง โปรแกรมจะยกเลิกการทำงานแล้ว กลับไปเริ่มรับข้อความใหม่ ถ้าใช่หมายเลขโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง โปรแกรมจะทำการตรวจสอบว่าคำสั่งคืออะไร ซึ่งแบ่งคำสั่งได้ 3 ประเภท ดังนี้

1. คำสั่งคือ “ON.” โปรแกรมจะสั่งให้ทำการเปิดเครื่องส่งวิทยุกระจายเสียง
2. คำสั่งคือ “OFF.” โปรแกรมจะสั่งให้ทำการปิดเครื่องส่งวิทยุกระจายเสียง
3. คำสั่งคือ “CHECK.” โปรแกรมจะสั่งให้ทำการส่งข้อความ SMS ตอบกลับแสดง สถานะเครื่องส่งวิทยุกระจายเสียง

นอกจากนี้หากนอกเหนือจาก 3 คำสั่งนี้ นั่นคือ พิมพ์ข้อความผิด โปรแกรมจะยกเลิกการ ทำงานแล้วกลับไปเริ่มรับข้อความใหม่

บทที่ 4
การใช้งานโครงงาน

4.1 การใช้งานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS

โครงงานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS มีลักษณะต่าง ๆ ดังรูปที่ 4.1

รูปที่ 4.1 ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS

ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านระบบ SMS นั้นมีไมโครคอนโทรลเลอร์ ARM7 เป็นตัวควบคุมการทำงาน และมี GSM Module ซึ่งมีประสิทธิภาพการทำงานเช่นเดียวกับโทรศัพท์เคลื่อนที่ คือ การโทรออก การรับสายเรียกเข้า และการส่งข้อความ SMS โดยการใช้ Sim Card จากเครือข่ายผู้ให้บริการที่สามารถรับสัญญาณโทรศัพท์ในท้องถิ่นนั้น ๆ โดยไมโครคอนโทรลเลอร์จะประมวลผลและส่งข้อความตอบกลับผ่านทาง GSM Module เพื่อแสดงข้อมูลต่าง ๆ ได้แก่ Date , Time , Status , Ratio , Forward , Reflect , Temp. และ

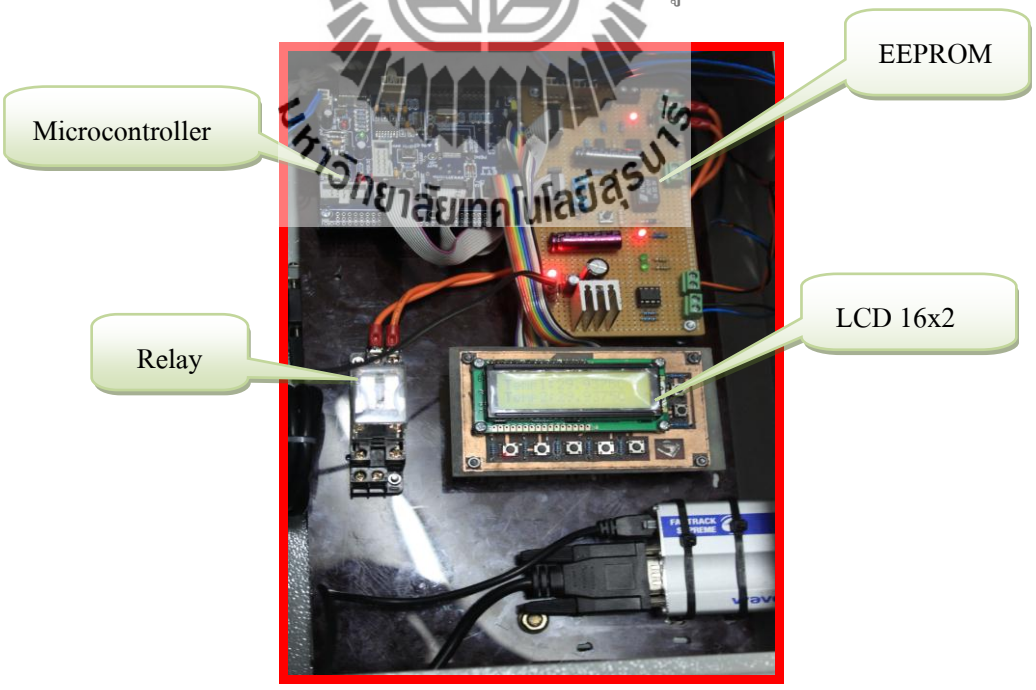
Temp. Room ของเครื่องส่งวิทยุกระจายเสียงไปยังหมายเลขโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง ได้ตามที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการ

4.1.1 การเริ่มใช้งานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านระบบ SMS

1. ใส่ Sim Card ที่ตัว GSM Module, เชื่อมต่อสายอากาศของตัว GSM Module เพื่อให้สามารถส่ง – รับ ข้อความ SMS , โทรออก รับสายเรียกเข้าได้ ดังรูปที่ 4.2

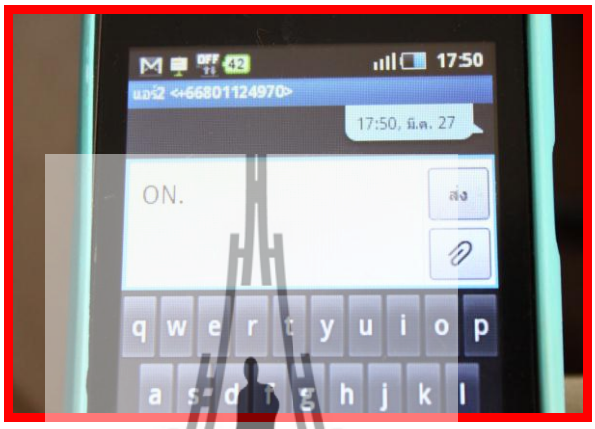
รูปที่ 4.2 แสดงการใส่ Sim Card ที่ตัว GSM Module

2. เมื่อเริ่มเปิดระบบจะมีไฟ LED บอกสถานะ ดังรูปที่ 4.3



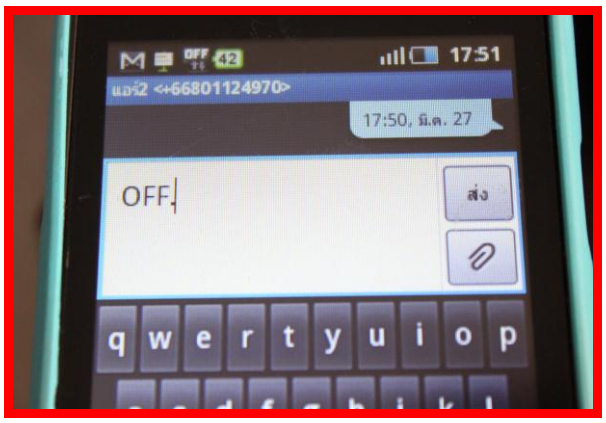
รูปที่ 4.3 แสดงลักษณะส่วนประกอบต่าง ๆ ของระบบ

3. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถสั่งให้เครื่องส่งวิทยุกระจายเสียงทำงานได้ โดยส่งข้อความ SMS คำว่า “ON.” จากโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง จากนั้นระบบจะสั่งการให้เครื่องส่งวิทยุกระจายเสียงเปิดเครื่อง



รูปที่ 4.4 รูปแสดงลักษณะที่ส่งข้อความ ON.

4. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถสั่งให้เครื่องส่งวิทยุกระจายเสียงทำงานได้ โดยส่งข้อความ SMS คำว่า “OFF.” จากโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง จากนั้นระบบจะสั่งการให้เครื่องส่งวิทยุกระจายเสียงปิดเครื่อง



รูปที่ 4.5 รูปแสดงลักษณะที่ส่งข้อความ OFF.

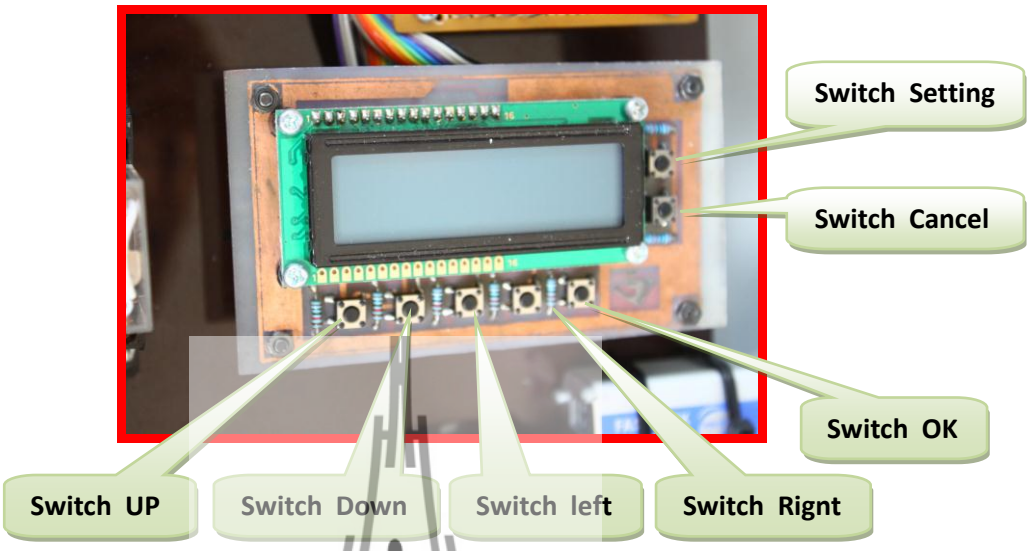
5. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถทราบสถานะของเครื่องส่งวิทยุกระจายเสียงได้ว่า ณ ขณะนั้นข้อมูลต่าง ๆ ได้แก่ Date , Time , Status , Ratio , Forward , Reflect , Temp. และ Temp. Room ของเครื่องส่งวิทยุกระจายเสียงมีค่าเท่าไร โดยการส่งข้อความ SMS คำว่า “CHECK.” จากโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียงไปที่ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง แล้วระบบจะส่งข้อความ SMS ตอบกลับแสดงข้อมูลดังกล่าวมายังโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง

รูปที่ 4.6 ข้อความ SMS แสดงข้อมูลของเครื่องส่งวิทยุกระจายเสียง

4.1.2 หน้าจอที่แสดง และ Switch ที่ใช้ในการตั้งค่า

หน้าจอ LCD จะทำให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียง ได้ทราบข้อมูลของเครื่องส่งวิทยุกระจายเสียงอัพเดตตลอดเวลา ได้แก่ Date , Time , Ratio , Forward , Reflect , Temp. Temp. Room และหมายเลยโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง ซึ่งผู้ดูแลเครื่องส่งวิทยุกระจายเสียงได้ตั้งค่าและบันทึกไว้ ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงสามารถกลับมาแก้ไขหรือเปลี่ยนแปลงค่าต่าง ๆ เหล่านี้ได้ตลอดเวลา

Switch ที่ใช้ในการตั้งค่า



รูปที่ 4.7 แสดงหน้าจอ LCD และ Switch ที่ใช้ในการตั้งค่า

4.1.3 การตั้งค่าหมายเลขโทรศัพท์ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง (เบอร์ Admin)

1. กด Switch setting แล้วเลือกไปที่หน้า เบอร์ Admin

1.1 การตั้งค่า เบอร์ Admin โดยกด switch right เรื่อยๆ มาจนถึงหน้า ที่มีคำว่า Admin1 : _ _ _ - _ _ _ _ _ _ _ และ Admin2 : _ _ _ - _ _ _ _ _ _ _ และทำการตั้งเบอร์ที่จะใช้เป็น เบอร์ Admin โดยกด Switch up หรือ down ทั้งสองเบอร์ ทั้ง Admin1 และ Admin2 แล้วกด Switch ok เพื่อที่ทำการ Save เมื่อตั้งค่าเสร็จแล้ว แสดงดังรูปที่ 4.8

รูปที่ 4.8 แสดงเบอร์ Admin1 และAdmin2 ที่ได้ทำการตั้งไว้ที่หน้าจอ LCD

1.2 การตั้งค่าและบันทึก วัน เวลา โดยกด switch right เรื่อยๆ มาจนถึงหน้าที่มีคำว่า DATE : _ _ : _ _ : _ _ _ _ และ TIME : _ _ : _ _ และทำการเลือก วันและ เวลาที่จะตั้ง โดยกด Switch up หรือ down เพิ่มค่าหรือ ลด วันที่และเวลา ตามที่ผู้ใช้งานจะเลือกว่าจะตั้งไว้ที่เท่าไหร่ แล้วกด Switch ok เพื่อที่ทำการ Save

