



ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง  
(FM Broadcast Status Checking System)

โดย

นายพิพัฒน์	พงษ์ศรีเพชร	รหัสนักศึกษา	B5106745
นางสาวสิริกัญญา	ประไพทรัพย์	รหัสนักศึกษา	B5110049
นายเสรี	แก้วตา	รหัสนักศึกษา	B5119431

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงการวิศวกรรมโทรคมนาคม

และวิชา 427494 โครงการวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

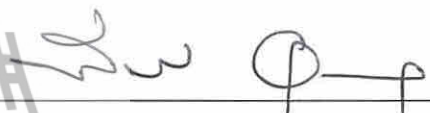
ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2554

ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง  
(FM Broadcast Status Checking System)

คณะกรรมการสอบโครงการ


---

(ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ ทองทา)  
กรรมการ/อาจารย์ที่ปรึกษาโครงการ



---

(ผู้ช่วยศาสตราจารย์ ดร.พีระพงษ์ อุฑารสกุล)  
กรรมการ



---

(อาจารย์ ดร. สมศักดิ์ วาณิชอนันต์ชัย)  
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นำรายงานโครงการฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม รายวิชา 427499 โครงการวิศวกรรมโทรคมนาคม และรายวิชา 427494 โครงการศึกษาวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2554

โครงการงาน	ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง (FM Broadcast Status Checking System)		
ผู้ดำเนินงาน	1. นายพิพัฒน์ พงษ์ศรีเพชร รหัสประจำตัว	B5106745	
	2. นางสาวสิริกัญญา ประไพทรัพย์ รหัสประจำตัว	B5110049	
	3. นายเสรี แก้วตา รหัสประจำตัว	B5119431	
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ ทองทา		
สาขาวิชา	วิศวกรรมโทรคมนาคม		
ภาคการศึกษาที่	3/ 2554		

### บทคัดย่อ

เนื่องจากสถานีวิทยุกระจายเสียง จะต้องมีการกระจายเสียงอย่างต่อเนื่องตลอดเวลา จึงอาจเกิด  
 ความผิดปกติอันเกิดจากเครื่องส่งวิทยุกระจายเสียงมีค่าพารามิเตอร์ต่างๆ ที่เกินขีดจำกัด ทำให้จะต้อง  
 มีบุคลากรคอยดูแลเครื่องส่งวิทยุกระจายเสียงอยู่ด้วย ซึ่งในการตรวจสอบค่าพารามิเตอร์จำเป็นที่  
 จะต้องใช้เครื่องมือตรวจสอบอีกหลายชิ้นตามชนิดของพารามิเตอร์ เราจึงนำปัญหาดังกล่าวมาคิด  
 พัฒนาเป็นระบบ ตรวจสอบ สถานะเครื่องส่งวิทยุกระจายเสียง นี้ขึ้น เพื่อให้บุคลากรผู้ดูแลเครื่องส่ง  
 วิทยุกระจายเสียงนั้น ทราบถึงค่าพารามิเตอร์ต่างๆ ได้โดยไม่ต้องเดินทางไปที่สถานีวิทยุด้วย  
 ตนเอง ซึ่งระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงนี้สามารถตรวจสอบค่า Ratio (อัตราส่วน  
 ระหว่าง Reflect และ Forward) และอุณหภูมิของเครื่องส่งและส่งข้อมูลผ่านทางระบบ SMS เป็น  
 รายวันได้ นอกจากนี้หากผู้ดูแลต้องการทราบค่าพารามิเตอร์สามารถส่งข้อความผ่าน SMS เพื่อให้  
 ระบบตรวจสอบสถานะส่งข้อมูลกลับมา รวมถึงเมื่อค่าพารามิเตอร์เกิน จากค่าที่กำหนดไว้ ระบบก็  
 จะส่งข้อมูลถึงผู้ดูแลได้ด้วยเช่นกัน

## กิตติกรรมประกาศ

จากการที่คณะจัดทำรายงานได้รับมอบหมายให้ทำโครงการเรื่อง ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง (FM Broadcast Status Checking System) ส่งผลให้คณะจัดทำรายงานได้รับความรู้และประสบการณ์ต่างๆ เกี่ยวกับการเขียนโปรแกรมด้วยโปรแกรม Keil uVision3 เป็นอย่างมาก บัดนี้โครงการดังกล่าวพร้อมทั้งรายงานได้สำเร็จลงแล้ว ทั้งนี้ด้วยความร่วมมือและสนับสนุนจากบุคคลต่างๆ ดังนี้

1. ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ ทองทา (อาจารย์ที่ปรึกษาโครงการ)
2. นายปัญญา หันตุลา (นักศึกษาบัณฑิตศึกษา

สาขาวิชาวิศวกรรมโทรคมนาคม)

ข้าพเจ้าใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูลและเป็นที่ปรึกษาในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับพื้นฐานการใช้งานโปรแกรม ซึ่งข้าพเจ้าขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

นายเสรี

แก้วตา



นายพิพัฒน์

พงษ์ศรีเพชร

นางสาวศิริกัญญา

ประไพทรัพย์

## สารบัญ

เรื่อง	หน้า
บทคัดย่อ	ก
กิตติกรรมประกาศ	ข
สารบัญ	ค
สารบัญรูปภาพ	ฉ
สารบัญตาราง	ญ
บทที่ 1 บทนำ	
1.1 ความเป็นมา	1
1.2 หลักการและเหตุผล	1
1.3 วัตถุประสงค์	2
1.4 ขอบเขตการทำงาน	2
1.5 การเลือกใช้ซอฟต์แวร์	3
1.6 ขั้นตอนการดำเนินการ	3
1.7 ประโยชน์ที่คาดว่าจะได้รับ	4
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง	
2.1 ไมโครคอนโทรลเลอร์ ARM7	5
2.1.1 ไมโครคอนโทรลเลอร์ ARM7 Philips LPC2148	5
2.1.2 ฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการทดลอง	10
2.1.3 การกำหนดค่าเริ่มต้นให้กับไมโครคอนโทรลเลอร์ ARM7	13
2.1.4 การต่อ GPIO เป็นเอาต์พุต	21
2.1.5 อินเทอร์รัปต์ไมโครคอนโทรลเลอร์ ARM7	27
2.1.6 การกำหนดค่าเริ่มต้นสำหรับ UART0	38
2.2 หน่วยความจำ EEPROM	45
2.2.1 คุณสมบัติของ I <sup>2</sup> C EEPROM 24LCXX	45
2.2.2 ขั้นตอนการเขียนข้อมูลลงใน I <sup>2</sup> C EEPROM 24LCXX	46
2.2.3 ขั้นตอนการอ่านข้อมูลใน I <sup>2</sup> C EEPROM 24LCXX	47

## สารบัญ (ต่อ)

เรื่อง	หน้า
2.3 เซนเซอร์ตรวจวัดอุณหภูมิ (DS18B20)	48
2.3.1 ข้อมูลทั่วไปของเทอร์โมมิเตอร์ DS18B20	48
2.3.2 การสื่อสารแบบ 1-wire	50
2.3.3 ขั้นตอนการเข้าใช้งาน DS18B20	52
2.3.4 ตัวอย่างการต่อใช้งาน DS18B20 กับไมโครคอนโทรลเลอร์	53
2.4 เครื่องวัด SWR และ Power	57
2.4.1 Wattmeter CN-101-L	57
2.4.2 คุณสมบัติของ Wattmeter CN-101-L	57
2.4.3 Description (ค่าเฉลี่ยของพลังงาน FM)	59
2.4.4 การใช้งานเครื่องวัด SWR และ Power	60
2.4.5 ข้อควรระวัง	60
2.5 วงจรขับ Relay	61
2.5.1 คุณลักษณะของรีเลย์	61
2.5.2 การใช้งานวงจรขับรีเลย์	61
2.6 อุปกรณ์แสดงผล (LCD 16x2)	63
2.6.1 การเชื่อมต่อทางฮาร์ดแวร์ และหน้าที่การใช้งาน	63
2.6.2 คำสั่งควบคุมการแสดงผล	64
2.7 GSM Modem	69
2.7.1 คุณสมบัติของ GSM Module (wave com)	69
2.7.2 ส่วนประกอบของ GSM Module	70
2.7.3 Power Supply Connector	71
2.7.4 การใช้งาน AT Command เพื่อสั่งงาน โมดูล Fastrack Supreme 20	72
2.7.5 การทดสอบการสั่งงาน โมดูล	73
2.7.6 การกำหนด Flow Chart	76
2.7.7 การ Setup และตรวจสอบค่า Configuration	77
2.7.8 ตัวอย่างการรับข้อความ SMS	78
2.7.9 ตัวอย่างการส่งข้อความ SMS	79

## สารบัญ (ต่อ)

เรื่อง	หน้า
บทที่ 3 การออกแบบโครงงาน	
3.1 การออกแบบฮาร์ดแวร์	81
3.1.1 ส่วนการอ่านค่าอุณหภูมิ	81
3.1.2 ส่วนการอ่านค่า forward & Reflect	82
3.1.3 ส่วนการควบคุมการเปิดปิดพัดลมระบายความร้อน	89
3.1.4 ส่วนการควบคุมการเปิดปิดเครื่องส่ง	90
3.1.5 ส่วนการแสดงผล	91
3.2 การออกแบบซอฟต์แวร์	92
3.3 การทำงานของโปรแกรม	92
3.3.1 Main Function	93
3.3.2 Check Time Function	94
3.3.3 Check Temperature Function	95
3.3.4 Check Forward & Reflect Function	96
3.4 อธิบายการทำงานของโครงงาน	97
บทที่ 4 การใช้งานโครงงาน	
4.1 การใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง	98
4.2 การทดลองการใช้งานโครงงาน	104
4.3 ผลการทดลองโครงงาน	106
บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ	
5.1 ปัญหา และอุปสรรค	112
5.2 สิ่งที่ได้รับจากการทำโครงงาน	112
ประวัติผู้จัดทำ	113
บรรณานุกรม	114
ภาคผนวก	115

## สารบัญรูปภาพ

รูปภาพ	หน้า
รูปที่ 2.1 Block Diagram LPC2148	6
รูปที่ 2.2 memory map LPC2148	8
รูปที่ 2.3 LPC2148 pinning	9
รูปที่ 2.4 ลักษณะของบอร์ด CP-JR ARM7 USB-LPC2148 EXP	12
รูปที่ 2.5 ผังวงจร PLL ภายในไมโครคอนโทรลเลอร์ ARM7	14
รูปที่ 2.6 Block Diagram วงจรกำเนิดสัญญาณภายใน ARM7	19
รูปที่ 2.7 ขาสัญญาณของ หน่วยความจำ EEPROM	45
รูปที่ 2.8 Control Byte Format	46
รูปที่ 2.9 Address Sequence Bit Assignments	46
รูปที่ 2.10 โครงสร้างและขาของ DS18B20 ตัวถังแบบ TO-92	48
รูปที่ 2.11 โครงสร้างรีจิสเตอร์ภายใน DS18B20	49
รูปที่ 2.12 โครงสร้างภายในรีจิสเตอร์ Temperature LSB และ MSB	49
รูปที่ 2.13 การต่อใช้งาน DS18B20	50
รูปที่ 2.14 การเริ่มการติดต่อสื่อสารแบบ 1-wire ด้วย Reset pulse และ Presence pulse	51
รูปที่ 2.15 การเขียนข้อมูลลง DS18B20	51
รูปที่ 2.16 การอ่านข้อมูลจาก DS18B20	52
รูปที่ 2.17 การต่อวงจรทดสอบการ ใช้งาน DS18B20	53
รูปที่ 2.18 ขั้นตอนการอ่าน ROM Code จาก DS18B20	54
รูปที่ 2.19 ผลการอ่าน ROM Code ขนาด 64 บิต จาก DS18B20	55
รูปที่ 2.20 ขั้นตอนการแปลง และอ่านอุณหภูมิจาก DS18B20	56
รูปที่ 2.21 ผลการอ่านอุณหภูมิจาก DS18B20	56
รูปที่ 2.22 เครื่องวัด SWR และ Power	57
รูปที่ 2.23 ตำแหน่งแสดงการอ่านค่าบนหน้าปัดเครื่อง Wattmeter	58
รูปที่ 2.24 จุดเชื่อมต่อบนเครื่อง Wattmeter (ด้านหน้า)	59
รูปที่ 2.25 จุดเชื่อมต่อบนเครื่อง Wattmeter (ด้านหลัง)	59
รูปที่ 2.26 สัญลักษณ์ของรีเลย์ และ โวลติสแตติคัลรีเลย์	61
รูปที่ 2.27 วงจรขับกระแสรีเลย์ด้วยทรานซิสเตอร์	62



## สารบัญรูปภาพ (ต่อ)

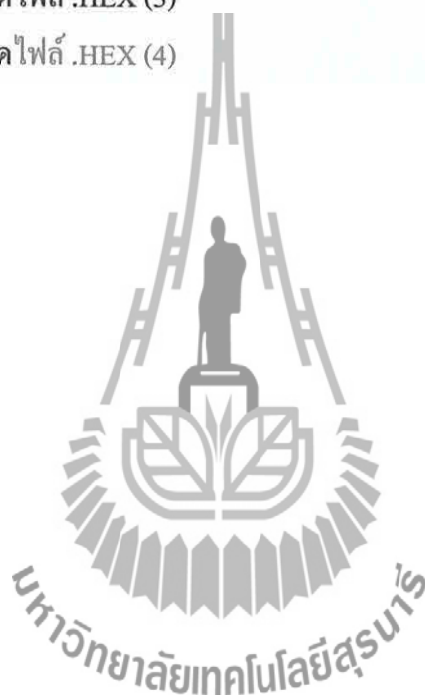
รูปภาพ	หน้า
รูปที่ 2.28 จอแสดงผล LCD 16x2 Line	63
รูปที่ 2.29 จุดเชื่อมต่อบน GSM Module	70
รูปที่ 2.30 Power Supply Connector	71
รูปที่ 2.31 หน้าต่าง Connection Description	74
รูปที่ 2.32 หน้าต่าง Connect To (ตั้งค่า Com Port)	74
รูปที่ 2.33 หน้าต่าง Port Settings (ตั้งค่า Baud rate)	75
รูปที่ 2.34 ลักษณะของหน้าจอ Hyper Terminal เมื่อโมดูลพร้อมทำงาน	76
รูปที่ 3.1 แสดงแผนผังของโครงการ	81
รูปที่ 3.2 การจัดแหล่งจ่ายไฟภายนอกของ DS18B20	81
รูปที่ 3.3 ผลการอ่านค่าอุณหภูมิที่ได้บน LCD	82
รูปที่ 3.4 การพล็อตกราฟของค่า Forward	84
รูปที่ 3.5 การพล็อตกราฟของค่า Reflect	84
รูปที่ 3.6 การเปรียบเทียบระหว่างค่าที่ได้จากการทดลองกับค่าที่ได้จากสมการของ Forward	86
รูปที่ 3.7 การเปรียบเทียบระหว่างค่าที่ได้จากการทดลองกับค่าที่ได้จากสมการของ Reflect	87
รูปที่ 3.8 แสดงค่าที่ SWR & Power Meter วัดได้	88
รูปที่ 3.9 แสดงการนำค่าจาก SWR & Power Meter มาคำนวณเป็นค่า Ratio	88
รูปที่ 3.10 แสดงค่า Forward และ Reflect ในหน่วยวัตต์	89
รูปที่ 3.11 วงจรควบคุมการเปิด-ปิดพัดลมระบายความร้อน	89
รูปที่ 3.12 วงจรควบคุมการเปิด-ปิดเครื่องส่ง	90
รูปที่ 3.13 ส่วนการแสดงผล	91
รูปที่ 3.14 Flow Chart ของฟังก์ชัน Main	93
รูปที่ 3.15 Flow Chart ของฟังก์ชัน Check Time	94
รูปที่ 3.16 Flow Chart ของฟังก์ชัน Check Temperature	95
รูปที่ 3.17 Flow Chart ของฟังก์ชัน Check Forward & Reflect	96
รูปที่ 4.1 ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง	98
รูปที่ 4.2 ส่วนประกอบของระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง	99

## สารบัญรูปภาพ (ต่อ)

รูปภาพ		หน้า
รูปที่ 4.3	หน้าจอ LCD สำหรับการเข้าตั้งค่าโปรแกรม	100
รูปที่ 4.4	การ Setting ในโหมด Ratio	101
รูปที่ 4.5	การใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงร่วมกับเครื่องส่งวิทยุ	102
รูปที่ 4.6	เข็มแสดงค่า Forward และ Reflect ของเครื่อง SWR & Power Meter เมื่อมีการส่งวิทยุ	102
รูปที่ 4.7	แสดงค่า Ratio จากการอ่านค่า Reflect ของเครื่องส่งวิทยุ	103
รูปที่ 4.8	การทำงานของพัดลมระบายอากาศ เมื่ออุณหภูมิเกินที่กำหนด	104
รูปที่ 4.9	จอแสดงผล เมื่อระบบส่งข้อมูลถึงผู้ใช้ตามหมายเลขที่กำหนดไว้	105
รูปที่ 4.10	จอแสดงผล เมื่อระบบส่งข้อมูลถึงผู้ใช้ได้สำเร็จ	105
รูปที่ 4.11	ข้อความที่ระบบส่งถึงผู้ใช้ เมื่ออุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียงเกินค่าที่กำหนดไว้	106
รูปที่ 4.12	ข้อความที่ระบบส่งถึงผู้ใช้ เมื่อค่า Ratio เกินค่าที่กำหนดไว้	107
รูปที่ 4.13	ข้อความที่ระบบส่งถึงผู้ใช้ เมื่อถึงกำหนดเวลาที่ตั้งค่าไว้	108
รูปที่ 4.14	การรับ – ส่งข้อความเพื่อตรวจสอบค่าพารามิเตอร์ของผู้ใช้	109
รูปที่ 5.1	หน้าต่างของโปรแกรม Keil uVision3	116
รูปที่ 5.2	การกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3	117
รูปที่ 5.3	หน้าต่างของการสร้าง Project ใหม่	118
รูปที่ 5.4	การกำหนดเบอร์ MCU ที่จะใช้งาน	118
รูปที่ 5.5	แสดงข้อความการกำหนด Startup File	119
รูปที่ 5.6	การกำหนดค่า XTAL ของ MCU ที่ใช้งาน	120
รูปที่ 5.7	การกำหนดค่าตัวเลือก Create HEX File	120
รูปที่ 5.8	หน้าต่างของโปรแกรม Keil uVision3 หลังจากตั้งค่าต่าง ๆ แล้ว	121
รูปที่ 5.9	หน้าต่างของการ save ไฟล์	122
รูปที่ 5.10	การสั่ง Add File ต่างๆเข้ากับ Project File	123
รูปที่ 5.11	การแปลงเป็น Hex File	124
รูปที่ 5.12	หน้าต่างโปรแกรม Ethernet Flash Utility	125

สารบัญรูปภาพ (ต่อ)

รูปภาพ	หน้า
รูปที่ 5.13 หน้าต่างรูปแบบการเชื่อมต่อ	125
รูปที่ 5.14 หน้าต่างการเชื่อมต่อสำเร็จ	126
รูปที่ 5.15 หน้าต่างการโหลดไฟล์ .HEX (1)	126
รูปที่ 5.16 หน้าต่างการโหลดไฟล์ .HEX (2)	127
รูปที่ 5.17 หน้าต่างการโหลดไฟล์ .HEX (3)	127
รูปที่ 5.18 หน้าต่างการโหลดไฟล์ .HEX (4)	128



## สารบัญตาราง

ตาราง	หน้า	
ตารางที่ 2.1.1	PLL Register	16
ตารางที่ 2.1.2	PLLCFG Register	17
ตารางที่ 2.1.3	PLLCON Register	17
ตารางที่ 2.1.4	ค่าแต่ละบิตของรีจิสเตอร์ PLLSTAT	18
ตารางที่ 2.1.5	โหมดการทำงานของวงจร PLL	18
ตารางที่ 2.1.6	VPBDIV Register	19
ตารางที่ 2.1.7	ค่าแต่ละบิตของรีจิสเตอร์ VPBDIV	20
ตารางที่ 2.1.8	MAM Register	20
ตารางที่ 2.1.9	การกำหนดค่าให้กับ MAMCR Register	21
ตารางที่ 2.1.10	การกำหนดค่าให้กับ MAMTIM Register	21
ตารางที่ 2.1.11	รีจิสเตอร์ PINSEL0, PINSEL1, PINSEL2	22
ตารางที่ 2.1.12	ค่าประจำบิตของรีจิสเตอร์ PINSEL0	23
ตารางที่ 2.1.13	ค่าประจำบิตของรีจิสเตอร์ PINSEL1	24
ตารางที่ 2.1.14	ค่าประจำบิตของรีจิสเตอร์ PINSEL2	25
ตารางที่ 2.1.15	ชื่อและแอดเดรสของรีจิสเตอร์ที่ใช้ในการควบคุม GPIO	29
ตารางที่ 2.1.16	แหล่งกำเนิดอินเตอร์รัปต์ทั้ง 32 ตัว	29
ตารางที่ 2.1.17	รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์	32
ตารางที่ 2.1.18	รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์จากภายนอก	35
ตารางที่ 2.1.19	ค่าประจำบิตของรีจิสเตอร์ EXTINT	35
ตารางที่ 2.1.20	ค่าประจำบิตของรีจิสเตอร์ EXTMODE	36
ตารางที่ 2.1.21	ค่าประจำบิตของรีจิสเตอร์ EXTPOLAR	37
ตารางที่ 2.1.22	ค่าประจำบิตของรีจิสเตอร์ EXTPILAR	38
ตารางที่ 2.1.23	รีจิสเตอร์ที่เกี่ยวข้องกับ UART0	41
ตารางที่ 2.1.24	ค่าประจำบิตของรีจิสเตอร์ UART0 Line Control Register (U0LCR)	42
ตารางที่ 2.1.25	แสดงค่าประจำบิตของ UART0 Line Status Register (U0LSR)	43

## สารบัญตาราง (ต่อ)

ตาราง		หน้า
ตารางที่ 2.4.1	คุณสมบัติของ Wattmeter CN-101-L	58
ตารางที่ 2.5.1	ข้อดี และข้อเสียของรีเลย์และ โซลิดสเตตรีเลย์	61
ตารางที่ 2.6.1	ตำแหน่งของขาและหน้าที่การใช้งานของ LCD โมดูล	63
ตารางที่ 2.6.2	คำสั่งควบคุมการแสดงผล LCD	64
ตารางที่ 2.7.1	Power supply connector pin description	71
ตารางที่ 2.7.2	รูปแบบการใช้งาน AT Command (เมื่อ <x> คือ รหัสคำสั่ง)	72
ตารางที่ 3.1	แสดงผลการทดสอบวัดค่า Forward (F)	83
ตารางที่ 3.2	แสดงผลการทดสอบวัดค่า Reflect (R)	83
ตารางที่ 3.3	แสดงการเชื่อมต่อระหว่าง Port กับ Display board	91



# บทที่ 1

## บทนำ

### 1.1 ความเป็นมา

เนื่องจากสถานีวิทยุกระจายเสียง จะต้องมีการกระจายเสียงอย่างต่อเนื่องตลอดเวลา จึงอาจเกิดความผิดปกติอันเกิดจากเครื่องส่งวิทยุกระจายเสียง ซึ่งมีค่าพารามิเตอร์ต่างๆ ที่เกินขีดจำกัด ได้ ทำให้จำเป็นต้องมีบุคลากรคอยดูแลเครื่องส่งวิทยุกระจายเสียงอยู่ด้วย และในการตรวจสอบค่าพารามิเตอร์ต่างๆ นั้น จำเป็นที่จะต้องใช้เครื่องมือตรวจสอบอีกหลายชิ้น ที่เป็นไปตามชนิดของพารามิเตอร์ นั้นๆ เราจึงนำปัญหาดังกล่าวมาคิดพัฒนาเป็นระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงขึ้น เพื่อให้บุคลากรผู้ดูแลเครื่องส่งวิทยุกระจายเสียงนั้นทราบถึงค่าพารามิเตอร์ต่างๆ ได้โดย ไม่จำเป็นต้องเดินทางไปสถานีวิทยุด้วยตนเอง

### 1.2 หลักการและเหตุผล

ในปัจจุบัน ได้มีสถานีวิทยุกระจายเสียงเกิดขึ้นเป็นจำนวนมากในชุมชนท้องถิ่น ซึ่งสถานีวิทยุจะต้องมีการกระจายเสียงตลอด 24 ชั่วโมง ทำให้ค่าพารามิเตอร์ต่างๆ ภายในเครื่อง ส่งวิทยุกระจายเสียง เช่น ค่ากำลังส่งของเครื่องส่งวิทยุกระจายเสียง , ค่า Reflect และอุณหภูมิภายในเครื่อง เกิดความผิดปกติได้ ซึ่งพารามิเตอร์ดังกล่าวมีความสำคัญต่อการทำงานของเครื่องส่งวิทยุกระจายเสียงเป็นอย่างมาก จึงจำเป็นต้องมีบุคลากรผู้ดูแลเครื่องส่งวิทยุกระจายเสียงอยู่ตลอดเวลา ในบางครั้งสถานีกระจายเสียงอยู่ในพื้นที่ที่เข้าถึงได้ยาก ทำให้ผู้ดูแลเครื่องมือมีอุปสรรคในการไปตรวจสอบค่าพารามิเตอร์มากขึ้น นอกเหนือจากการที่ต้องใช้เครื่องมือวัดอีกหลายชิ้น ที่เป็นไปตามชนิดของค่าพารามิเตอร์ ด้วยเหตุนี้ เราจึงคิดค้นโครงการระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงนี้ขึ้น เพื่อเป็นการประหยัดเวลาของบุคลากรที่ต้องไปตรวจดูแลเครื่องส่งวิทยุกระจายเสียง และลดจำนวนของเครื่องมือวัดค่าพารามิเตอร์ได้

โดยอุปกรณ์การรายงานสถานะของเครื่องส่งวิทยุกระจายเสียงนี้ สามารถตรวจสอบค่ากำลังส่งของเครื่องส่งวิทยุกระจายเสียง , ค่า Reflect รวมถึงอุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียงได้โดยอัตโนมัติ ผ่านทางบอร์ดไมโครคอนโทรลเลอร์และเซนเซอร์ที่ติดตั้งอยู่ในเครื่องส่ง นอกจากนี้ยัง

สามารถส่งข้อมูล จากการตรวจสอบ ผ่านทางระบบ SMS ถึงบุคลากรผู้ดูแลได้ ทั้งในแบบรายวันและ  
ในกรณีที่ผู้ดูแลต้องการทราบค่าพารามิเตอร์ดังกล่าว รวมถึงกรณีที่เครื่องส่งวิทยุกระจายเสียงเกิด  
ความผิดปกติ ขึ้น นั่นคือ ค่าพารามิเตอร์ต่างๆ เกินจากค่าที่กำหนดไว้ ซึ่งในการรับ-ส่งข้อมูลผ่าน  
ระบบ SMS นั้น มีความสะดวกต่อผู้ดูแลเป็นอย่างมาก เนื่องจากสามารถส่งข้อมูลได้ไม่จำกัด  
ระยะทาง และไม่จำกัดเครือข่าย จึงทำให้เครื่องส่งวิทยุกระจายเสียงสามารถทำงานได้อย่างมี  
ประสิทธิภาพมากขึ้น

### 1.3 วัตถุประสงค์

- 1.3.1 เพื่อศึกษาโปรแกรมควบคุมและการทำงานของบอร์ดไมโครคอนโทรลเลอร์  
รุ่น CP-JR ARM7 USB-LPC2148
- 1.3.2 เพื่อทำการวัดค่าพารามิเตอร์ต่างๆ ดังนี้ คือ ค่ากำลังส่งของเครื่องส่ง  
วิทยุกระจายเสียง , ค่า Reflect และอุณหภูมิภายในเครื่องส่งได้
- 1.3.3 เพื่อศึกษาการทำงานของระบบการส่งข้อมูลผ่านทางระบบ SMS
- 1.3.4 เพื่อให้เครื่องส่งวิทยุกระจายเสียงในสถานีวิทยุชุมชน ทำงานได้อย่างมี  
ประสิทธิภาพ โดยไม่จำเป็นต้องมีผู้ดูแลตลอดเวลา
- 1.3.5 เพื่อนำความรู้จากการศึกษาภาคทฤษฎีของวิชาต่างๆ ที่ได้ศึกษามาปฏิบัติมา  
ประยุกต์ใช้ เพื่อสร้างชิ้นงาน และสามารถนำไปใช้งานได้จริง

### 1.4 ขอบเขตการทำงาน

- 1.4.1 สามารถวัดกำลังส่งของเครื่องส่งวิทยุกระจายเสียง , ค่า Reflect รวมถึงอุณหภูมิ  
ภายในเครื่องส่งวิทยุกระจายเสียง และให้แสดงผลผ่านทางจอ LCD ได้
- 1.4.2 สามารถส่งค่าที่ตรวจสอบ คือ ค่ากำลังส่ง , ค่า Reflect และค่า อุณหภูมิของ  
เครื่องส่งวิทยุกระจายเสียงผ่านระบบ SMS เป็นรายวันได้
- 1.4.3 ในกรณีที่ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงต้องการทราบค่าที่ตรวจสอบ  
สามารถส่ง ข้อความผ่านทางระบบ SMS เพื่อให้ระบบรายงานสถานะเครื่องส่ง  
วิทยุกระจายเสียงกลับมา

- 1.4.4 ในกรณีที่ค่าพารามิเตอร์ภายในเครื่องส่งวิทยุกระจายเสียงเกินจากค่าที่กำหนดไว้ ระบบจะส่งข้อมูลผ่านทางระบบ SMS ไปยังผู้ดูแลได้ รวมถึงจะทำการปิดเครื่องส่ง วิทยุกระจายเสียงได้โดยอัตโนมัติ

## 1.5 การเลือกใช้ซอฟต์แวร์

ซอฟต์แวร์ที่ใช้ในโครงการนี้ ผู้จัดทำได้เลือกใช้ภาษาซีในการควบคุมสั่งงาน ไมโครคอนโทรลเลอร์ เนื่องจากภาษาซีเป็นภาษาโครงสร้างง่ายต่อการทำความเข้าใจ ปรับปรุงพัฒนาต่อ นอกจากนั้นภาษาซียังเป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์ (ไมโครคอนโทรลเลอร์) มีความยืดหยุ่นในการโยกย้ายไปใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย

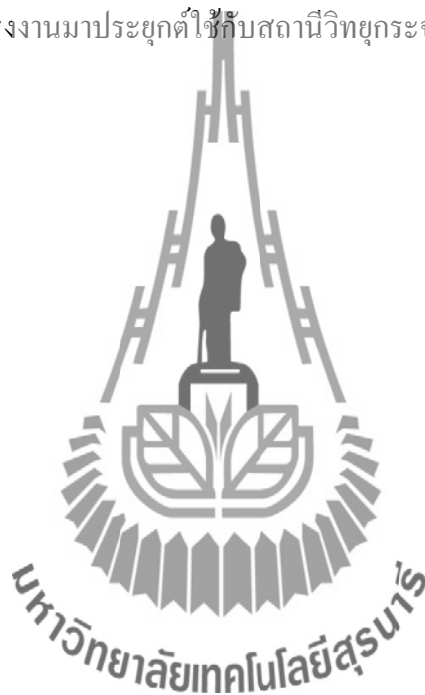
## 1.6 ขั้นตอนการดำเนินการ

- 1.6.1 ปรึกษาอาจารย์ที่ปรึกษาโครงการเกี่ยวกับขอบเขตของโครงการที่จะทำ
- 1.6.2 ศึกษาข้อมูลเกี่ยวกับอุปกรณ์แต่ละตัวที่ต้องใช้ในโครงการ ได้แก่  
Microcontroller, GSM Module, IC Temperature (DS18B20) และ EEPROM 24LC256
- 1.6.3 สั่งซื้ออุปกรณ์ที่เกี่ยวข้องสำหรับโครงการ
- 1.6.4 ฝึกใช้งานโปรแกรมไมโครคอนโทรลเลอร์
- 1.6.5 ประกอบวงจรอิเล็กทรอนิกส์
- 1.6.6 เขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ให้ทำงานได้ตามวัตถุประสงค์
- 1.6.7 ทดลองใช้งานและแก้ไขข้อผิดพลาด
- 1.6.8 จัดทำรูปเล่มรายงานของโครงการเพื่อเสนออาจารย์ประจำสาขาวิชา



## 1.7 ประโยชน์ที่คาดว่าจะได้รับ

- 1.7.1 ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์
- 1.7.2 ได้เรียนรู้การทำงานของเครื่องส่งวิทยุกระจายเสียง และการส่งข้อมูลผ่านทางระบบ SMS
- 1.7.3 ได้เรียนรู้การสร้างอุปกรณ์อิเล็กทรอนิกส์ให้สามารถทำงานตามที่ต้องการได้
- 1.7.4 สามารถนำความรู้ที่ได้จากทางทฤษฎีมาประยุกต์ใช้ในทางปฏิบัติ
- 1.7.5 ได้เรียนรู้วิธีการหาความรู้ด้วยตนเอง เพื่อนำมาปฏิบัติและใช้งานจริง
- 1.7.6 สามารถนำโครงการมาประยุกต์ใช้กับสถานีวิทยุกระจายเสียงได้