4.2 การทดลองใช้งานโครงงาน

การทดลองใช้งานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS คณะผู้จัดทำโครงงานได้ทำการทดสอบระบบ ดังนี้

1. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “ON.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง
2. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “OFF.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง
3. ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “CHECK.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง

4.3 ผลการทดสอบโครงงาน

เมื่อนำโครงงานเครื่องส่งวิทยุกระจายเสียงผ่านเครือข่ายโทรศัพท์เคลื่อนที่มาทดสอบการใช้งาน ได้ผลการทดสอบดังนี้

1. เมื่อผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “ON.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง มีผลทำให้เครื่องส่งวิทยุกระจายเสียงอยู่ใน Status : ON พร้อมใช้งาน
2. เมื่อผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “ON.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง มีผลทำให้เครื่องส่งวิทยุกระจายเสียงอยู่ใน Status : OFF ไม่พร้อมใช้งาน
3. เมื่อผู้ดูแลเครื่องส่งวิทยุกระจายเสียงส่งข้อความ SMS คำว่า “CHECK.” ไปยังระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียง ระบบจะส่งข้อความ SMS ตอบกลับมายังโทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง เพื่อแสดงข้อมูลต่าง ๆ ได้แก่ Date , Time , Status , Ratio , Forward , Reflect , Temp. และ Temp. Room

4. เมื่อเครื่องส่งวิทยุกระจายเสียงทำงานมาได้ ณ ขณะเวลาหนึ่ง ถ้าค่าพารามิเตอร์ต่าง ๆ เกินกว่าค่าที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงได้ตั้งค่าไว้ในตอนแรก ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงจะทำการส่งข้อความ SMS เตือนไปที่โทรศัพท์เคลื่อนที่ของผู้ดูแลเครื่องส่งวิทยุกระจายเสียง เพื่อให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงได้ทราบและทำการแก้ไข

รูปที่ 4.9 ข้อความ SMS เตือนเมื่อพารามิเตอร์ของเครื่องส่งวิทยุกระจายเสียงเกิดความผิดปกติ

บทที่ 5
สรุปผลการทดสอบและข้อเสนอแนะ

โครงงานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS สร้างขึ้นเพื่อใช้เป็นระบบในการรายงานสถานะการทำงานและค่าพารามิเตอร์ต่าง ๆ ของเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS ได้ตามที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการ แม้ว่าผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะไม่ได้ประจำอยู่บริเวณหน้าเครื่องตลอดเวลา โดยผู้ดูแลเครื่องส่งวิทยุกระจายเสียงจะทำการส่งข้อความ SMS ไปที่ระบบด้วยคำว่า ON. , OFF. เพื่อสั่งการให้ระบบทำการเปิด/ปิดเครื่องส่งวิทยุกระจายเสียง และหากส่งข้อความ SMS คำว่า CHECK. จะเป็นการสั่งการให้ระบบส่งข้อความ SMS ตอบกลับเพื่อรายงานสถานะการทำงานของเครื่องส่งวิทยุกระจายเสียง นอกจากนี้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงยังสามารถได้รับข้อความ SMS แสดงสถานะการทำงานและค่าพารามิเตอร์ต่าง ๆ ของเครื่องส่งวิทยุกระจายเสียงได้ตามรอบเวลาหรือเมื่อเครื่องส่งวิทยุกระจายเสียงเกิดความผิดปกติ และจากการทดสอบการใช้งานระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS สามารถสรุปผลการทดสอบระบบได้ดังนี้คือ

1. เมื่อส่งคำสั่งจากโทรศัพท์เคลื่อนที่ว่า “ON.” จะทำให้เครื่องส่งวิทยุกระจายเสียงมีสถานะเปิด
2. เมื่อส่งคำสั่งจากโทรศัพท์เคลื่อนที่ว่า “OFF.” จะทำให้เครื่องส่งวิทยุกระจายเสียงมีสถานะปิด
3. เมื่อส่งคำสั่งจากโทรศัพท์เคลื่อนที่ว่า “CHECK.” จะสามารถทราบถึง วัน เวลา Forward Reflect และอุณหภูมิ ของเครื่องส่งวิทยุกระจายเสียง
4. เมื่อต้องการทราบข้อมูลอย่างสม่ำเสมอ ระบบสามารถส่งข้อความ SMS ตามรอบเวลา
5. เมื่อเกิดความผิดพลาดกับเครื่องส่งวิทยุกระจายเสียง ระบบจะสามารถส่งข้อความ SMS เตือนไปยังโทรศัพท์เคลื่อนที่ได้

5.1 ปัญหาและอุปสรรค

5.1.1 เมื่ออยู่ในบริเวณที่ไม่มีสัญญาณโทรศัพท์ ไม่สามารถส่งข้อความ SMS ไปยังเครื่องส่งวิทยุกระจายเสียงได้

5.1.2 ในบางครั้งมีความล่าช้าในการส่งข้อความ SMS จากเครื่องส่งวิทยุกระจายเสียงไปยังโทรศัพท์เคลื่อนที่

5.1.3 การส่งข้อความ SMS จากโทรศัพท์เคลื่อนที่ไปยังเครื่องส่งวิทยุกระจายเสียงต้องพิมพ์ตามที่กำหนดไว้เท่านั้น ระบบจึงจะทำงานตามคำสั่ง

5.1.4 เมื่อมีการส่งข้อความ SMS กลับมายังโทรศัพท์เคลื่อนที่ บางครั้งเกิดเป็นภาษาที่ไม่สามารถอ่านข้อมูลได้

5.1.5 เมื่อเกิดไฟฟ้าขัดข้อง ไม่สามารถทราบสถานะของเครื่องส่งวิทยุกระจายเสียงได้

5.2 สิ่งที่ได้รับจากการทำโครงงาน

5.2.1 ได้รับความรู้เกี่ยวกับหลักการทำงานของอุปกรณ์ รับ-ส่ง ข้อความ (GSM Module)

5.2.2 ได้รับความรู้เกี่ยวกับเรื่อง GSM (Global System for Mobile Communications)

5.2.3 ได้รับความรู้เกี่ยวกับการใช้งานไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์ต่าง ๆ และการประยุกต์งานไมโครคอนโทรลเลอร์สำหรับใช้งานอย่างอื่น

5.2.4 ได้เรียนรู้การการทำงานร่วมกับผู้อื่น

5.2.5 สามารถนำความรู้ที่ได้จากการศึกษาทฤษฎีมาปฏิบัติได้จริง

5.2.6 สามารถนำความรู้และประสบการณ์ที่ได้จากการทำโครงงานไปประยุกต์ใช้ในชีวิตจริง

ประวัติผู้เขียน

นางสาวสุชญา ชะมังพล

เกิดเมื่อวันที่ 4 พฤศจิกายน พ.ศ.2532

ภูมิลำเนาอยู่ที่ ตำบลจอหอ อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา

สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนสุรนารีวิทยา

อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา เมื่อปี พ.ศ.2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวมัลลิกา แสงจันทร์

เกิดเมื่อวันที่ 4 มกราคม พ.ศ.2533

ภูมิลำเนาอยู่ที่ ตำบลสระกะเทียม อำเภอเมืองนครปฐม จังหวัดนครปฐม

สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนสิรินธรราชวิทยาลัย

อำเภอเมืองนครปฐม จังหวัดนครปฐม เมื่อปี พ.ศ.2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวอัจฉรียา คงบรรทัด

เกิดเมื่อวันที่ 22 กุมภาพันธ์ พ.ศ.2533

ภูมิลำเนาอยู่ที่ ตำบลในเมือง อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา

สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนสุรนารีวิทยา

อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา เมื่อปี พ.ศ.2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

บรรณานุกรม

[1] นคร ภักดีชาติ, อรรถพล บุญยะโภคาม, โอภาส ศิริครรชิตถาวร และชัยวัฒน์ ลิ้มพรวิไล (ม.ป.ป.).
คู่มือทดลองไมโครคอนโทรลเลอร์ 32 บิตตระกูล ARM7 เบื้องต้น ฉบับ LPC2148.
บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด.

[2] บริษัท อีทีที จำกัด. คู่มือการใช้งานบอร์ด CP-JR ARM7 USB-LPC2148 EXP [On line] จาก : http://www.ett.co.th/product/ARM/images/CP_JR_ARM7_LPC2138/MAN_CP_JR_ARM7_LPC2138.pdf

[3] การใช้งาน LCD โมดูล [On line] จาก : http://thaimicrotron.com/Referrence/LCD/LCD-Module1.htm

ตารางที่ 2.2.1 PLL Register

ชื่อทั่วไป	ความหมาย	ติดต่อ	ค่าหลังรีเซต	ชื่อและแอดเดรสของ PLL0	ชื่อและแอดเดรสของ PLL1
PLLCON	PLL Control Register เก็บค่ารีจิสเตอร์สำหรับการอัพเดทค่า PLL control bit ค่าที่เขียนให้ยังไม่มีผลจนกว่าจะเขียน PLL feed sequence ที่ถูกต้องให้	อ่าน/เขียน	0	PLL0CON 0xE01FC080	PLL1CON 0xE01FC0A0
PLLCFG	PLL Configuration Register เก็บค่าสำหรับอัพเดทค่าของ PLL Configuration ค่าทีเขียนให้ยังไม่มีผลจนกว่าจะเขียน PLL feed sequence	อ่าน/เขียน	0	PLL0CFG 0xE01FC084	PLL1CFG 0xE01FC0A4
PLLSTAT	PLL Configuration อ่านค่าสถานะของ PLL Control และ configuration	อ่าน/เขียน	0	PLL0STAT 0xE01FC088	PLL1STAT 0xE01FC0A8
PLLFEED	PLL Feed Register เป็นการสั่งโหลดค่าที่เก็บใน PLLCON และ PLLCFG เพื่อสั่งให้ทำงานตามค่าที่กำหนดให้	อ่าน/เขียน	ไม่มี	PLL0FEED 0xE01FC08C	PLL1FEED 0xE01FC0AC

ตารางที่ 2.2.2 PLLCFG Register

ค่าบิตของ PLLCFG	หน้าที่	ความหมาย	ค่าหลังรีเซต
4:0	MSEL 4:0	ค่าตัวคูณ M ของวงจร PLL 00000 คือ M=1 , 00001 คือ M=2,…..,11110 คือ M =31 และ 11111 คือ M=32	0
6:5	PSEL 1:0	ค่าตัวหาร P ของวงจร PLL 00 คือ P=1 ,01 คือ P=2 10 คือ P=2 , 11 คือ P=4	0
7	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่ได้กำหนด

ตารางที่ 2.2.3 PLLCON Register

ค่าบิตของ PLLCON	หน้าที่	ความหมาย	ค่าหลังรีเซต
0	PLLE	PLL Enable สั่งให้ PLL ทำงาน โดยเขียนค่าเป็น 1 และต้องเขียนค่า PLLFEED ที่เหมาะสมด้วย	0
1	PLLC	PLL Connect เมื่อ PLLC และ PLLE มีค่าเป็น 1 และหลังจากเขียนค่า PLLFEED ที่เหมาะสม จะเป็นการสั่งให้ต่อ PLL เป็นสัญญาณนาฬิกาหลังของวงจร	0
7:2	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่ได้กำหนด

ตารางที่ 2.2.4 ค่าแต่ละบิตของรีจิสเตอร์ PLLSTAT

ค่าบิตของ PLLSTAT	หน้าที่	ความหมาย	ค่าหลังรีเซต
4:0	MSEL 4:0	อ่านค่าว่าปัจจุบันวงจร PLL ใช้ค่าตัวคูณความถี่เท่ากับเท่าไร	0
6:5	PSEL 1:0	อ่านค่าว่าปัจจุบันวงจร PLL ใช้ค่าตัวหารความถี่เท่ากับเท่าไร	0
7	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่กำหนด
8	PLLE	อ่านค่าว่าปัจจุบันบิต PLLE มีค่าเป็น 0 หรือ 1	0
9	PLLC	อ่านค่าว่าปัจจุบันบิต PLLC มีค่าเป็น 0 หรือ 1	0
10	PLOCK	ถ้าเป็น 0 แสดงว่า PLL ยังไม่สามารถล็อคค่าความถี่ ถ้าเป็น 1 แสดงว่าสามารถล็อคค่าความถี่ตามที่กำหนดได้แล้ว	0
15:11	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่กำหนด

ตารางที่ 2.2.5 โหมดการทำงานของวงจรPLL

บิต PLLC	บิต PLLE	การทำงานของ PLL
0	0	PLL ไม่ทำงาน ไม่ได้ต่อเข้ากับไมโครคอนโทรเลอร์
0	1	PLL ทำงานแต่ไม่ได้ต่อกับไมโครคอนโทรเลอร์
1	0	เหมือนกับกรณีค่า 00 เพื่อป้องกันไม่ให้ต่อ PLL เข้ากับระบบถ้ายังไม่สั่งให้มันทำงาน
1	1	PLL ทำงาน และต่อเป็นสัญญาณนาฬิกาหลัก

ตารางที่ 2.2.6 VPBDIV Register

แอดเดรส	ชื่อ	ความหมาย	ติดต่อ
0xE10FC100	VPBDIV	ควบคุมค่าตัวหารความถี่ของ VPB Bus	อ่าน/เขียน

ตารางที่ 2.2.7 ค่าแต่ละบิตของรีจิสเตอร์ VPBDIV

ค่าบิตของ VPBDIV	หน้าที่	ความหมาย	ค่าหลังรีเซต
1:0	VPBDIV	VPB bus clock = CCLK/4 VPB bus clock = CCLK VPB bus clock = CCLK/2 สงวนไว้	0
7:2	สงวนไว้	ห้ามเขียนค่า 1 ให้บิตเหล่านี้	0

ตารางที่ 2.2.8 MAM Register

แอดเดรส	ชื่อ	ความหมาย	ติดต่อ
0xE01FC000	MAMCR	Memory Accelerator Module Control Register ใช้กำหนดโหมดการทำงานของ MAM	อ่าน/ เขียน
0xE01FC004	MAMTIM	Memory Accelerator Module Timing Control ใช้กำหนดจำนวนสัญญาณนาฬิกาที่ใช้ในการติดต่อกับหน่วยความจำ Flash	อ่าน/ เขียน

ตารางที่ 2.2.9 การกำหนดค่าให้กับ MAMCR Register

MAMCR bit	ฟังก์ชัน	ความหมาย
1:0	MAN mode control	00-MAM Disable หยุดการทำงานของ MAM 01-MAM partially enable เปิดการทำงาน MAMบางส่วน 10-MAM fully enable เปิดการทำงาน MAMสมบรูณ์แบบ 11-สงวนไว้
7:2	สงวนไว้	สงวนไว้ ห้ามเขียนค่า 1 ให้บิตเหล่านี้

ตารางที่ 2.2.10 การกำหนดค่าให้กับ MAMTIM Register

MAMTIM bit	ฟังก์ชัน	ความหมาย
2:0	MAN Fetch Cycle timing	000-สงวนไว้ 001-MAM fetch cycles = 1 * CCLK 010-MAM fetch cycles = 2 * CCLK 011-MAM fetch cycles = 3 * CCLK 100-MAM fetch cycles = 4 * CCLK 101-MAM fetch cycles = 5 * CCLK 110-MAM fetch cycles = 6 * CCLK 111-MAM fetch cycles = 7 * CCLK
7:3	สงวนไว้	สงวนไว้ ห้ามเขียนค่า 1 ให้บิตเหล่านี้

ตารางที่ 2.2.11 รีจิสเตอร์ PINSEL0, PINSEL1, PINSEL2

ชื่อ	ความหมาย	การติดต่อ	แอดเดรส
PINSEL0	Pin function select register กำหนดการทำงานของ P0.0-P0.15	อ่าน/เขียน	0xE002 C000
PINSEL1	Pin Function select register 1 กำหนดการทำงานของ P0.16-P0.31	อ่าน/เขียน	0xE002 C004
PINSEL2	Pin Function select register 2 กำหนดการทำงานของ P01.0 –P1.31	อ่าน/เขียน	0xE002 C014

ตารางที่ 2.2.12 ค่าประจำบิตของรีจิสเตอร์ PINSEL0

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า 10	การทำงาน เมื่อมีค่า 11	ค่าหลัง รีเซต
1:0	P0.0	GPIO Port0.0	TxD0(UART0)	PWM1	Reserved	00
3:2	P0.1	GPIO Port0.1	RxD0(UART0)	PWM3	EINT0	00
5:4	P0.2	GPIO Port0.2	SCL0(I^2C)	Capture 0.0(TIMER0)	Reserved	00
7:6	P0.3	GPIO Port0.3	SDA0(I^2C)	Match0.0 (TIMER0)	EINT1	00
9:8	P0.4	GPIO Port0.4	SCK0(SPI0)	Capture 0.1(TIMER0)	AD0.6	00
11:10	P0.5	GPIO Port0.5	MISO0(SPI0)	Match0.1 (TIMER0)	AD0.7	00
13:12	P0.6	GPIO Port0.6	MISI0(SPI0)	Capture 0.2(TIMER0)	AD1.0	00
15:14	P0.7	GPIO Port0.7	SSEL0(SPI0)	PWM2	EINT2	00
17:16	P0.8	GPIO Port0.8	TxD1 (UART1)	PWM4	AD1.1	00
19:18	P0.9	GPIO Port0.9	RxD1 (UART1)	PWM6	EINT3	00
21:20	P0.10	GPIO Port0.10	RTS1(UART1)	Capture 1.0(TIMER1)	AD1.2	00
23:22	P0.11	GPIO Port0.11	CTS1(UART1)	Capture 1.1(TIMER1)	SCL1(I^2C1)	00
25:24	P0.12	GPIO Port0.12	DSR1(UART1)	Match1.0 (TIMER1)	AD1.3	00
27:26	P0.13	GPIO Port0.13	DTR1(UART1)	Match1.1 (TIMER1)	AD1.4	00
29:28	P0.14	GPIO Port0.14	DCD1(UART1)	EINT1	SDA1(I^2C1)	00
31:30	P0.15	GPIO Port0.15	RI1(UART1)	EINT2	AD1.5	00

หมายเหตุ :

- ขา P0.0 และ P0.1 ทำหน้าที่เป็น UART0 เพื่อติดต่อกับคอมพิวเตอร์ในการเขียน โปรแกรม
- การกำหนด ค่ารีจิสเตอร์ PINSEL0 ต้องระวังให้ P0.0, P0.1 มีหน้าที่การทำงานเป็นขา TxD0, RxD0, ของ UART0 เสมอ มิฉะนั้นอาจจะทำให้วงจรเสียได้ ดังนั้นจึงกำหนดค่าเริ่มต้นของ PINSEL0 ได้ดังนี้

```
PINSEL0 = 0x00000005; // set P0.2 –P0.15 to GPIO Function
```

ตารางที่ 2.2.13 ค่าประจำบิตของรีจิสเตอร์ PINSEL1

PINSEL1	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า 10	การทำงาน เมื่อมีค่า 11	ค่าหลัง รีเซต
1:0	P0.16	GPIO Port0.16	EINT0	Match0.2(TIMER0)	Capture0.2 (TIMER0)	00
3:2	P0.17	GPIO Port0.17	Capture1.2(TIMER1)	SCK1(SPI1)	Match1.2(TIMER1)	00
5:4	P0.18	GPIO Port0.18	Capture1.3(TIMER1)	MISO1(SPI1)	Match1.3(TIMER1)	00
7:6	P0.19	GPIO Port0.19	Match1.2(TIMER1)	MOIS1(SPI1)	Capture1.2 (TIMER1)	00
9:8	P0.20	GPIO Port0.20	Match1.3(TIMER1)	SSEL(SPI1)	EINT3	00
11:10	P0.21	GPIO Port0.21	PWM5	AD1.6	Capture1.3 (TIMER1)	00
13:12	P0.22	GPIO Port0.22	V_{BUS}	Capture0.0 (TIMER0)	Match0.0(TIMER0)	00
15:14	P0.23	GPIO Port0.23	AD0.7	สงวนไว้	สงวนไว้	00
17:16	P0.24	สงวนไว้				00
19:18	P0.25	GPIO Port0.25	AD0.4	AOUT	สงวนไว้	00
21:20	P0.26	สงวนไว้				00
23:22	P0.27	สงวนไว้				00
25:24	P0.28	GPIO Port0.28	AD0.1	Capture0.2 (TIMER0)	Match0.2(TIMER0)	00
27:26	P0.29	GPIO Port0.29	AD0.2	Capture0.3 (TIMER0)	Match0.3(TIMER0)	00
29:28	P0.30	GPIO Port0.30	AD0.3	EINT3	Match0.0(TIMER0)	00
31:30	P0.31	GPIO Port0.31	UP_LED	CONNECT		00

ตารางที่ 2.2.14 ค่าประจำบิตของรีจิสเตอร์ PINSEL2

PINSEL2	ความหมาย	ค่าหลังรีเซต
1:0	สงวนไว้	00
2	เป็น 0 ขา P1.31:26 จะใช้เป็น GPIO ถ้าเป็น1ขา P1.31:26จะเป็น Debug port	P1.26/RTCK
3	เป็น 0 ขา P1.25:16 จะใช้เป็น GPIO ถ้าเป็น1ขา P1.25:16จะเป็นTrace port	P1.20/ TRACESYNC

ตารางที่ 2.2.15 ชื่อและแอดเดรสของรีจิสเตอร์ที่ใช้ในการควบคุม GPIO

ชื่อทั่วไป	ความหมาย	การติดต่อ	ชื่อ และ แอดเดรสของPort0	ชื่อ และ แอดเดรสของPort0
IOPIN	GPIO Port Pin value register ใช้อ่านสถานะปัจจุบันของพอร์ต	อ่าน	0XE002 8000 IOPIN0	0XE002 8010 IOPIN1
IOSET	GPIO Port Output set register ใช้กำหนดค่าให้ขาของพอร์ตมีค่าเป็น 1 ถ้าบิตใดเป็น 1 ขาของพอร์ตที่ตรงกันจะมีค่าเป็น 1	อ่าน/เขียน	0XE002 8004 IOSET0	0XE002 8014 IOSET1
IODIR	GPIO Port Direction control register ใช้กำหนดว่าขาใดเป็นอินพุตขาใดเป็นเอาต์พุต	อ่าน/เขียน	0XE002 8008 IODIR0	0XE002 8018 IODIR1
IOCLR	GPIO Port Output clear register ใช้กำหนดให้ขาของพอร์ตมีค่าเป็น 0	เขียน	0XE002 800C IOCLR0	0XE002 801C IOCLR1

ตารางที่ 2.2.16 แหล่งกำเนิดอินเตอร์รัปต์ทั้ง 32 ตัว

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel#
WDT	Watchdog Interupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embeded ICE, DbgCommRx	2
ARM Core	Embeded ICE, DbgcommTx	3
TIME0	Match 0-3 (MR0, MR1, MR2, MR3) Capture0-3 (CR0, CR1, CR2, CR3)	4
TIME1	Match0-3 (MR0, MR1, MR2, MR3) Capture 0-3 (CR0,CR1,CR2 CR3)	5
UART0	Rx Line Status (RLS) Transmit Holding Register Empty(THRE) Rx Data Available (RDA) Charater Time-out Indicator(CTI)	6
UART1	Rx Line Status (RLS) Transmit Holding Register Empty(THRE) Rx Data Available (RDA) Charater Time-out Indicator(CTI) Modem Status Interrupt(MSI)	7
PWM0	Match 0-6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I^2C0	SI(state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault(MODF)	10
SPI1(SSP)	SPI Interrupt Flag(SPIF) Mode Fault (MODF)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Indrement(RTCCIF) Alarm(RTCALF)	13

อุปกรณ์	แหล่งกำเนิดอินเตอร์รัปต์	VIC Channel#
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16
System Control	External Interrupt 3 (EINT3)	17
ADC0	A/D Converter 0 end of conversion	18
I^2C1	SI(state change)	19
BOD	Brown our detect	20
ADC1	A/D Converter 1 end of conversion	21
USB	USB interrupt, DMA Interrupt	22
	Reserved	23-31

ตารางที่ 2.2.17 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซต	แอดเดรส
VICIRQStatus	IRQ Status Request อ่านค่าสถานะว่าอนุญาตให้มี interrupt request จากตัวใด และถูกจัดเป็น IRQ	อ่าน	0	0xFFFF F000
VICFIQStatus	FIQ Status Request อ่านค่าสถานะว่าอนุญาตให้มีอินเตอร์รัปต์รีเควสจากตัวใด และถูกจัดเป็น FIQ	อ่าน	0	0xFFFF F004
VICRawltr	Raw Interrupt Status Register ใช้อ่านสถานะของอินเตอร์รัปต์จากทั้ง 32 แหแหล่งหรือจากซอฟต์แวร์โดยไม่สนว่าถูกเปิดใช้หรือถูกจัดประเภทหรือไม่	อ่าน/เขียน	0	0xFFFF F008
VICintselect	Interrupt Select Register ใช้ในการระบุว่าอินเตอร์รัปต์ทั้ง 32 แหล่งแต่ละตัวเป็น FIQ หรือ IRQ	เขียน	0	0xFFFF F00C