## บทที่ 2

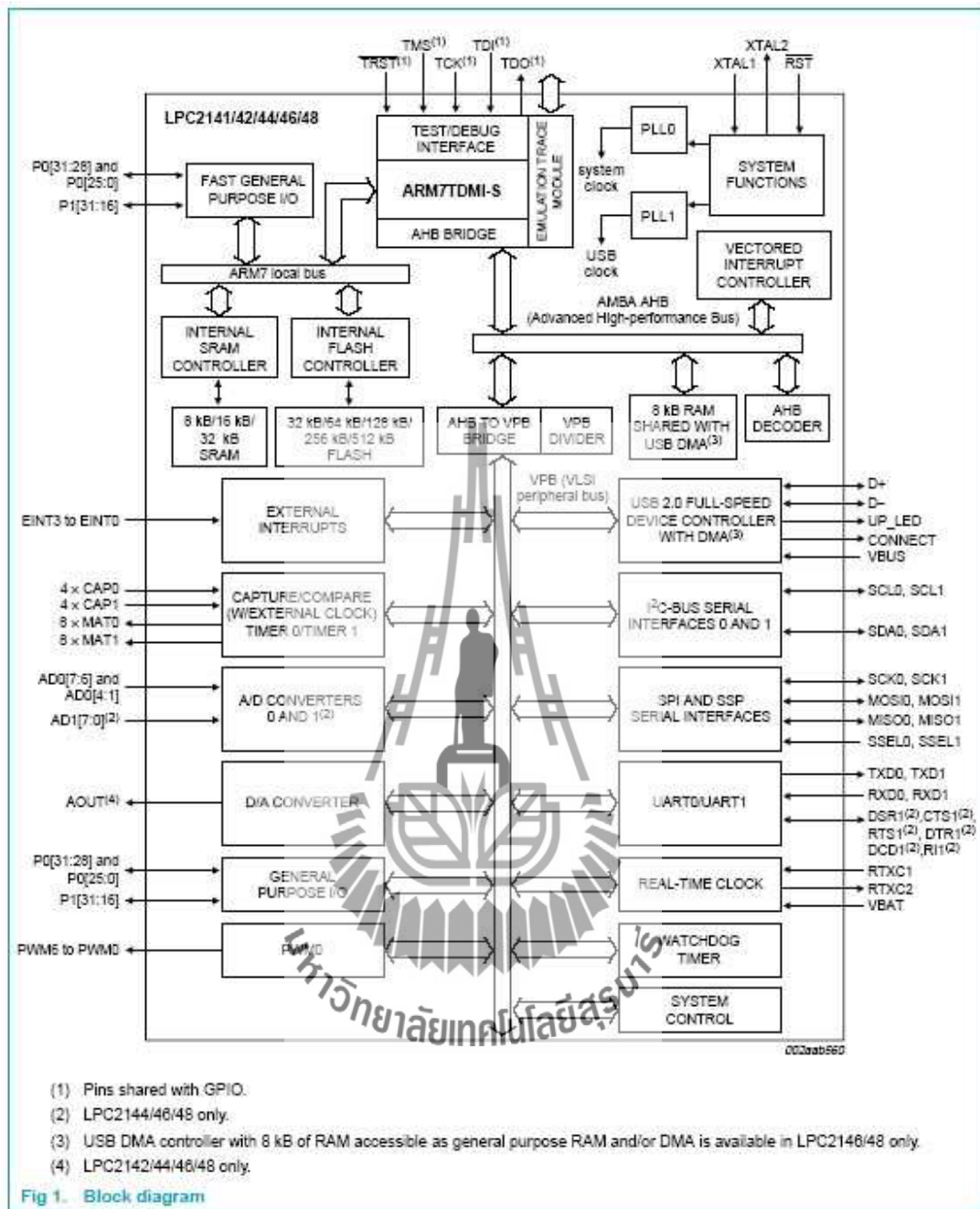
### ทฤษฎีที่เกี่ยวข้อง

#### 2.1 ไมโครคอนโทรลเลอร์ ARM7

##### 2.1.1 ไมโครคอนโทรลเลอร์ ARM7 Philips LPC2148

ความสามารถของไมโครคอนโทรลเลอร์ Philips LPC2148

- ไมโครคอนโทรลเลอร์ขนาด 16/32 บิต ARM7TDMI-S มี LQFP 64 ขา
- หน่วยความจำ Static RAM มีขนาด 40 kB
- หน่วยความจำ Flash Program Memory มีขนาด 512 kB อยู่ภายในชิปที่สามารถลบเขียนซ้ำได้ ถึง 10000 ครั้ง โปรแกรมชิปสามารถได้ทันทีผ่าน In-System Programming (ISP) และ In-Application Programming (IAP) โดยใช้ซอฟต์แวร์ boot loader ที่อยู่ในชิปตัวควบคุม USB 2.0 Full speed โดยมี ARM สำหรับ endpoint ขนาด 8 kB สำหรับการติดต่อแบบ DMA วงจรแปลง อนุลอกเป็นดิจิทัล ความละเอียด 10 บิต จำนวน 2 ชุด ที่รับอินพุตได้ถึง 14 อินพุต โดยมีเวลาในการแปลงค่าต่ำถึง 2.44 us วงจรแปลงดิจิทัลเป็นอนุลอกความละเอียด 10 บิต 1 ตัว วงจร ไทมเมอร์ขนาด 32 บิต 2 ชุด (มี 4 capture และ 4 compare channel) PWM (Pulse Width Modulation) 6 เอาท์พุท โมดูลนาฬิกาเวลาจริง (Real Time Clock) ที่สามารถ ติดต่อกับคริสตอลความถี่ 32 KHz และแบตเตอรี่ภายนอกได้ และวอชด็อกซ์ (Watchdog)
- วงจรสื่อสารอนุกรม UART (16C550) จำนวน 2 ชุดสื่อสารอนุกรม I<sup>2</sup>C ความเร็วสูง (400 Kbits/s) วงจรสื่อสารอนุกรม SPI และ SSP มีวงจร Phase lock loop ภายในเพื่อคูณค่าให้สัญญาณนาฬิกา ภายในทำงานในความเร็วสูงสุด 60 MHz Vectored Interrupt Controller ที่สามารถกำหนดลำดับความสำคัญ และกำหนดแอดเดรสของเวกเตอร์ได้ใช้กับแหล่งจ่ายไฟชุดเดียวขนาด 3.0 V และ 3.6 V ( $3.3 \pm 10\%$ )
- มี I/O pin อเนกประสงค์ที่สามารถใช้กับระดับแรงดัน 5 V ได้สูงสุด 45 ขา โดยสามารถจัดเป็นขาอินเทอร์รับต์จากภายนอกได้สูงสุด 21 ขามีโหมดประหยัดพลังงาน 2 โหมดได้แก่ Idle และ Power-Down



รูปที่ 2.1 Block Diagram LPC2148

จากรูปที่ 2.1 เป็นไมโครโปรเซสเซอร์ ARM7TDMI-S ซึ่งเป็นปัจจัยหลัก ด้านซ้ายมือเป็นส่วน ของ ARM7 Local Bus ที่ใช้ในการติดต่อกับหน่วยความจำแบบ Flash ที่ใช้เก็บโปรแกรมและ หน่วยความจำ SRAM ที่ใช้เก็บข้อมูล ส่วนที่ใชในการติดต่อกับหน่วยความจำภายนอกมีการติดต่อ

ผ่านบัส AMBA AHB (Advance High-performance Bus) ซึ่งใน LPC2148 ไม่สามารถต่อกับ หน่วยความจำภายนอกได้

ในการติดต่อกับอุปกรณ์ที่เกี่ยวข้อง เช่น GPIO, IC, SPI, UART จะติดต่อผ่านบัส VPB (VLSI peripheral BUS) ซึ่ง VPS บัสต่อกับ AHB to VPB Bridge โดยสามารถปรับลด ค่าความถี่ ของ VPB บัสให้ทำงานช้ากว่าความถี่ของซีพียูได้เพื่อให้ทำงานร่วมกับ อุปกรณ์เสริมต่างๆที่มี ความเร็วต่ำกว่าได้

### การจำกัดหน่วยความจำของ LPC2148

เนื่องจาก ARM7 เป็นซีพียูขนาด 32บิต ที่มีขาแอดเดรสต่อกับหน่วยความจำ จำนวน 32 ขาให้ สามารถอ้างถึงหน่วยความจำได้ 4 GB อุปกรณ์หลักของ ARM7TDMI จะมีสถาปัตยกรรมแบบ Von Neumann ที่ใช้บัสขนาด 32 บิต ชุดเดียวกันสำหรับ คำสั่งของโปรแกรมและข้อมูลโดยมีแค่คำสั่ง Load, Store, Swap เท่านั้น ที่ใช้ในการเรียกข้อมูลที่เก็บในหน่วยความจำ การติดต่อกับคอมพิวเตอร์ อินพุต หรือ เอาต์พุตก็จะใช้คำสั่งเดียวกันกับการใช้คำสั่งจัดการเกี่ยวกับหน่วยความจำ ไมโครคอนโทรลเลอร์ LPC2148 ได้จัดสรรหน่วยความจำในรูปที่ 2.6 เมื่อซีพียูถูกรีเซตจะเริ่มต้น ทำงานที่หน่วยความจำ 0x0000 0000 จากหน่วยความจำทั้งหมด 4 GB ได้แบ่งออกเป็น 4 ส่วน ดังนี้

- 1 GB แรก (แอดเดรส 0x0000 0000 - 0x3FFF FFFF) จัดเป็นส่วนของหน่วยความจำ สำหรับเก็บโปรแกรม หรือหน่วยความจำ Flash Memory ขนาด 512 kB ซึ่งมีแอดเดรส 0x0000 0000 0x0007 FFFF
- หน่วยความจำในช่วง 1-2 GB (แอดเดรส 0x4000 0000 0x7FFF FFFF) จัดเป็นส่วนของ หน่วยความจำ ARM จะเป็น SRAM ขนาด 40 kB โดยแบ่งออกเป็น 2 ส่วน
  - ส่วนที่ 1 คือ 32 kB อยู่ที่แอดเดรส 0x4000 0000 0x4000 7FFF
  - ส่วนที่ 2 คือ 8 kB เป็นหน่วยความจำที่ใช้สำหรับ USB โดยมีแอดเดรส 0x7FD0 0000- 0x7FD0 1FFF ในกรณีที่ไมใช้การติดต่อกับ USB สามารถนำ หน่วยความจำ ARM ส่วนนี้มาใช้สำหรับงานทั่วไปได้หน่วยความจำที่ ใกล้เคียง 2 GB จะเป็นส่วนของ Boot Block ขนาด 12 kB ซึ่งเป็นโปรแกรมที่ทำงานเพื่อเขียน โปรแกรมลงหน่วยความจำ Flash

- หน่วยความจำ 2-3 GB (แอดเดรส 0x8000 0000 0xDFFF FFFF) สงวนไว้สำหรับต่อกับหน่วยความจำภายนอก ซึ่งไม่ได้ใช้งาน
- หน่วยความจำ 2-3 GB จะเป็นพื้นที่สำหรับติดต่อกับอุปกรณ์อื่นๆ ที่อยู่ใน ชิพ โดยแบ่งเป็นอุปกรณ์ที่ต่อกับ VPB บัสจะติดต่อกับหน่วยความจำช่วง 0xE000 0000 0xEFFF FFFF ถ้าเป็นอุปกรณ์ที่ติดต่อผ่านทาง AHB จะมีช่วงแอดเดรส 0xF000 0000 0xFFFF FFFF ไม่สามารถติดต่อกับหน่วยความจำภายนอก จึงไม่ได้ใช้หน่วยความจำในส่วนนี้

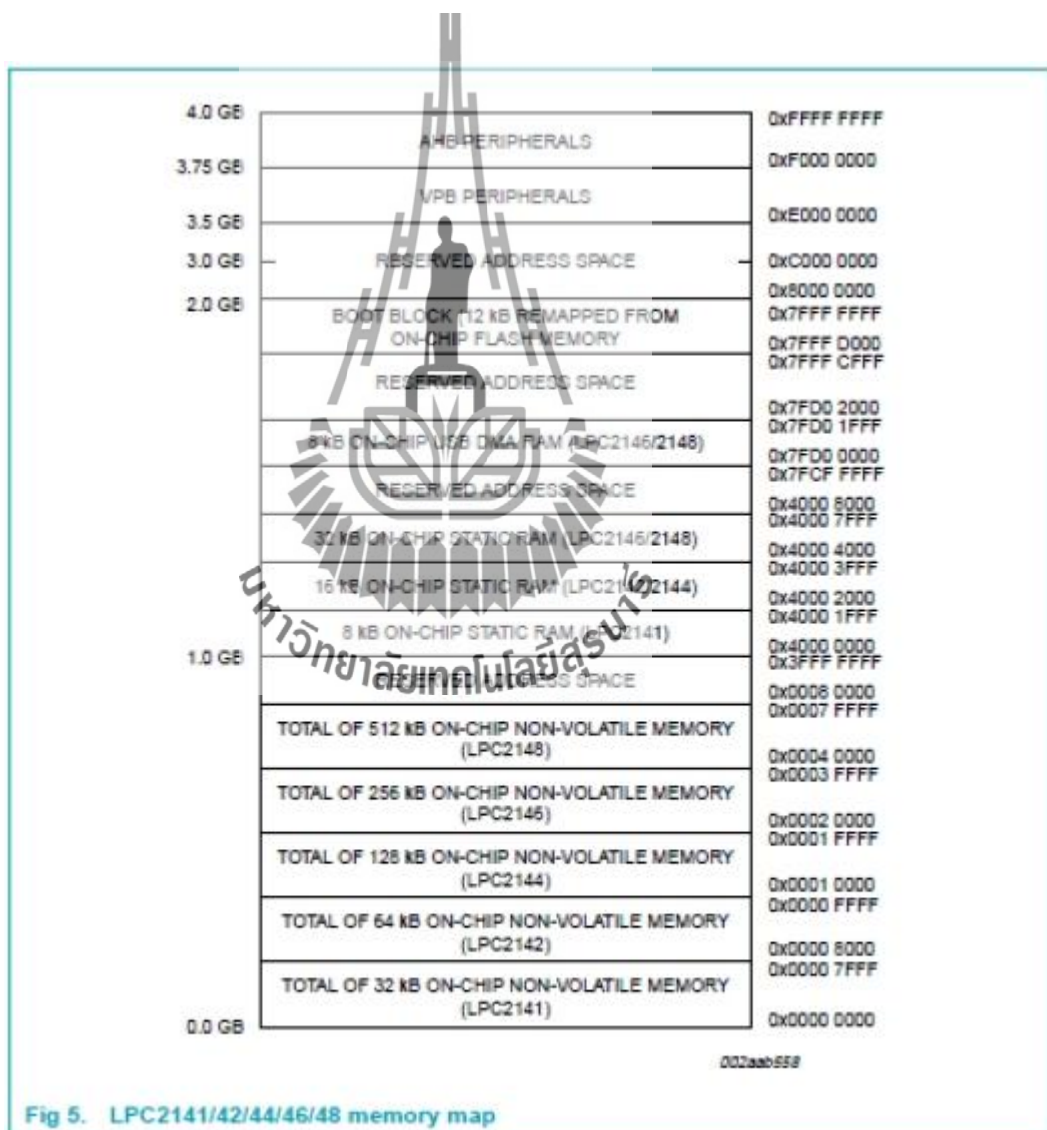


Fig 5. LPC2141/42/44/46/48 memory map

รูปที่ 2.2 Memory map LPC2148

## การจัดการขาของ LPC2148

มีจำนวนขาทั้งหมด 64 ขา โดยมีการแสดงขาผังรูป 2.7 ดังนี้

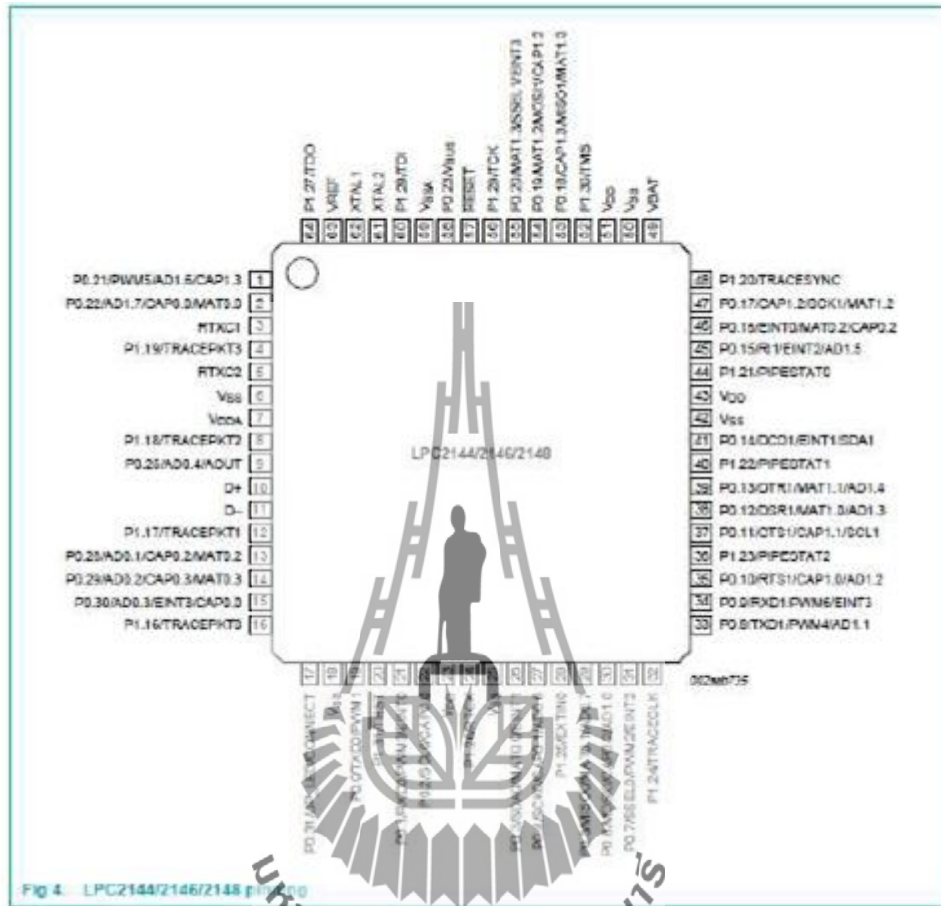


Fig 4. LPC2144/2146/2148 pin

รูปที่ 2.3 LPC2148 pinning

หลังจากการรีเซตขาพอ ร์ตทั้งหมดจะถูกกำหนดให้ทำหน้าที่เป็นอินพุตขาแต่ละขา จะมีการทำงาน เช่น ขาที่ 14 สามารถทำหน้าที่ได้ 4 หน้าที่ คือ

1. เป็น P0.29/AD0.2/CAP0.3/MAT0.3
2. ถ้าเป็น อินพุต เอาต์พุต จะเรียกว่า General Purpose Input Output: GPIO ก็คือขา P0.23
3. ถ้าใช้วงจรแปลง Analog เป็น Digital ขานี้คือ AD0.2 เป็นขาอินพุตสำหรับรับสัญญาณ Analog ADC0 อินพุตสอง

- ถ้าใช้งาน Time0 ขานี้จะเป็น CAP0.3 คือ Capture input for timer 0, Channel3 หรือ MAT0.3 ซึ่งเป็น Match output for timer0, Channel3

### 2.1.2 ฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการทดลอง

ฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการเขียนไมโครคอนโทรลเลอร์ LPC2148 เป็นไมโครคอนโทรลเลอร์ตระกูล ARM7TDMI-S ฮาร์ดแวร์ที่ใช้ประกอบด้วย แผงวงจร CP-JR ARM7 USB-LPC2148 EXP บริษัท อีทีที จำกัด ส่วนของซอฟต์แวร์คอมพิวเตอร์เพื่อแปลภาษาซีเป็นภาษาเครื่องของ LPC2148 จะใช้ชุดพัฒนา Real View Microcontroller Development Kit Version 3.02a โดยใช้คอมพิวเตอร์ CARM ของบริษัท Keil เป็นรุ่น Evaluation ที่อนุญาตให้ดาวน์โหลดมาใช้ทดสอบโปรแกรมฟรี โดยมีข้อกำหนดว่าโปรแกรมที่แปลเป็นภาษาเครื่องแล้ว จะมีขนาดไม่เกิน 16kB หลังจากคอมไพล์เป็นภาษาเครื่องจะทำการลงโปรแกรมชิปไมโครคอนโทรลเลอร์ด้วยโปรแกรม LPC2000 Flash Utility ของบริษัท Philips ซึ่งอนุญาตให้ใช้งานได้ฟรี

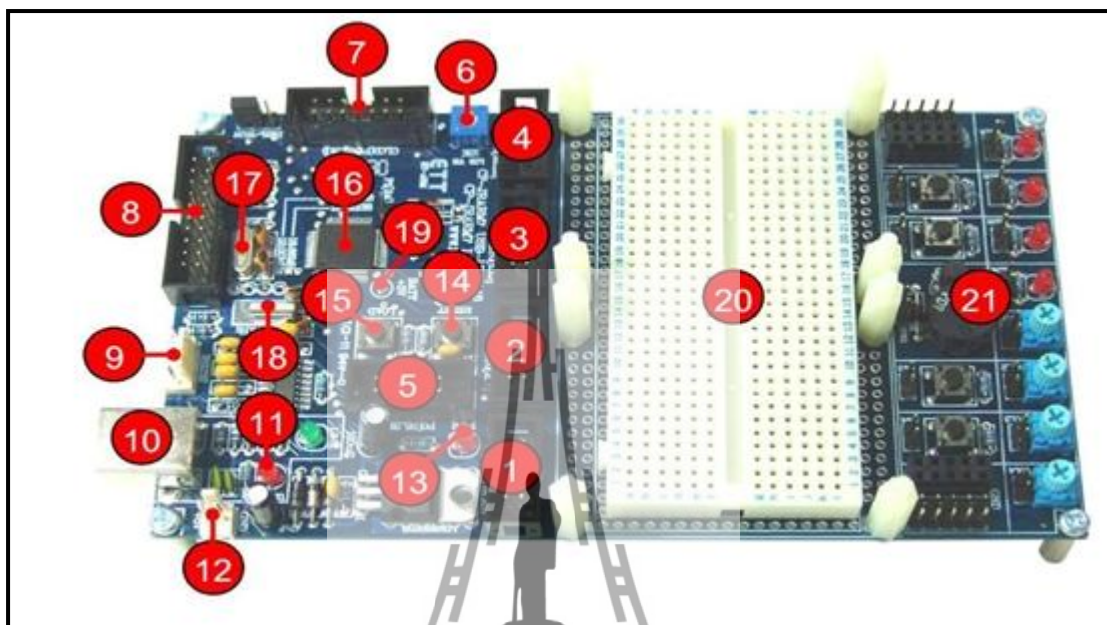
#### คุณสมบัติของบอร์ด

- ใช้ MCU ตระกูล ARM7TDMI-S เบอร์ LPC2148 ของ Philips ซึ่งเป็น MCU ขนาด 16/32-Bit
- ใช้ Crystal 12.00 MHz โดย MCU สามารถประมวลผลด้วยความเร็วสูงสุดที่ 60 MHz เมื่อใช้งานร่วมกับ Phase-Locked Loop (PLL) ภายในตัว MCU เอง
- รองรับการโปรแกรมแบบ In-System Programming (ISP) และ In-Application Programming (IAP) ผ่านทั้ง On-Chip Boot-Loader Software ทางพอร์ต UART0 (RS232)
- Power Supply ใช้แรงดันไฟฟ้า +5VDC โดยใช้ขั้วต่อแบบ CPA-2PIN จากภายนอก หรือ ใช้พลังงานจาก USB Port ได้ (ในกรณีใช้กระแสไม่เกิน 500mA)
- ภายใน MCU มีหน่วยความจำโปรแกรมแบบ Flash ขนาด 512kB, Static RAM ขนาด 40kB
- มีวงจร USB มาตรฐาน 2.0 แบบ Full Speed ภายในตัว (USB Function มี 32 End Point)
- จำนวน GPIO สูงสุดถึง 47 I/O Pins สามารถเชื่อมต่อกับระบบ I/O ที่เป็นสัญญาณ 5V ได้ ซึ่งขาสัญญาณ GPIO จะมีการใช้งานร่วมกันของ Function อื่นๆอีกด้วย  
  - วงจรรีจิสเตอร์อนุกรมแบบ SPI จำนวน 2 ช่อง และ วงจรรีจิสเตอร์อนุกรมแบบ I2C จำนวน 2 ช่อง

- วงจร ADC ขนาด 10 Bit จำนวน 14 ชุด และ วงจร DAC ขนาด 10 Bit จำนวน 1 ชุด
  - วงจร UART แบบ Full-Duplex จำนวน 2 ช่อง คือ UART-0 มาตรฐาน 4 Pin ETT เป็นสัญญาณระดับ RS232 Level และ UART-1 เป็นสัญญาณระดับ TTL Level
  - Timer 32-bit จำนวน 2 ช่อง (4 Input Capture / 4 Output Compare), 6 Channels PWM
  - Output, Watchdog Timer และ Real Time Clock
8. มีวงจรเชื่อมต่อกับ Character LCD โดยใช้วงจรการเชื่อมต่อแบบ 4 บิต จาก GPIO1 [25...31] หรือมีวงจรปรับความสว่างหน้าจอ
9. มีวงจรเชื่อมต่อกับ JTAG ARM ขนาด 20 Pin มาตรฐาน เพื่อทำการ Debug แบบ Real Time
10. มีวงจรทดลองขั้นพื้นฐานสำหรับสนับสนุนการใช้งานและทดลองเรียนรู้ขั้นพื้นฐานอย่างครบถ้วนจัดเตรียมไว้ในบอร์ด (ติดตั้งไว้เฉพาะรุ่น CP-JR ARM7 USB-LPC2148EXP) 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 ปรับได้แก่ 0-3.3V สำหรับทดสอบการทำงานของ ADC จำนวน 4 ชุด
  - ชุดกำเนิดสัญญาณเสียง Mini Speaker สำหรับทดสอบการเสียงแบบต่างๆ จำนวน 1 ชุด
  - แผงต่อวงจร Project Board รุ่น AD-100 ขนาด 360 จุด สำหรับเป็นพื้นที่ต่อทดลองวงจรขนาดเล็กๆ เพื่อใช้งานร่วมกับ CPU ได้อย่างอิสระ
  - จุดต่อแหล่งจ่ายไฟ +3.3V และ GND สำหรับเชื่อมต่อไปยังวงจรรายนอกอื่นๆ
11. ทนอุณหภูมิใช้งานระหว่าง -40 ถึง +85°C



## โครงสร้างบอร์ดไมโครคอนโทรลเลอร์ CP-JR ARM7 USB LPC2148



รูปที่ 2.4 ลักษณะของบอร์ด 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 สถานะของการเชื่อมต่อกับ 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 ถังถ่าน Battery ขนาด +3V (อยู่ด้านใต้บอร์ด) สำหรับต่อให้กับ RTC 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 ต่างๆ ของ CPU (มีติดตั้งไว้เฉพาะในรุ่น CP-JR ARM7 USB-LPC2148 EXP)

มีรายละเอียดดังนี้คือ

- LED สำหรับแสดงผลการทำงานของ Output แบบ Sink Current มีทั้งหมด 4 ชุด
- Push Button Switch สำหรับกำหนด Logic เพื่อทดสอบการทำงานของ 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

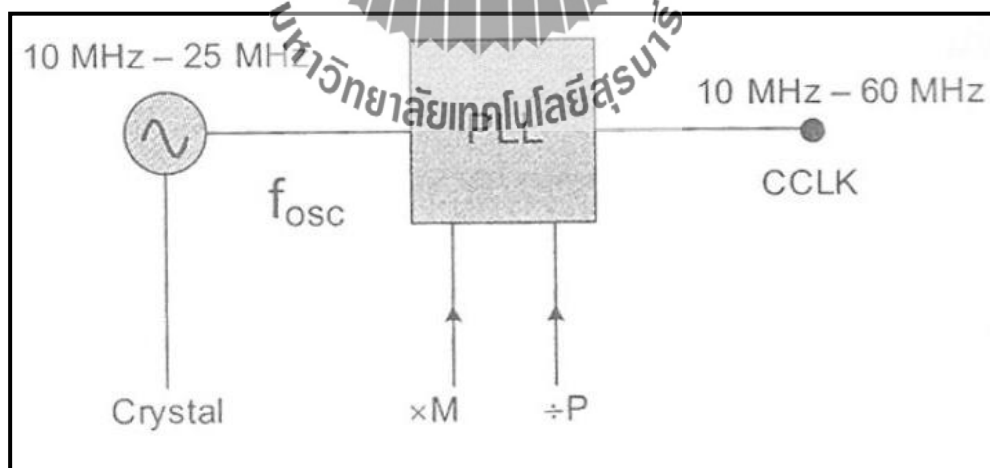
### 2.1.3 การกำหนดค่าเริ่มต้นให้กับไมโครคอนโทรลเลอร์ ARM7

การกำหนดค่าเริ่มต้นให้ก่อนโดยค่าที่ต้องการกำหนดคือ ส่วนของหน่วยความจำ RAM ที่ใช้เป็น Stack Pointer การกำหนดค่าของ Stack Pointer ต้องเขียนคำสั่งเป็นโปรแกรมภาษาแอสเซมบลีภายในของไมโครคอนโทรลเลอร์ ARM มีวงจร Phase Lock Loop: PLL สำหรับคูณค่าความถี่ของสัญญาณนาฬิกาจากภายนอก เพื่อให้ตัวไมโครคอนโทรลเลอร์สามารถทำงานที่ความถี่สูงๆ ได้โดย

การกำหนดค่าตัวคูณความถี่ สำหรับ LPC2148 ภายในจะมีวงจร PLL อยู่สองวงจรโดยตั้งชื่อเป็น PLL0 และ PLL1 โดย PLL0 ใช้คูณค่าความถี่ของสัญญาณนาฬิกาภายนอกให้กับซีพียู ARM7 และ PLL1 ที่ทำหน้าที่คูณความถี่ให้กับวงจรส่วนของ USB โดยต้องคูณค่าให้ได้ความถี่ 48 MHz หลังจากที่ไม่ไมโครคอนโทรลเลอร์ทำงานที่ความถี่สูงแล้วอุปกรณ์ประกอบ (Peripheral) ต่างๆ ที่อยู่ภายนอกไมโครคอนโทรลเลอร์ เช่น UART, I<sup>2</sup>C ฯลฯ จะทำงานไม่ทัน ดังนั้นภายในไมโครคอนโทรลเลอร์ ARM7 จะมีวงจรหารความถี่เพื่อลดความถี่ที่ป้อนให้อุปกรณ์ต่างๆ ในหน่วยความจำแบบ Flash ภายในของ LPC2148 มีการทำงานช้ากว่าตัวซีพียู ดังนั้นภายในของ LPC2148 จะมีวงจร Memory Accelerator Module (MAM) เพื่อทำหน้าที่เร่งความเร็วของการติดต่อกับหน่วยความจำแฟลต (Flash) ภายในชิป

#### วงจร Phase Lock Loop: PLL

วงจรคริสตัลออสซิลเลเตอร์ภายในของไมโครคอนโทรลเลอร์ตระกูล LPC2000 ใช้ได้กับคริสตัลค่าความถี่ 1 MHz – 30 MHz เอาต์พุตของวงจรเรียกว่า  $f_{osc}$  ส่วนความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์จะมีชื่อเรียกว่า CCLK โดยปกติถ้าไม่ได้เปิดใช้วงจร PLL ไมโครคอนโทรลเลอร์ ARM7 จะนำความถี่  $f_{osc}$  ไปใช้กับไมโครคอนโทรลเลอร์ ARM7 เช่นเดียวกัน (กรณีนี้ ค่าของ  $f_{osc}$  และ CCLK จะมีค่าเท่ากัน)



รูปที่ 2.5 ผังวงจร PLL ภายในไมโครคอนโทรลเลอร์ ARM7

วงจร PLL จะรับค่าความถี่สัญญาณนาฬิกาจากคริสตัลออสซิลเลเตอร์ที่ถูกควบคุมค่าโดยคริสตัลภายนอก แล้วนำมาคูณด้วยค่าคงที่  $M$  ให้เป็นความถี่ 10 MHz – 60 MHz โดยใช้วงจร Current

Controlled Oscillator (CCO) ทำหน้าที่คูณความถี่ ค่าตัวคูณ M จะมีค่าตั้งแต่ 1 ถึง 32 วงจร CCO ทำงานในช่วงความถี่ 156 MHz – 320 MHz ดังนั้นภายในรูปของ CCO จะต้องมียังวงจรหารค่าอีกหนึ่งตัวเพื่อให้เอาต์พุตของ PLL สร้างความถี่ให้ได้ค่าตามที่ต้องการ ค่าตัวหารคือ P กำหนดค่าได้เป็น 2, 4, 8 หรือ 16 เมื่อไมโครคอนโทรลเลอร์ถูกรีเซ็ต วงจร PLL จะถูกปิดการทำงาน ซึ่งต้องให้ซอฟต์แวร์เปิดการทำงานของโปรแกรม ตัวโปรแกรมจะต้องกำหนดค่าตัวคูณ M ตัวหาร P และกระตุ้นการทำงานของ PLL และรอให้ PLL ล็อกความถี่ได้ก่อนจึงจะสั่งต่อให้ PLL เป็นสัญญาณนาฬิกาของไมโครคอนโทรลเลอร์ ARM7 ค่าเวลาในการเซต PLL เป็น 10  $\mu$ sec เอาต์พุตของ PLL กำหนดได้ดังนี้

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

ค่าความถี่ CCO กำหนดได้ดังนี้

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

ค่า P เมื่อนำไปคูณแล้ว  $f_{cco}$  จะมีเงื่อนไขดังนี้

$$156 \text{ MHz} < f_{cco} < 320 \text{ MHz}$$

แผงวงจร CP-JR ARM7 USB-LPC2148 EXP ใช้ความถี่ 12.00 MHz ดังนั้น  $f_{osc} = 12 \text{ MHz}$  ต้องการคูณให้ได้ความถี่สูงสุดไม่เกิน 60 MHz จะได้ตัวคูณ  $M = 5$  ดังนั้น  $CCLK = 5 * 12 = 60 \text{ MHz}$  ในการกำหนดค่า PLL1 เพื่อกำหนดสัญญาณนาฬิกาสำหรับวงจร USB จะต้องกำหนดค่า M ที่นำไปคูณกับความถี่ของสัญญาณนาฬิกาให้ได้ความถี่ 48 MHz จากแผงวงจร CP-JR ARM7 USB-LPC2148 EXP จะกำหนดได้  $M = 48 \text{ MHz} / f_{osc} = 48 \text{ MHz} / 12 \text{ MHz} = 4$

การหาค่าของ P ให้ใช้ตามค่าที่ได้จากกรณีของ PPL0 คือให้ค่า  $P = 2$  จะได้  $f_{cco} = 48 \text{ MHz} * 2 * 2 = 192 \text{ MHz}$  ซึ่งตรงตามเงื่อนไขในการกำหนดค่าการทำงานของวงจร PLL จะส่งผ่านทางรีจิสเตอร์ ที่เกี่ยวข้องกับ PLL ดังแสดงในตารางที่ 2.1.1

การกำหนดค่า P, M กำหนดที่รีจิสเตอร์ PLLCFG ซึ่งแสดงรายละเอียดแต่ละบิตของรีจิสเตอร์ PLLCFG ได้ดังตารางที่ 2.1.2

เมื่อกำหนดค่าตัวหาร P และตัวคูณ M แล้วคำสั่งให้วงจร PLL ทำงานและต่อ PLL เป็นสัญญาณนาฬิกาหลักได้ที่รีจิสเตอร์ PLLCON ซึ่งมีค่าดังตารางที่ 2.1.3 ค่าแต่ละบิตของรีจิสเตอร์ PLLSTAT เพื่อดูสถานะของ PLL ซึ่งมีค่าดังตารางที่ 2.1.4

เราสามารถกำหนดโหมดการทำงานของวงจร PLL ได้จากบิต PLLC และ PLLE ของรีจิสเตอร์ PLLCON ได้ดังตารางที่ 2.1.5

ตารางที่ 2.1.1 PLL Register

ชื่อทั่วไป	ความหมาย	ติดต่อ	ค่าหลังรีเซ็ต	ชื่อและแอดเดรสของ PLL0	ชื่อและแอดเดรสของ PLL1
PLLCON	PLL Control Register เก็บค่ารีจิสเตอร์สำหรับการอัปเดตค่า PLL control bit ค่าที่เขียนให้ยังไม่มีผลจนกว่าจะเขียน PLL feed sequence ที่ถูกต้องให้	อ่าน/เขียน	0	PLL0CON 0xE01FC080	PLL1CON 0xE01FC0A0
PLLCFG	PLL Configuration Register เก็บค่าสำหรับอัปเดตค่าของ PLL Configuration ค่าที่เขียนให้ยังไม่มีผลจนกว่าจะเขียน 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 เป็นการส่งโหลดค่าที่เก็บใน PLLCON และ PLLCFG เพื่อสั่งให้	อ่าน/เขียน	ไม่มี	PLL0FEED 0xE01FC08C	PLL1FEED 0xE01FC0AC

	ทำงานตามค่าที่กำหนดให้				
--	------------------------	--	--	--	--

ตารางที่ 2.1.2 PLLCFG Register

ค่าบิตของ PLLCFG	หน้าที่	ความหมาย	ค่าหลังรีเซต
4:0	MSEL 4:0	ค่าตัวคูณ M ของวงจรร 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 ของวงจรร 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 2.1.3 PLLCON Register

ค่าบิตของ PLLCON	หน้าที่	ความหมาย	ค่าหลังรีเซต
0	PLLE	PLL Enable สั่งให้ PLL ทำงาน โดยเขียนค่าเป็น 1 และต้องเขียน ค่า PLLFEED ที่เหมาะสมด้วย	0
1	PLLC	PLL Connect เมื่อ PLLC และ PLLE มีค่าเป็น 1 และหลังจาก เขียนค่า PLLFEED ที่เหมาะสม จะเป็นการสั่งให้ต่อ PLL เป็น สัญญาณนาฬิกาหลังของวงจรร	0
7:2	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่ได้กำหนด

ตารางที่ 2.1.4 ค่าแต่ละบิตของรีจิสเตอร์ PLLSTAT

ค่าบิตของ PLLSTAT	หน้าที่	ความหมาย	ค่าหลังรีเซต
4:0	MSEL 4:0	อ่านค่าว่าปัจจุบันวงจร PLL ใช้ ค่าตัวคูณความถี่เท่ากับเท่าไร	0
6:5	PSEL 1:0	อ่านค่าว่าปัจจุบันวงจร PLL ใช้ ค่าตัวหารความถี่เท่ากับเท่าไร	0
7	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่กำหนด
8	PLLE	อ่านค่าว่าปัจจุบันบิต PLLE มีค่า เป็น 0 หรือ 1	0
9	PLLC	อ่านค่าว่าปัจจุบันบิต PLLC มีค่า เป็น 0 หรือ 1	0
10	PLOCK	ถ้าเป็น 0 แสดงว่า PLL ยังไม่ สามารถล๊อคค่าความถี่ ถ้าเป็น 1 แสดงว่าสามารถล๊อคค่าความถี่ ตามที่กำหนดได้แล้ว	0
15:11	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตนี้	ไม่กำหนด

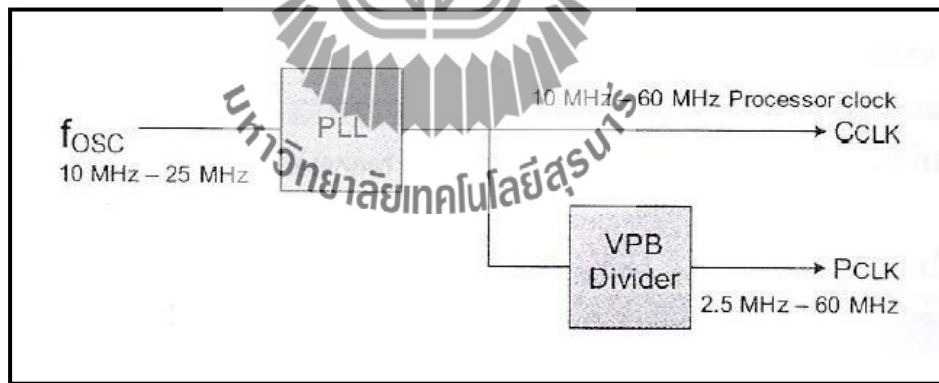
ตารางที่ 2.1.5 โหมดการทำงานของวงจร PLL

บิต PLLC	บิต PLLE	การทำงานของ PLL
----------	----------	-----------------

0	0	PLL ไม่ทำงาน ไม่ได้ต่อเข้ากับไมโครคอนโทรลเลอร์
0	1	PLL ทำงานแต่ไม่ได้ต่อเข้ากับไมโครคอนโทรลเลอร์
1	0	เหมือนกับกรณีค่า 00 เพื่อป้องกันไม่ให้ต่อ PLL เข้ากับระบบ ถ้ายังไม่สั่งให้มันทำงาน
1	1	PLL ทำงาน และต่อเป็นสัญญาณนาฬิกาหลัก

### การกำหนดค่าความถี่ให้กับ VLSI Peripheral Bus

เมื่อกำหนดให้กับวงจร PLL ภายในไมโครคอนโทรลเลอร์ ARM7 ทำงานสร้างความถี่สูงและต่อให้เป็นสัญญาณนาฬิกาของโปรเซสเซอร์แล้ว อุปกรณ์ประกอบต่างๆ เช่น UART, GPIO, I<sup>2</sup>C ที่ต่อกับบัส VPB จะทำงานไม่ทัน ดังนั้นภายในไมโครคอนโทรลเลอร์ ARM7 จะมีวงจรหารลดความถี่ของ CCLK ให้เป็นความถี่ของ VPB Bus ที่เรียกว่า PCLK โดยบล็อกไดอะแกรมภายในแสดงดังรูปที่ 2.6



รูปที่ 2.6 Block Diagram วงจรกำเนิดสัญญาณภายใน ARM7

รีจิสเตอร์ที่เกี่ยวข้องคือ VPBDIV ที่อยู่แอดเดรส 0xE10F C100 โดยสามารถอ่านและเขียนได้ ค่าของรีจิสเตอร์ VPBDIV จะมีค่า 2 บิตสุดท้าย โดยกำหนดได้ดังตารางที่ 2.1.7

#### ตารางที่ 2.1.6 VPBDIV Register



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

ตารางที่ 2.1.7 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

#### การกำหนดค่าให้กับ Memory Accelerator Module (MAM)

การกำหนดค่าการทำงานของ MAM มีรีจิสเตอร์ที่เกี่ยวข้อง 2 ตัวคือ MAMCR, MAMTIM ดังแสดงในตารางที่ 2.1.8

การกำหนดค่าโหมดการทำงานของ MAM ผ่านทาง MAMCR ได้ดังตารางที่ 2.1.9

การกำหนดค่ารีจิสเตอร์ MAMTIM กำหนดได้ดังตารางที่ 2.1.10

ในการกำหนดค่าเริ่มต้นการทำงานของ MAM เริ่มต้นด้วยการหยุดการทำงานของ 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  $MAMTIM = 3 * CCLK$

ตารางที่ 2.1.8 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 2.1.9 การกำหนดค่าให้กับ MAMCR Register

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

ตารางที่ 2.1.10 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

## 2.1.4 การต่อ GPIO เป็นเอาต์พุต

ภายในไมโครคอนโทรลเลอร์ มีพอร์ตอเนกประสงค์ ขนาด 32 บิต ให้ใช้งาน 2 พอร์ต คือ Port0, 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

หลังจากเกิดการรีเซ็ตไมโครคอนโทรลเลอร์ ARM7 วงจรภายในสั่งรีเซ็ตค่ารีจิสเตอร์ PINSEL ทุกตัวกำหนดค่าให้ Port0, Port1 ทุกขาเป็น GPIO และให้ทุกขาเป็นอินพุต ขาแต่ละขาของ Port0, Port1 มีหน้าที่การทำงานได้หลายหน้าที่ โดยกำหนดการทำงานของ Port ที่รีจิสเตอร์ PINSEL0, PINSEL1 และกำหนดหน้าที่การทำงานของ Port1 ที่รีจิสเตอร์ PINSEL2 โดยรีจิสเตอร์แต่ละตัวที่มีแอดเดรสแสดงในตารางที่ 2.1.11

ตัวอย่างการกำหนดค่าให้ขา P0.16 – P0.31 มีการทำงานเป็น GPIO ทำโดยการเขียนค่า 0 ทุกบิตให้กับ PINSEL1 ดังนี้

```
PINSEL1 = 0x0000 0000; //set P0.16 – P0.31 to GPIO Function
```

ค่าแต่ละบิตของรีจิสเตอร์ PINSEL2 ที่ใช้กำหนดหน้าที่การทำงานของ Port P1 แสดงได้ในตารางที่ 2.1.14 ตัวอย่างการกำหนดค่าให้ขา P1.16 – P1.31 มีการทำงานเป็น GPIO ทำการเขียนค่า 0 ทุกบิตให้กับ PINSEL1

```
PINSEL2 = 0x0000 0000; //set P1.16 – P1.31 to GPIO Function
```

ตารางที่ 2.1.11 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 P0.10 –P1.31	อ่าน/เขียน	0xE002 C014

ตารางที่ 2.1.12 ค่าประจำบิตของรีจิสเตอร์ PINSEL0

PINSEL0	ชื่อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า 10	การทำงาน เมื่อมีค่า 11	ค่าหลัง รีเซ็ต
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 <sup>2</sup> C)	Capture 0.0(TIMER0)	Reserved	00
7:6	P0.3	GPIO Port0.3	SDA0(I <sup>2</sup> C)	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 <sup>2</sup> C1)	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 <sup>2</sup> C1)	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 Tx/D0, Rx/D0, ของ UART0 เสมอ มิฉะนั้นอาจจะทำให้วงจรเสียได้ ดังนั้นจึงกำหนดค่าเริ่มต้นของ PINSEL0 ได้ดังนี้

PINSEL0 = 0x0000 0005; // set P0.2 –P0.15 to GPIO Function

ตารางที่ 2.1.13 ค่าประจำบิตของรีจิสเตอร์ PINSEL1

PINSEL1	ข้อขา	การทำงาน เมื่อมีค่า 00	การทำงาน เมื่อมีค่า 01	การทำงาน เมื่อมีค่า 10	การทำงาน เมื่อมีค่า 11	ค่าหลัง รีเซ็ต
1:0	P0.16	GPIO Port0.16	EINT0	Match 0.2 (TIMER0)	Capture 0.2 (TIMER0)	00
3:2	P0.17	GPIO Port0.17	Capture 1.2 (TIMER1)	SCK1(SPI1)	Match 1.2 (TIMER1)	00
5:4	P0.18	GPIO Port0.18	Capture 1.3 (TIMER1)	MISO1(SPI1)	Match 1.3 (TIMER1)	00
7:6	P0.19	GPIO Port0.19	Match 1.2 (TIMER1)	MOIS1(SPI1)	Capture 1.2 (TIMER1)	00
9:8	P0.20	GPIO Port0.20	Match 1.3 (TIMER1)	SSEL(SPI1)	EINT3	00

11:10	P0.21	GPIO Port0.21	PWM5	AD1.6	Capture 1.3 (TIMER1)	00
13:12	P0.22	GPIO Port0.22	V <sub>BUS</sub>	Capture 0.0 (TIMER0)	Match 0.0 (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
<b>PINSEL1</b>	<b>ชื่อขา</b>	<b>การทำงาน เมื่อมีค่า 00</b>	<b>การทำงาน เมื่อมีค่า 01</b>	<b>การทำงาน เมื่อมีค่า 10</b>	<b>การทำงาน เมื่อมีค่า 11</b>	<b>ค่าหลัง รีเซ็ต</b>
23:22	P0.27	สงวนไว้				00
25:24	P0.28	GPIO Port0.28	AD0.1	Capture 0.2 (TIMER0)	Match 0.2 (TIMER0)	00
27:26	P0.29	GPIO Port0.29	AD0.2	Capture 0.3 (TIMER0)	Match 0.3 (TIMER0)	00
29:28	P0.30	GPIO Port0.30	AD0.3	EINT3	Match 0.0 (TIMER0)	00
31:30	P0.31	GPIO Port0.31	UP_LED	CONNECT		00

ตารางที่ 2.1.14 ค่าประจำบิตของรีจิสเตอร์ PINSEL2

PINSEL2	ความหมาย	ค่าหลังรีเซ็ต
1:0	สงวนไว้	00
2	เป็น 0 ขา P1.31:26 จะใช้เป็น GPIO ถ้าเป็น 1 ขา P1.31:26 จะเป็น Debug port	<u>P1.26</u> RTCK

3	เป็น 0 ขา P1.25:16 จะใช้เป็น GPIO ถ้าเป็น1ขา P1.25:16จะเป็นTrace port	P1.20 TRACESYNC
---	--	--------------------

### การกำหนดค่าควบคุมการทำงานพอร์ตอเนกประสงค์ (GPIO)

หลังจากที่กำหนดค่าให้พอร์ตแต่ละตัวมีการทำงานเป็น GPIO แล้ว ควบคุมการทำงานของ GPIO จะสั่งงานผ่านทางรีจิสเตอร์ 4 ตัวได้แก่ IOPN, IOSET, IOCKR, IODIR โดยที่รีจิสเตอร์แต่ละตัวจะมีแอดเดรสแสดงในตารางที่ 2.1.13

ในไมโครคอนโทรลเลอร์ LPC2148 มีพอร์ตอเนกประสงค์ที่ให้ทำงานได้เร็วขึ้นจะเรียกว่าเป็น Fast GPIO โดยให้กำหนดรีจิสเตอร์เพิ่มเติมขึ้นอีก แต่การทำงานที่รวดเร็วจน จะต้องเขียนโปรแกรมเป็นภาษาแอสเซมบลีเท่านั้น

การสั่งงานของ GPIO เริ่มต้นด้วยการกำหนดทิศทางของพอร์ตก่อนว่าจะให้เป็นอินพุตหรือเอาต์พุต โดยกำหนดค่ารีจิสเตอร์ IODIR โดยค่าบิตใดเป็น 0 คือให้ขาของบิตนั้นเป็นอินพุต ถ้าให้ค่าเป็น 1 ของบิตนั้นจะเป็นเอาต์พุต

ตัวอย่างเช่น ต้องการให้ขา P0.22, P0.20, P0.19 และ P0.16 เป็นเอาต์พุต 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

```
IODIR0= 0X0059 0000;
```

หลังจากการกำหนดให้พอร์ตเป็นเอาท์พุทแล้ว ถ้าต้องการสั่งให้พอร์ตที่เป็นเอาท์พุทนี้มีค่าเป็น 1 ต้องสั่งที่รีจิสเตอร์ IOSET จากตัวอย่างต้องสั่งให้ขา P0.16 และ P0.19 เป็น 1 จะต้องกำหนดให้ค่ารีจิสเตอร์ 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

IOSET0 = 0X0009 0000;

การเขียนค่าเป็น 0 รีจิสเตอร์ IOSET0 จะไม่มีผลต่อขาของพอร์ตที่ตรงกับบิตนั้น ค่าของพอร์ตจะมีค่าคงเดิม ถ้าต้องการสั่งให้พอร์ตเป็นเอาท์พุท ที่มีค่าเป็น 0 ต้องสั่งที่รีจิสเตอร์ IOCLR ด้วยจากตัวอย่าง ถ้าต้องการสั่งให้ขา P0.22 และ P.0.20 เป็น 0 จะต้องกำหนดค่าให้กับ รีจิสเตอร์ IOCLR 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 = 0X0050 0000;

**ข้อควรระวัง** ในการสั่งให้ขาพอร์ตมีค่าเป็น 0 นี้ไม่ได้สั่งให้เขียนค่า 0 ให้กับรีจิสเตอร์ IOCLR ต้องเขียนค่าเป็น 1 เท่านั้น

ในการเขียนค่าให้พอร์ต จะต้องแยกข้อมูล ถ้าบิตใดเป็น 0 จะต้องเขียนให้รีจิสเตอร์ IOCLR ถ้าบิตใดเป็น 1 จะต้องเขียนสิ่งที่รีจิสเตอร์ IOSET

## 2.1.5 อินเทอร์รัปต์ไมโครคอนโทรลเลอร์ ARM7

ในไมโครคอนโทรลเลอร์ ARM7 มีโมดูล Vectored Interrupt Controller (VIC) เป็นตัวควบคุมกลไกของการอินเทอร์รัปต์ โดยสามารถรับอินเทอร์รัปต์จากอุปกรณ์ภายในและภายนอกไมโครคอนโทรลเลอร์ ARM7 ได้ทั้งหมด 32 อินพุตตามที่แสดงในตารางที่ 2.1.14 VIC จะนำอินพุตจากการอินเทอร์รัปต์ 32 แหล่งมาจัดแบ่งตามประเภทได้เป็น 3 ประเภท ทำให้สามารถปรับระดับความสำคัญของอินเทอร์รัปต์จากอุปกรณ์ประกอบต่างๆ ได้ตามความต้องการ คือ

- Fast Interrupt Request (FIR) จัดลำดับความสำคัญสูงสุดโดยตอบสนองเร็วที่สุด
- Vectored IRQs จะมีการลำดับความสำคัญอยู่กึ่งกลาง โดยสามารถนำอินเทอร์รัปต์ 16 บิต หรือ 32 แหล่งมาจัดให้เป็น Vectored IRQs ได้ 16 ตัว โดย Slot 0 จะมีความสำคัญสูงสุด Slot 15 มีความสำคัญต่ำสุด
- Non-Vectored IRQs มีความสำคัญต่ำสุด

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเทอร์รัปต์มีทั้งหมด 43 ตัว ดังแสดงในตารางที่ 2.1.15 โดยในหัวข้อนี้จะเป็นการเลือกเฉพาะรีจิสเตอร์ที่เกี่ยวข้องกับการอินเทอร์รัปต์ในแบบ Vector IRQ คือรีจิสเตอร์ VICVectAddr0-15, VICVectCntl0-15 และ VICIntEnable รีจิสเตอร์ VICVectAddr0-15 เป็นรีจิสเตอร์ที่เก็บค่าแอดเดรสของโปรแกรมที่ตอบสนองต่ออินเทอร์รัปต์โดย VICVectAddr0 จะเป็นอินเทอร์รัปต์ Slot 0 ที่มีความสำคัญสูงสุด ตามลำดับไปจนถึง VICVectAddr15 จะเป็นอินเทอร์รัปต์ Slot 15 ที่มีความสำคัญต่ำสุด

VICVectCntl0-15 ใช้ในการควบคุม คือ Slot หมายเลขที่ตรงกับ VICVectCntl นี้จะรับอินเทอร์รัปต์จากแหล่งใด จากทั้งหมด 32 แหล่ง ตามตารางที่ 2.1.16 โดยกำหนดค่าในบิตที่ 4 :0 โดย บิตที่ 5 เป็น 1 หมายถึง อนุญาตให้อินเทอร์รัปต์จากอุปกรณ์ที่กำหนดใน Slot

VICIntEnable ใช้เปิดอินเทอร์รัปต์จากแหล่งต่างๆ ทั้ง 32 แหล่ง ตามตารางที่ 2.1.17 โดย บิตใดมีค่าเป็น 1 หมายถึง อนุญาตให้อินเทอร์รัปต์ได้ ถ้าบิตใดเป็น 0 ไม่อนุญาตให้อุปกรณ์นั้นๆ อินเทอร์รัปต์

ตารางที่ 2.1.15 ชื่อและแอดเดรสของรีจิสเตอร์ที่ใช้ในการควบคุม GPIO

ชื่อทั่วไป	ความหมาย	การติดต่อ	ชื่อ และ แอดเดรส ของPort0	ชื่อ และ แอดเดรส ของPort0
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 2.1.16 แหล่งกำเนิดอินเทอร์รัปต์ทั้ง 32 ตัว

อุปกรณ์	แหล่งกำเนิดอินเทอร์รัปต์	VIC Channel#
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgcommTx	3
TIME0	Match 0-3 (MR0, MR1, MR2, MR3) Capture 0-3 (CR0, CR1, CR2, CR3)	4
TIME1	Match 0-3 (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
อุปกรณ์	แหล่งกำเนิดอินเทอร์รัปต์	VIC Channel#
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 <sup>2</sup> C0	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)	13

	Alarm (RTCALF)	
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 <sup>2</sup> C1	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

### การอินเตอร์รัปต์จากอินพุตภายนอกของไมโครคอนโทรลเลอร์ ARM7

ในไมโครคอนโทรลเลอร์ LPC2148 จะรับอินเตอร์รัปต์จากภายนอกได้สูงสุด 4 แหล่ง คือ EINT0 , EINT1, EINT2 และ EINT3 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 มาใช้ในการกระตุ้นให้ไมโครคอนโทรลเลอร์ ออกจากการทำงานในโหมด Power Down ได้

### รีจิสเตอร์ที่เกี่ยวข้องกับอินเตอร์รัปต์จากภายนอก

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์จากภายนอกมีทั้งหมด 4 ตัวคือ EXTINT, EXTWAKE, EXTMODE และ EXTPOLAR ดังแสดงในตารางที่ 2.1.18 เมื่อเกิดการอินเตอร์รัปต์จากภายนอกเกิดขึ้น จะทำให้ค่าบิตบางบิตของรีจิสเตอร์ EXTINT มีค่าเป็น 1 และส่งต่อการอินเตอร์รัปต์ไปยัง VIC เพื่อสร้างอินเตอร์รัปต์จัดการทำงานของไมโครคอนโทรลเลอร์ เมื่อโปรแกรมตอบสนองต่อ

การอินเทอร์รัปต์ ทำงานเสร็จแล้วจะต้องเขียนค่า 1 ให้กับบิตเหล่านี้เพื่อหยุดการอินเทอร์รัปต์ค่าแต่ละบิตของรีจิสเตอร์ EXTINT แสดงในตารางที่ 2.1.19

ในการกำหนดว่าสัญญาณที่อินเทอร์รัปต์นี้ทำงานที่ระดับสัญญาณหรือที่ขาของสัญญาณ กำหนดได้ที่รีจิสเตอร์ EXTMODE ซึ่งมีค่าดังตารางที่ 2.1.20

หลังจากที่กำหนดโหมดการทำงานของสัญญาณที่ใช้อินเทอร์รัปต์ว่าเป็นระดับของสัญญาณหรือขาของสัญญาณแล้วที่รีจิสเตอร์ EXTMODE แล้วต้องกำหนดว่าสัญญาณที่อินเทอร์รัปต์นี้ทำงานในระดับลอจิก 1 หรือ 0 หรือทำงานที่ขอบขาขึ้นหรือขาลงของสัญญาณ โดยกำหนดที่รีจิสเตอร์ EXTPOLAR ซึ่งแสดงรายละเอียดดังตารางที่ 2. 1.21 ถ้าต้องการให้สัญญาณที่ขา EINT0 – EINT3 สามารถกระตุ้นให้ไมโครคอนโทรลเลอร์ ARM7 ออกจากการทำงานในโหมด Power Down จะต้องตั้งที่รีจิสเตอร์ EXTWAKE ซึ่งมีค่าบิตของรีจิสเตอร์แสดงในตารางที่ 2.1.22

ตารางที่ 2.1.17 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเทอร์รัปต์

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICIRQStatus	IRQ Status Request อนุญาตให้มี interrupt request จากตัวใด และถูกจัดเป็น IRQ	อ่าน	0	0xFFFF F000
VICFIQStatus	FIQ Status Request อ่านค่าสถานะว่า อนุญาตให้มีอินเทอร์รัปต์รีควีส จากตัวใด และถูกจัดเป็น FIQ	อ่าน	0	0xFFFF F004
VICRawItr	Raw Interrupt Status Register ใช้อ่านสถานะของอินเทอร์รัปต์จากทั้ง 32 แหแหล่งหรือจากซอฟต์แวร์โดยไม่ว่าถูกเปิดใช้หรือถูกจัดประเภทหรือไม่	อ่าน/ เขียน	0	0xFFFF F008

VICintselect	Interrupt Select Register ใช้ในการ ระบุว่าอินเทอร์รัปต์ทั้ง 32 แหล่งแต่ละ ตัวเป็น FIQ หรือ IRQ	เขียน	0	0xFFFF F00C
VICIntEnable	Inerrupt E able Register ควบคุมว่าให้ อินเทอร์รัปต์จากทั้ง 32 แหล่งตัวไหน ทำงาน และให้เป็น FIQ หรือ IRQ	อ่าน/ เขียน	0	0xFFFF F010
VICIntEnClr	Interrupt Enable Clear Register ใช้ล้างค่าบิตหนึ่งบิตหรือ มากกว่าหนึ่งบิตของรีจิสเตอร์ Interrupt Enable Register	เขียน	0	0xFFFF F014
VicSoftInt	Software Interrupt Register ค่าของ รีจิสเตอร์นี้จะถูกนำไป OR กับอินเทอร์ รัปต์จากอุปกรณ์ต่างๆ ทั้ง 32 แหล่ง	อ่าน/ เขียน	0	0xFFFF F018

ชื่อ	ความหมาย	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
VICProtection	Protection enable register ใช้จำกัดการเข้าถึง VIC Register	อ่าน/เขียน	0	0xFFFF F020
VICVectAddr	Vector Address Register เมื่อเกิดการ อินเทอร์รัปต์แบบ IRQ ตัว IRQ Service routine จะอ่านค่าจาก รีจิสเตอร์เพื่อกระโดดไปยังแอดเดรส ที่ระบุไว้	อ่าน/เขียน	0	0xFFFF F030
VICDefVectAddr	Default Vector Address Register ใช้เก็บค่าแอดเดรสของ Interrupt service routine (ISR) สำหรับ IRQ non-vectorred IRQ	อ่าน/เขียน	0	0xFFFF F100
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
<b>ชื่อ</b>	<b>ความหมาย</b>	<b>การติดต่อ</b>	<b>ค่าหลังรีเซ็ต</b>	<b>แอดเดรส</b>
VICVectCntl0	Vector control 0 register รีจิสเตอร์ Vector Control Register0-15 แต่ละ ตัวควบคุม 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
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 2.1.18 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเทอร์รัปต์จากภายนอก

แอดเดรส	ชื่อ	ความหมาย	การติดต่อ
0xE01F C140	EXTINT	External Interrupt Flag Register เก็บค่าอินเทอร์รัปต์เฟล็กของ EINT0, EINT1 และ EINT2	อ่าน/เขียน
0xE01F C144	EXTWAKE	External Interrupt Wakeup Register เก็บค่าบิตที่ควบคุมว่าจะให้อินเทอร์ รัปต์จากภายนอกนี้สามารถกระตุ้น ซีพียูจาก Power mode หรือไม่	อ่าน/เขียน
0xE01F C148	EXTMODE	External Interrupt Mode Register ควบคุมให้แต่ละระดับจะตอบสนอง ต่อระดับลอคจิกหรือขอบสัญญาณ	อ่าน/เขียน
0xE01F C14C	EXTPOLAR	External Interrupt Polarity Register ควบคุมว่าแต่ละขาจะตอบสนองระดับ	อ่าน/เขียน



		ลอกจิก 0 หรือ 1 หรือตอบสนองต่อ ขอข่าจั้งขอขบขาลงของสัญญาณ	
--	--	--	--

ตารางที่ 2.1.19 ค่าประจำบิตของรีจิสเตอร์ EXTINT

ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลัง รีเซต
0	EINT0	เมื่อขา EINT0 ทำงานมีระดับลอกจิกหรือมีขอขบของ สัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อ เขียนค่า 1 ให้มันเป็นการล้างค่า ยกเว้นกรณีที่ทำงา ตามระดับลอกจิกต้องรอให้ค่าลอกจิกหยุดการทำงาก่อน ขาที่เป็น EINT0 คือ P0.1 และ P0.16	0
1	EINT1	เมื่อขา EINT1 ทำงานมีระดับลอกจิกหรือมีขอขบของ สัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อ เขียนค่า 1 ให้มันเป็นการล้างค่า ยกเว้นกรณีที่ทำงา ตามระดับลอกจิกต้องรอให้ค่าลอกจิกหยุดการทำงาก่อน ขาที่เป็น EINT1 คือ P0.3 และ P0.14	0
ค่าบิตของ EXTINT	หน้าที่	ความหมาย	ค่าหลัง รีเซต
2	EINT2	เมื่อขา EINT2 ทำงานมีระดับลอกจิกหรือมีขอขบของ สัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อ เขียนค่า 1 ให้มันเป็นการล้างค่า ยกเว้นกรณีที่ทำงา ตามระดับลอกจิกต้องรอให้ค่าลอกจิกหยุดการทำงาก่อน ขาที่เป็น EINT2 คือ P0.7 และ P0.15	0
3	EINT3	เมื่อขา EINT3 ทำงานมีระดับลอกจิกหรือมีขอขบของ สัญญาณตามที่กำหนดเกิดขึ้น บิตนี้จะมีค่าเป็น 1 เมื่อ เขียนค่า 1 ให้มันเป็นการล้างค่า ยกเว้นกรณีที่ทำงา ตามระดับลอกจิกต้องรอให้ค่าลอกจิกหยุดการทำงาก่อน ขาที่เป็น EINT3 คือ P0.9, P0.20 และ P0.30	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้อับบิตเหล่านี้	ไม่กำหนด

		ค่าที่อ่านได้ไม่มีความหมาย	
--	--	----------------------------	--

ตารางที่ 2.1.20 ค่าประจำบิตของรีจิสเตอร์ EXTMODE

ค่าบิตของ EXTWAKE	หน้าที่	ความหมาย	ค่าหลัง รีเซต
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
ค่าบิตของ EXTWAKE	หน้าที่	ความหมาย	ค่าหลัง รีเซต
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่กำหนด

ตารางที่ 2.1.21 ค่าประจำบิตของรีจิสเตอร์ EXTPOLAR

ค่าบิตของ EXTPOLAR	หน้าที่	ความหมาย	ค่าหลัง รีเซต
0	EXTPOLAR0	เป็น 0 EINT0 ทำงานที่ลอจิก 0 หรือขอบขาด ของสัญญาณเป็น 1 EINT0 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0

1	EXTPOLAR1	เป็น 0 EINT1 ทำงานที่ลอจิก 0 หรือขอบขาลง ของสัญญาณเป็น 1 EINT1 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
2	EXTPOLAR2	เป็น 0 EINT2 ทำงานที่ลอจิก 0 หรือขอบขาลง ของสัญญาณเป็น 1 EINT2 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
3	EXTPOLAR3	เป็น 0 EINT3 ทำงานที่ลอจิก 0 หรือขอบขาลง ของสัญญาณเป็น 1 EINT3 ทำงานที่ลอจิก 1 หรือขอบขาขึ้นของสัญญาณ	0
7:4	สงวนไว้	ห้ามเขียนค่า 1 ให้กับบิตเหล่านี้ ค่าที่อ่านได้ไม่มีความหมาย	ไม่ กำหนด



ตารางที่ 2.1.22 ค่าประจำบิตของรีจิสเตอร์ EXTPILAR

ค่าบิตของ EXTMODE	หน้าที่	ความหมาย	ค่าหลัง รีเซ็ต
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 ค่าที่อ่านได้ไม่มี	ไม่กำหนด

		ความหมาย	
--	--	----------	--

## 2.1.6 การกำหนดค่าเริ่มต้นสำหรับ UART0

UART0 มีขาสัญญาณ 2 ขา คือ

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

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

```
PINSEL0 = 0x0000 0005;
```

รีจิสเตอร์ที่เกี่ยวข้องกับ UART0 มีทั้งหมด 10 ตัว โดยแต่ละตัวมีขนาด 8 บิต ดังแสดงในตารางที่ 2.1.23

หลังจากที่กำหนดให้ขา P0.0 และ P0.1 ทำงานเป็น UART0 แล้วถัดมาเป็นการกำหนดรูปแบบการติดต่อเช่น ติดต่อแบบ 8 ใช้การตรวจสอบบิตผิดพลาดแบบใด เช่น even parity จำนวนของ Stop bit โดยการกำหนดค่าผ่านทางรีจิสเตอร์ UART Line Control Register: LCR ซึ่งมีรายละเอียดของรีจิสเตอร์ดังแสดงในตารางที่ 2.1.24