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซต	แอดเดรส
VICIntEnable	Inerrupt E able Register ควบคุมว่าให้อินเตอร์รัปต์จากทั้ง32 แหล่งตัวไหนทำงาน และให้เป็น FIQ หรือ IRQ	อ่าน/เขียน	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิตของรีจิสเตอร์ Interrupt Enable Register	เขียน	0	0xFFFF F014
VicSoftlnt	Software Interrupt Register ค่าของรีจิสเตอร์นี้จะถูกจำไป OR กับอินเตอร์รัปต์จากอุปกรณ์ต่างๆ ทั้ง 32 แหล่ง	อ่าน/เขียน	0	0xFFFF F018
VICSoftlntClear	Software Interrupt Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือมากกว่าหนึ่งบิตของรีจิสเตอร์ Software Interrupt Register	เขียน	0	0xFFFF F01C
VICProtection	Protection enable register ใช้จำกัดการเข้าถึง VIC Register	อ่าน/เขียน	0	0xFFFF F020
VICVectAddr	Vector Address Register เมื่อเกิดการอินเตอร์รัปต์แบบ IRQ ตัว IRQ Sevice routine จะอ่านค่าจากรีจิสเตอร์เพื่อกระโดดไปยังแอดเดรสที่ระบุไว้	อ่าน/เขียน	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register ใช้เก็บค่าแอดเดรสของ Interrupt service routine(ISR) สำหรับ IRQ non-vectored IRQ	อ่าน/เขียน	0	0xFFFF F100

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซต	แอดเดรส
VICVEctAddr1	Vector address 1 register	อ่าน/เขียน	0	0xFFFF F104
VICVEctAddr2	Vector address 2 register	อ่าน/เขียน	0	0xFFFF F108
VICVEctAddr3	Vector address 3 register	อ่าน/เขียน	0	0xFFFF F10C
VICVEctAddr4	Vector address 4 register	อ่าน/เขียน	0	0xFFFF F110
VICVEctAddr5	Vector address 5 register	อ่าน/เขียน	0	0xFFFF F114
VICVEctAddr6	Vector address 6 register	อ่าน/เขียน	0	0xFFFF F118
VICVEctAddr7	Vector address 7 register	อ่าน/เขียน	0	0xFFFF F11C
VICVEctAddr8	Vector address 8 register	อ่าน/เขียน	0	0xFFFF F120
VICVEctAddr9	Vector address 9 register	อ่าน/เขียน	0	0xFFFF F124
VICVEctAddr10	Vector address 10 register	อ่าน/เขียน	0	0xFFFF F128
VICVEctAddr11	Vector address 11 register	อ่าน/เขียน	0	0xFFFF F12C
VICVEctAddr12	Vector address 12 register	อ่าน/เขียน	0	0xFFFF F130
VICVEctAddr13	Vector address 13 register	อ่าน/เขียน	0	0xFFFF F134
VICVEctAddr14	Vector address 14 register	อ่าน/เขียน	0	0xFFFF F138
VICVEctAddr15	Vector address 15 register	อ่าน/เขียน	0	0xFFFF F13C
VICVectCntl0	Vector control 0 register รีจิสเตอร์ Vector Control Register0-15 แต่ละตัวควบคุม IRQ Slot แต่ละตัว โดย Slot 0 มีความสำคัญสูงสุดและ Slot 15 มีความสำคัญต่ำสุด	อ่าน/เขียน	0	0xFFFF F200
VICVectCntl1	Vector control 1 register	อ่าน/เขียน	0	0xFFFF F204
VICVectCntl2	Vector control 2 register	อ่าน/เขียน	0	0xFFFF F208
VICVectCntl3	Vector control 3 register	อ่าน/เขียน	0	0xFFFF F20C
VICVectCntl4	Vector control 4 register	อ่าน/เขียน	0	0xFFFF F210
VICVectCntl5	Vector control 5 register	อ่าน/เขียน	0	0xFFFF F214
VICVectCntl6	Vector control 6 register	อ่าน/เขียน	0	0xFFFF F218

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซต	แอดเดรส
VICVectCntl7	Vector control 7 register	อ่าน/เขียน	0	0xFFFF F21C
VICVectCntl8	Vector control 8 register	อ่าน/เขียน	0	0xFFFF F220
VICVectCntl9	Vector control 9 register	อ่าน/เขียน	0	0xFFFF F224
VICVectCntl10	Vector control10 register	อ่าน/เขียน	0	0xFFFF F228
VICVectCntl11	Vector control 11 register	อ่าน/เขียน	0	0xFFFF F22C
VICVectCntl12	Vector control 12 register	อ่าน/เขียน	0	0xFFFF F230
VICVectCntl13	Vector control 13 register	อ่าน/เขียน	0	0xFFFF F234
VICVectCntl14	Vector control 14 register	อ่าน/เขียน	0	0xFFFF F238
VICVectCntl15	Vector control 15 register	อ่าน/เขียน	0	0xFFFF F23C

ตารางที่ 2.2.18 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์จากภายนอก

แอดเดรส	ชื่อ	ความหมาย	การติดต่อ
0xE01FC140	EXTINT	External Interrupt Flag Register เก็บค่าอินเตอร์รัปต์แฟล็กของ EINT0, EINT1 และ EINT2	อ่าน/เขียน
0xE01FC144	EXTWAKE	External Interrupt Wakeup Register เก็บค่าบิตที่ควบคุมว่าจะให้อินเตอร์รัปต์จากภายนอกนี้สามารถกระตุ้นซีพียูจาก Power mode หรือไม่	อ่าน/เขียน
0xE01FC148	EXTMODE	External Interrupt Mode Register ควบคุมให้แต่ละระดับจะตอบสนองต่อระดับลอกจิกหรือขอบสัญญาณ	อ่าน/เขียน
0xE01FC14C	EXTPOLAR	External Interrupt Polarity Register ควบคุมว่าแต่ละขาจะตอบสนองระดับลอกจิก 0 หรือ1 หรือตอบสนองต่อขอบขาขึ้นขอบขาลงของสัญญาณ	อ่าน/เขียน

ตารางที่ 2.2.19 ค่าประจำบิตของรีจิสเตอร์ EXTINT

ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EINT0	เมื่อขา EINT0 ทำงานมีระดับลอจิกหรือมีขอบของสัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1 ให้มันเป็นการล้างค่า ยกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าลอจิกหยุดการทำงานก่อน ขาที่เป็น EINT0 คือ P0.1 และ P0.16	0
1	EINT1	เมื่อขา EINT1 ทำงานมีระดับลอจิกหรือมีขอบของสัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1 ให้มนเป็นการล้างค่า ยกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าบอจิกหยุดการทำงานก่อน ขาที่เป็น EINT1 คือ P0.3 และ P0.14	0
2	EINT2	เมื่อขา EINT2 ทำงานมีระดับลอจิกหรือมีขอบของสัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1 ให้มนเป็นการล้างค่า ยกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าบอจิกหยุดการทำงานก่อน ขาที่เป็น EINT2 คือ P0.7 และ P0.15	0
3	EINT3	เมื่อขา EINT3 ทำงานมีระดับลอจิกหรือมีขอบของสัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อเขียนค่า 1 ให้มนเป็นการล้างค่า ยกเว้นกรณีที่ทำงานตามระดับลอจิกต้องรอให้ค่าบอจิกหยุดการทำงานก่อน ขาที่เป็น EINT3 คือ P0.9, P0.20 และ P0.30	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ตารางที่ 2.2.20 ค่าประจำบิตของรีจิสเตอร์ EXTMODE

ค่าบิตของ EXTWAKE	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EXTWAKE0	โหมดการทำงานของ EINT0 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
1	EXTWAKE1	โหมดการทำงานของ EINT1 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
2	EXTWAKE2	โหมดการทำงานของ EINT2 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
3	EXTWAKE3	โหมดการทำงานของ EINT3 เมื่อมีค่าเป็น 0 ทำงานที่ระดับสัญญาณ เมื่อมีค่าเป็น 1 ทำงานที่ขอบของสัญญาณ	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ตารางที่ 2.2.21 ค่าประจำบิตของรีจิสเตอร์ EXTPOLAR

ค่าบิตของ EXTPOLAR	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EXTPOLAR0	เป็น 0 EINT0 ทำงานที่ลอจิก 0 หรือขอบขาลงของสัญญาณ เป็น1 EINT0 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
1	EXTPOLAR1	เป็น 0 EINT1 ทำงานที่ลอจิก 0 หรือขอบขาลงของสัญญาณ เป็น1 EINT1 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0

ค่าบิตของ EXTPOLAR	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
2	EXTPOLAR2	เป็น 0 EINT2 ทำงานที่ลอจิก 0 หรือขอบขาลงของสัญญาณ เป็น1 EINT2 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
3	EXTPOLAR3	เป็น 0 EINT3 ทำงานที่ลอจิก 0 หรือขอบขาลงของสัญญาณ เป็น1 EINT3 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ตารางที่ 2.2.22 ค่าประจำบิตของรีจิสเตอร์ EXTPILAR

ค่าบิตของ EXTMODE	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
0	EXTMODE0	เป็น 1 สัญญาณที่ขา EINT0 สามารถกระตุ้นให้ออกจาก Power Down ได้	0
1	EXTMODE1	เป็น 1 สัญญาณที่ขา EINT1 สามารถกระตุ้นให้ออกจาก Power Down ได้	0
2	EXTMODE2	เป็น 1 สัญญาณที่ขา EINT2 สามารถกระตุ้นให้ออกจาก Power Down ได้	0
3	EXTMODE3	เป็น 1 สัญญาณที่ขา EINT0 สามารถกระตุ้นให้ออกจาก Power Down ได้	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ตารางที่ 2.2.23 รีจิสเตอร์ที่เกี่ยวข้องกับ UART0

ชื่อ	ความหมาย	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0	การติดต่อ	ค่าหลังรีเซต	แอดเดรส
U0RBR	Receiver Buffer Register	MSB	อ่านข้อมูล						LSB	อ่าน	ไม่กำหนด	0xE000C000 DLAB=0
U0THR	Transmit Holding Register	MSB	เขียนข้อมูล						LSB	เขียน	ไม่กำหนด	0xE000C000 DLAB=0
U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	อ่าน/เขียน	0	0xE000C004 DLAB=0
U0IIR	Interrupt ID Register	FIFOs Enable		0	0	IIR3	IIR2	IIR1	IIR0	อ่าน	0x01	0xE000C008
U0FCR	FIFO Control Register	Rx Trigger		สงวนไว้		-	TX FIFO Reset	Rx FIFO Reset	FIFO Enable	เขียน	0	0xE000C008
U0LCR	Line Control Register	DLAB	Set Break	Stuck Parity	Even Parity Select	Parity Enable	Number of Stop Bit	World Length Select		อ่าน/เขียน	0	0xE000C00C
U0LSR	Line Status Register	Rx FIFO Error	TE MT	TH RE	BI	FE	PE	OE	DR	อ่าน	0X60	0xE000C014
U0SCR	Scratch Pad Register	MSB LSB								อ่าน/เขียน	0	0xE000C01C
U0DLL	Divisor Latch LSB	MSB LSB								อ่าน/เขียน	0x01	0xE000C000 DLAB=1
U0DLM	Divisor Latch MSB	MSB LSB								อ่าน/เขียน	0	0xE000C004 DLAB=1

ตารางที่ 2.2.24 ค่าประจำบิตของรีจิสเตอร์ UART0 Line Control Register(U0LCR)

ค่าของบิต U0LCR	หน้าที่	ความหมาย	ค่าหลังรีเซ็ต
1:0	World Length Select	ตัวอักษรขนาด 5 บิต ตัวอักษรขนาด 6 บิต ตัวอักษรขนาด 7 บิต ตัวอักษรขนาด 8 บิต	0
2	Stop Bit Select	0 Stop bit 1 บิต 1 Stop Bit 2 บิต(1.5บิต ถ้า U0LCR[1:0]=00)	0
3	Parity Enable	0 ยกเลิกการเพิ่มพาริตี้บิตและการตรวจพาริตี้ 1 ใช้งานการเพิ่มพาริตี้บิตและการตรวจพาริตี้	0
5:4	Parity Select	Odd parity Even Parity ให้พาริตี้บิตมีค่าเป็น 1 ตลอดเวลา ให้พาริตี้บิตมีค่าเป็น 0 ตลอดเวลา	0
6	Break Control	ยกเลิกการหยุดการส่ง อนุญาตให้หยุดการส่งให้ ขาที่เอาต์พุตของ UART0 TxD จะมีค่าลอจิก 0	0
7	Divisor Latch Access Bit	ไม่อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch	0