ในรีจิสเตอร์ LCR มีบิตที่เรียกว่า DLAB (Divisor Latch Accesses bit) ถ้าต้องการปรับค่าของวงจรถ่าย Broad Rate ต้องเขียนบิตนี้ให้เป็น 1 แล้วเขียน Baud rate generator เป็นค่าของตัวหารขนาด 16 บิต เพื่อนำไปใช้หาค่าของ PCLK เพื่อให้ได้ความถี่สูงกว่าความเร็ว Broad Rate 16 เท่า ทำให้สมการค่าตัวหารเป็นดังนี้

$$\text{Divisor} = \text{PCLK} / (16 * \text{Baud})$$

แผงวงจร CP-JR ARM7 USB-LPC2148EXP ใช้คริสตัลความถี่ 12.000 MHz ซึ่ง PLL คูณ 5 จะได้ CCLK = 60.0 MHz และกำหนดให้ VPBDIV = 2 จึงได้ PCLK = 30.0 MHz ด้วยถ้าต้องการอัตราบอर्ड์ที่ 9600 bps ตัวหารจะมีค่าเท่ากับ

$$\text{Divisor} = 30,000,000 / (60 * 9600) = 195.3125 \text{ ปัดเศษลง}$$

$$\text{Divisor} = 195 \text{ หรือ } 0xC3$$

นำค่าที่ได้ไปคำนวณหาค่าอัตรบอर्ड์จะได้

$$\text{Baud} = \text{PCLK} / (16 * \text{Divisor}) = 30,000,000 / (16 * 195) = 9615 \text{ bps}$$

ซึ่งพลาดไป 0.156 % สามารถใช้งานได้ เนื่องมาตรฐานของการสื่อสารแบบอนุกรมสามารถรับอัตราบอ์ดที่พลาดได้ถึง 5%

นำค่าที่ได้ไปเก็บลงในรีจิสเตอร์ขนาด 8 บิตสองตัว คือ Divisor Latch MAB (DLM) และ Divisor Latch LSB (DLL) ในขนาดที่เขียนค่าเก็บในรีจิสเตอร์ DLM และ DLL ค่า DLAB ต้องมีค่าเป็น 1 เมื่อเขียนเสร็จแล้วต้องรีเซตค่าบิตนี้ให้กลับเป็น 0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
U0DLM = 0x00;
```

```
U0DLL = 0xC3;
```

```
U0LCR = 0x7F;
```

ในการเรียกใช้ฟังก์ชัน `uart0_init ()` จะต้องส่งค่าอัตราบอ์ดที่ต้องการให้ฟังก์ชัน ตัวอย่างเช่น ต้องการอัตราบอ์ดที่ 9600 bps จะต้องเรียกใช้ฟังก์ชันดังนี้

```
uart0_init (9600);
```

เมื่อกำหนดค่าการทำงานให้กับ UART แล้ว จะสามารถรับส่งค่าผ่านพอร์ตอนุกรมได้ในการส่งข้อมูลต้องเขียนข้อมูลไปยังรีจิสเตอร์ Transmit Holding Register (THR) ถ้าต้องการอ่านข้อมูลที่รับพอร์ตอนุกรมต้องอ่านค่าจากรีจิสเตอร์ Receiver Buffer Register (RBR) จากตารางที่ 2.2.24 จะพบว่าค่าแอดเดรสของรีจิสเตอร์ทั้งสองมีค่าอยู่ที่ตำแหน่งเดียวกัน แต่คงว่าการเขียนค่าให้กับ THR เป็นการเขียนค่าลงในบัฟเฟอร์แบบ FIFO ของ UART0 การอ่านค่าจากรีจิสเตอร์ RBR เป็นการอ่านค่าบัฟเฟอร์แบบ FIFO

ก่อนที่จะอ่านหรือเขียนค่าลงในรีจิสเตอร์ THR หรือ RBR จะต้องอ่านค่าสถานะของ UART ก่อนว่ามีการผิดพลาดหรือไม่ โดยอ่านค่าสถานะที่รีจิสเตอร์ Line Status Register (LSR) ก่อน โดยค่าประจำบิตของรีจิสเตอร์แสดงได้ในตารางที่ 2.1.25

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

สามารถเขียนเป็นฟังก์ชัน `putchar();` สำหรับเขียนข้อมูลจำนวน 1 ไบต์ ให้กับ UART และเขียนเป็นฟังก์ชัน `getchar();` สำหรับอ่านค่าจาก UART ได้

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



ตารางที่ 2.1.23 รีจิสเตอร์ที่เกี่ยวข้องกับ UART0

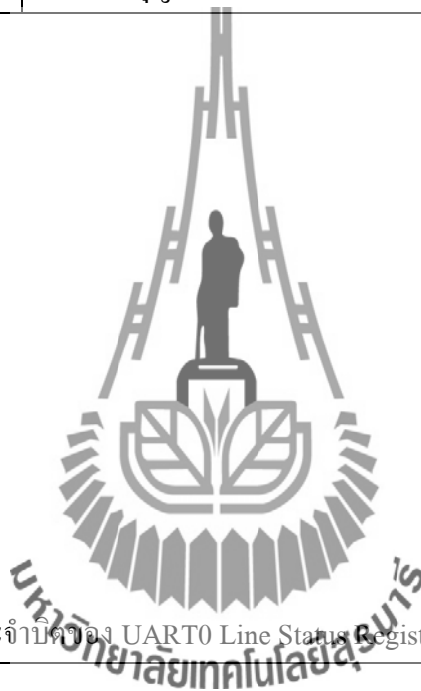
ชื่อ	ความหมาย	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0	การติดต่อ	ค่าหลังรีเซ็ต	แอดเดรส
U0RBR	Receiver Buffer Register	MSB	อ่านข้อมูล						LSB	อ่าน	ไม่กำหนด	0xE00C000 DLAB=0
U0THR	Transmit Holding Register	MSB	เขียนข้อมูล						LSB	เขียน	ไม่กำหนด	0xE00C000 DLAB=0
U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line status Interrupt Enable THRE	Interrupt Enable Rx Data Available Interrupt		อ่าน/ เขียน	0	0xE00C004 DLAB=0
U0IIR	Interrupt ID	FIFOs		0	0	IIR3	IIR2	IIR1	IIR0	อ่าน	0x01	

	Register	Enable											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	Stick 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	THRE	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 2.1.24 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 ยกเลิกการเพิ่มพาริตีบิตและการตรวจพาริตี 1 ใช้งานการเพิ่มพาริตีบิตและการตรวจพาริตี	0
5:4	Parity Select	Odd parity	0

		Even Parity ให้พาริตีบิตมีค่าเป็น 1 ตลอดเวลา ให้พาริตีบิตมีค่าเป็น 0 ตลอดเวลา	
6	Break Control	ยกเลิกการหยุดการส่ง อนุญาตให้หยุดการส่งให้ ขาที่เอาต์พุตของ UART0 TxD จะมีค่าลอจิก 0	0
7	Divisor Latch Access Bit	ไม่อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch อนุญาตให้แก้ไขค่าตัวหาร Divisor Latch	0



ตารางที่ 2.1.25 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	0



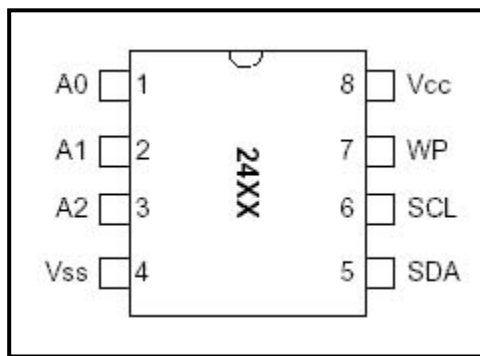
		เมื่ออ่านค่าของ UOLSR จะล้างค่าบิตนี้	
2	Parity Error (PE)	0 : ไม่มี Parity Error 1 : มี Parity Error เกิดขึ้น ใช้ตรวจสอบว่าข้อมูลที่รับมาใน UART0 RBR FIFO มีการผิดพลาดที่พาริตีบิตหรือไม่ เมื่ออ่านค่าของ UOLSR จะล้างค่าของบิตนี้	0
3	Framing Error (FE)	0 : ไม่มี Framing Error 1 : มี Framing Error เกิดขึ้น 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 UOLSR	หน้าที่		ค่าหลังรีเซต
5	Transmitter Holding Register Empty (THRE)	0 : U0THR มีข้อมูลที่ต้อง 1 : U0THR ว่าง บิต THRE จะถูกเซตค่าเป็น 1 ทันทีที่พบว่า UART0 THR ว่าง และถูกล้างค่าทันทีที่มีการเขียนค่าไปยัง U0THR	1
6	Transmitter Empty (TEMT)	0 : U0THR และ/หรือ U0TSR มีข้อมูลที่ต้อง 1 : U0THR และ U0TSR ว่าง บิตนี้จะถูกล้างค่าเมื่อ U0THR หรือ U0TSR มีข้อมูลที่ต้อง	1
7	Error in Rx	0 : ข้อมูลที่เก็บใน UART0 RBR FIFO ไม่ผิดพลาด	0

	FIFO (RXFE)	1 : ข้อมูลที่เก็บใน UART0 RBR FIFO อย่างน้อยหนึ่งตัว มีการผิดพลาด	
--	-------------	---	--



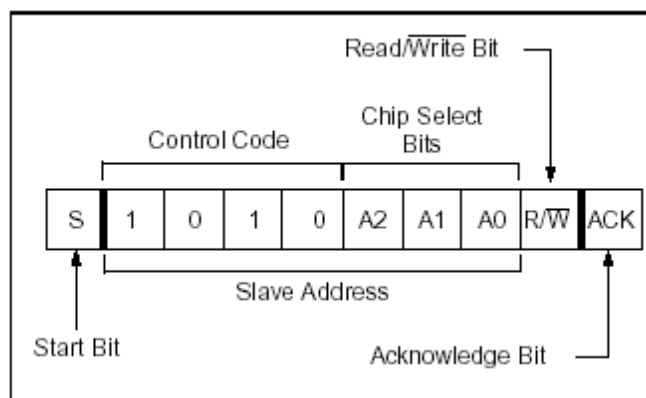
## 2.2 หน่วยความจำ EEPROM

### 2.2.1 คุณสมบัติของ IC EEPROM24LCXX

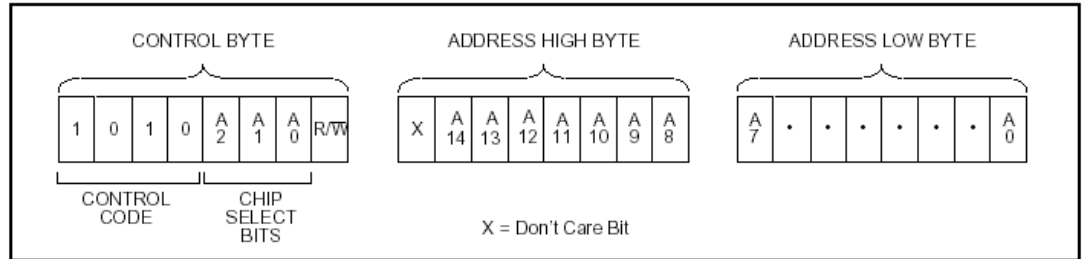


รูปที่ 2.7 ขาสัญญาณของ หน่วยความจำ EEPROM

- **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<sup>2</sup>C ใช้สำหรับรับส่งข้อมูลระหว่าง EEPROM และไมโครคอนโทรลเลอร์ โดยจะทำหน้าที่เป็น Input ในการรับข้อมูลจากไมโครคอนโทรลเลอร์ที่จะส่งให้กับ EEPROM และในทางกลับกันก็จะทำหน้าที่เป็น Output สำหรับส่งข้อมูลจาก EEPROM ให้กับไมโครคอนโทรลเลอร์
- **SCL** เป็นขาสัญญาณนาฬิกา Input ของ I<sup>2</sup>C ใช้สำหรับควบคุมการรับส่งหรืออ่านเขียนข้อมูลระหว่างไมโครคอนโทรลเลอร์และ EEPROM
- **WP** เป็นขาสัญญาณ Write Protect โดยมีสถานะเป็น Input ทำหน้าที่ป้องกันการเขียนข้อมูลให้กับ EEPROM โดยถ้าขานี้มีสถานะเป็น “0” จะสามารถส่งเขียนข้อมูลให้กับ EEPROM ได้ แต่ถ้าขานี้มีสถานะเป็น “1” จะไม่สามารถส่งเขียนข้อมูลให้กับ EEPROM ได้
- **VDD** เป็นขาสัญญาณไฟเลี้ยงวงจรของ EEPROM 24LC256 ขนาดของจำนวนความจุในการเก็บข้อมูล เป็น 256 Kbit ข้อมูลที่เก็บได้ครั้งละ 8 Bit ดังนั้นจะเก็บข้อมูลได้ เป็น (32K \*8) สามารถอ้าง Address ได้ เป็น 32 \* 1024 = 32768 นั่นก็จะมีค่าเท่ากับ 215 (A0 – A14)



รูปที่ 2.8 Control Byte Format



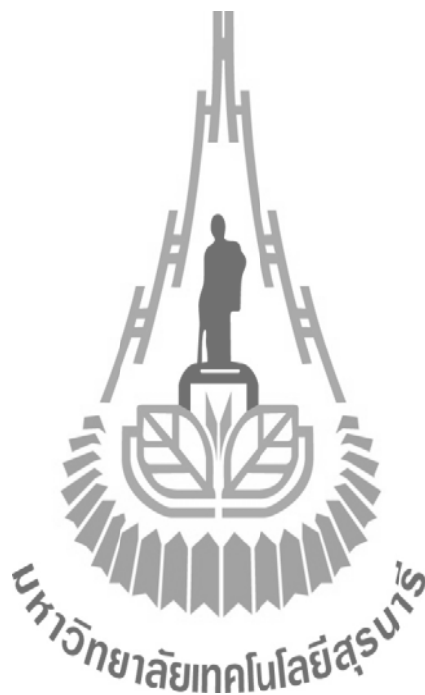
รูปที่ 2.9 Address Sequence Bit Assignments

### 2.2.2 ขั้นตอนการเขียนข้อมูลลงใน 24LCXX

1. ส่งสถานะเริ่มต้น (Start Condition) ไปยังบัส เพื่อเริ่มต้นการสื่อสาร
2. ส่งรหัส Control Byte ของ EEPROM สำหรับการเขียน ซึ่งก็คือ “10101110”
3. ส่งค่าตำแหน่งแอดเดรสไบต์สูงที่ต้องการเขียนข้อมูลไปให้ EEPROM
4. ส่งค่าตำแหน่งแอดเดรสไบต์ต่ำที่ต้องการเขียนข้อมูลไปให้ EEPROM
5. ส่งค่าข้อมูลที่ต้องการเขียนไปยัง EEPROM ตำแหน่งแอดเดรสที่ระบุในข้อ 3 และ 4
6. ส่งค่าข้อมูลไบต์ถัดไปที่ต้องการเขียนไปยัง EEPROM จนกว่าจะครบ Page หรือส่งค่าสถานะสิ้นสุด (Stop Condition) เพื่อจบการสื่อสารถ้าต้องการเขียนเพียง 1 ไบต์ (Byte Write)
7. ส่งสถานะสิ้นสุด (Stop Condition) ไปยังบัสเพื่อจบการสื่อสาร

### 2.2.3 ขั้นตอนการอ่านข้อมูลลงใน 24LCXX

1. ส่งสถานะเริ่มต้น (Start Condition) ไปยังบัส เพื่อเริ่มต้นการสื่อสาร
2. ส่งรหัส Control Byte ของ EEPROM สำหรับการเขียน ซึ่งก็คือ “10101110”
3. ส่งค่าตำแหน่งแอดเดรสไบต์สูงที่ต้องการเริ่มต้นการอ่านข้อมูลไปให้ EEPROM
4. ส่งค่าตำแหน่งแอดเดรสไบต์ต่ำที่ต้องการเริ่มต้นการอ่านข้อมูลไปให้ EEPROM
5. ส่งรหัส Control Byte ของ EEPROM สำหรับการอ่าน ซึ่งก็คือ “10101111”
6. อ่านข้อมูลจากหน่วยความจำจากแอดเดรสที่ระบุในข้อ 1.3 และ 1.4 จำนวน 1 ไบต์
7. ส่งสถานะสิ้นสุด (Stop Condition) ไปยังบัสเพื่อจบการสื่อสาร

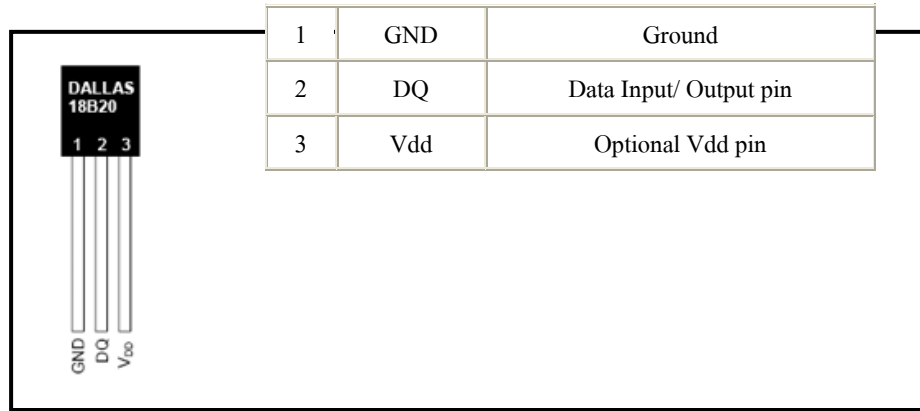


## 2.3 เซนเซอร์ตรวจวัดอุณหภูมิ (DS18B20)

### 2.3.1 ข้อมูลทั่วไปของ เทอร์โมมิเตอร์ DS18B20

DS18B20 เป็น IC วัดอุณหภูมิแบบดิจิตอล ของ Dallas Semiconductor สามารถวัดอุณหภูมิเป็นหน่วยองศา C ในช่วง  $-55^{\circ}\text{C}$  ถึง  $125^{\circ}\text{C}$  ที่ความละเอียด 9-12 บิต และมีความแม่นยำอยู่ที่  $0.5^{\circ}\text{C}$  ในช่วง  $-10^{\circ}\text{C}$  ถึง  $85^{\circ}\text{C}$  ในกรณีที่เป็นตัวถังแบบ TO-92 นั้นจะมีโครงสร้าง และขาแสดงในรูปที่ 2.10

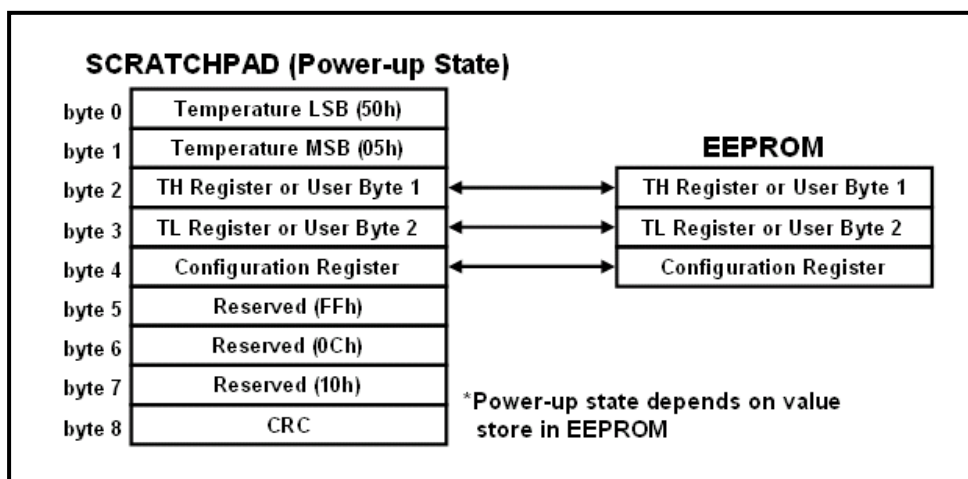
PIN	SYMBOL	Description
-----	--------	-------------



รูปที่ 2.10 โครงสร้าง และขาของ DS18B20 ตัวถังแบบ TO-92

การสื่อสารและควบคุม DS18B20 นั้นสามารถทำได้โดยใช้บัสข้อมูลแบบ 1-wire ของ Dallas Semiconductor ซึ่งใช้สายสัญญาณเพียงเส้นเดียวเท่านั้น ภายใน DS18B20 แต่ละตัวจะมีโค้ดประจำตัวขนาด 64 บิต ทำให้สามารถใช้งาน DS18B20 หลายตัวทำงานบนบัสแบบ 1-wire พร้อมกันได้ นอกจากนี้ DS18B20 ยังสามารถทำงานในโหมดพาราสิต (Parasite Power -Mode) ซึ่งเป็นการทำงานโดยไม่ใช่ไฟเลี้ยง แต่ใช้พลังงานจากสายสัญญาณ 1-wire ซึ่งมีประโยชน์มากสำหรับการวัดอุณหภูมิระยะไกล หรือในการใช้งานในที่ๆ มีเนื้อที่จำกัด แต่ในที่นี้จะกล่าวถึงรายละเอียดการใช้งานขั้นพื้นฐานในโหมดธรรมดาเท่านั้น

โครงสร้างรีจิสเตอร์ภายในของ DS18B20 มีลักษณะดังแสดงในรูปที่ 2.11 จะเห็นได้ว่าประกอบไปด้วย SRAM Scratchpad ขนาด 9 ไบต์ และ EEPROM ขนาด 3 ไบต์ ซึ่งใช้เก็บค่าอุณหภูมิสูงสุด ( TH ) ต่ำสุด ( TL ) สำหรับเปรียบเทียบการเกิดสัญญาณเตือนและรีจิสเตอร์ควบคุม (Configuration Register)



รูปที่ 2.11 โครงสร้างรีจิสเตอร์ภายในของ DS18B20

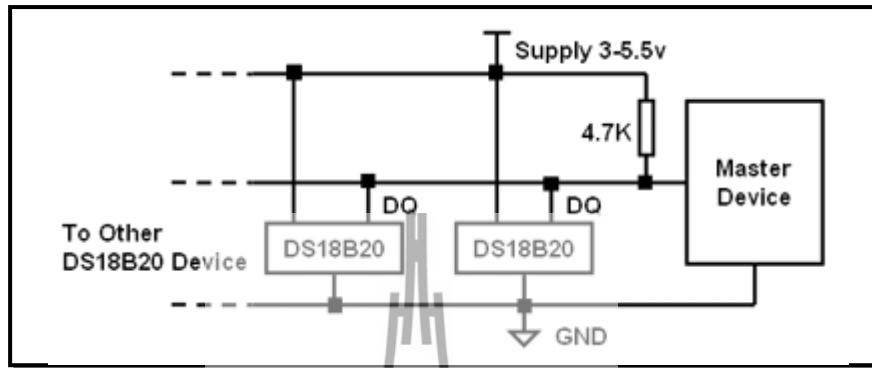
ข้อมูลอุณหภูมิที่วัดได้จะถูกเก็บอยู่ในรีจิสเตอร์ Temperature ซึ่งมีขนาด 16 บิต ดังแสดงในรูปที่ 2.12 ถ้าข้อมูลอุณหภูมิเป็นบวก S จะเป็น “1” แต่ถ้าข้อมูลอุณหภูมิเป็นลบ S จะเป็น “0” ในกรณีที่ DS18B20 ทำงานในโหมดความละเอียด 12 บิต บิตทุกบิตในรีจิสเตอร์ Temperature จะถูกใช้หมด แต่ในกรณีที่ทำงานในโหมด 9-11 บิต บิตล่าง (บิต 0 – บิต 2) จะไม่ถูกใช้งาน ซึ่งในการกำหนดโหมดความละเอียดการทำงานของ DS18B20 นั้นสามารถกำหนดได้ที่รีจิสเตอร์ Configuration ซึ่งโดยปกติเริ่มต้น DS18B20 จะทำงานในโหมด 12 บิต

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
LS Byte	$2^3$	$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
MS Byte	S	S	S	S	S	$2^6$	$2^5$	$2^4$

รูปที่ 2.12 โครงสร้างภายในรีจิสเตอร์ Temperature LSB และ MSB

การสื่อสารแบบ 1-wire เป็นระบบบัสข้อมูลแบบ Half-duplex นั่นคือสามารถสื่อสารได้ 2 ทิศทาง แต่ไม่สามารถรับ และส่งข้อมูลได้พร้อมกันในช่วงเวลาเดียวกันได้ ระบบบัสมีการทำงานเป็นแบบ Master/Slave โดยอุปกรณ์ Master จะเป็นตัวควบคุมสถานะ และจังหวะการรับส่งของบัสข้อมูล ในขณะที่อุปกรณ์ Slave จะทำงานตามการควบคุมของอุปกรณ์ Master เท่านั้น

ในการใช้งานบัสแบบ 1-wire นี้ สายสัญญาณข้อมูล DQ จะต้องมีสถานะปกติที่ลอจิกสูง สามารถทำได้โดยการต่อตัวต้านทานประมาณ 5 กิโลโอห์ม Pull Up ไว้กับไฟเลี้ยง หรือในกรณีที่ใช้งานบัสแบบ 1-wire ร่วมกับอุปกรณ์ DS18B20 หลายตัว ก็สามารถทำได้ดังแสดงในรูปที่ 2.13

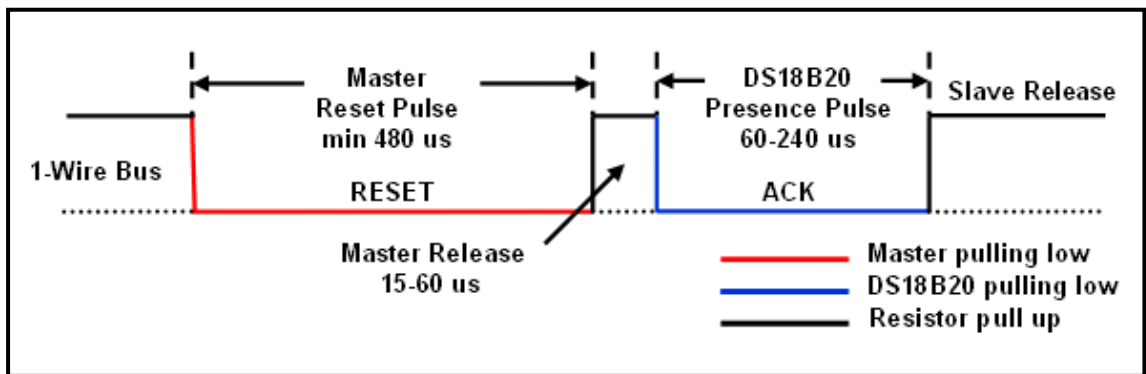


รูปที่ 2.13 การต่อใช้งาน DS18B20

รูปแบบของสัญญาณบนบัส 1-wire สามารถแบ่งออกได้เป็น 6 รูปแบบ คือ Reset pulse, Presence pulse, write 0, write 1, read 0, read 1

### 2.3.2 การสื่อสารแบบ 1-wire

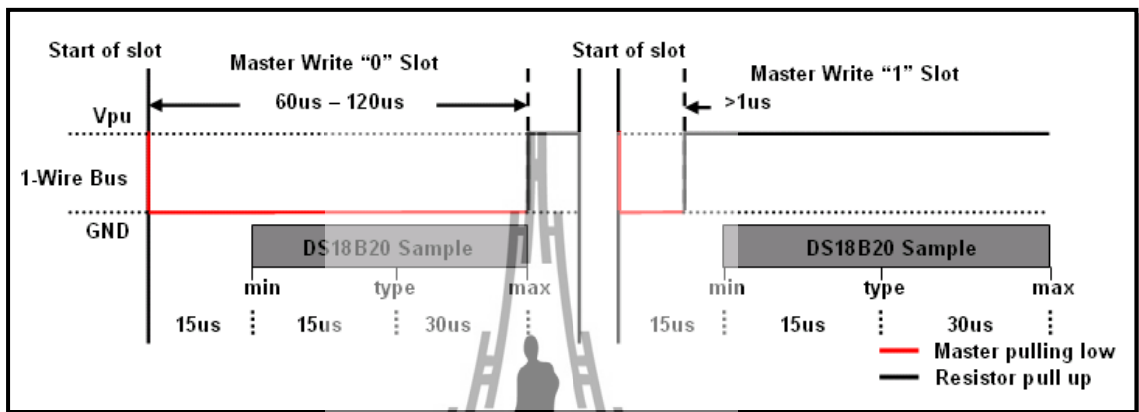
ในกระบวนการเริ่มต้นการสื่อสารแบบ 1-wire ทั้งหมดนั้น อุปกรณ์ Master ต้องขอเริ่มการสื่อสารด้วยการสร้าง Reset pulse ก่อน เมื่ออุปกรณ์ Slave ได้รับ Reset pulse ก็จะสร้าง Presence pulse เพื่อตอบรับการขอเริ่มการสื่อสารนั้น ซึ่งมีรายละเอียดของช่วงเวลาต่างๆ ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 การเริ่มการติดต่อสื่อสารแบบ 1-wire ด้วย Reset pulse และ Presence pulse



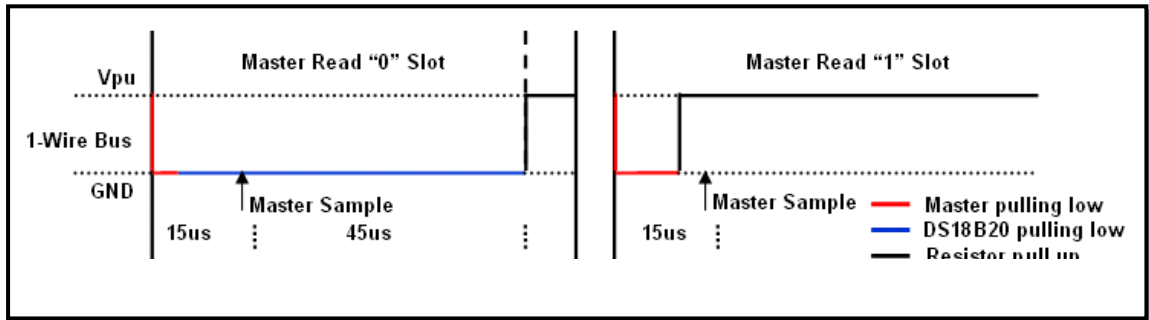
ในการเขียนข้อมูลแบ่งออกเป็น 2 ชนิด คือ การเขียนข้อมูล “ 1” และการเขียนข้อมูล “ 0” ดังแสดงในรูปที่ 2.15 การเขียนข้อมูลลง DS18B20 ต้องใช้ช่วงเวลาของ Time slot อย่างต่ำ 60  $\mu$ sec และต้องมีช่วงเวลาระหว่าง Time slot อย่างต่ำ 1  $\mu$ sec



รูปที่ 2.15 การเขียนข้อมูลลง DS18B20

การเขียนข้อมูลทั้ง 2 ชนิด เริ่มแรกอุปกรณ์ Master ต้องดึงสัญญาณบนบัส 1-wire ลงมาให้อยู่ในสถานะลอจิกต่ำก่อน ในกรณีที่ต้องการเขียนข้อมูล “ 0” ลงใน DS18B20 อุปกรณ์ Master ต้องดึงสัญญาณบนบัสให้เป็นลอจิกต่ำต่อจนกว่าจะครบช่วงเวลา Time slot (อย่างต่ำ 60  $\mu$ sec) ส่วนในกรณีที่ต้องการเขียนข้อมูล “ 1” ลง DS18B20 อุปกรณ์ Master ต้องปล่อยบัส เพื่อให้บัสกลับไปอยู่ในสถานะลอจิกสูงก่อนการ Sampling ของ DS18B20 ซึ่งอยู่ในช่วง 15  $\mu$ sec – 60  $\mu$ sec หลังจากให้อุปกรณ์ Master ดึงสัญญาณบัส 1-wire ลงมา

ในการอ่านค่าภายใน SRAM ของ DS18B20 สามารถทำได้ก็ต่อเมื่ออุปกรณ์ Master ได้เขียนข้อมูลเพื่อขอทำการอ่านค่าใน SRAM (Read Scratchpad) ซึ่งมีค่าเป็น 0xBE ลงไปที่ DS18B20 เสียก่อน จากนั้นจึงเริ่มอ่านข้อมูลจากบัส 1-wire โดย Time slot ของการอ่านต้องมีช่วงเวลาอย่างต่ำ 60  $\mu$ sec และต้องมีช่วงเวลาระหว่าง Time slot อย่างต่ำ 1  $\mu$ sec ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 การอ่านข้อมูลจาก DS18B20

การอ่านข้อมูลจากบัส 1-wire เริ่มแรกอุปกรณ์ Master จะต้องดึงบัส 1-wire ลงให้อยู่ในสถานะลอจิกต่ำเป็นช่วงเวลาอย่างน้อย 1  $\mu\text{sec}$  จากนั้นจึงปล่อยบัส ในกรณีที่ DS18B20 ส่งข้อมูล “0” DS18B20 จะดึงบัสให้เป็นลอจิกต่ำจนกว่าจะสิ้นสุด Time slot ถึงจะปล่อยบัสให้กลับไปอยู่ในสถานะลอจิกสูง ส่วนในกรณีที่ DS18B20 ส่งข้อมูล “1” DS18B20 จะปล่อยบัสให้อยู่ในสถานะลอจิกสูงตลอด ในการ sampling เพื่อรับข้อมูลจาก DS18B20 ควรทำภายใน 15  $\mu\text{sec}$  หลังจากจุดเริ่มของ Time slot ดังแสดงในรูปที่ 2.16

### 2.3.3 ขั้นตอนการเข้าใช้งาน DS18B20

ขั้นตอนการเข้าใช้งาน DS18B20 มี 3 ขั้นตอน คือ

- Initialization
- ROM Command
- DS18B20 Function Command

การ Initialization ประกอบไปด้วยการส่ง Reset pulse จากอุปกรณ์ Master ตามด้วย Presence pulse ซึ่งตอบรับโดย DS18B20 เพื่อบ่งบอกว่าอุปกรณ์พร้อมทำงาน หลังจากการทำ Initialization เสร็จเรียบร้อยแล้ว อุปกรณ์ Master ต้องส่ง ROM Command ไปยัง DS18B20 ROM Command นั้นแบ่งออกได้เป็น 5 คำสั่งด้วยกันคือ SEARCH ROM [F0h], READ ROM [33h], MATCH ROM [55h], SKIP ROM [CCh], ALARM SEARCH [ECh] ซึ่งในกรณีที่ต่อใช้งาน DS18B20 เพียงตัวเดียวนั้นจะใช้ ROM Command ได้แค่ 2 คำสั่ง นั่นคือ READ ROM ซึ่งเป็นการอ่านค่า ROM Code ขนาด

64 บิต บนตัว DS18B20 อีกคำสั่งคือ SKIP ROM ซึ่งเป็นคำสั่งที่ใช้ในกรณีที่อุปกรณ์ Master ต้องการส่งคำสั่งควบคุม DS18B20 ทุกตัว ซึ่งไม่จำเป็นต้องระบุ ROM Code

หลังจากที่อุปกรณ์ Master ส่ง ROM Command ไปยัง DS18B20 แล้ว อุปกรณ์ Master จะสามารถใช้ Function Command เพื่อเข้าไปควบคุมการทำงานของ DS18B20 ได้ Function Command ประกอบไปด้วย CONVERT T [44h], WRITE SCRATCHPAD [4Eh], READ SCRATCHPAD [BEh], COPY SCRATCHPAD [48h], RECALL E2 [B8h], READ POWER SUPPLY [B4h]

### 2.3.4 ตัวอย่างการต่อใช้งาน DS18B20 กับไมโครคอนโทรลเลอร์

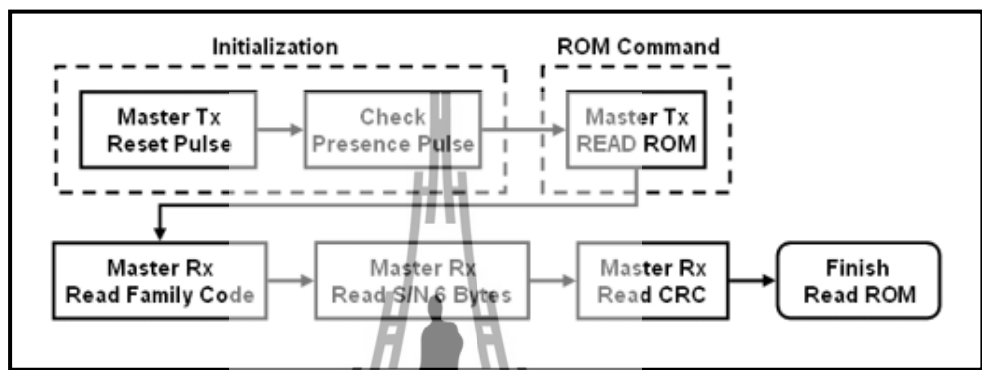
ในตัวอย่างการต่อใช้งานนี้ได้ต่อ DS18B20 กับไมโครคอนโทรลเลอร์ P89V51RD2 โดยใช้พอร์ต P1.1 เพื่อเป็นพอร์ตสื่อสารแบบ 1-wire ดังแสดงในรูปที่ 2.17 และแสดงผลการทำงานโดยใช้โปรแกรม Hyper Terminal สื่อสารกับพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ด้วยอัตราข้อมูล 9600 bps



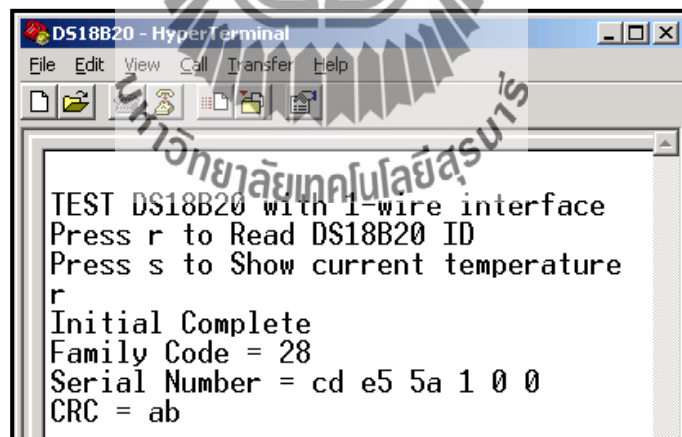
รูปที่ 2.17 การต่อวงจรทดสอบการใช้งาน DS18B20

ในส่วนโปรแกรม ไมโครคอนโทรลเลอร์จะคอยตรวจสอบพอร์ตอนุกรม ในกรณีที่ผู้ใช้งานพิมพ์ตัวอักษร “r” ผ่านโปรแกรม Hyper Terminal เข้ามาทางพอร์ตอนุกรม ไมโครคอนโทรลเลอร์จะอ่าน ROM Code จาก DS18B20 ซึ่งมีขั้นตอนการทำงานตามรูปที่ 2.18 จะเห็นได้ว่าไมโครคอนโทรลเลอร์ซึ่งเป็นอุปกรณ์ Master จะต้องทำการ Initialization โดยส่ง Reset pulse ไปที่บัส 1-wire ก่อน จากนั้นจึงคอยตรวจสอบ Presence pulse แสดงว่า DS18B20 นั้นพร้อมที่จะทำงานและเป็นการสิ้นสุดกระบวนการ Initialization ไมโครคอนโทรลเลอร์จึงค่อยส่งคำสั่ง READ ROM

[33h] ซึ่งเป็น ROM Command ไปยัง DS18B20 จากนั้นจึงค่อยอ่านค่า ROM Code จาก DS18B20 ซึ่งมีขนาด 8 ไบต์ (64 บิต) ทีละไบต์ โดยไบต์แรกนั้นเป็น Family Code ไบต์ที่ 2-7 เป็น Serial Number ส่วนไบต์สุดท้ายเป็น CRC ตามลำดับ พร้อมกับส่งข้อมูล ROM Code ที่อ่านได้นั้นออกมาทางพอร์ตอนุกรมเพื่อแสดงในโปรแกรม Hyper Terminal ในรูปที่ 2.19



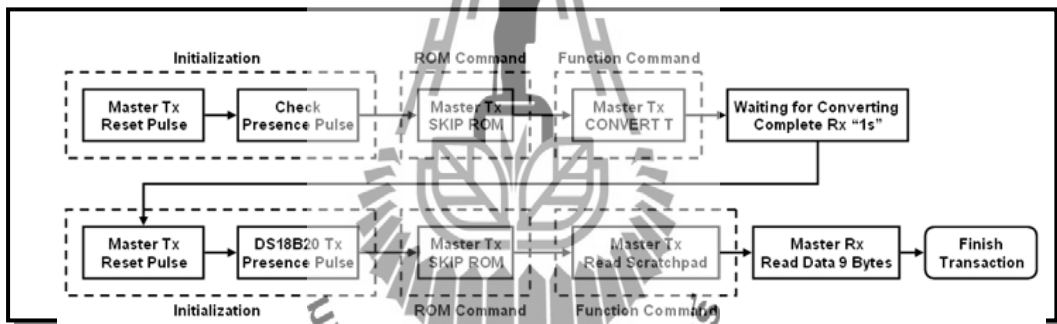
รูปที่ 2.18 ขั้นตอนการอ่าน ROM Code จาก DS18B20



รูปที่ 2.19 ผลการอ่าน ROM Code ขนาด 64 บิต จาก DS18B20

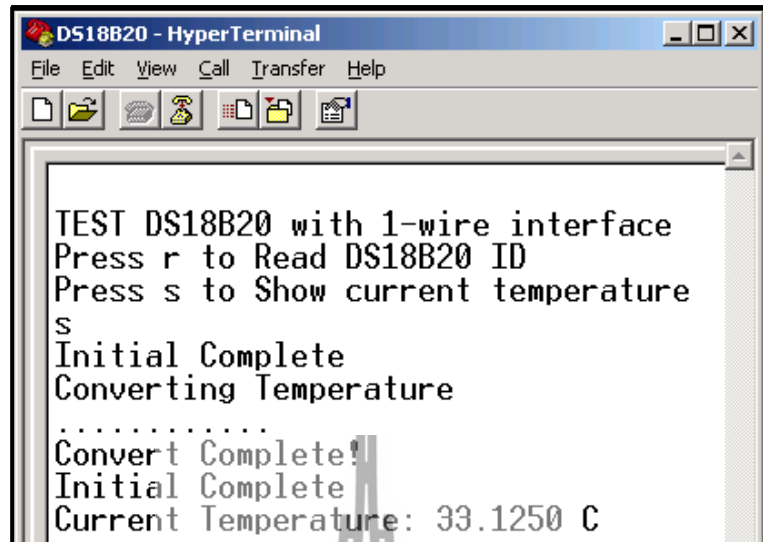
ในกรณีผู้ใช้งานพิมพ์ตัวอักษร “ r ” ผ่านทางโปรแกรม Hyper Terminal เข้ามาทางพอร์ตอนุกรม ไมโครคอนโทรลเลอร์จะอ่านค่าอุณหภูมิจาก DS18B20 และแสดงค่าอุณหภูมิที่อ่านได้ขณะนั้นออกมาทางพอร์ตอนุกรมดังแสดงในโปรแกรม Hyper Terminal ในรูปที่ 2.21

ขั้นตอนการอ่านค่าอุณหภูมิจาก DS18B20 นั้น แบ่งออกเป็น 2 ขั้นตอนใหญ่ๆ คือ Converting Temperature ซึ่งเป็นการแปลงอุณหภูมิให้อยู่ในรูปแบบข้อมูลดิจิทัลเก็บไว้ใน Scratchpad และ Read Scratchpad ซึ่งเป็นการอ่านข้อมูลที่เก็บใน Scratchpad นั้นออกมา ดังแสดงในรูปที่ 2.20 จะเห็นได้ว่าไมโครคอนโทรลเลอร์ซึ่งเป็นอุปกรณ์ Master จะต้องทำการ Initialization จากนั้นจึงค่อยใช้คำสั่ง SKIP ROM [CCh] ซึ่งเป็น ROM Command ที่ใช้ในกรณีที่ต่อใช้งาน DS18B20 เพียงตัวเดียว จากนั้นจึงค่อยส่ง Function Command CONVERT T [44h] เพื่อสั่งให้ DS18B20 ทำการแปลงอุณหภูมิให้อยู่ในรูปแบบดิจิทัลและเก็บไว้ใน Scratchpad กระบวนการแปลงนี้จะใช้เวลาประมาณ 750 ms ซึ่งเราสามารถตรวจสอบสถานะการแปลงข้อมูลได้จากการเช็คสถานะของบัส 1-wire ในกรณีที่ DS18B20 ยังทำการแปลงข้อมูลอยู่บัส 1-wire จะมีสถานะเป็นลอจิกต่ำ แต่ถ้า DS18B20 ได้ทำการแปลงข้อมูลเสร็จเรียบร้อยแล้ว บัส 1-wire จะกลับมาอยู่ในสถานะปกติ คือ ลอจิกสูง



รูปที่ 2.20 ขั้นตอนการแปลง และอ่านอุณหภูมิจาก DS18B20

เมื่อ DS18B20 แปลงข้อมูลอุณหภูมิเป็นดิจิทัลเก็บไว้ใน Scratchpad แล้วไมโครคอนโทรลเลอร์สามารถอ่านข้อมูลภายใน Scratchpad นั้น โดยทำการ Initialization บัส 1-wire อีกครั้ง จากนั้นจึงส่งคำสั่ง SKIP ROM [CCh] ซึ่งเป็น ROM Command และส่งคำสั่ง READ SCRATCHPAD [BEh] ซึ่งเป็น Function Command เพื่อขออ่านข้อมูล Scratchpad ภายใน DS18B20 จากนั้นไมโครคอนโทรลเลอร์จึงทำการอ่านข้อมูลภายใน Scratchpad ของ DS18B20 มาทีละไบต์มาจนครบ 9 ไบต์ และแสดงผลข้อมูลอุณหภูมิที่เป็นข้อมูลในไบต์แรก และไบต์ 2 ที่อ่านมาจาก Scratchpad นั้นออกมาทางพอร์ตอนุกรม ดังแสดงในรูปที่ 2.20



รูปที่ 2.21 ผลการอ่านอุณหภูมิจาก DS18B20

## 2.4 เครื่องวัด SWR และ Power

### 2.4.1 Wattmeter CN-101-L



รูปที่ 2.22 เครื่องวัด SWR และ Power

Daiwa CN-101L คือ SWR ข้ามเข็ม / wattmeter ครอบคลุม 1.8-150 MHz การเคลื่อนไหวก้ามเข็มช่วยให้การตรวจสอบพร้อมกันของอำนาจไปข้างหน้าไฟสะท้อนและ SWR ทั้งหมดโดยไม่ต้องของขั้นตอนการสอบเทียบเสริมเมตรเป็น switchable สำหรับ 15, 150 หรือ 1500 อ่านขนาดวัดได้เต็มและมีโคมไฟในตัวที่สามารถขับเคลื่อนจากภายนอกที่มา VDC 13.8 มาพร้อมกับสายไฟ DC ซึ่งเป็นสิ่งจำเป็นสำหรับการเปิดเครื่องเพียงหลอดไฟในตัว

แม้ว่าจะมีการเปลี่ยนโหมด PEP, CN-101L ไม่ได้ถูกออกแบบอย่างดีที่สุดสำหรับอำนาจสูงสุดหรืออำนาจสูงสุดของ (PEP) วัด

#### 2.4.2 คุณสมบัติของ Wattmeter CN-101-L

CN101L DAIWA Power / SWR เหมาะสำหรับการปรับแต่งระบบเสาอากาศและการวัดพลังงาน SWR อื่นๆ ต้องมีรายการสำหรับการปรับที่เหมาะสมและจุดส่งสัญญาณและวงจรรขยาย RF หน่วยนี้จะครอบคลุม ทั้งสเปกตรัมจาก 1.8 ถึง 150MHz ด้วยความถูกต้อง 5% ผลการดำเนินงานเกี่ยวกับคลื่นขนาดกลาง ที่เป็นที่ยอมรับทำให้หน่วยนี้มากมีประโยชน์สำหรับเครื่องส่งสัญญาณ MAX ของเราเช่นกัน ขนาดมิเตอร์ขนาดใหญ่จะมีการสอบเทียบสำหรับไปข้างหน้าและ reverse power และอัตราส่วนคลื่นนิ่ง (SWR) สลับเฉลี่ยที่เลือก (AVG) และเพาเวอร์ของ Peak (PEP) เมตรมีสามช่วงพลังงานที่มีสวิทช์เลือกตามที่ระบุไว้ ( 15W/150W และ 1500W) เมตรนี้ได้ อย่างถูกต้องสามารถ อ่านระดับพลังงานต่ำเป็น 50 มิลลิวัตต์ ปัจจัยการผลิตและผลที่มีความต้านทาน 50 โอห์ม

โดยใช้การเชื่อมต่อ SO239

##### Specifications ทางเทคนิค

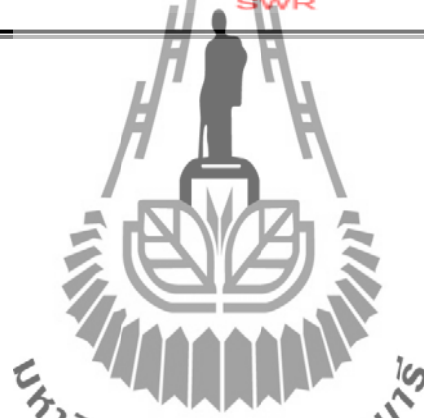
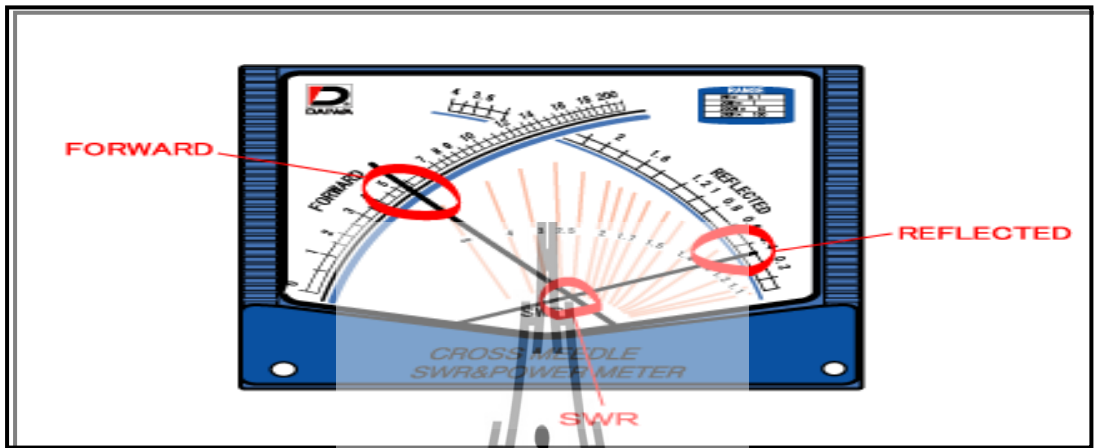
- ความต้านทาน: 50 โอห์ม
- Freq ช่วง: 1.8-150 MHz
- Power ขนาดเต็ม: 15W/150W/1500W ดำเนินการได้ดีภายใต้ 1.8 MHz

มีความไวค่อนข้างลดลง

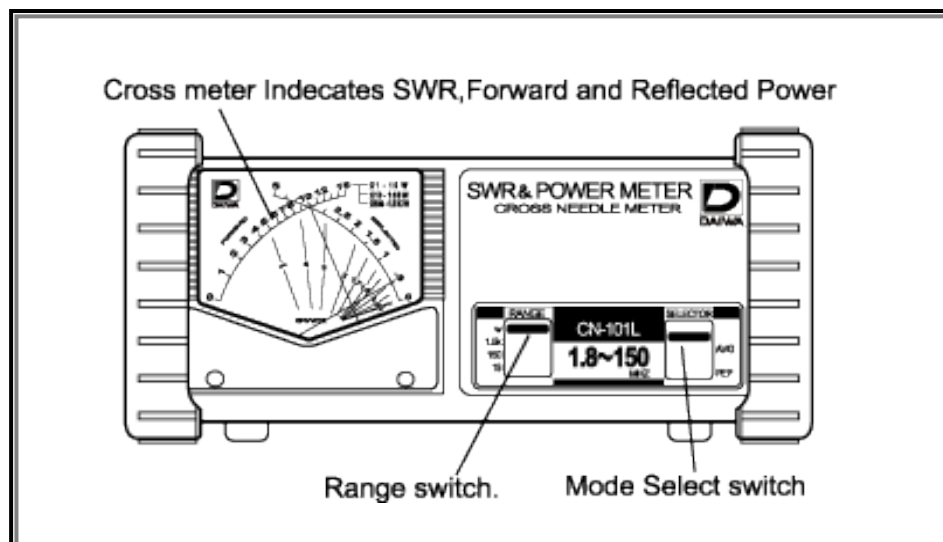
#### ตารางที่ 2.4.1 คุณสมบัติของ Wattmeter CN-101-L

ช่วงความถี่	1.8 – 150 MHz
ช่วง Power (หน้า)	15/ 150/ 1500 วัตต์
การประเมินพลังงาน	1500 วัตต์ (1.8 – 60 MHz) 1000 วัตต์ (60 – 150 MHz)
ความถูกต้อง	+ เต็มสเกล% - 10
การตรวจสอบความไว SWR	4 วัตต์ขั้นต่ำ
ความต้านทาน	50 โอห์ม

ตัวเชื่อมต่อ	การป้อนข้อมูล: SO-239 Output: SO-239
ขนาด	3" x 5" W x 4" D

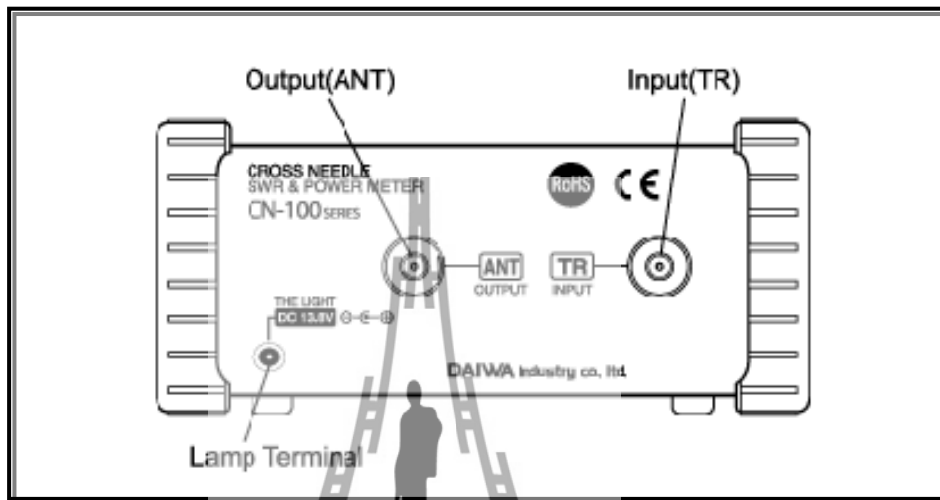


รูปที่ 2.23 ตำแหน่งแสดงที่อ่านค่าบนหน้าปัดเครื่อง Wattmeter





รูปที่ 2.24 จุดเชื่อมต่อบนเครื่อง Wattmeter (ด้านหน้า)



รูปที่ 2.25 จุดเชื่อมต่อบนเครื่อง Wattmeter (ด้านหลัง)