ตารางที่ 2.2.25 แสดงค่าประจำบิตของ UART0 Line Status Register (U0LSR)

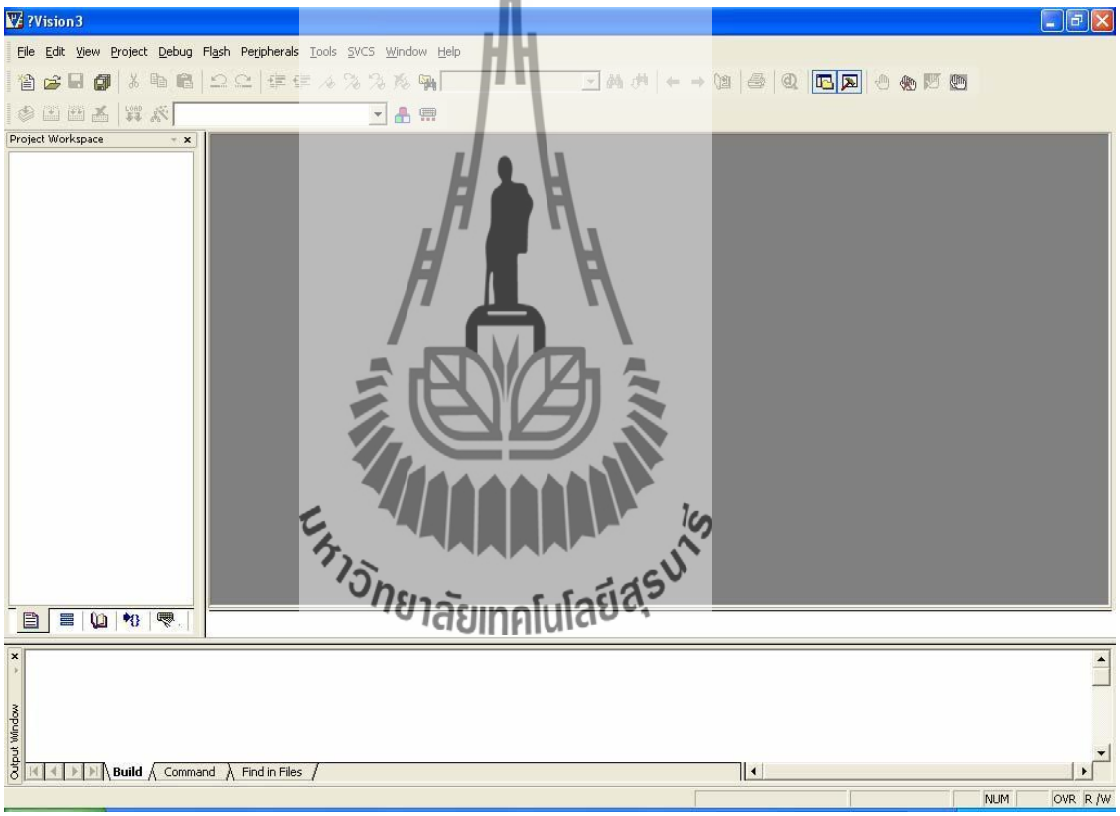
ค่าบิตของ U0LSR	หน้าที่	ความหมาย	ค่าหลังรีเซต
0	Receiver Data Ready (RDR)	0 : U0RBR ว่าง 1 : U0RBR มีรับข้อมูลที่ถูกต้อง U0LSR0 เป็น 1 เมื่อ U0RBR มีค่าตัวอักษรที่ยังไม่อ่าน และถูกล้างค่าเมื่อ UART0 RBR FIFO ว่าง	0
1	Overrun Error (OE)	0 : ไม่มี Overrun error 1 : มี Overrun error เกิดขึ้น Overrun error เกิดขึ้นเมื่อ UART0 RSR ได้รับตัวอักษรตัวใหม่ในขณะที่บัฟเฟอร์ FIFO เต็ม ในกรณีนี้ข้อมูลตัวใหม่ใน UART0 RSR จะสูญหายไป เมื่ออ่านค่าของ U0LSR จะล้างค่าบิตนี้	0
2	Parity Error (PE)	0 : ไม่มี Parity Error 1 : มี Parity Error เกิดขึ้น ใช้ตรวจสอบว่าข้อมูลที่รับมาใน UART0 RBR FIFO มีการผิดพลาดที่พาริตี้บิตหรือไม่ เมื่ออ่านค่าของ U0LSR จะล้างค่าของบิตนี้	0
3	Framing Error (FE)	0 : ไม่มี Framing Error 1 : มี Framing Error เกิดขึ้น เมื่อ Stop bit ของข้อมูลที่รับมามีค่าเป็น 0 แสดงว่าเกิดการผิดพลาดที่เฟรมข้อมูลในขณะที่เกิดการผิดพลาดที่เฟรมข้อมูลนี้ UART0 จะพยายามรับข้อมูลโดยนำ Stop bit ที่ผิดพลาดนี้มาใช้เป็น Start bit ข้อมูลตัวถัดไป ซึ่งอาจจะอ่านข้อมูลได้ถูกต้องหรือไม่ถูกต้อง	0
4	Break Interrupt (BI)	0 : ไม่ใช้ Break Interrupt 1 : ใช้ Break Interrupt เมื่อ RxD0 ได้รับข้อมูลที่เป็น 0 ทุกบิต (start, data, parity, stop) จะเกิด Break Interrupt ภาครับจะหยุดทำงานจนกว่าจะได้รับข้อมูลที่เป็น 1 ทุกบิตอีกครั้งหนึ่ง	0

ค่าบิตของ U0LSR	หน้าที่	ความหมาย	ค่าหลังรีเซต
5	Transmitter Holding Register Empty (THRE)	0 : U0THR มีข้อมูลที่ถูกต้อง 1 : U0THR ว่าง บิต THRE จะถูกเซ็ตค่าเป็น 1 ทันทีที่พบว่า UART0 THR ว่าง และถูกล้างค่าทันทีมีการเขียนค่าไปยัง U0THR	1
6	Transmitter Empty (TEMT)	0 : U0THRและ/หรือ U0TSR มีข้อมูลที่ถูกต้อง 1 : U0THR และ U0TSR ว่าง บิตนี้จะถูกล้างค่าเมื่อ U0THR หรือ U0TSR มีข้อมูลที่ถูกต้อง	1
7	Error in Rx FIFO (RXFE)	0 : ข้อมูลที่เก็บใน UART0 RBR FIFO ไม่ผิดพลาด 1 : ข้อมูลที่เก็บใน UART0 RBR FIFO อย่างน้อยหนึ่งตัวมีการผิดพลาด	0

การใช้ Keil uVision3 ในการสร้าง Project File ของ Keil ARM

ในที่นี้จะขอแสดงแนวทางการเขียนโปรแกรมภาษาซีโดยใช้ Keil-CARMในการแปลคำสั่งภายใต้โปรแกรม Text Editor ของ Keil (Keil uVision3) โดยจะขออธิบายถึงเฉพาะวิธีการกำหนดค่า Option สำหรับเชื่อมโยงคำสั่งในการสั่งแปลโปรแกรมด้วย Keil-CARM ผ่านทาง Keil uVision3 เท□านั้น ส□วนรายละเอียดคำสั่งและการใช้งานฟังก์ชันต□างๆในการเขียนโปรแกรมของ Keil-CARM

นั้นขอให□ผู□ใช□ศึกษาจากคู□มือคำสั่งของ Keil-CARM เอง โดยวิธีการกำหนดค□าตัวเลือกของ Keil uVision3 ให□ใช□งานกับ Keil-CARM นั้นมีขั้นตอนพอสังเขปดังนี้คือ

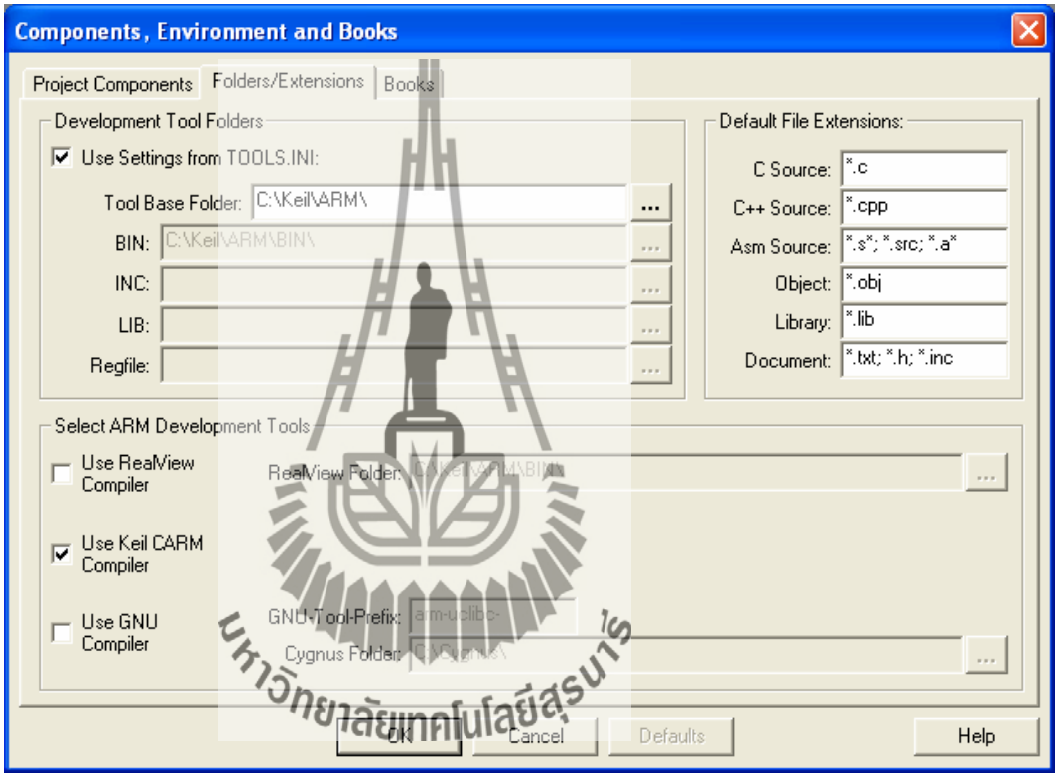


1. เป□ดโปรแกรม Keil uVision3 ซึ่งเป□นโปรแกรม Text Editor ของ Keil-CARM ใช□สำหรับใช□ในการเขียนโปรแกรมที่เป□น Source Code ภาษาซี โดยจะมีลักษณะดังรูป

รูปที่ 5.1 หน้าต่างของโปรแกรม Keil uVision3

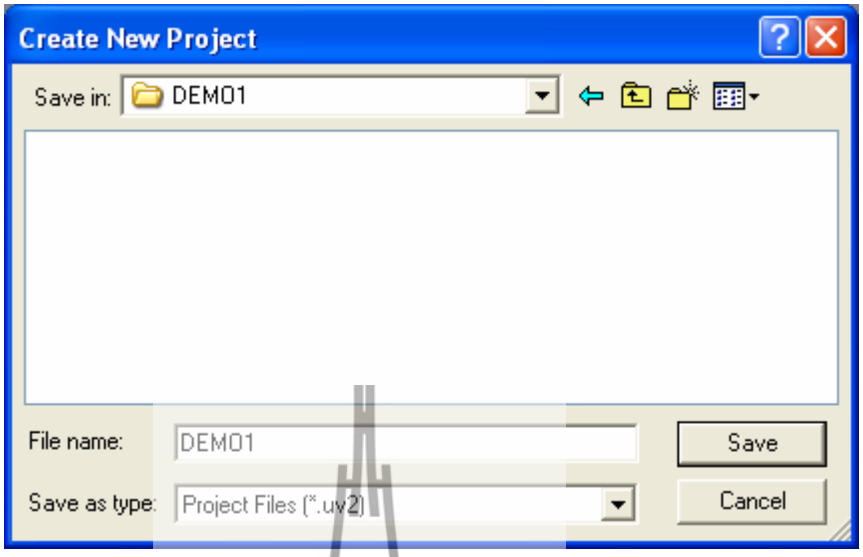
2. ทำการกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3 ให้ใช้งานกับโปรแกรม Keil uVision3 และ Keil-CARM โดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง Project ➔ Components