#### 2.4.3 DESCRIPTION (ค่าเฉลี่ยของพลังงาน FM)

ซีรีส์ CN-101 เป็นเครื่องมือที่มีคุณภาพสูงที่มีเอกลักษณ์เฉพาะที่ทำการวัด SWR และ Power ในระหว่างการทดสอบเสาอากาศที่ตรงกันและใช้บนเครื่องส่งสัญญาณที่ใช้งานง่ายตัววัด SWR และ Power มีการติดตั้งในหน่วยหนึ่งเมตร หนึ่งขนาดจะระบุกำลังไฟไปข้างหน้าในระดับอื่น สะท้อนพาวเวอร์และ SWR จะแสดงที่จุดข้ามจาก 2 คุณลักษณะเฉพาะ needles. This ทำให้สามารถที่จะอ่านไปข้างหน้าพาวเวอร์, ไฟสะท้อนและ SWR ทั้งหมดในเวลาเดียวกัน

#### 2.4.4 การใช้งานเครื่องวัด SWR และ Power

1. เลือกสลับโหมดไปที่ "AVG" ตำแหน่ง FM

ขนาด "Forward" บ่งชี้ไปข้างหน้า Power Level สะท้อนบ่งบอกถึงพลังสะท้อน มีผลบังคับใช้รังสีพลังงานในการวัดประสิทธิภาพแผ่พลังงานลบไฟสะท้อนจาก Power Forward

2. อำนวยการตรวจสอบ PEP เปิดสวิตช์โหมดไปที่ตำแหน่ง PEP SSB เมื่อส่งสัญญาณเป็นผู้ดำเนินการและสวิตช์อยู่ใน PEP ตำแหน่งเข็มมิเตอร์ PEP การตรวจสอบของสัญญาณ SSB สำหรับการตรวจสอบ PEP, condenser จะวางอยู่ในวงจรเครื่องตรวจจับ

#### 2.4.5 ข้อควรระวัง

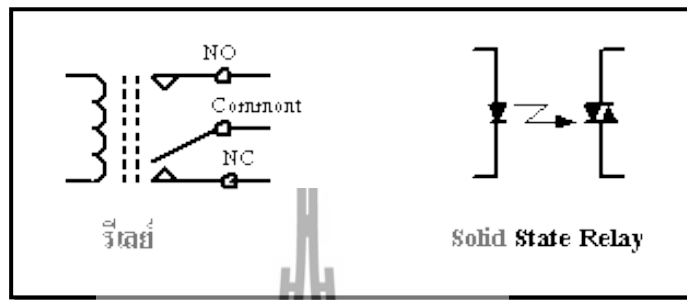
1. ใช้เฉพาะ 50 ohms เล้าโลมสายสำหรับการเชื่อมต่อ นี้จะรักษาความถูกต้องของเครื่องวัด
2. สำหรับการตรวจวัดพลังงานที่ถูกต้องให้ใช้ 50 ohms บริสุทธิ์ resistance โหลด dummy
3. การเคลื่อนไหวของมิเตอร์มีความไวสูง การป้องกันไม่ให้ mechanical ช็อคและการสั่นสะเทือน
4. การวัดพลังงานที่จับคู่กับเสาอากาศได้ไม่ดีหรือ disconnecting การส่งออกของสะพาน ในขณะที่ปฏิบัติการอาจเกิดความเสียหาย



## 2.5 วงจรขั้วรีเลย์

### 2.5.1 คุณลักษณะของรีเลย์

รีเลย์และ โซลิดสเตตรีเลย์ (Solid state relay) เป็นอุปกรณ์แยกสัญญาณ อินพุตและเอาต์พุตออกจากกันโดยเด็ดขาด นิยมมาควบคุมงานที่ใช้กับไฟ AC 220 Volt ในขณะที่ชุดควบคุมใช้ไฟ DC ต่ำๆ มีสัญลักษณ์ดังรูปที่ 2.26



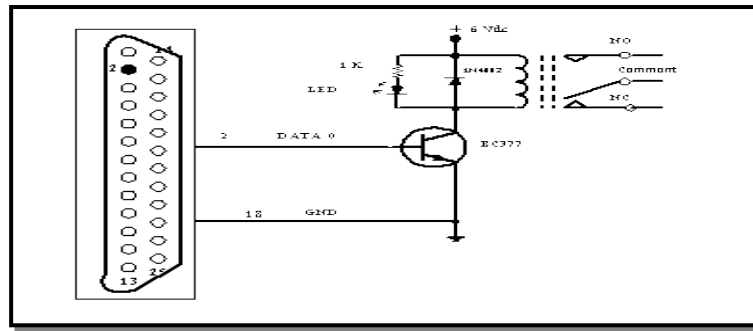
รูปที่ 2.26 แสดงสัญลักษณ์ของรีเลย์ และ โซลิดสเตตรีเลย์

ตารางที่ 2.5.1 ข้อดี และข้อเสียของรีเลย์และ โซลิดสเตตรีเลย์

Relay	Solid state Relay
1. รับ Load มากแต่กินกำลังไฟฟ้า	1. มีอายุการใช้งานนาน
2. ทำงานช้า	2. ทำงานเร็วมาก, ราคาแพง
3. อาจเกิดสัญญาณแก๊วการกระเด็น (Debounce)	3. ไม่มีปัญหาการแก๊วการกระเด็น
4. ตรวจสอบได้ง่าย	4. ใช้เวลานาน ๆ จะร้อนมาก

2.5.2 การใช้งานวงจรจับรีเลย์

ข้อมูลที่ส่งออกจาก CPU ผ่านพอร์ตขนาน (D0) นั้นอาจจะไม่มีกระแสขับสูงพอให้รีเลย์ทำงานได้ ดังนั้นนิยมนำเอาทรานซิสเตอร์ มาเป็นตัวขยายและขับกระแสก่อนถึงรีเลย์ หรืออาจใช้ตัวเชื่อมต่อผ่านแสง (Optocouple) ก็ได้ดังรูปที่ 2.27

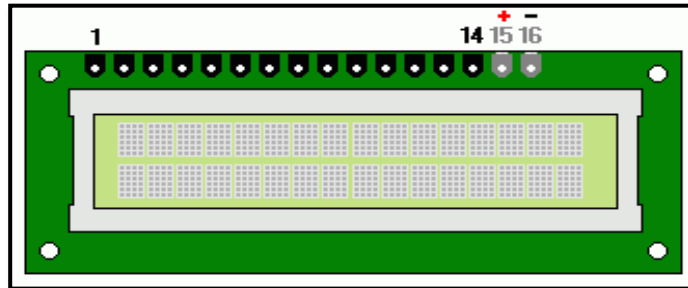


รูปที่ 2.27 แสดงวงจรขับกระแสรีเลย์ด้วยทรานซิสเตอร์

การทำงานของ CPU ต้องส่งข้อมูล “1” ออกทางบัสข้อมูล D0 เพื่อกระตุ้นให้ทรานซิสเตอร์ทำงาน มีกระแสไหลจาก +6 Vdc ผ่านรีเลย์ลิ่งกราวด์ที่ขั้วมีเตอร์ของทรานซิสเตอร์ ทำให้รีเลย์ “ON” ส่งผลให้หน้าสัมผัสของรีเลย์ขาร่วม (Common) ต่อกับ No อยู่จนครบเท่าที่ CPU จำข้อมูล “1” แก่ชุดขับกระแสไดโอด 1N4002 ต่อไว้เป็นทางผ่านของกระแสไหลย้อนกลับ กรณีที่ทรานซิสเตอร์หยุดทำงานเป็นการยืดอายุการใช้งานของรีเลย์ด้วย



## 2.6 อุปกรณ์แสดงผล (LCD 16x2)



รูปที่ 2.28 จอแสดงผล LCD 16x2 Line

### 2.6.1 การเชื่อมต่อทางฮาร์ดแวร์ และหน้าที่การใช้งาน

ตารางที่ 2.6.1 ตำแหน่งของขาและหน้าที่การใช้งานของ LCD โมดูล

Pin No.	Symbol	Description	Level	Function
1	VSS	Ground	-	Ground
2	VDD	Power Supply	-	ต่อกับแรงดันไฟเลี้ยง +5 V
3	VO	LCD Contr	-	ต่อกับแรงดันเพื่อปรับความเข้มของการแสดงผล
4	RS	Register Select	H/L	RS = 0 หมายถึงต้องการติดต่อกับรีจิสเตอร์คำสั่ง (Instruction Register) RS = 1 หมายถึงต้องการติดต่อกับรีจิสเตอร์ข้อมูล (Data Register)
5	R/W	Read/Write	H/L	R/W = 0 หมายถึงต้องการเขียนข้อมูลไปยัง LCD โมดูล R/W = 1 หมายถึงต้องการอ่านข้อมูลจาก LCD โมดูล
6	E	Enable	H, H->L	Enable Signal
7 - 14	DB0-DB7	Data Bus	H/L	Data Bus Line
15	A	Back Light A	-	Back Light +5 V (สำหรับรุ่นที่มี Back Light)
16	K	Back Light K	-	Back Light 0 V (สำหรับรุ่นที่มี Back Light)

## 2.6.2 คำสั่งควบคุมการแสดงผล

ตารางที่ 2.6.2 คำสั่งควบคุมการแสดงผล LCD

Instruction	RS	R/W	Command Code (binary)								Description		
			7	6	5	4	3	2	1	0			
1	Clear Display	0	0	0	0	0	0	0	0	0	0	1	Clear entire display and move cursor home (address 0)
2	Home Display	0	0	0	0	0	0	0	0	0	1	0	Move cursor home and return display to home position.
3	Entry Mode Set	0	0	0	0	0	0	0	1	M	S		Sets cursor direction (M: 0=left, 1=right) and display scrolling (S: 0=no scroll, 1=scroll)
4	Display/Cursor	0	0	0	0	0	0	1	D	C	B		Sets display on/off (D), cursor on/off (C) and blinking cursor (B). (0=off, 1=on)
5	Cursor or Display Shift	0	0	0	0	0	1	C	M	0	0		Cursor or Display Shift (C: 0=cursor, 1=display) left or right (M: 0=left, 1=right).
6	Function Set	0	0	0	0	0	1	D	N	F	0		Data bus size (D: 0=4-bits, 1=8-bits), lines No.(N: 0=1-line, 1=2-lines) and font size (F: 0=5x7, 1=5x10)
7	Set CG-RAM Address	0	0	0	1	CGRAM ADDRESS						Move pointer to Character Generator RAM location specified by address (ADDRESS)	
8	Set DD-RAM Address	0	0	1	DDRAM ADDRESS						Move cursor to Display Data RAM location specified by address (ADDRESS)		
9	Busy, ADD Read	0	1	BF	ADDRESS						Read Busy flag, And Address Read		
10	CGRAM,DDRAM WR	1	0	WRITE DATA						Write Data to DDRAM or CGRAM			
11	CGRAM,DDRAM RD	1	1	READ DATA						Read Data to DDRAM or CGRAM			

## รายละเอียดของคำสั่ง

1. เคลียร์การแสดงผล (Clear Display)

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

Bit0 = 1; เคลียร์การแสดงผล

เคอร์เซอร์จะกลับไปอยู่ที่มุมซ้ายสุด

2. Home Display

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	-

RS = 0, R/W = 0

Bit1 = 1; เคอร์เซอร์กลับไปอยู่ที่มุมซ้ายสุด

ข้อมูลไม่มีการเปลี่ยนแปลง

3. โหมดการป้อนข้อมูล (Entry Mode Set)

7	6	5	4	3	2	1	0
0	0	0	0	0	1	M	S

RS = 0, R/W = 0

Bit2 = 1

**M** (Address Increase/Decrease), M = 0 ลดตำแหน่งแอดเดรส, M = 1 เพิ่มตำแหน่งแอดเดรส

**S** (Shift bit) การเลื่อนข้อมูล, S = 0 เคอร์เซอร์จะเลื่อนไปทางขวา, S = 1 เคอร์เซอร์จะอยู่กับที่

4. การควบคุมการแสดงผล (Display/Cursor)

7	6	5	4	3	2	1	0
0	0	0	0	1	D	C	B

RS = 0, R/W = 0

Bit3 = 1

**D** (Display ON/OFF) D = 0 OFF, D = 1 ON

**C** (Cursor ON/OFF) C = 0 OFF, C = 1 ON

**B** (Blinking Cursor ON/OFF) B = 0 OFF, B = 1 ON

5. การควบคุมการเลื่อนเคอร์เซอร์ (Cursor or Display Shift)

7	6	5	4	3	2	1	0
0	0	0	1	C	M	-	-

RS = 0, R/W = 0

Bit4 = 1

**C** (Cursor or Display Shift) C = 0 shift cursor, C = 1 shift display

**M** (Move left/right) M = 0 left, M = 1 right

6. ฟังก์ชันเซต (Function Set)

7	6	5	4	3	2	1	0
0	0	1	D	N	F	-	-

RS = 0, R/W = 0

Bit5 = 1

**D** (Data bus size) D = 0 is 4-Bits, D = 1 is 8-Bits

**N** (lines No.) N = 0 is 1-line, N = 1 is 2-lines

**F** (font size) F = 0 is 5x7, F = 1 is 5x10

7. เซตตำแหน่งใน CG-RAM (Set CG-RAM Address)

7	6	5	4	3	2	1	0
0	1	A5	A4	A3	A2	A1	A0

RS = 0, R/W = 0

Bit6 = 1

หน่วยความจำชั่วคราว เก็บข้อมูลตัวอักษร CG-RAM (Character Generator RAM)

A0-A5 เป็นตำแหน่งแอดเดรสใน CG-RAM



8. เซตตำแหน่งใน DD-RAM (Set DD-RAM Address)

7	6	5	4	3	2	1	0
1	A6	A5	A4	A3	A2	A1	A0

RS = 0, R/W = 0

Bit7 = 1

หน่วยความจำชั่วคราว เก็บข้อมูลการแสดงผล DD-RAM (Display Data RAM)

A0-A6 เป็นตำแหน่งแอดเดรสใน DD-RAM ซึ่งจะถูกคัดลอกไปยัง Address Counter (AC)

9. การอ่าน BUSY Flag and Address Counter (BF and AC)

7	6	5	4	3	2	1	0
BF	A6	A5	A4	A3	A2	A1	A0

RS = 0, R/W = 1

BF = Bit7; เป็นตัวบอกสถานะของ LCD

R/W = 1; กำหนดให้เป็น Read mode

BF = 0 ว่าง, BF = 1 ไม่ว่าง

A0 - A6 = Address Counter (AC)

10. การเขียนข้อมูลใน CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

RS = 1, R/W = 0

RS = 1; กำหนดให้เป็นข้อมูล

R/W = 0; กำหนดให้เป็น Write mode

D0 - D7 = ข้อมูลที่ต้องการเขียน

11. การอ่านข้อมูลจาก CG or DD-RAM

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

RS = 1, R/W = 0

RS = 1; กำหนดให้เป็นข้อมูล, R/W = 1; กำหนดให้เป็น Read mode

D0-D7 = ข้อมูลที่อ่านได้

โดยที่ \* หากต้องการเขียนข้อมูลใน CG-RAM ให้เซตตำแหน่ง CG-RAM ในคำสั่งที่ 7 ก่อน

\* หากต้องการเขียนข้อมูลใน DD-RAM ให้เซตตำแหน่ง DD-RAM ในคำสั่งที่ 8 ก่อน

DD-RAM คือหน่วยความจำที่เก็บข้อมูลการแสดงผล หากเขียนรหัส ASCII ลงในหน่วยความจำนี้ก็จะปรากฏที่จอ LCD ทันที

ตำแหน่ง Address ของ LCD (2 x 16 Displays)

Line 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Line 2	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79

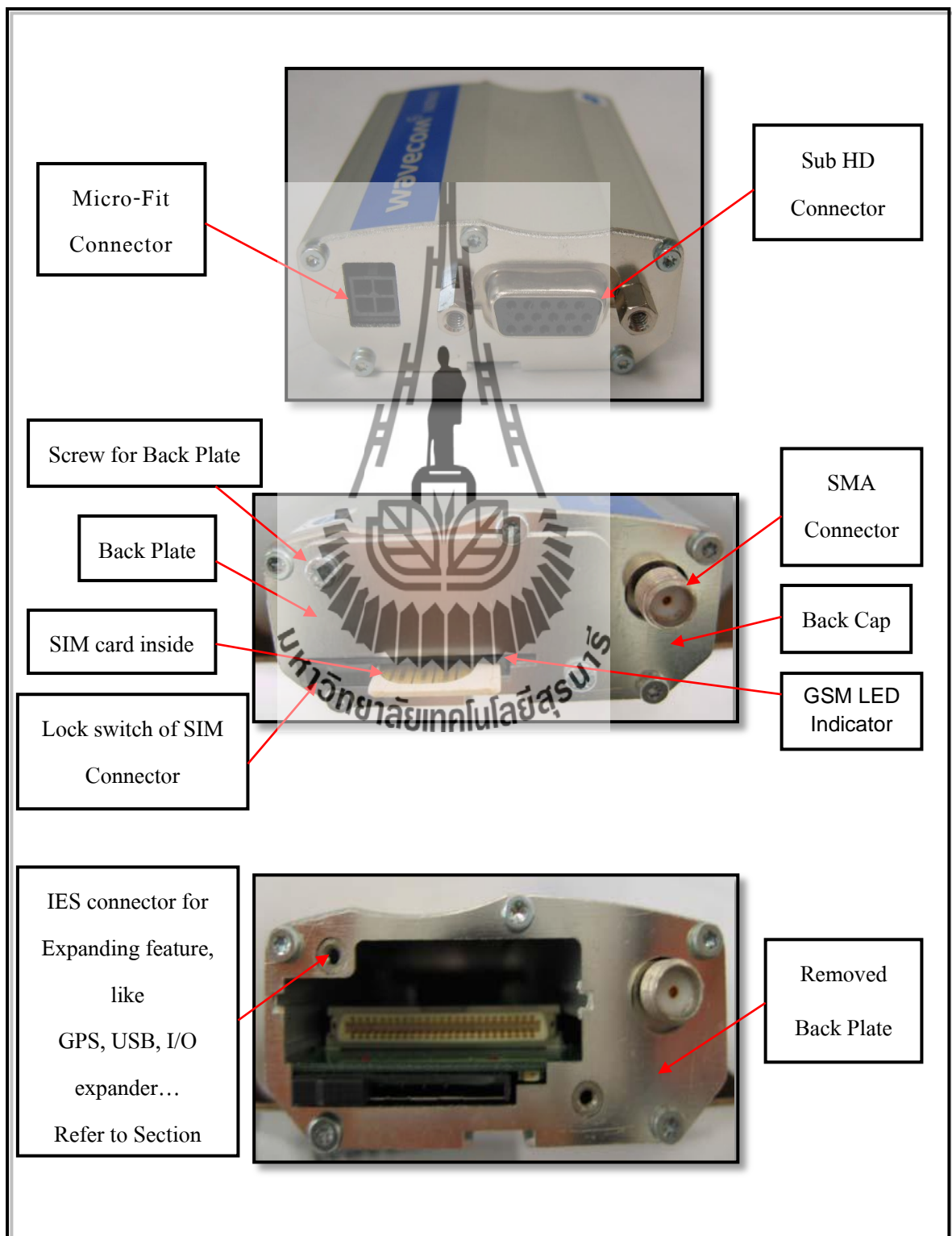
## 2.7 GSM Modem

### 2.7.1 คุณสมบัติ ของ GSM Module (wave com)

GSM Module (wave com) เป็นชุดเรียนรู้และพัฒนาาระบบการสื่อสารไร้สาย โดยใช้โมดูล GSM/GPRS รุ่น Fastrack Supreme 20 ของ “Wave come 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 ซึ่งตามปกติแล้ว ถึงแม้ว่าโมดูล Fastrack Supreme 20 จะมีวงจร และ Firmware บรรจุไว้ภายในตัวเป็นที่เรียบร้อยแล้วก็ยังไม่สามารถนำไปใช้งานได้โดยตรงทันที เนื่องจากในการใช้งานจริงๆ นั้น ผู้ใช้งานเองจำเป็นต้องออกแบบวงจรรอบนอกที่จำเป็นมาเชื่อมต่อกับขาสัญญาณของตัวโมดูลอีกในบางส่วน ไม่ว่าจะเป็น วงจรภาค Power Supply, วงจรเชื่อมต่อกับ SIM Card รวมไปถึงวงจร Line Driver ของ RS232 เป็นต้น ดังนั้น จึงได้จัดสร้างบอร์ดสำหรับเป็นตัวกลางในการเชื่อมต่อระหว่างโมดูล Fastrack Supreme 20 กับอุปกรณ์ภายนอกเพื่อให้ผู้ใช้งานสามารถนำโมดูล GSM ของ Fastrack Supreme 20 ไปทำการทดลองและศึกษาเรียนรู้การสั่งงานต่างๆ ได้โดยสะดวก ก่อนที่จะนำเอาโมดูลตัวนี้ไปออกแบบตัดแปลงและประยุกต์ใช้งานในด้านต่างๆ ได้ต่อไปในอนาคต ซึ่งถึงแม้ว่าวงจรการเชื่อมต่อทั้งหมดที่ทาง wave come ได้จัดทำขึ้นมาจะยังไม่สามารถรองรับการใช้งานทรัพยากรต่างๆ ที่มีอยู่ภายในโมดูลได้ครบถ้วนทั้งหมดก็ตามที แต่ในส่วนของการใช้งานโมดูลในส่วนที่เป็นความสามารถหลักๆ ที่จำเป็นนั้น มีไว้รองรับอย่างครบถ้วนเพียงพอแล้ว

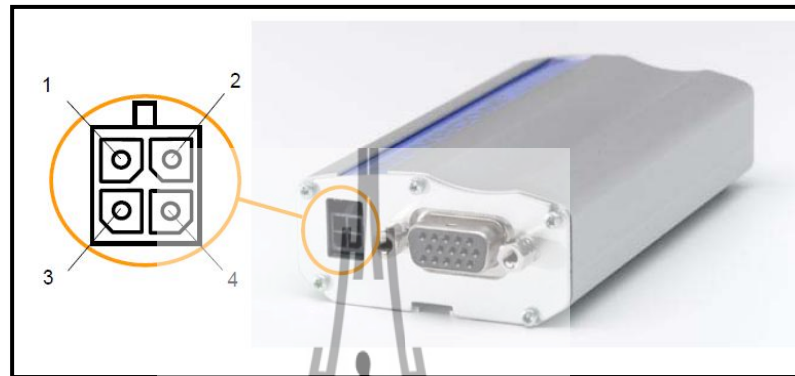
อย่างไรก็ตาม ถ้าผู้ใช้งานต้องการพัฒนา Application ที่สูงขึ้น ก็ยังสามารถประยุกต์ตัดแปลงหรือทำการเชื่อมต่ออุปกรณ์เพิ่มเติมให้กับบอร์ดได้โดยง่าย ทั้งนี้ก็เพราะว่าขาสัญญาณต่างๆ จากโมดูลในส่วนที่ยังไม่ได้ทำการออกแบบวงจรเตรียมไว้ให้ภายในบอร์ด

## 2.7.2 ส่วนประกอบของ GSM Module



รูปที่ 2.29 แสดงจุดเชื่อมต่อบน GSM Module

### 2.7.3 Power Supply Connector



รูปที่ 2.30 Power Supply Connector

ตารางที่ 2.7.1 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: - 2.7 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

#### 2.7.4 การใช้งาน AT Command เพื่อสั่งงานโมดูล Fastrack Supreme 20

โมดูล GSM/GPRS รุ่น Fastrack Supreme 20 ถูกออกแบบให้ทำหน้าที่เหมือน Modem โดยจะใช้การติดต่อสั่งงานและสื่อสารกับโมดูล ผ่านทางพอร์ตสื่อสาร RS232 รองรับ Baud rate ตั้งแต่ 9600 Bit Per Sec โดยใช้ชุดคำสั่งแบบ AT Command ซึ่งจะมีรูปแบบการใช้งานเหมือนกับ Modem มาตรฐานทั่วไป ไปเพียงแต่จะมีการเพิ่มเติม Option และคำสั่งพิเศษอื่นๆเพิ่มเติมขึ้นมาอีก เพื่อให้เหมาะสมและสอดคล้องกับความสามารถในการทำงานของโมดูลได้อย่างครบถ้วน โดยรูปแบบของคำสั่งต่างๆ ที่เป็น AT Command นั้น จะเริ่มต้นคำสั่งด้วยรหัส ASCII ของตัวอักษร 2 ตัวคือ “A” และ “T” ซึ่งจะใช้ตัวอักษรแบบพิมพ์เล็กหรือพิมพ์ใหญ่ก็ได้ มีความหมายเหมือนกัน จากนั้น ก็จะตามด้วยรหัสคำสั่งและ Option ต่างๆ ของคำสั่ง (ถ้ามี) โดยทุกๆคำสั่งจะต้องจบด้วยรหัส Enter หรือ 0DH (13) เสมอ เช่น คำสั่ง รีเซตจะใช้รูปแบบคำสั่งเป็น “ATZ” หรือ “atz” ก็สามารถใช้งานได้ถูกต้องเหมือนกัน โดยรูปแบบคำสั่งทั้งหมดจะแบ่งออกเป็น 4 แบบด้วยกัน คือ

#### ตารางที่ 2.7.2 รูปแบบการใช้งาน 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 Baud rate
สั่งให้ทำงาน	AT+<x>	รูปแบบการใช้คำสั่งแบบนี้ จะใช้สำหรับสั่งงานให้โมดูลปฏิบัติตามคำสั่งที่ต้องการ เช่น การสั่งรีเซต (ATZ)

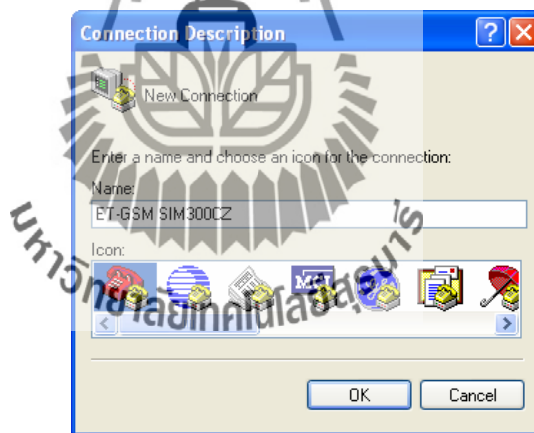
### 2.7.5 การทดสอบการสั่งงานโมดูล

คงได้ทราบแล้วว่าในการสั่งงานโมดูล Fastrack Supreme 20 นั้น จะใช้วิธีการส่งคำสั่งคำสั่งในรูปแบบของ AT Command ผ่านทางพอร์ตสื่อสารอนุกรมไปให้กับโมดูล ซึ่งตามปกติจะต้องเขียนโปรแกรมเพื่อส่งรหัสคำสั่งต่างๆ ไปให้กับโมดูลเอง ขึ้นอยู่กับว่าจะใช้อุปกรณ์ใดเป็นตัวควบคุมการทำงานของโมดูล ซึ่งไม่ได้จำกัดว่าเป็นอุปกรณ์แบบใด อาจจะเป็นคอมพิวเตอร์ PC หรือไมโครคอนโทรลเลอร์ตระกูลใดๆ ก็ได้ ขอให้มีพอร์ตสื่อสารอนุกรม RS232 อยู่ก็สามารถนำมาเชื่อมต่อเพื่อสั่งงานโมดูล Fastrack Supreme 20 ได้แล้ว ส่วนที่ว่าจะเขียนโปรแกรมอย่างไร และจะใช้ภาษาใดในการเขียนนั้น ขึ้นอยู่กับผู้พัฒนาโปรแกรมว่า มีความถนัดอย่างไรและมีพื้นฐานอะไรอยู่บ้าง ซึ่งหลักสำคัญก็คือ ผู้พัฒนาต้องหาคำตอบให้ได้ว่าการจะเขียนโปรแกรมสั่งงานอุปกรณ์ทำการส่งและรับข้อมูล ด้วยพอร์ตสื่อสารอนุกรม RS232 นั้นจะต้องทำอย่างไร ซึ่งจะไม่ขอกล่าวถึงในที่นี้ด้วย สำหรับในการศึกษาเบื้องต้นนั้น ยังไม่จำเป็นต้องใช้วิธีการเขียนโปรแกรมก็ได้ แต่สามารถใช้โปรแกรมสำเร็จรูปจำพวก Serial Terminal ต่างๆ ของคอมพิวเตอร์เป็น ตัวทดสอบการทำงานเพื่อทำความเข้าใจกับรูปแบบคำสั่งและผลของการทำงานต่างๆ ให้เข้าใจเสียก่อน ตัวอย่างเช่น ถ้าต้องการจะสั่งให้โมดูล Fastrack Supreme 20 โทรออกไปยังโทรศัพท์มือถือหมายเลข 0811234567 นั้น ในอันดับแรกจะต้องศึกษารูปแบบการทำงานของคำสั่งให้เข้าใจเสียก่อน จนสามารถเข้าใจแล้วว่าจะต้องใช้คำสั่ง “ATD0811234567;” เพื่อสั่งให้โทรออก จากนั้นจึงค่อยปรับเปลี่ยนไปเป็นการเขียนโปรแกรมในภายหลัง ซึ่งผู้ใช้ก็จะต้องไปศึกษาหาคำตอบต่อไปอีกว่าการที่จะเขียนโปรแกรมเพื่อสั่งให้อุปกรณ์ส่งคำสั่ง “ATD0811234567;” ออกไปทางพอร์ตสื่อสารอนุกรมนั้นต้องทำอะไรบ้าง ซึ่งในที่นี้ จะขอแนะนำให้ใช้โปรแกรม Hyper -Terminal ของ Windows เป็นเครื่องมือในการทดลองในเบื้องต้นไปก่อนโดย Hyper Terminal เป็นโปรแกรม Terminal สำเร็จรูป ซึ่งแถมมาพร้อมกับระบบปฏิบัติการ Windows อยู่แล้ว โดยความสามารถของโปรแกรมตัวนี้จะมีอยู่มากมายหลายส่วน ซึ่งในที่นี้ เราจะใช้ประโยชน์เฉพาะในส่วนของการทำหน้าที่เป็น Serial Terminal ใน Text Mode เท่านั้น โดยหลังจากสั่ง Run โปรแกรมแล้ว ข้อมูลใดๆที่

รับได้จากสัญญาณด้านรับ (RXD) ของพอร์ตสื่อสารอนุกรม ในย่านที่เป็นรหัส ASCII Code (20H..FFH) จะถูกนำมาแปลงเป็นตัวอักษรและแสดงผลที่หน้าจอของโปรแกรมให้เห็นทันที ส่วนรหัสของข้อมูลที่มีค่าต่ำกว่า 20H (00H-1FH) จะไม่ถูกนำมาแสดงผล แต่จะถือว่าเป็นคำสั่ง เช่น เมื่อได้รับรหัส 0DH โปรแกรม Hyper Terminal จะถือว่าเป็นคำสั่งให้เลื่อน Cursor ของการแสดงผลไว้ในตำแหน่งเริ่มต้นของบรรทัดหรือเมื่อได้รับรหัส 0AH ก็จะทำการเลื่อน Cursor ของการแสดงผลให้ขึ้นบรรทัดใหม่แทนดังนี้ เป็นต้นและ ในทางตรงกันข้าม เมื่อเราทำการกดคีย์ใดๆ โปรแกรมก็จะแปลค่าการกดคีย์นั้นให้เป็นรหัส ASCII ของตัวอักษรของตำแหน่งคีย์นั้นๆส่งออกไปยังขา TXD ของพอร์ตสื่อสารอนุกรมโดยอัตโนมัติ โดยการใช้งานโปรแกรม สามารถทำได้ดังนี้



ในขั้นตอนแรกจะปรากฏหน้าต่าง Connection Description ขึ้นมา ให้คลิกเมาส์เลือกรูปแบบของ ICON และกำหนดชื่อของการเชื่อมต่อตามต้องการ แล้วเลือก “OK” ดังตัวอย่างใน



รูปที่ 2.31 หน้าต่าง Connection Description

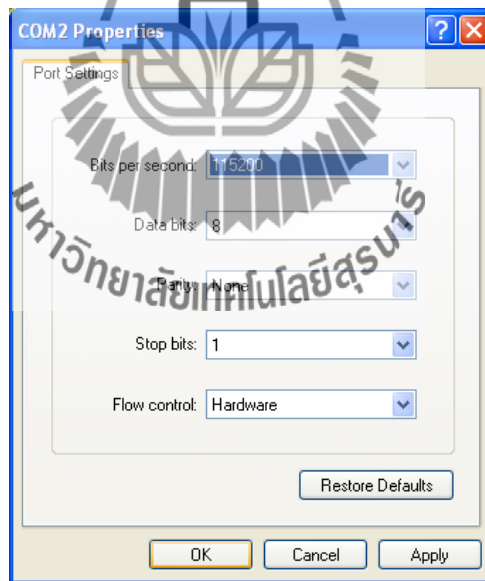
ในขั้นตอนนี้ให้คลิกเมาส์ที่ Connect using แล้วเลื่อนเมาส์ไปยังหมายเลข Comport ที่ใช้ในการเชื่อมต่อกับบอร์ดตามจริง แล้วเลือก “OK” ดังตัวอย่าง



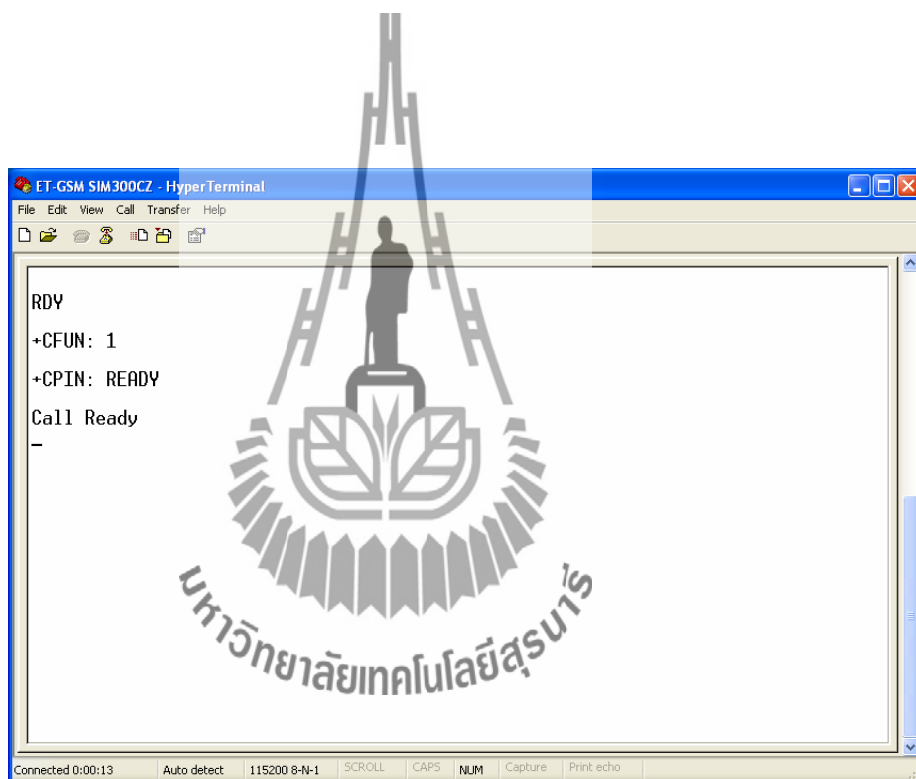


รูปที่ 2.32 หน้าต่าง Connect To (ตั้งค่า Com Port)

ในขั้นตอนนี้ให้เลือกค่า Baud rate ให้ตรงและสอดคล้องกับที่กำหนดให้กับโมดูลไว้หรือในกรณีที่กำหนดค่า Baud rate ของโมดูลเป็น Auto-Baud rate ไว้ก็สามารถกำหนดค่าใดๆที่โมดูลสามารถรองรับได้ระหว่าง 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 2.33 หน้าต่าง Port Settings (ตั้งค่า Baud rate)



รูปที่ 2.34 ลักษณะของหน้าจอ Hyper Terminal เมื่อโมดูลพร้อมทำงาน

### 2.7.6 การกำหนด Flow Control

โมดูล Fastrack Supreme 20 สามารถกำหนด Flow Control หรือ รูปแบบการตรวจสอบความพร้อมในการสื่อสารและรับส่งข้อมูลได้ด้วย ซึ่ง Flow Control จะมีความสำคัญเป็นอย่างมาก เนื่องจากการประมวลผลของอุปกรณ์ต่างๆ จะมีความช้าเร็วที่แตกต่างกัน เมื่อมีการรับส่งข้อมูลที่มีจำนวนข้อมูลมากๆ แบบต่อเนื่องนั้น ถ้าฝ่ายรับไม่พร้อมรับข้อมูลแต่ฝ่ายส่งยังคงส่งข้อมูลออกไป จะ

ทำให้ข้อมูลสูญหายและเกิดความผิดพลาดขึ้นได้ โดย Fastrack Supreme 20 เองรองรับการตรวจสอบความพร้อมหรือ Flow Control ได้ 2 แบบ คือ

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

#### 2.7.7 การ Setup และตรวจสอบค่า Configuration

ตามปกติแล้วการทำงานของโมดูล Fastrack Supreme 20 นั้นจะสามารถกำหนดรูปแบบการทำงานได้มากมายหลายลักษณะ เช่น เงื่อนไขในการติดต่อสื่อสารกับโมดูล ผู้ใช้สามารถเปลี่ยนแปลงค่าต่างๆได้มากมาย ไม่ว่าจะเป็นค่า Baud rate หรือรูปแบบของการ Handshake ต่างๆที่จะใช้ในการสื่อสาร เป็นต้น ดังนั้นจึงจำเป็นต้องมีการกำหนดรูปแบบการทำงานของโมดูลให้ตรงกับความต้องการ ซึ่งตามปกติแล้ว เงื่อนไขต่างๆเหล่านี้จะมีค่าที่แน่นอนอยู่ค่าหนึ่งเสมอหลังการ Reset หรือ Power ON โดยโมดูลจะกำหนดค่าเงื่อนไขต่างๆ ให้กับตัวมันเองในตอนเริ่มต้นการทำงานด้วยค่าที่กำหนดไว้ใน Configuration ที่ถูกบันทึกไว้แล้ว แต่อย่างไรก็ตามผู้ใช้สามารถสั่งเปลี่ยนแปลงแก้ไขค่า Configuration ต่างๆได้เองตามต้องการ ซึ่งวิธีการกำหนดเงื่อนไขการทำงานให้กับโมดูลนั้นสามารถทำได้ 2 แบบ

- **การกำหนดค่าแบบถาวร** จะเป็นการสั่งบันทึกค่าเงื่อนไขการทำงานต่างๆ ของโมดูลตามรูปแบบที่เราที่กำหนดไว้ในหน่วยความจำถาวรภายในตัวโมดูล โดยใช้คำสั่ง “AT&W” ซึ่งหลังจากโมดูลเริ่มต้นทำงานใหม่ หรือหลังการ Reset โมดูลแต่ละครั้งค่าการทำงานต่างๆของโมดูลจะถูกกำหนดเงื่อนไขตามที่เรากำหนดไว้แล้วเสมอ

- การกำหนดค่าแบบชั่วคราว เป็นการใช้คำสั่ง AT Command ต่างๆ เพื่อกำหนดเงื่อนไขการทำงาน ให้กับโมดูล แต่ไม่มีการตั้งบันทึกค่า Configuration ด้วยคำสั่ง “AT&W” ซึ่งการทำงานของโมดูลก็จะปรับเปลี่ยนไปตามการสั่งงานในขณะนั้นๆ แต่เมื่อสั่ง Reset การทำงานของโมดูล หรือ มีการ Power ON ใหม่คุณสมบัติการทำงานของโมดูลจะถูกเปลี่ยนกลับเป็นค่าเดิมอีก โดยเราสามารถ ใช้คำสั่ง AT Command ในการตั้งตรวจสอบ และบันทึกค่า Configuration ต่างๆ ให้กับโมดูล Fastrack Supreme 20 ได้ดังนี้
- ใช้คำสั่ง “AT&V” เพื่อสั่งให้โมดูลแสดงค่า Configuration ปัจจุบันให้ทราบ
- ใช้คำสั่ง “AT&F” เพื่อสั่งกำหนดค่า Configuration ทั้งหมดให้กลับเป็นค่ามาตรฐาน
- ใช้คำสั่ง “AT&W” เพื่อตั้งบันทึกค่า Configuration ด้วยค่าที่เรากำหนดไว้ในขณะนั้นๆ

#### ค่า Configuration ที่แนะนำ

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

#### 2.7.8 ตัวอย่างการรับข้อความ SMS

ตามปกติแล้วโมดูล Fastrack Supreme 20 จะสามารถกำหนดโหมดการทำงานของข้อความ หรือ SMS ได้ 2 โหมด คือ PDU Mode และ Text Mode โดย PDU Mode การรับและแสดงผลการทำงานของคำสั่งจะเป็นรูปแบบของรหัสตัวเลขแบบ Binary Code ส่วน Text Mode การรับและแสดงผลการทำงานของคำสั่งจะเป็นข้อความ ซึ่งจะง่ายต่อการแปลความหมายและทำความเข้าใจมากกว่า PDU Mode ซึ่งในการทดสอบ จะขอแสดงให้เห็นด้วย Text Mode

- ใช้คำสั่ง “AT+CMGF=1” เพื่อกำหนดรูปแบบของข้อความ เป็น Text Mode ซึ่งเมื่อมีการส่งข้อความ SMS มาถึงโมดูลจะมีข้อความแจ้งให้ทราบเช่น +CMTI: “SM”, 3 ซึ่งหมายความว่า มีข้อความส่งเข้าและเก็บไว้ในหน่วยความจำลำดับที่ 3
- ใช้คำสั่ง “AT+CMGR” เพื่อสั่งอ่านข้อความ เช่นถ้าต้องการอ่านข้อความลำดับที่ 3 ก็ให้ใช้คำสั่งเป็น “AT+CMGR=3”

- ใช้คำสั่ง “AT+CMGL” เพื่อส่งแสดงข้อความทั้งหมดที่เก็บไว้ในหน่วยความจำ โดยสามารถเลือกประเภทของข้อความได้ เช่น ข้อความใหม่ ข้อความทั้งหมด
- ใช้คำสั่ง “AT+CMGD” เพื่อส่งลบข้อความออกจากหน่วยความจำ เช่น ถ้าต้องการส่งลบข้อความลำดับที่ 3 ก็ให้ใช้คำสั่งเป็น “AT+CMGD=3”