,Environment, Books… จากนั้นให้เลือกค่าตัวเลือกสำหรับกำหนดการใช้งาน Complier จากหัวข้อ Select ARM Development Tools ซึ่งจะมีค่าตัวเลือกอยู่ 3 แบบ คือ Use Keil-CARM Tools ,Use GNU Tools และ Use ARM Tools โดยให้เลือกเป็น “Use Keil ARM Tools” จากนั้นให้ทำการกำหนดตำแหน่ง Folder สำหรับเก็บค่าตัวเลือกการทำงานของโปรแกรม Keil ARM ซึ่งตามปรกติแล้วจะเป็น “C:\Keil\ARM\” แต่ถ้าติดตั้ง Keil ไว้ที่อื่นก็ต้องปรับเปลี่ยนให้ถูกต้องตามความเป็นจริงด้วยดังรูป



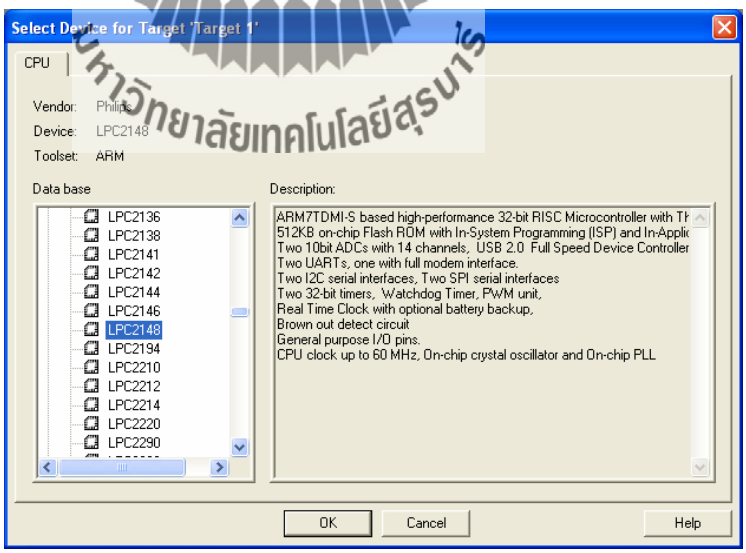
รูปที่ 5.2 การกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3

3. ทำการสร้าง Project File ขึ้นมาใหม่ โดยเรียกเมนูคำสั่ง Project ➔ New Project จากนั้นให้เลือกกำหนดหรือสร้างตำแหน่ง Folder ที่จะบันทึก Project File พร้อมกับกำหนดชื่อ Project File ตามต้องการ เช่น ถ้าต้องการสร้างProject File ชื่อ DEMO1 โดยเก็บไว้ใน Folder ชื่อ DEMO1 ก็สามารถกำหนดตำแหน่ง Folder และชื่อ Project Fileได้เอง โดยเมื่อกำหนดชื่อในช่อง File name แล้วให้เลือก save เพื่อบันทึก Project File ไว้ ดังรูป



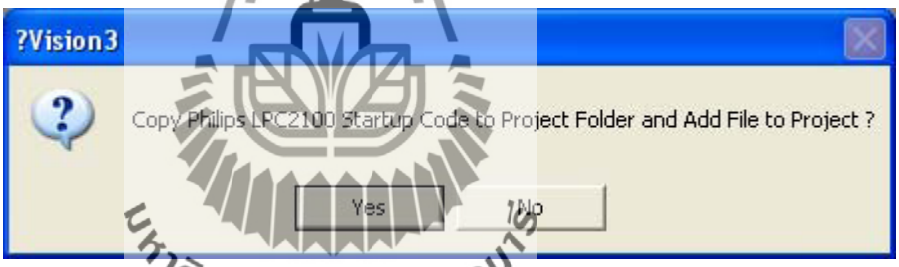
รูปที่ 5.3 หน้าต่างของการสร้าง Project ใหม่

หลังจากกำหนดชื่อและสั่ง Save Project File แล้ว โปรแกรมจะรอให้ผู้ใช้ทำการกำหนดเบอร์ MCU ที่จะใช้งานใน Project ที่สั่ง Save นั้น ซึ่งในกรณีที่ใช้งานกับบอร์ด CP-JR ARM7 USB-LPC2148 นั้น ให้เลือกกำหนดเบอร์ของ MCU เป็น LPC2148 ของ Philips แล้วเลือก OK ดังรูป



รูปที่ 5.4 การกำหนดเบอร์ MCU ที่จะใช้งาน

หลังจากเลือกกำหนดเบอร์ของ MCU เป็นที่เรียบร้อยแล้ว ในขั้นตอนนี้โปรแกรมจะรอให้ผู้ใช้ยืนยันว่าต้องการจะทำการ Copy ไฟล์ Startup ของ Keil เพื่อใช้งานกับ MCU ของ Philips มาใช้ใน Project ด้วยหรือไม่ โดยStartup ไฟล์จะเป็นส่วนของการกำหนดค่าเริ่มต้นการทำงานให้กับ MCU เช่น การกำหนดค่า Stack และการกำหนดค่าการทำงานให้กับ Phase-Lock-Loop ต่างๆ ก่อนที่จะเริ่มต้นทำงานตามโปรแกรมที่เราเขียนขึ้น ไม่เช่นนั้นแล้วโปรแกรมที่เราเขียนขึ้นมานั้นจะต้องเพิ่มคำสั่งในการเตรียมการทำงานส่วนเหล่านี้ให้ MCU เองทั้งหมดแต่เนื่องจากไฟล์ Startup ของ Keil-ARM นั้น เป็นไฟล์ภาษา แอสแซมบลี้(Assembly) ซึ่งกำหนดค่าการทำงานไว้กับชุดพัฒนาของ Keil เอง ดังนั้นข้อกำหนดและการกำหนดค่าบางอย่างจะมีความแตกต่างกันอยู่กับค่าที่ต้องการสำหรับบอร์ด “CP-JR ARM7 USB-LPC2148” ไม่สามารถใช้งานไฟล์ Startup ได้ทันทีต้องมีการแก้ไขค่าตัวเลือกใหม่ดังนั้นก่อนที่จะใช้โปรแกรม Keil-CARM ในการแปลคำสั่งให้นั้นผู้ใช้จะต้องเข้าไปแก้ไขไฟล์ Startup ใหม่โดยต้องกำหนดรูปแบบให้ถูกต้องตรงกับความต้องการของบอร์ดด้วย ดังนั้นในที่นี้ขอแนะนำให้เลือก “No” เพื่อไม่ให้ Keil uVision3 ทำการ Copy ไฟล์ Startup ของ Keil-CARM มาใช้ใน Project นี้ด้วย

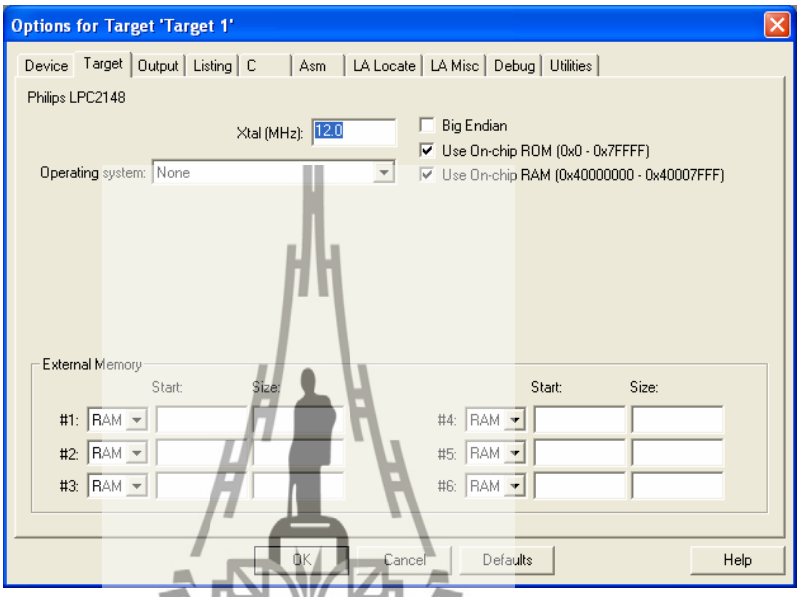


รูปที่ 5.3 แสดงหน้าต่างการกำหนด Startup File

4. ให้ทำการ Copy File ชื่อ “Startup.s” ที่ทาง ผู้จัดทำจัดเตรียมไว้ใน CD-ROM ชื่อ “Startup.s” มาไว้ในตำแหน่ง Folder เดียวกันกับ Project File ที่สร้างขึ้นมาใหม่นี้โดยไฟล์ “Startup.s” จะเป็นไฟล์ซึ่งบรรจุคำสั่งภาษาแอสแซมบลี้ของ ARM7 สำหรับทำหน้าที่กำหนดค่าเริ่มต้นการทำงานที่จำเป็นให้กับ MCU ซึ่งได้แก่การ กำหนดค่า Stack ให้กับ MCU การ Initial Phase-Lock-Loopการกำหนดค่าให้กับ MAM Function และการกำหนดตำแหน่ง Vector ต่างๆของ MCU สำหรับใช้งานร่วมกับบอร์ด “CP-JR ARM7 USB-LPC2148” ซึ่งถ้าสั่ง Add ไฟล์ “Startup.s” จาก Keil หรือ Copy ไฟล์ดังกล่าวมาจากแหล่งอื่นๆ อาจมีการทำงานของโปรแกรมใน Startup ไม่เหมือนกัน ซึ่งจะส่งผลต่อการทำงานของโปรแกรมด้วย

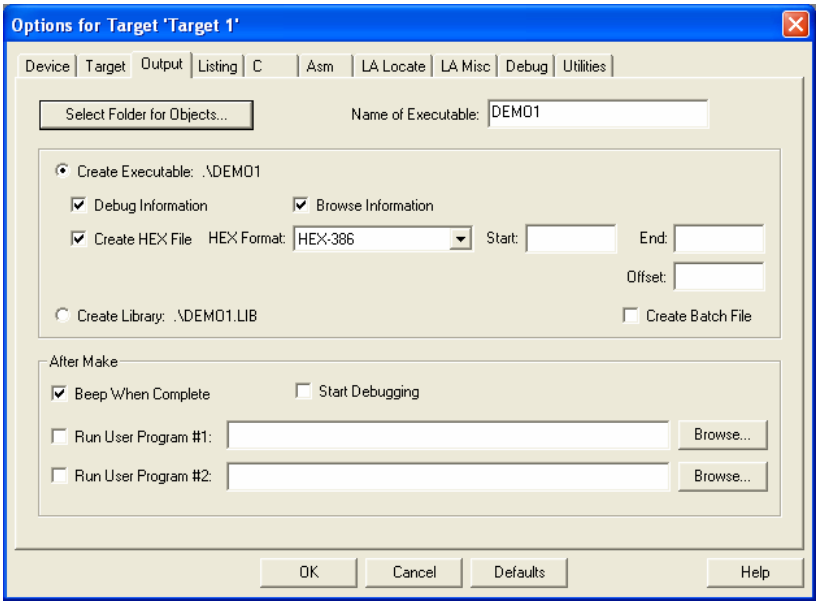
5. ให้ทำการกำหนดค่า Option ของ Project File โดยเลือกเมนูคำสั่ง Project ➔ Option for Target 'Target 1' จากนั้นเลือกที่ Tab ของ Target เพื่อกำหนดค่าของ MCU Target โดยให้กำหนดดังนี้

5.1 X-TAL ให้กำหนดเป็น 12 MHz พร้อมกับเลือกกำหนดให้ใช้หน่วยความจำที่มีอยู่ใน MCU เป็นเงื่อนไข ในการแปลโปรแกรมของ Keil-CARM ด้วย ดังรูป