```
+CMTI: "SM",3
AT+CMGR=3<Ent>
+CMGR: "REC UNREAD","+66812505187", "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
```

### 2.7.9 ตัวอย่างการส่งข้อความ SMS

ในการส่งข้อความ SMS นั้นจะใช้คำสั่ง “AT+CMGS” ในการสั่งงาน โดยในกรณีที่ใช้ Text Mode นั้นให้ใช้รูปแบบคำสั่งเป็น “AT+CMGS=” +เบอร์ผู้รับ” โดยเบอร์ของผู้รับต้องใส่รหัสประเทศ นำหน้าแทนศูนย์ด้วยเสมอ ซึ่งในกรณีที่เป็นประเทศไทยจะใช้รหัสประเทศเป็น “66” ดังนั้นถ้าต้องการส่งข้อความ SMS ให้กับเบอร์ที่ใช้งานอยู่ในประเทศไทย เช่น 081-1234567 ก็จะต้องกำหนดหมายเลขของเบอร์ผู้รับปลายทางเป็น 6681-1234567 แทน ซึ่งในกรณีนี้จะได้รับรหัสเบอร์ผู้รับข้อความ เป็น “+66811234567” ซึ่งเมื่อโมดูล Fastrack Supreme 20 ได้รับคำสั่ง “AT+CMGS” เรียบร้อยแล้ว มันจะตอบรับด้วยการส่งเครื่องหมาย “>” กลับมาบอก ซึ่งหลังจากนี้เป็นต้นไปผู้ใช้ก็สามารถจะทำ

การพิมพ์ข้อความต่างๆ ที่ต้องการจะส่งให้กับโมดูลได้ทันที โดยให้ ปิดท้ายข้อความด้วยรหัส  
“Ctrl+Z” ตามด้วย “Enter” เช่น ถ้าต้องการส่งข้อความ SMS ให้กับหมายเลข 0811234567 ด้วย  
ข้อความ “Hello Test SMS” จะเป็นดังนี้