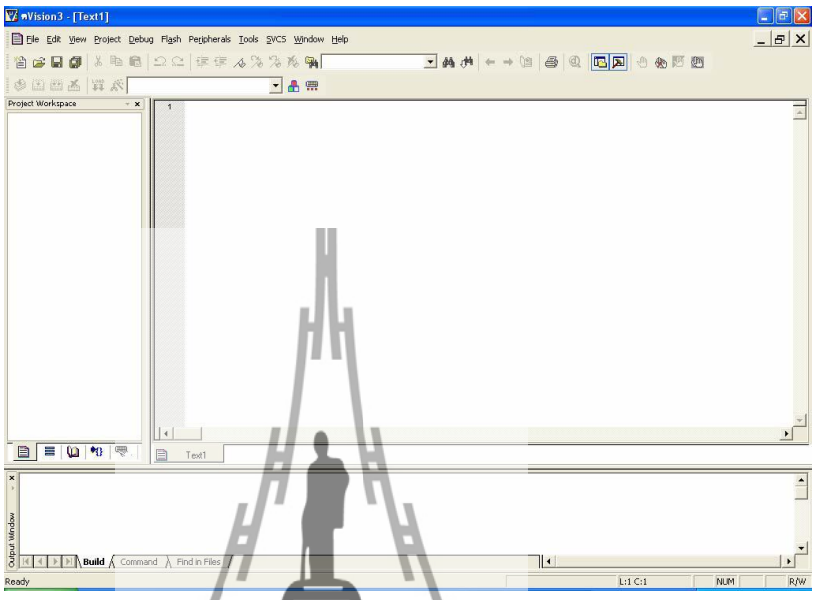
รูปที่ 5.6 การกำหนดค่า XTAL ของ MCU ที่ใช้งาน

5.2 เลือกที่ Output ให้เลือกคลิกเมาส์ที่ค่าตัวเลือก Create HEX File พร้อมกับเลือกกำหนดรูปแบบของ Hex ให้เป็นแบบ HEX-386 แล้วเลือก OK ดังรูป



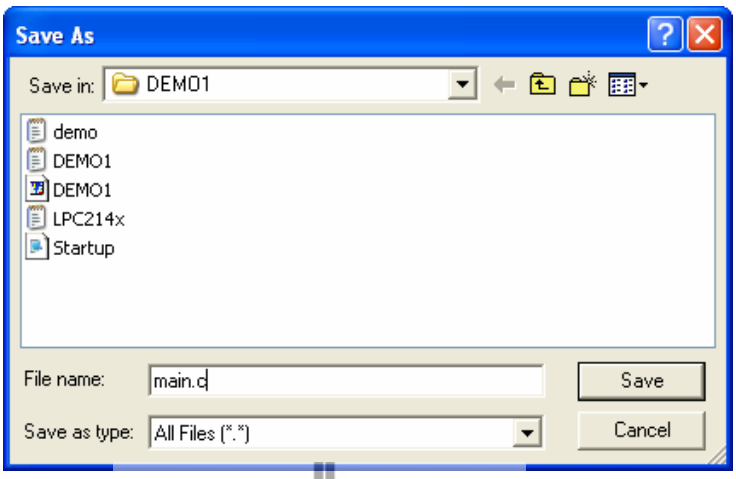
รูปที่ 5.7 การกำหนดค่าตัวเลือก Create HEX File

6. เริ่มต้นเขียน Source Code ภาษาซี โดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง File ➔ New… ซึ่งจะได้พื้นที่ในการเขียน Text File เกิดขึ้นมา โดยในครั้งแรกจะกำหนดชื่อตามค่า Default เป็น “Text1” ดังรูป



รูปที่ 5.8 หน้าต่างของโปรแกรม Keil uVision3 หลังจากตั้งค่าต่าง ๆแล้ว

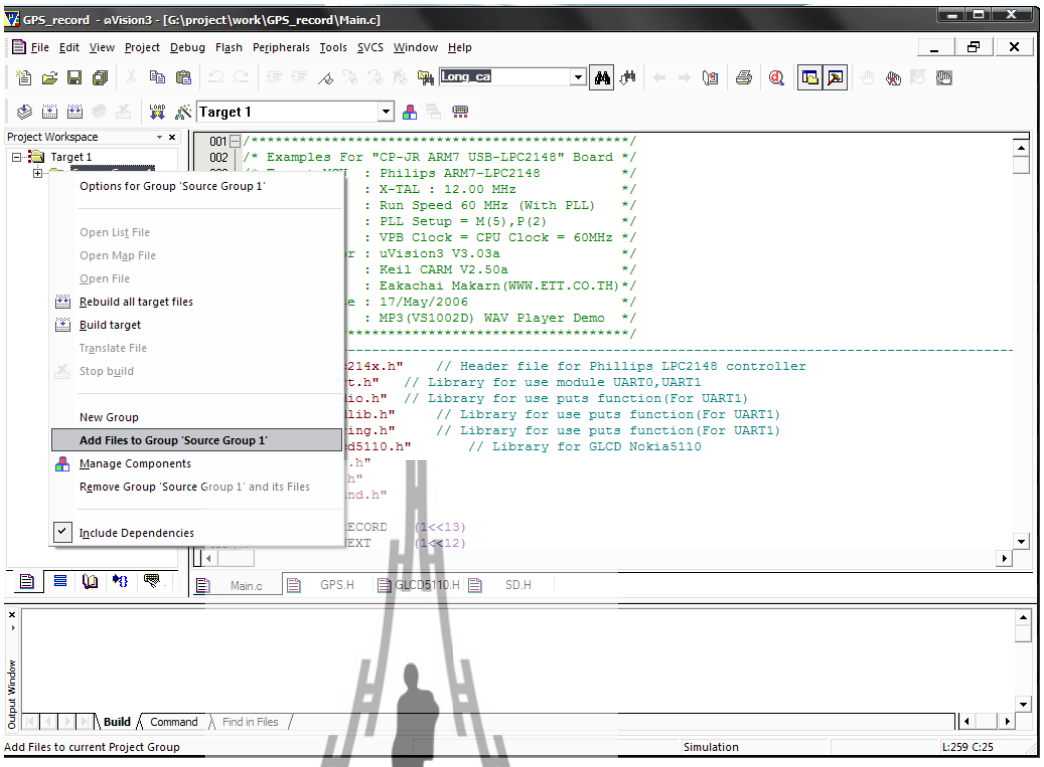
ในขั้นตอนนี้ให้ทำการพิมพ์ Source Code ภาษาซี ตามข้อกำหนดของ Keil-CARM ในพื้นที่เขียนโปรแกรมตามต้องต้องการ หลังจากพิมพ์คำสั่งภาษาซีเสร็จเรียบร้อยตามต้องการแล้ว ให้สั่ง Save ไฟล์ดังกล่าว โดยต้องกำหนดเป็นไฟล์ที่มีนามสกุลเป็น “.C” ในที่นี้ขอแนะนำให้สั่ง Save โดยใช้คำสั่ง File ➔ Save As… แล้วกำหนดชื่อและนามสกุลของไฟล์เป็น .main.c” ดังรูป



รูปที่ 5.9 หน้าต่างของการ save ไฟล์

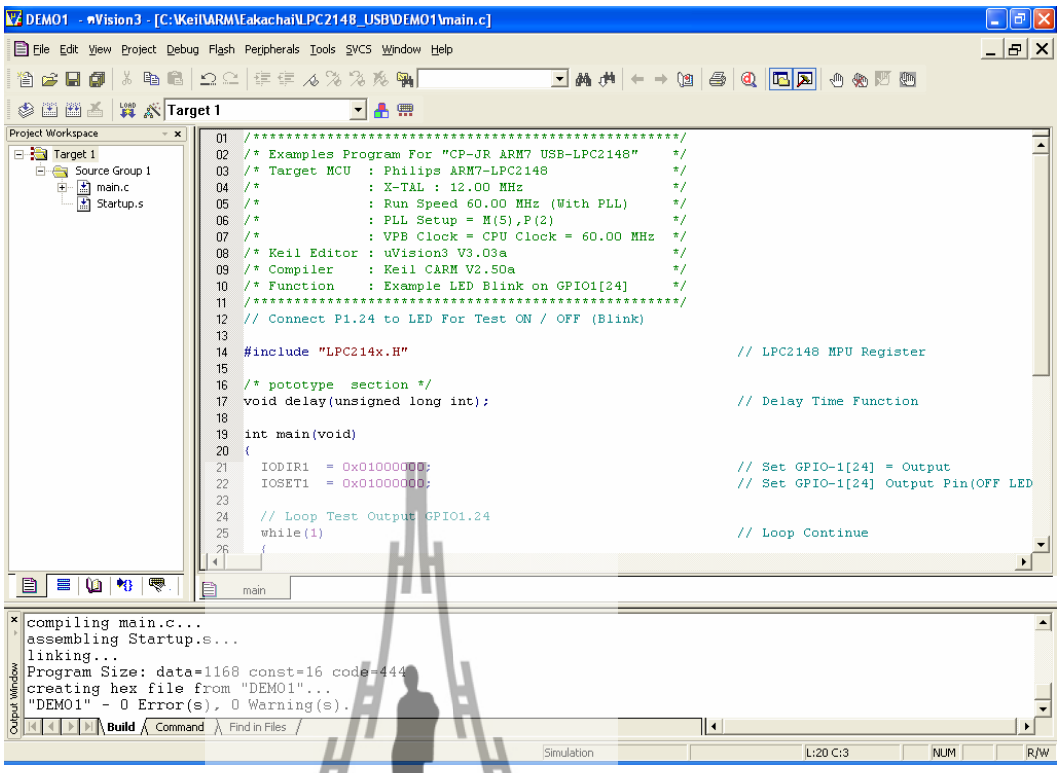
ซึ่งหลังจากที่สั่ง Save ไฟล์เป็น “main.c” แล้วจะเห็นว่าลักษณะสีของตัวอักขระต่างๆในโปรแกรมจะเกิดการเปลี่ยนแปลงไปตามหน้าที่ เช่น Comment, ตัวแปร และ คำสั่ง เป็นต้น ซึ่งส่วนนี้เป็นข้อดีของ Keil uVision3 ซึ่งสามารถแยกและแสดงตัวอักขระได้อย่างเป็นหมวดหมู่ ทำให้ง่ายต่อการอ่านโปรแกรมด้วย

7. ทำการสั่ง Add File ต่างๆเข้ากับ Project File โดยให้เลือกคลิกเมาส์ที่ด้านซ้ายของหน้าต่าง จากนั้นให้เลือกที่ Add Files to Group “Source Group 1” แล้วเลือกที่ Add File ที่ต้องการจะเพิ่มเข้าไปใช้งานร่วมกับ Project File โดยในครั้งแรกให้เลือก Files of type เป็น “C Source files(*.c)” ซึ่งจะปรากฏชื่อไฟล์ต่างๆที่เป็น Source Code ภาษาซีให้เห็น โดยในที่นี้ให้เลือกคลิกเมาส์ที่ไอคอนของไฟล์ชื่อ “main.c” แล้วเลือก Add เพื่อสั่งเพิ่มไฟล์ชื่อ “Startup.s” เข้าไปรวมกับ Project Files ที่เราสร้างไว้



รูปที่ 5.10 การสั่ง Add File ต่างๆเข้ากับ Project File

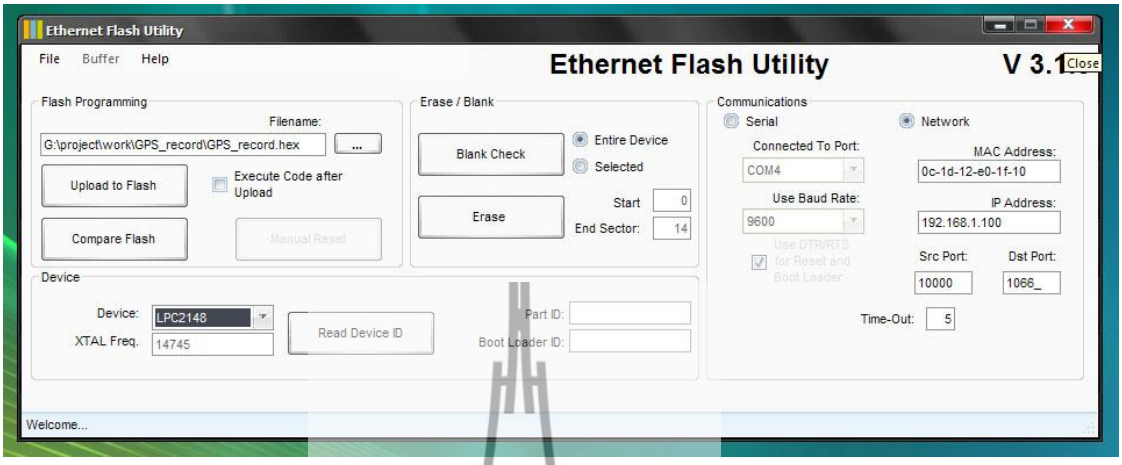
8. ให้ทำการสั่งแปลโปรแกรมที่เราเขียนขึ้นเรียบร้อยแล้ว โดยให้คลิกเมาส์ที่เมนูคำสั่ง Projects ➔ Rebuild all target files ซึ่งโปรแกรม Keil uVision3 จะทำการสั่งให้โปรแกรม Keil-CARM ทำการแปลคำสั่งให้ทันที ซึ่งหลังจากสั่งแปลโปรแกรมแล้วได้ผลถูกต้องและไม่เกิดข้อผิดพลาดใดๆขึ้น (0 Error และ 0 Warning) ก็ได้ Hex File ซึ่งมีชื่อเหมือนกันกับชื่อของ Project File ที่สร้างไว้ ซึ่งผู้ใช้สามารถนำ Hex File ดังกล่าวไปทำการ Download ให้กับ MCU ได้ทันที



รูปที่ 5.11 การแปลงเป็น Hex File

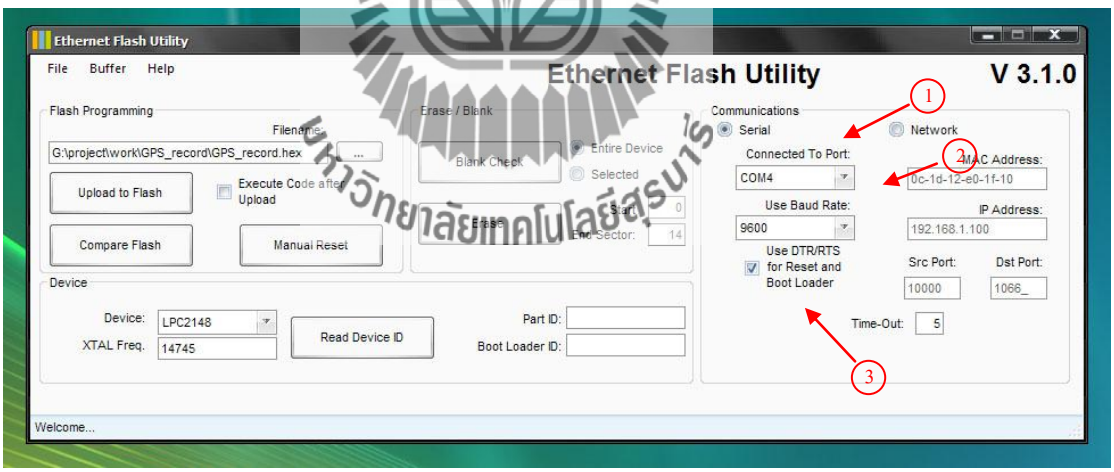
การโหลดโปรแกรมทดสอบ

เปิดโปรกแกรม Ethernet Flash Utility จะได้หน้าต่างดังรูป



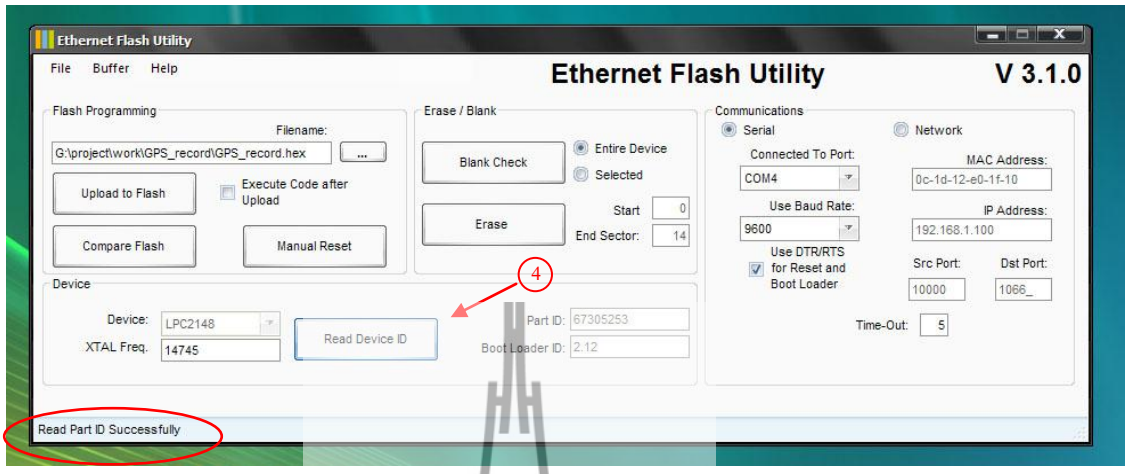
รูปที่ 5.12 หน้าต่างโปรแกรม Ethernet Flash Utility

เลือกรูปแบบการเชื่อมต่อเป็นแบบ Serial ➔ เลือก COM ให้ถูกต้อง ➔ Baud Rate 9600 ➔ เลือกที่ช่อง Use DTR/RTS for Reset and Boot Loader



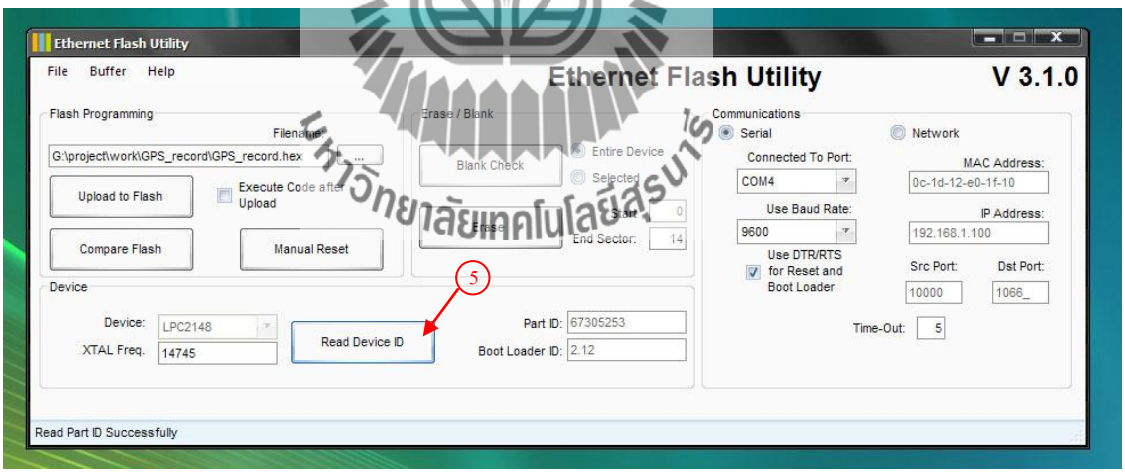
รูปที่ 5.13 หน้าต่างรูปแบบการเชื่อมต่อ

กด Switch Reset ที่ บอร์ดไมโครคอนโทรลเลอร์ แล้วเลือกที่ Read Device ID หากเชื่อมต่อกับไมโครคอนโทรลเลอร์สำเร็จจะแสดงข้อความ Read Part ID Successfully

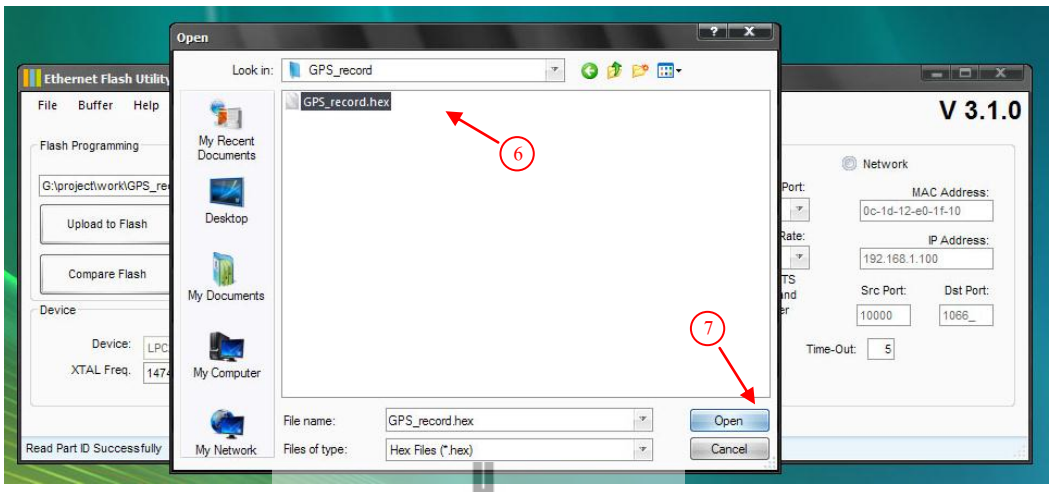


รูปที่ 5.14 หน้าต่างการเชื่อมต่อสำเร็จ

เมื่อเชื่อมต่อเรียบร้อยแล้ว ทำการโหลดไฟล์ .HEX โดยเลือกที่ Brown เลือกไฟล์ .HEX ที่ต้องการ แล้วเลือก Open ตามรูปที่ 5.15 และรูปที่ 5.16

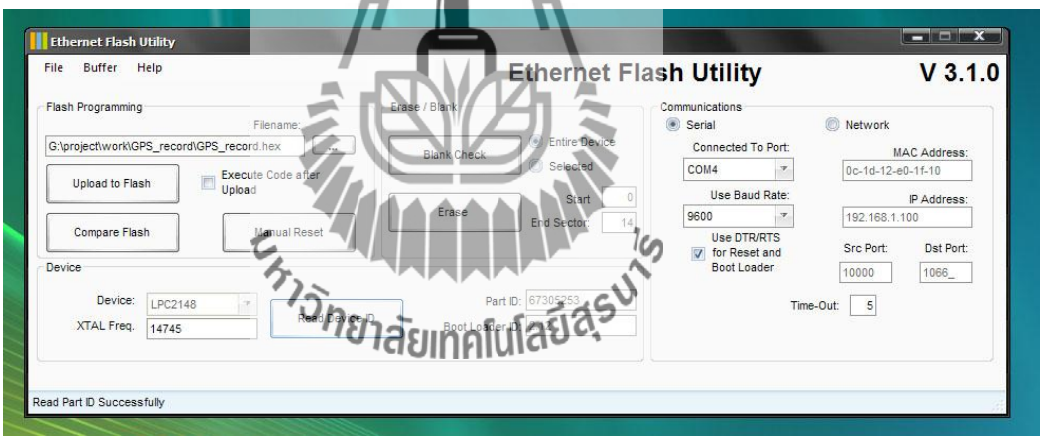


รูปที่ 5.15 หน้าต่างการโหลดไฟล์ .HEX

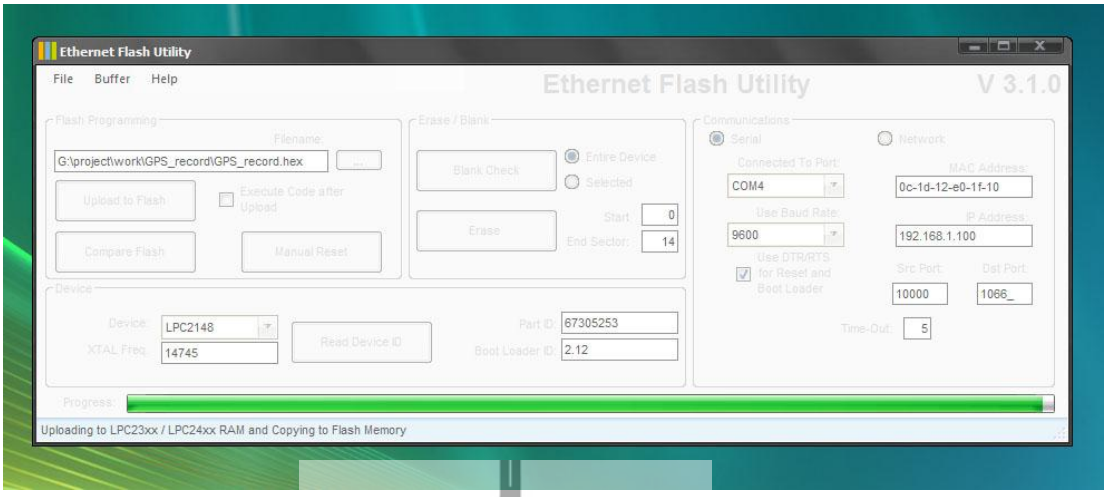


รูปที่ 5.16 หน้าต่างการโหลดไฟล์ .HEX

เลือกที่ Upload to flash และรอจนกว่าโปรแกรมจะทำการโหลดเสร็จตามรูปที่ 5.17 และรูปที่ 5.18



รูปที่ 5.17 หน้าต่างการโหลดไฟล์ .HEX



รูปที่ 5.18 หน้าต่างการโหลดไฟล์ .HEX