```
AT+CMGS="+66811234567"<Ent>
```

```
> Hello Test SMS<Ctrl+Z><Ent>
```

```
+CMGS: 6
```

```
OK
```



### บทที่ 3

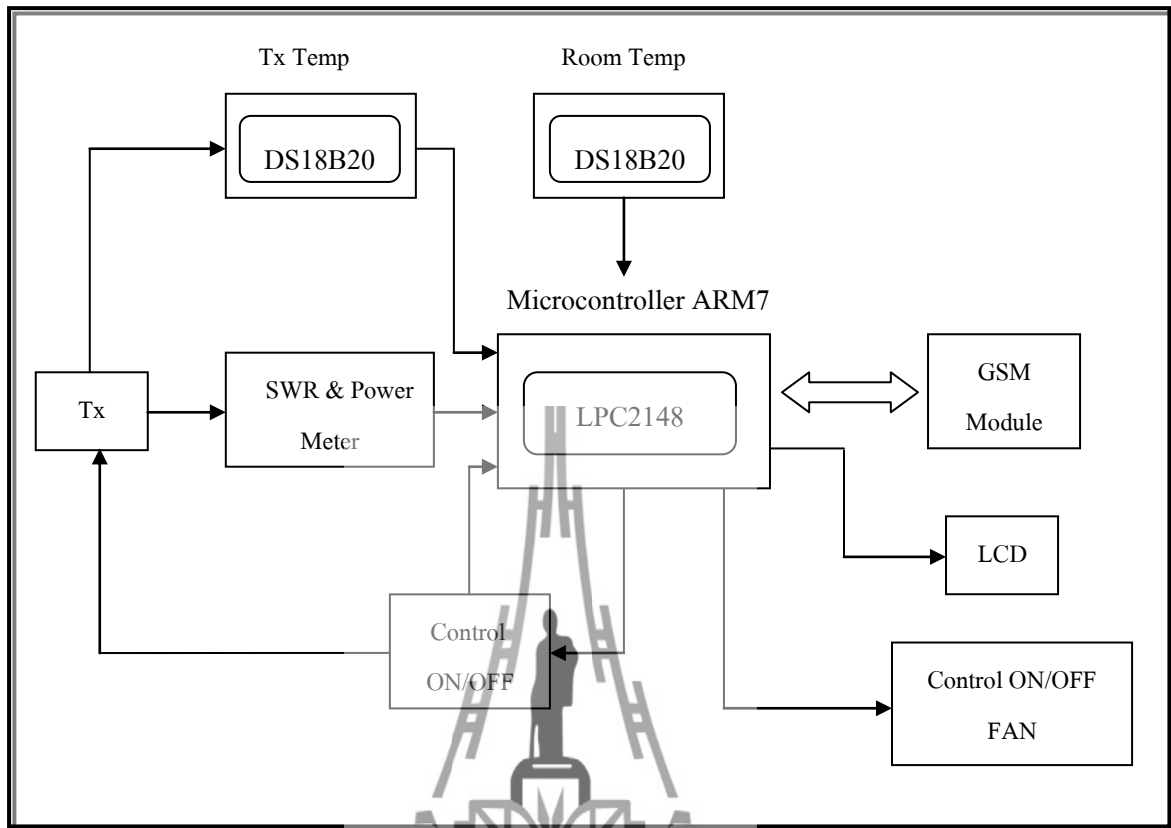
#### การออกแบบโครงการ

โครงการเรื่องระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง ได้มีการออกแบบมาเพื่อตรวจสอบและรายงานสถานะของเครื่องส่งวิทยุกระจายเสียง ซึ่งโครงการนี้ประกอบด้วย เซนเซอร์วัดอุณหภูมิ, SWR & Power Meter, วงจรควบคุมการเปิดปิดพัดลมระบายความร้อน, วงจรควบคุมการเปิดปิดเครื่องส่ง และไมโครคอนโทรลเลอร์ ARM7 โดยไมโครคอนโทรลเลอร์จะรับค่าอุณหภูมิ ที่ได้จากเซนเซอร์วัดอุณหภูมิและรับค่า Forward & Reflect ที่ได้จากเครื่องวัด และเปรียบเทียบข้อมูลกับค่าที่บันทึกไว้ หลังจากนั้น ไมโครคอนโทรลเลอร์จะทำการประมวลผลและรวบรวมข้อมูลเพื่อรายงานผลทั้งหมด นอกจากนี้ ไมโครคอนโทรลเลอร์สามารถตรวจเช็คเงื่อนไขของอุณหภูมิเพื่อควบคุมพัดลมระบายอากาศ และควบคุมการเปิดปิดเครื่องส่งวิทยุกระจายเสียงจากระบบ SMS โดยโครงการนี้จะประกอบไปด้วยส่วนของการทำงานต่างๆ ดังนี้

1. ส่วนการอ่านค่าอุณหภูมิ
2. ส่วนการอ่านค่า Forward & Reflect
3. ส่วนการควบคุมการเปิดปิดเครื่องส่งวิทยุกระจายเสียง
4. ส่วนการควบคุมการเปิดปิดพัดลมระบายความร้อน
5. ส่วนการแสดงผล

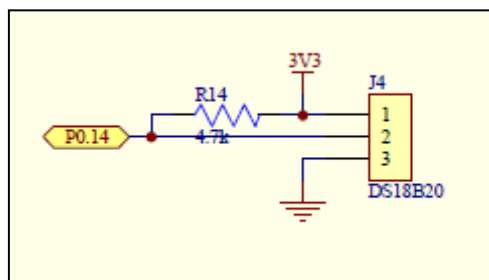
ซึ่งในแต่ละส่วนจะมีไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงานหลักเพื่อให้สามารถทำงานได้ตามต้องการ ดังนั้นเพื่อให้การทำงานในแต่ละส่วนเป็นไปอย่างถูกต้องและสมบูรณ์ จึงทำการออกแบบการทำงานในแต่ละส่วนดังนี้

### 3.1 การออกแบบฮาร์ดแวร์



รูปที่ 3.1 แสดงแผนผังของโครงการ

#### 3.1.1 ส่วนการอ่านค่าอุณหภูมิ



รูปที่ 3.2 แสดงการจัดแหล่งจ่ายไฟภายนอกของ DS18B20



ส่วนของการอ่านค่าอุณหภูมินั้น เราสามารถอ่านค่าอุณหภูมิจาก IC วัดอุณหภูมิแบบดิจิตอล ซึ่งจะต้องทำการเชื่อมต่อสายจากขา DQ เข้ากับตัวต้านทาน Pull Up เพื่อให้อ่านค่าสัญญาณได้โดยถูกต้องและแม่นยำ

การอ่านค่าอุณหภูมินั้นไมโครคอนโทรลเลอร์จะรับค่าอุณหภูมิจาก IC วัดอุณหภูมิ (DS18B20) โดยเชื่อมต่อขา DQ (2) เข้ากับ พอร์ตอนุกรมของบอร์ดไมโครคอนโทรลเลอร์ ( P0.14) และจัมป์ไฟฟ้เพียงผ่านตัวต้านทาน Pull Up (R14) เข้าระหว่างขา DQ (2) และ พอร์ตอนุกรม (P0.14) ดังรูปที่ 3.1



รูปที่ 3.3 ผลการอ่านค่าอุณหภูมิที่ได้นจจอ LCD

### 3.1.2 ส่วนการอ่านค่า Forward & Reflect

ส่วนของการอ่านค่า Forward และ Reflect นั้น จะเป็นการนำค่าการส่งผ่าน (Forward) และค่าที่สะท้อนกลับ (Reflect) ที่วัดได้จากเครื่อง SWR & Power Meter มาคำนวณหาอัตราส่วน (Ratio) ของเครื่องวิทยุกระจายเสียง โดยค่าที่วัดได้จากเครื่อง SWR & Power Meter นั้นจะทำการวัดค่าแรงดันที่เข็มแสดงสถานะของเครื่องวัด ดังนั้นการอ่านค่า Forward และ Reflect เพื่อให้ได้ค่า Power จริงนั้นจะต้องนำค่าแรงดันที่อ่านได้มาคำนวณเพื่อที่จะนำค่าที่ได้มาคำนวณเป็นค่า Ratio ต่อไป

การทดสอบนั้นจะทำการทดสอบโดยการวัดค่า Power ที่ค่าต่างๆ เพื่อให้ได้ค่าแรงดันที่เข็มแสดงสถานะออกมา โดยค่าที่วัดได้จะแสดงดังตาราง

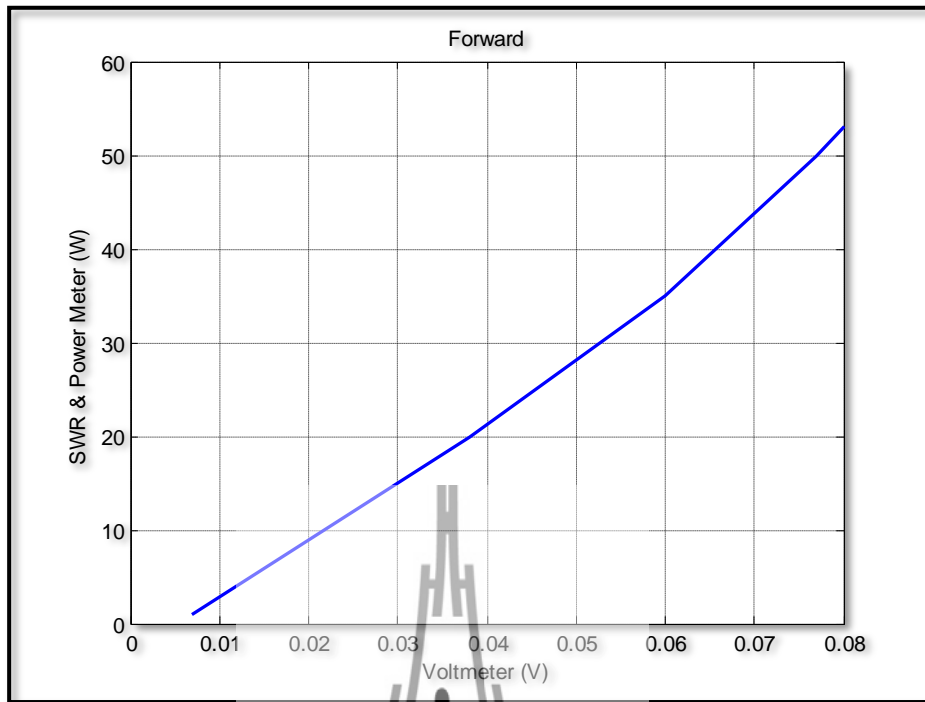
ตารางที่ 3.1 แสดงผลการทดสอบวัดค่า Forward (F)

Voltmeter (mV)	SWR & Power Meter (W)
80	53
80	53
77	50
60	35
38	20
7	1

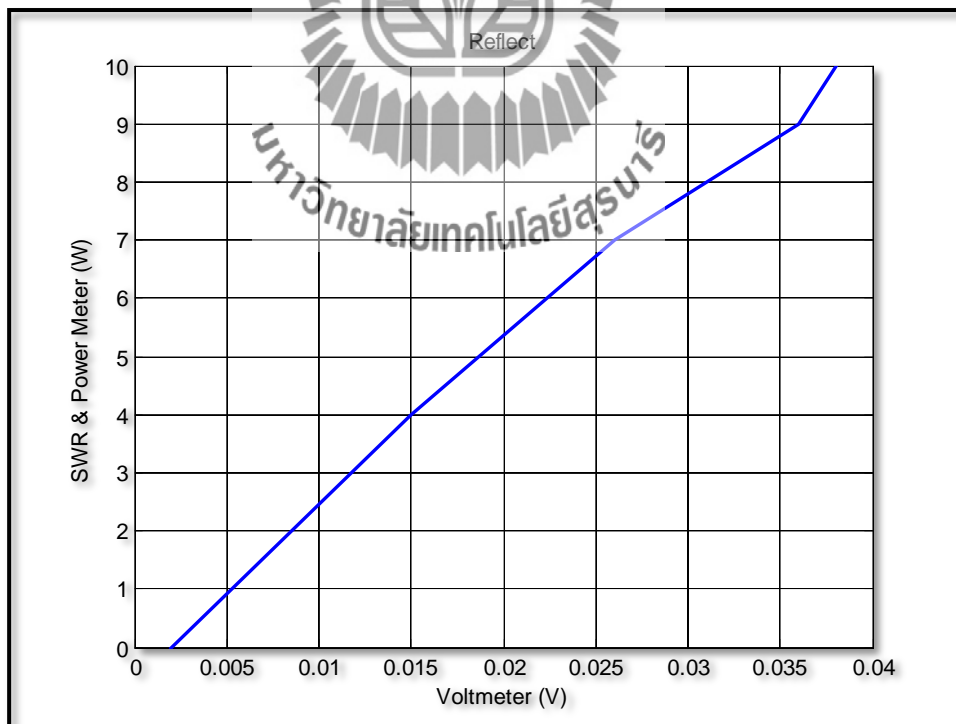
ตารางที่ 3.2 แสดงผลการทดสอบวัดค่า Reflect (R)

Voltmeter (mV)	SWR & Power Meter (W)
38	10
38	10
36	9
26	7
15	4
2	0

จากค่าในตารางสามารถนำมาแสดงได้ดังกราฟ เพื่อแสดงความสัมพันธ์ระหว่างค่าแรงดันและค่า Power



รูปที่ 3.4 แสดงการพล็อตกราฟของค่า Forward



รูปที่ 3.5 แสดงการพล็อตกราฟของค่า Reflect

เมื่อพิจารณาจากลักษณะกราฟทั้งสองรูปแสดงให้เห็นว่า กราฟที่ได้เป็นกราฟเส้นตรง ดังนั้น เราจึงต้องหาสมการเส้นตรง โดยเราจะหาสมการเส้นตรงของค่า Forward ก่อน ซึ่งหาได้จาก

$$y = mx + c$$

โดยให้  $y$  คือ ค่าที่วัดได้จาก SWR & Power Meter

$x$  คือ ค่าที่วัดได้จาก โวลต์มิเตอร์

$m$  คือ ความชันของกราฟ ซึ่งหาได้จาก  $\frac{y_{max} - y_{min}}{x_{max} - x_{min}}$  จะได้

จากตารางได้  $m = \frac{53-20}{(80-38) \times 10^{-3}} = 785.714$

และจากสมการ  $y - y_1 = m(x - x_1)$

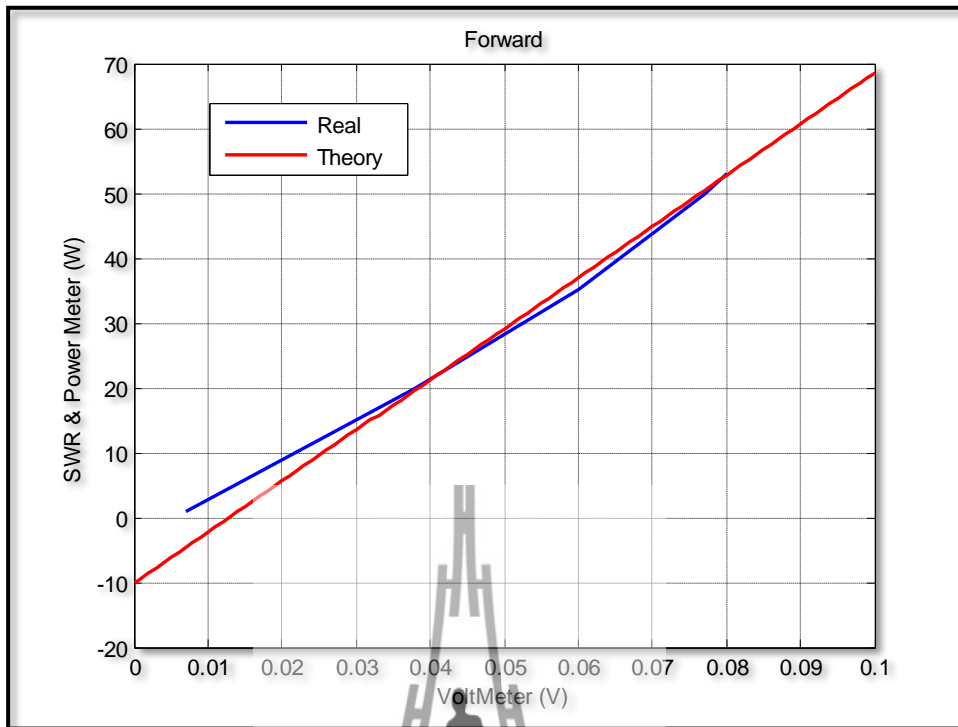
สมมติที่  $x = 60 \times 10^{-3}$ ,  $y = 35$  และ  $m = 785.714$  จะได้สมการดังนี้

$$y - 35 = 785.714(x - (60 \times 10^{-3}))$$

ดังนั้นสมการเส้นตรงของค่า Forward คือ

$$y = 785.714x + 10.14$$

ซึ่งจากสมการนี้ เรานำสมการมาเปรียบเทียบกับค่าที่วัดได้ ซึ่งจะแสดงดังรูป



รูปที่ 3.6 แสดงการเปรียบเทียบระหว่างค่าที่ได้จากการทดลองกับค่าที่ได้จากสมการของ Forward

ต่อมาเราจะหาสมการเส้นตรงของค่า Reflect ซึ่งวิธีการจะทำเหมือนการหาสมการเส้นตรงของค่า Forward ดังนี้



โดยให้  $y$  คือ ค่าที่วัดได้จาก SWR & Power Meter

$x$  คือ ค่าที่วัดได้จาก โวลต์มิเตอร์

$m$  คือ ความชันของกราฟ ซึ่งหาได้จาก  $\frac{y_{max} - y_{min}}{x_{max} - x_{min}}$  จะได้

จากตารางได้ 
$$m = \frac{10^{-4}}{(38-15) \times 10^{-3}} = 260.869$$

และจากสมการ 
$$y - y_1 = m(x - x_1)$$

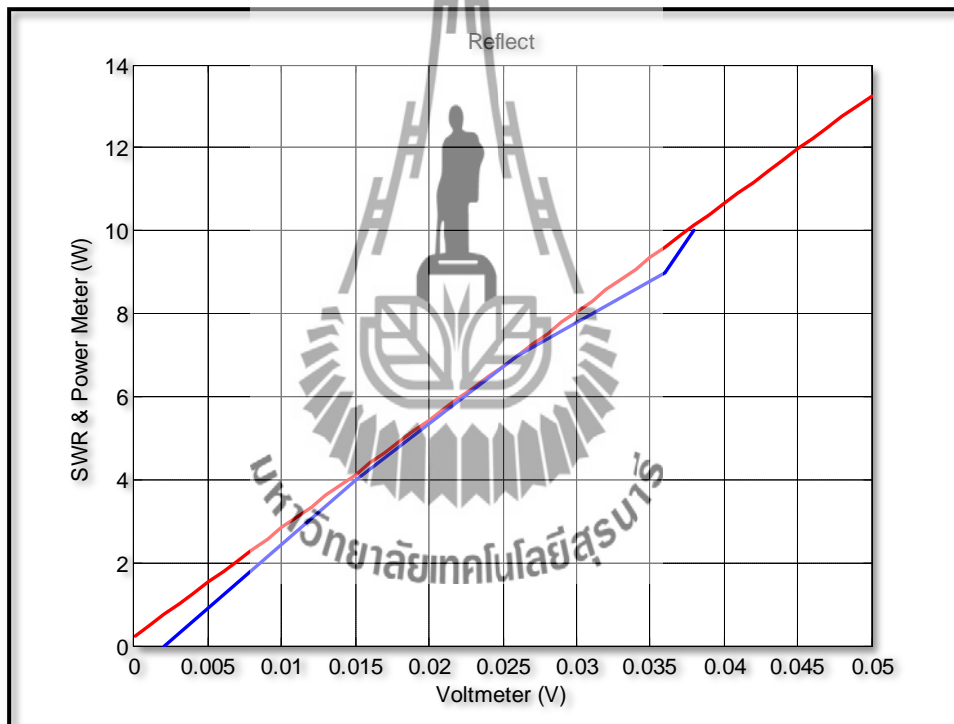
สมมติที่  $x = 26 \times 10^{-3}$ ,  $y = 7$  และ  $m = 260.869$  จะได้สมการดังนี้

$$y - 7 = 260.869(x - (26 \times 10^{-3}))$$

ดังนั้นสมการเส้นตรงของค่า Reflect คือ

$$y = 260.869x + 0.217$$

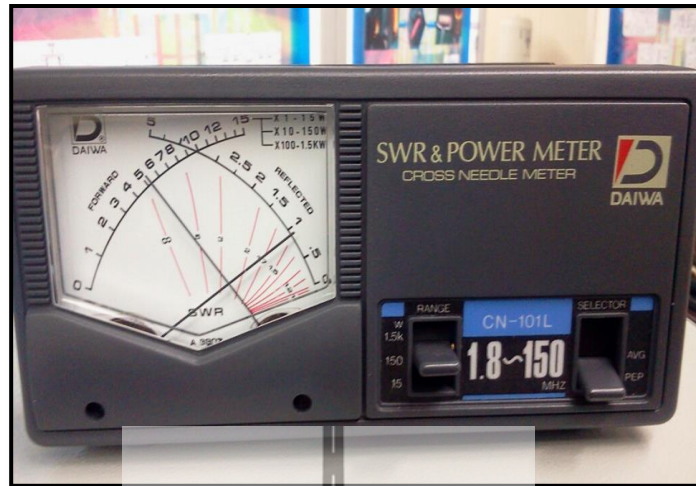
ซึ่งจากสมการนี้ จะได้กราฟเป็นกราฟเส้นตรงตามรูปด้านล่าง



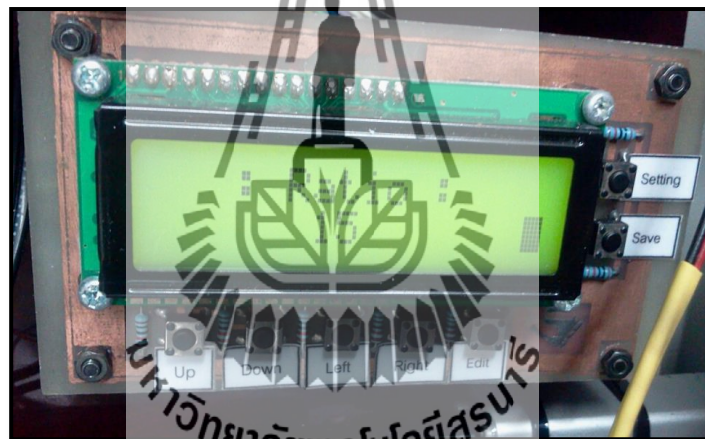
รูปที่ 3.7 แสดงการเปรียบเทียบระหว่างค่าที่ได้จากการทดลองกับค่าที่ได้จากสมการของ Reflect

เมื่อเราได้อ่านค่า Forward & Reflect ที่อยู่ในหน่วยของวัตต์ (Watt) แล้ว ต่อมาจะทำการนำค่าที่ได้มาใช้คำนวณเป็นค่า Ratio จากสมการ

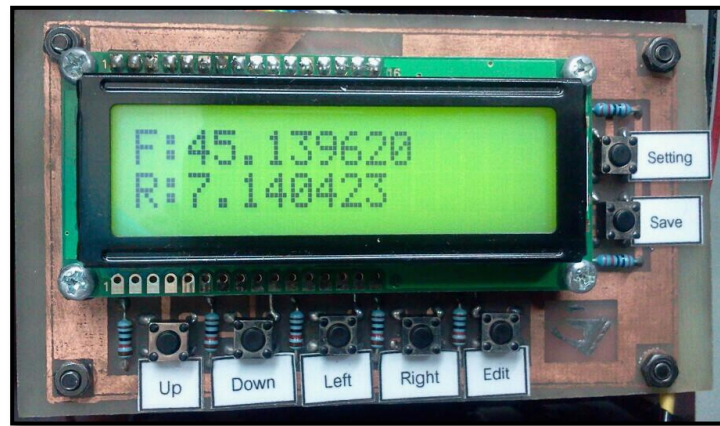
$$Ratio = (R/F) \times 100 \quad (\%)$$



รูปที่ 3.8 แสดงค่าที่ SWR & Power Meter วัดได้

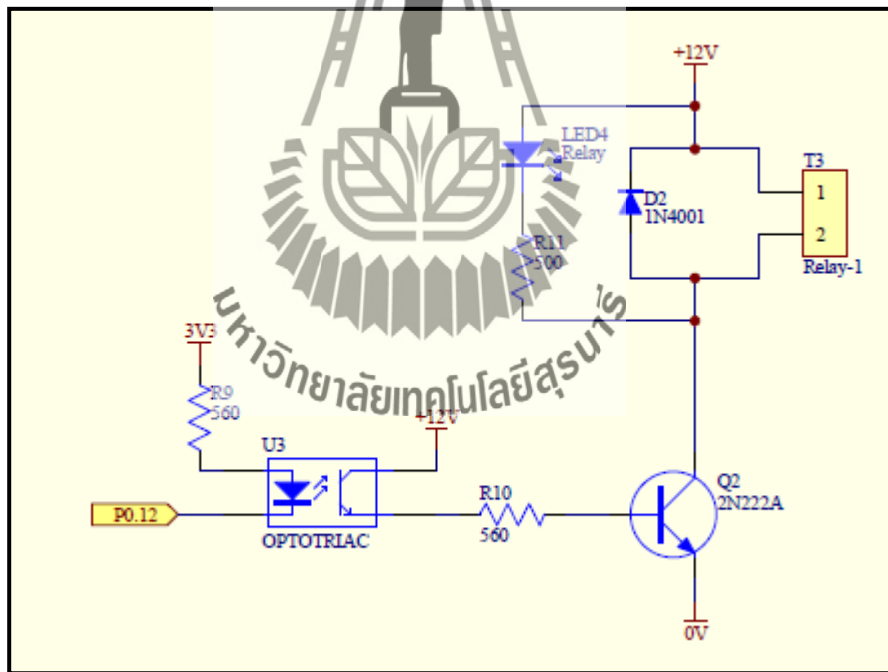


รูปที่ 3.9 แสดงการนำค่าจาก SWR & Power Meter มาคำนวณเป็นค่าRatio



รูปที่ 3.10 แสดงค่า Forward และ Reflect ในหน่วยวัตต์

### 3.1.3 ส่วนการควบคุมการเปิดปิดพัลลวมระบายความร้อน



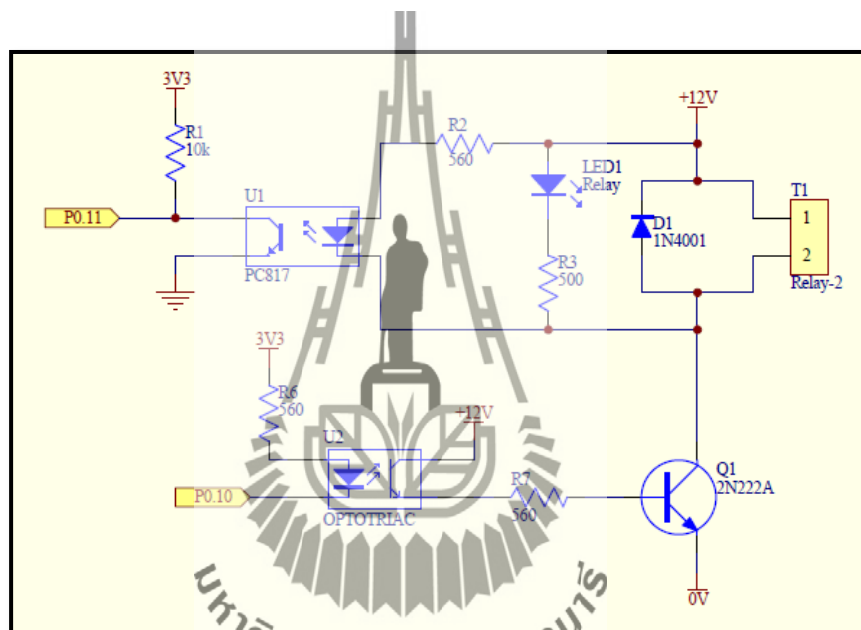
รูปที่ 3.11 แสดงวงจรควบคุมการเปิด - ปิดพัลลวมระบายความร้อน

ส่วนของการควบคุมการเปิด-ปิดพัลลวมระบายความร้อนนั้น จะเป็นส่วนของการควบคุมการเปิด-ปิดพัลลวมเมื่อค่าอุณหภูมิของบอร์ดไมโครคอนโทรลเลอร์มีค่าเกินกว่าที่กำหนดไว้



เนื่องจากตัว Optotriac ต้องเปลี่ยนแรงดันจาก 3 โวลต์ เป็น 12 โวลต์ เพื่อทำการไบแอสแรงดัน 12 โวลต์ ให้กับตัวทรานซิสเตอร์ทำงาน ดังนั้นจึงต้องมีการต่อขาของตัว Optotriac เข้ากับบอร์ดไมโครคอนโทรลเลอร์ ( P0.12) เพื่อป้อนลอจิกให้ตัว Optotriac ทำงาน โดยถ้าป้อนลอจิก “0” ตัว Optotriac “ทำงาน” และถ้าป้อนลอจิก “1” ตัว Optotriac “ไม่ทำงาน” เมื่อป้อนลอจิกให้ตัว Optotriac ทำงานแล้ว แรงดัน 12 โวลต์ จะไบแอสให้ทรานซิสเตอร์ทำงานและทรานซิสเตอร์จะขับกระแสให้รีเลย์ของพัดลมทำงานอีกที

### 3.1.4 ส่วนการควบคุมการเปิด-ปิดเครื่องส่ง

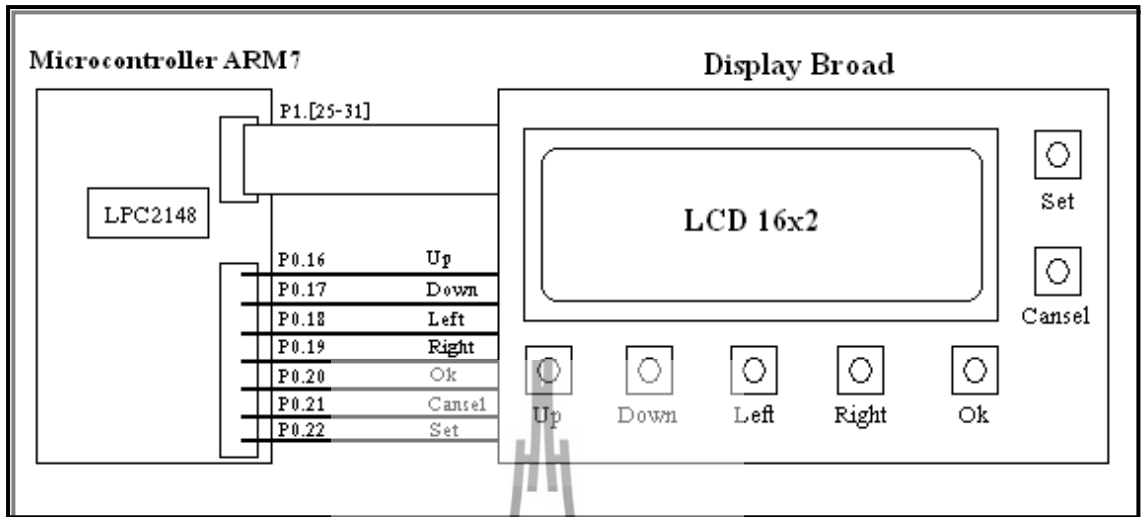


รูปที่ 3.12 แสดงวงจรควบคุมการเปิด-ปิดเครื่องส่ง

ส่วนของการควบคุมการเปิด-ปิดเครื่องส่งนั้น จะเป็นส่วนที่ใช้ควบคุมการเปิด-ปิดของเครื่องส่ง โดยจะใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการสั่งเปิด-ปิด

เนื่องจากเครื่องส่งนั้นจะต้องมีการเปิด-ปิด เราจึงต้องมีตัว Optotriac (U1) เพื่อแปลงแรงดันจากวงจรขั้วรีเลย์ที่มีแรงดัน 12 โวลต์ เป็นแรงดัน 3 โวลต์ เพื่อป้อนให้กับบอร์ดไมโครคอนโทรลเลอร์ และเป็นการบอกสถานะของเครื่องได้ว่าที่เวลานี้เครื่องเปิดหรือปิดอยู่ โดยวงจรขั้วรีเลย์จะเป็นเหมือนกับวงจรขั้วรีเลย์พัดลมระบายความร้อน ดังนั้นถ้าบอร์ดไมโครคอนโทรลเลอร์ ( P0.10) ส่งลอจิก “ 0 ” มาจะทำให้ตัวขั้วรีเลย์ทำงาน เมื่อตัวขั้วรีเลย์ทำงานตัว Optotriac (U1) ก็จะมีแรงดัน 3 โวลต์ตกคร่อมตัวต้านทาน 10kΩ ทำให้ลอจิกที่ส่งให้บอร์ดไมโครคอนโทรลเลอร์ มีค่าเป็น “0” หมายความว่า Relay กำลัง ON อยู่

### 3.1.5 ส่วนการแสดงผล



รูปที่ 3.13 ส่วนการแสดงผล

Display Broad เป็นบอร์ดที่ประกอบด้วย สวิตช์ และจอแสดงผล LCD เพื่อใช้ในการตั้งค่าที่หน้าจอแสดงผลและแสดงค่าค่าต่างๆที่บอร์ดไมโครคอนโทรลเลอร์อ่านได้ ซึ่งจำเป็นต้องจ่ายพลังงานไฟฟ้าให้กับ Display Broad โดยใช้แรงดัน 3.3V และการเชื่อมต่อ Display Broad กับบอร์ดไมโครคอนโทรลเลอร์ นั้นสามารถที่ได้ดังรูปที่ 3.6 ดังนี้

ตารางที่ 3.3 แสดงการเชื่อมต่อระหว่าง Port กับ Display broad

ไมโครคอนโทรลเลอร์ ARM7	Display Broad
P0.16	สวิตช์ Up
P0.17	สวิตช์ Down
P0.18	สวิตช์ Left
P0.19	สวิตช์ Right
P0.20	สวิตช์ Ok
P0.21	สวิตช์ Cancel
P0.22	สวิตช์ Setting

การเชื่อมต่อ LCD กับไมโครคอนโทรลเลอร์ สามารถทำได้ดังนี้

P1.25	เชื่อมต่อ	RS
P1.26	เชื่อมต่อ	RW
P1.27	เชื่อมต่อ	EN
P1.28	เชื่อมต่อ	D4
P1.29	เชื่อมต่อ	D5
P1.30	เชื่อมต่อ	D6
P1.31	เชื่อมต่อ	D7

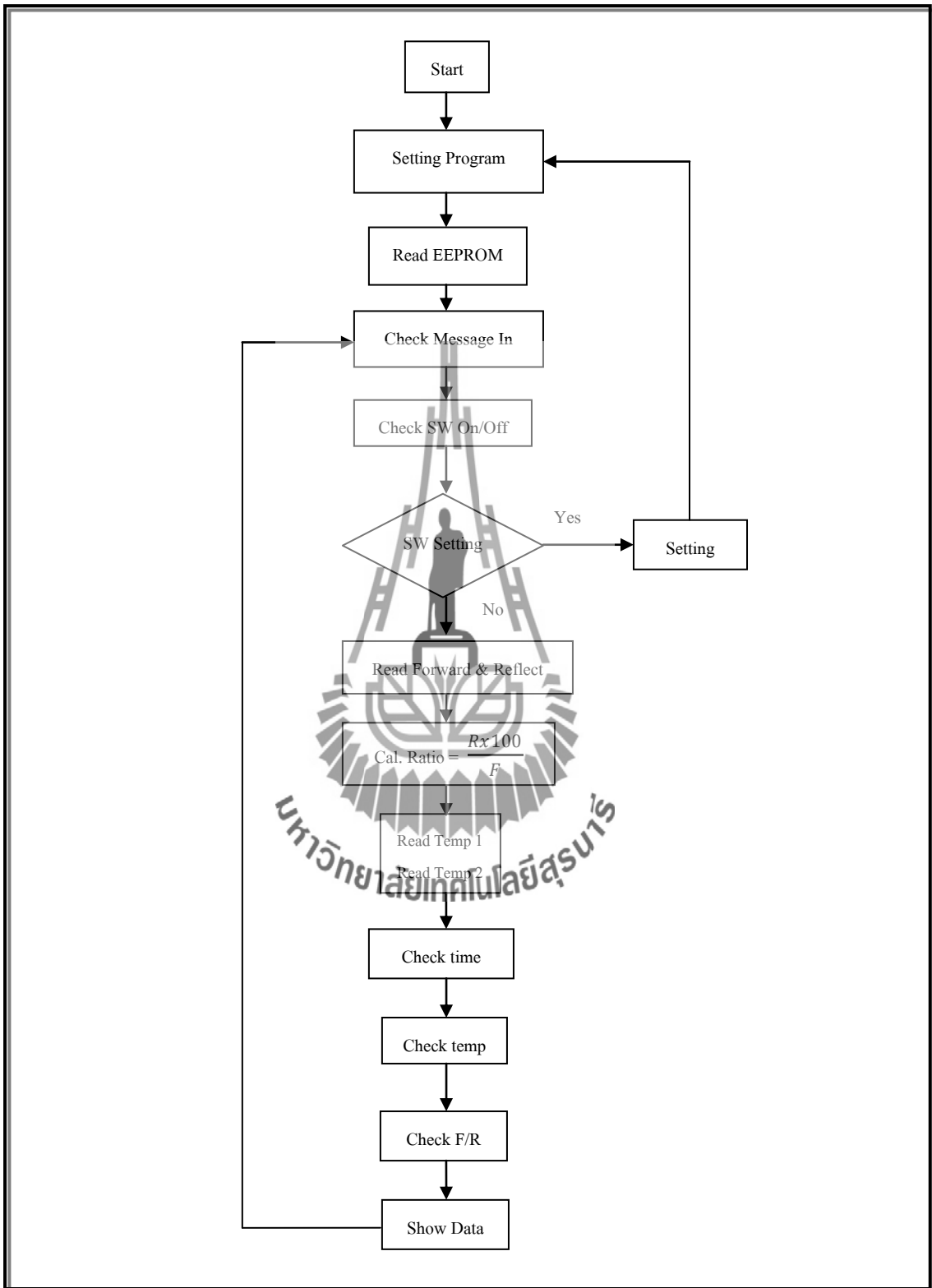
### 3.2 การออกแบบซอฟต์แวร์

ในโครงการนี้ จะมีไมโครคอนโทรลเลอร์ ARM7 เป็นตัวควบคุมการทำงานหลัก โดยต้องมีซอฟต์แวร์ที่ใช้ในการเขียนคำสั่งควบคุม ดังนั้นผู้จัดทำได้เลือกใช้ภาษาซีโดยใช้โปรแกรม Keil uVision ในการเขียนคำสั่งควบคุม เนื่องจากภาษาซีเป็นภาษาที่มีโครงสร้างง่ายต่อการทำความเข้าใจ และสามารถปรับปรุงพัฒนาต่อได้ง่ายนอกจากนั้นภาษาซียังเป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์ (ไมโครคอนโทรลเลอร์) มีความยืดหยุ่นในการใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย

### 3.3 การทำงานของโปรแกรม

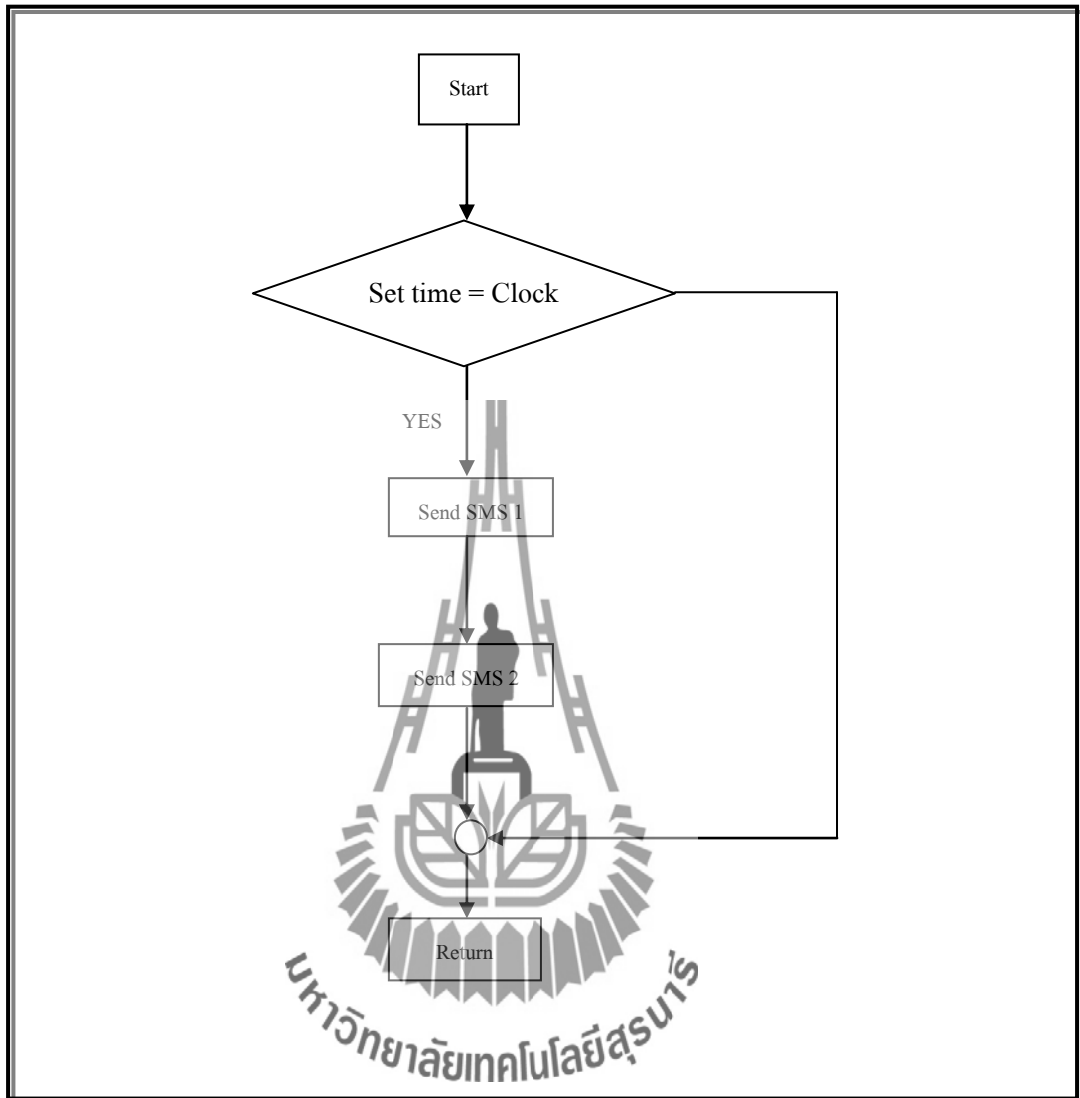
เนื่องจากการทำงานของเครื่องตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงนี้ ( FM Broadcast Status Checking System) สามารถส่งค่า Ratio, อุณหภูมิ, ความถี่, สถานะเปิดปิดของเครื่องส่ง และเวลาให้กับผู้ดูแลทั้งสองหมายเลขได้โดยผู้ดูแลจะต้องทำการตั้งค่าพารามิเตอร์ต่างๆ ทางหน้าจอแสดงผล หลังจากนั้น โปรแกรมจะอ่านค่าและทำการตรวจสอบกับค่าพารามิเตอร์ต่างๆ ที่ตั้งไว้เรื่อยๆ โดยในกรณีปกติ ระบบจะทำการส่งค่าสถานะของเครื่องส่งให้กับผู้ดูแลตามเวลาที่กำหนดไว้ หากมีกรณีที่ผิดปกติ เช่น ค่าอุณหภูมิ หรือ ค่า Ratio เกินกว่าค่าที่ตั้งไว้ ระบบก็จะทำการส่งข้อความแจ้งเตือนให้ผู้ดูแลได้ทราบ นอกจากนี้ เมื่ออุณหภูมิเกินกว่าที่กำหนด ระบบยังสามารถส่งเปิด - ปิดพัดลมระบายความร้อนได้ และถ้าหากผู้ดูแลต้องการทราบสถานะของเครื่องส่งโดยทันทีก็สามารถส่งข้อความ SMS มาที่เครื่องได้เช่นกัน

### 3.3.1 Main Function



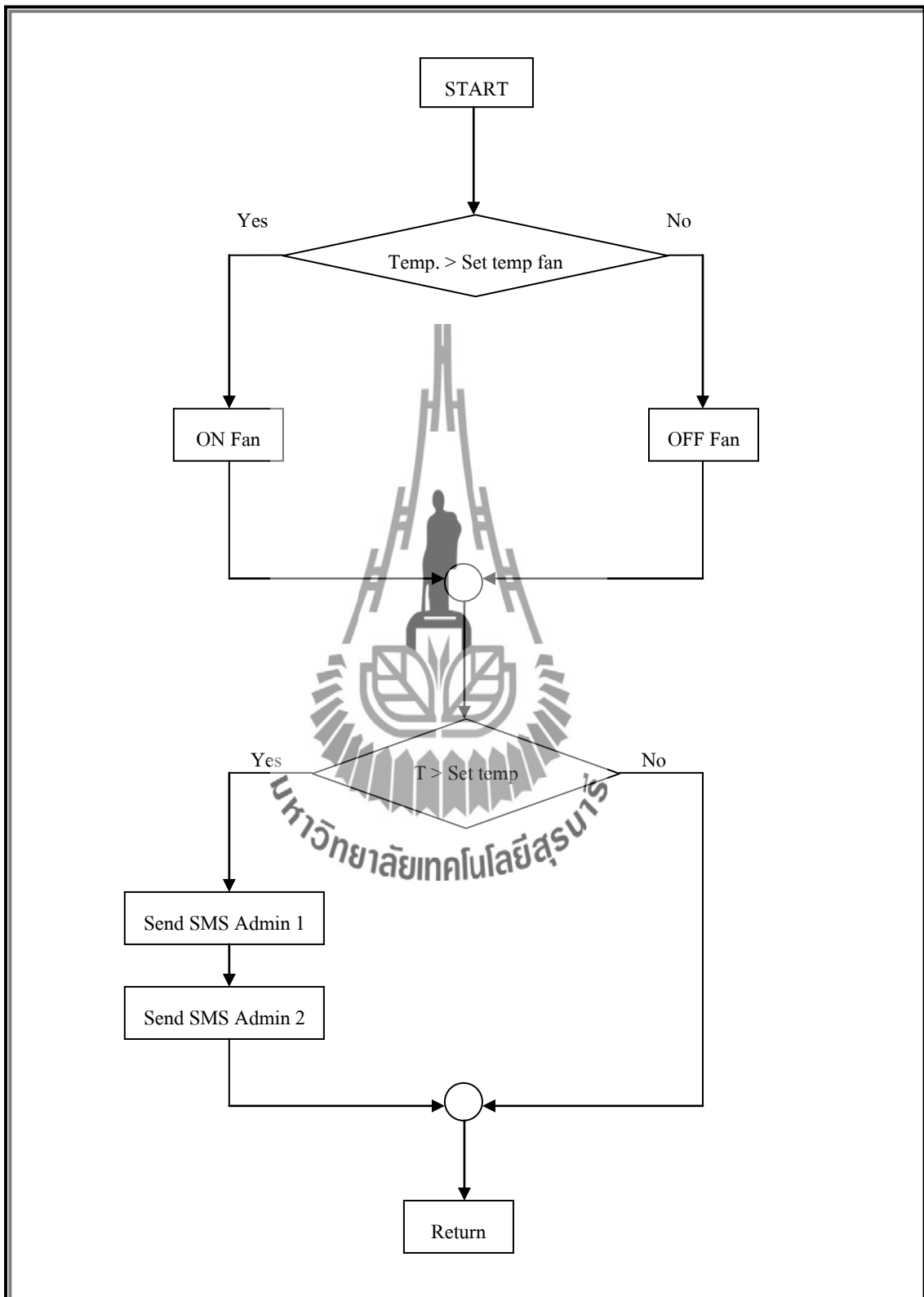
รูปที่ 3.14 Flow Chart ของฟังก์ชัน Main

### 3.3.2 Check Time Function



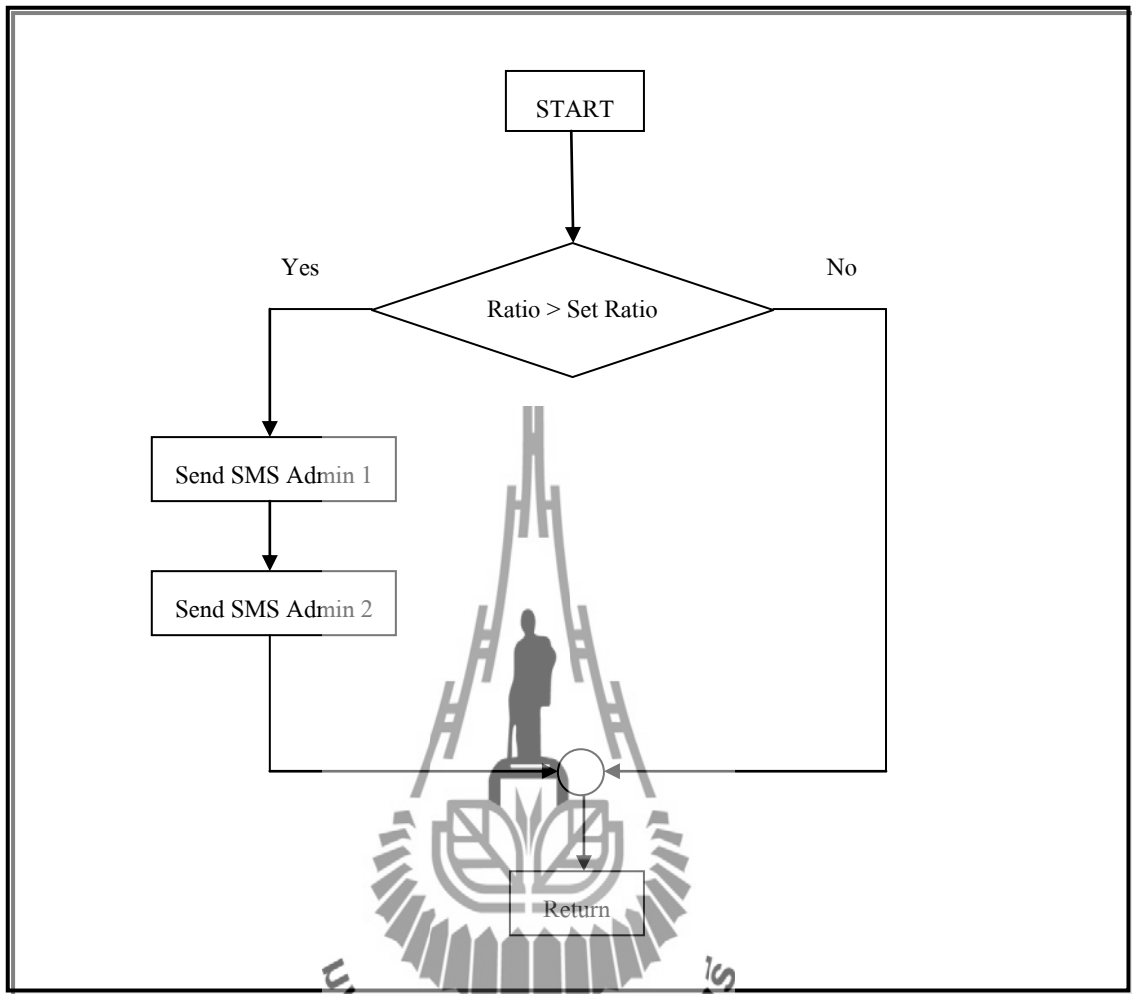
รูปที่ 3.15 Flow Chart ของฟังก์ชัน Check Time

### 3.3.3 Check Temperature Function



รูปที่ 3.16 Flow Chart ของฟังก์ชัน Check Temperature

### 3.3.4 Check Forward & Reflect Function



รูปที่ 3.17 Flow Chart สำหรับฟังก์ชัน Check Forward & Reflect

### 3.4 อธิบายการทำงานของโครงการ

ในส่วนของการประมวลผลหลักขั้นแรกจะทำการตรวจสอบก่อนว่ามีการตั้งค่าใหม่มาหรือไม่ ถ้ามีข้อมูลที่ตั้งค่าใหม่จะถูกบันทึกลงหน่วยความจำ (EEPROM) ถ้าไม่มีโปรแกรมก็จะนำค่าที่ตั้งไว้ก่อนหน้านี้จากหน่วยความจำ (EEPROM) มาใช้อีกครั้ง ต่อมาจะทำการตรวจสอบว่ามีข้อความจากผู้ใช้หรือไม่ เช่น ข้อความขอให้ส่งสถานะไปให้ , ข้อความสั่งเปิดเครื่องส่ง หรือ ข้อความคำสั่งให้ปิดเครื่องส่ง ต่อมาโปรแกรมจะตรวจสอบว่าสวิทช์ของเครื่องส่งนั้นเปิด หรือปิดอยู่ จากนั้นโปรแกรมจะทำการถามว่า มีการกดสวิทช์ตั้งค่าใหม่หรือไม่ ถ้ามี โปรแกรมจะกลับไปเริ่มใหม่อีกครั้ง ถ้าไม่มี โปรแกรมนี้จะทำการอ่านค่า Forward กับ Reflect จากเครื่องวัด SWR & Power Meter และนำค่าที่อ่านได้มาคำนวณเป็นค่า Ratio โดยจะเอาค่า Reflect หาค่าด้วย ค่า Forward แล้วคูณด้วย 100 ค่าที่ได้จะออกมาเป็น เปอร์เซนต์ ถัดมาโปรแกรมจะอ่านค่าอุณหภูมิจากไอซีอุณหภูมิทั้ง 2 ตัว เมื่ออ่านค่าได้หมดแล้ว ขั้นต่อมาโปรแกรมจะทำการตรวจสอบเวลาการส่งข้อมูลสถานะ โดยจะอ้างอิงจากเวลาที่มีการตั้งไว้ตั้งแต่เริ่มต้นโปรแกรม ถ้าถึงเวลา โปรแกรมก็จะทำการส่งค่าสถานะต่างๆ ที่เครื่องอ่านได้ให้ผู้ใช้ (GSM Module) ถัดมาโปรแกรมจะทำการตรวจสอบอุณหภูมิของเครื่องส่ง โดยแบ่งเป็น 2 กรณี ดังนี้ กรณีแรก คือ อุณหภูมิเกิน 30 องศา โปรแกรมจะสั่งให้พัดลมระบายความร้อน และถ้าอุณหภูมิเริ่มลดลง พัดลมก็จะดับเอง กรณีที่สองคือ หลังจากกรณีที่หนึ่งแล้ว อุณหภูมิ ยังสูงขึ้นเรื่อยๆ จนเกิน 35 องศา โปรแกรมจะส่งข้อความสถานะเครื่อง ไปยังผู้ใช้ (GSM Module) ถัดมาโปรแกรมจะทำการตรวจสอบค่า F/R (Forward & Reflect) ว่าเกินค่าที่ตั้งไว้ตั้งแต่เริ่มโปรแกรมหรือไม่ ถ้าเกิน โปรแกรมจะส่งข้อความสถานะเครื่อง ไปให้ผู้ใช้ โดยค่าทั้งหมดนี้จะแสดงผลที่จอแสดงผล และโปรแกรมจะกลับไปเริ่มต้นใหม่อีกครั้ง



## บทที่ 4

### การใช้งานโครงงาน

#### 4.1 การใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง

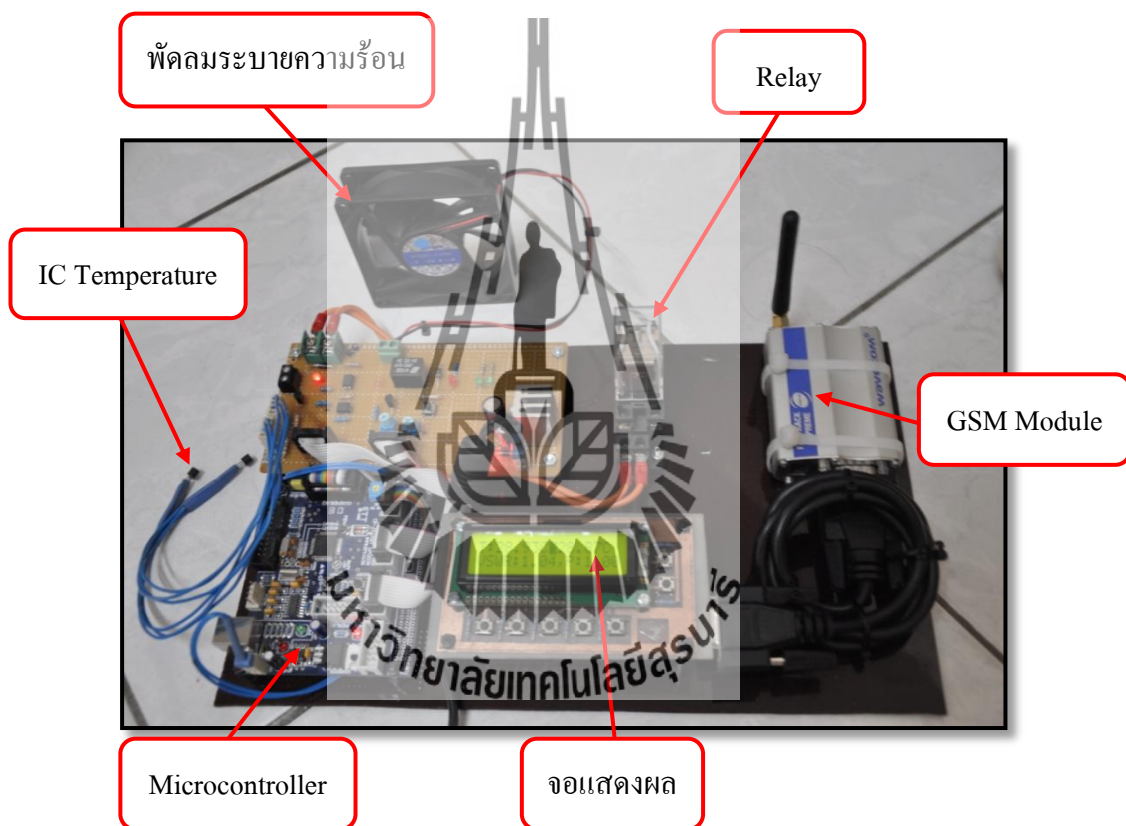
ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง มีลักษณะดังรูปที่ 4.1



รูปที่ 4.1 ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง

ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง นั้นมีไมโครคอนโทรลเลอร์ ARM7 เป็นตัวควบคุมการทำงาน โดยจะอ่านค่าอุณหภูมิและอัตราส่วนของกำลังส่งกับค่าสะท้อนกลับ ( Ratio) จากเซนเซอร์ที่ถูกติดตั้งไว้ภายในเครื่องส่ง เมื่อไมโครคอนโทรลเลอร์ประมวลผลแล้วพบว่า เครื่องส่งวิทยุกระจายเสียงมีค่าพารามิเตอร์ดังกล่าวเกินจากค่าที่กำหนดไว้ (ผู้ใช้เป็นผู้กำหนด) จะทำการส่งข้อมูลผ่านทางระบบ SMS ถึงผู้ใช้ในทันที

#### 4.1.1 การเริ่มต้นใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง



รูปที่ 4.2 ส่วนประกอบของระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง

1. เมื่อต่อ Power Supply ให้กับอุปกรณ์แล้ว ระบบจะทำการโหลดข้อมูล เพื่อเตรียมพร้อมเข้าสู่การทำงาน
2. หากมีการ Setting จะแสดงการตั้งค่า ผ่านทางจอแสดงผล ดังนี้
  - ตั้งค่าเวลา ที่ต้องการให้ระบบส่งข้อมูลให้ เป็นรายวัน
  - ตั้งค่า Ratio
  - ตั้งค่า Temperature ของเครื่องส่งวิทยุกระจายเสียง
  - ตั้งค่าเวลาและวันที่ปัจจุบัน
  - ตั้งค่าหมายเลขโทรศัพท์ของผู้ใช้

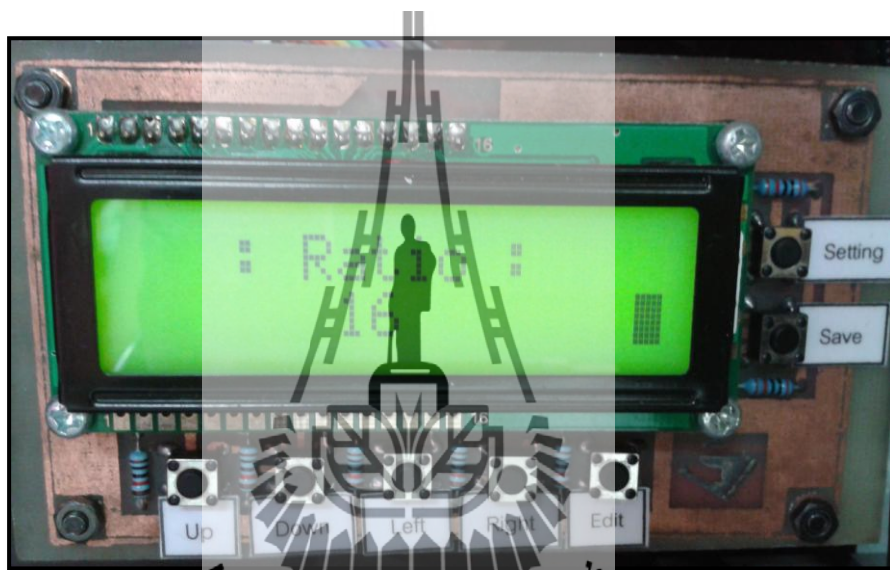
เมื่อทำการ Setting ค่าพารามิเตอร์ดังกล่าวแล้ว ให้กดปุ่ม OK เพื่อบันทึก

ตัวอย่างการ Setting ค่าพารามิเตอร์ (ในที่นี้ จะทำการ Setting ค่า Ratio)



รูปที่ 4.3 แสดงหน้าจอ LCD สำหรับการเข้าตั้งค่าโปรแกรม

- กดปุ่ม Setting ค้างไว้ ระบบจะนำเข้าสู่ โหมด Setting Program ดังรูปที่ 4.3
- จากนั้น กดปุ่ม Left หรือ Right ไปเรื่อยๆ จนเจอโหมด Ratio ดังรูปที่ 4.4
- กดปุ่ม Edit เพื่อทำการแก้ไขข้อมูล โดยการกดปุ่ม Up หรือ Down ไปเรื่อยๆ
- เมื่อได้ค่าที่ต้องการแล้ว ให้กดปุ่ม Edit อีกครั้ง เพื่อให้กลับสู่เมนู Setting Program
- เมื่อเสร็จสิ้นการตั้งค่า ให้กดปุ่ม OK เพื่อกลับเป็นโหมดแสดงผลดังเดิม



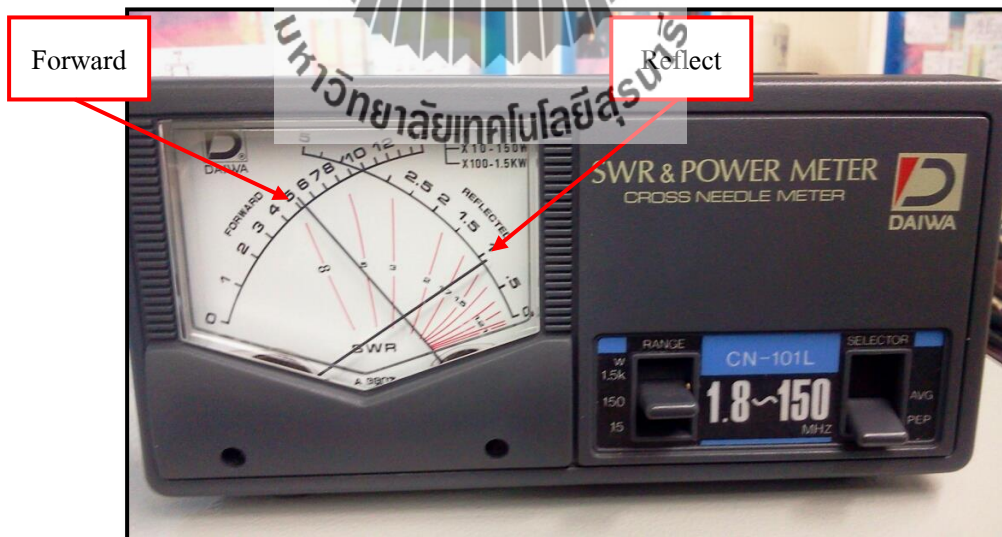
รูปที่ 4.4 แสดงการ Setting ในโหมด Ratio

ทั้งนี้ ในการรับ - ส่งข้อมูลผ่านทางระบบ SMS ผู้ใช้จะต้องทำการลงทะเบียนหมายเลขโทรศัพท์ไว้กับระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง โดยการ setting หมายเลขผ่านทางหน้าจอแสดงผล เพื่อป้องกันไม่ให้บุคคลอื่นเข้าถึงการทำงานของเครื่องส่งวิทยุกระจายเสียงได้

ตัวอย่าง การใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงร่วมกับเครื่องส่งวิทยุ

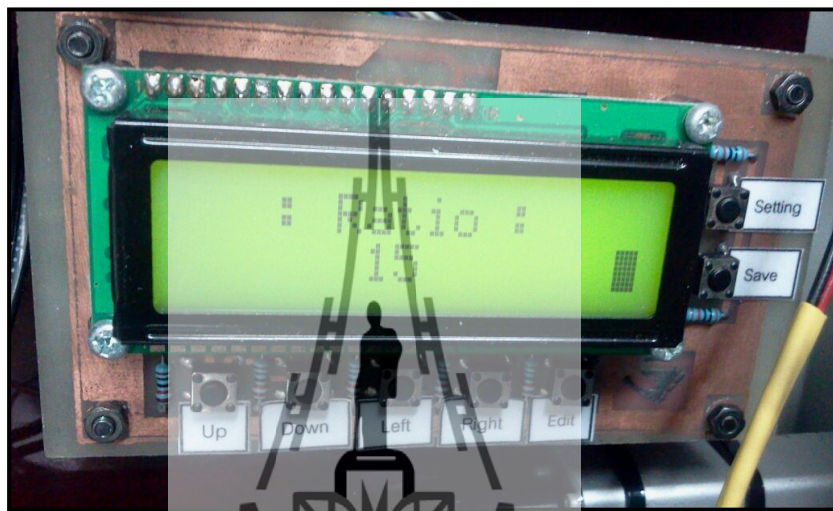


รูปที่ 4.5 การใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงร่วมกับเครื่องส่งวิทยุ



รูปที่ 4.6 เข็มแสดงค่า Forward และ Reflect ของเครื่อง SWR & Power Meter  
เมื่อมีการส่งวิทยุ

จากรูปที่ 4.5 และ 4.6 จะเห็นว่า เมื่อเครื่องส่งวิทยุส่งสัญญาณออกไป เครื่อง SWR & Power Meter จะสามารถวัดค่ากำลังสะท้อนกลับ (Reflect) ได้ ดังรูปที่ 4.6 และนำมาคำนวณค่า Ratio จากอัตราส่วนระหว่าง Forward และ Reflect เพื่อแสดงผลผ่านทางหน้าจอ LCD ดังรูปที่ 4.7



รูปที่ 4.7 แสดงค่า Ratio จากการอ่านค่า Reflect ของเครื่องส่งวิทยุ



## 4.2 การทดลองการใช้งานโครงการ

หลังจากที่ผู้ใช้ ทำการ Setting ค่าพารามิเตอร์แล้ว ระบบรายงานสถานะจะทำการอ่านค่าและประมวลผลผ่านทางไมโครคอนโทรลเลอร์อยู่ตลอดเวลา หากค่าพารามิเตอร์ในเครื่องส่งวิทยุกระจายเสียงเกินจากค่าที่ผู้ใช้กำหนดไว้ ไมโครคอนโทรลเลอร์จะทำการสั่งงานดังนี้

- 4.2.1 เมื่ออุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียง เกินค่าที่กำหนดไว้ ให้พัฒนาระบายอากาศทำงาน ดังตัวอย่าง ในที่นี้ กำหนดค่าอุณหภูมิไว้ที่ 30 C



รูปที่ 4.8 แสดงการทำงานของพัฒนาระบายอากาศ เมื่ออุณหภูมิเกินที่กำหนดไว้

- 4.2.2 เมื่ออุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียง เกินค่าที่กำหนดไว้ ให้ระบบรายงานสถานะส่งข้อความถึงผู้ใช้ทันที
- 4.2.3 เมื่อค่า Ratio ของเครื่องส่งวิทยุกระจายเสียง เกินค่าที่กำหนดไว้ ให้ระบบรายงานสถานะส่งข้อความถึงผู้ใช้ทันที
- 4.2.4 เมื่อถึงเวลาที่ผู้ใช้กำหนดให้ส่งข้อมูลทั้งหมด เป็นรายวัน ให้ระบบรายงานสถานะส่งข้อความถึงผู้ใช้ทันที



รูปที่ 4.9 จอแสดงผล เมื่อระบบส่งข้อความถึงผู้ใช้ตามหมายเลขที่กำหนดไว้





รูปที่ 4.10 จอแสดงผล เมื่อระบบส่งข้อมูลถึงผู้ใช้ได้สำเร็จ

### 4.3 ผลการทดลองโครงการ

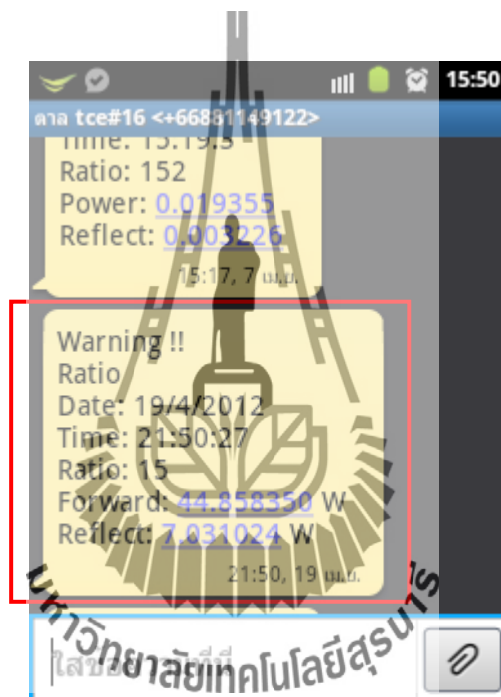
จากการทดลอง ใช้งานระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง หลังจากที่ไม่โครคอนโทรลเลอร์ประมวลผลข้อมูลจากเซนเซอร์ที่ติดตั้งไว้ในเครื่องส่งวิทยุกระจายเสียง แล้วพบว่า ค่าพารามิเตอร์ที่รับได้ เกินจากที่ผู้ใช้กำหนดไว้ ไมโครคอนโทรลเลอร์จะสั่งการให้ส่งข้อมูลผ่านทางระบบ SMS ถึงผู้ใช้ได้ทันที ซึ่งได้ผลการรับข้อมูลดังนี้

4.3.1 ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง ส่งข้อมูลถึงผู้ใช้ เมื่ออุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียงเกินที่ผู้ใช้กำหนดไว้ ดังรูปที่ 4.11



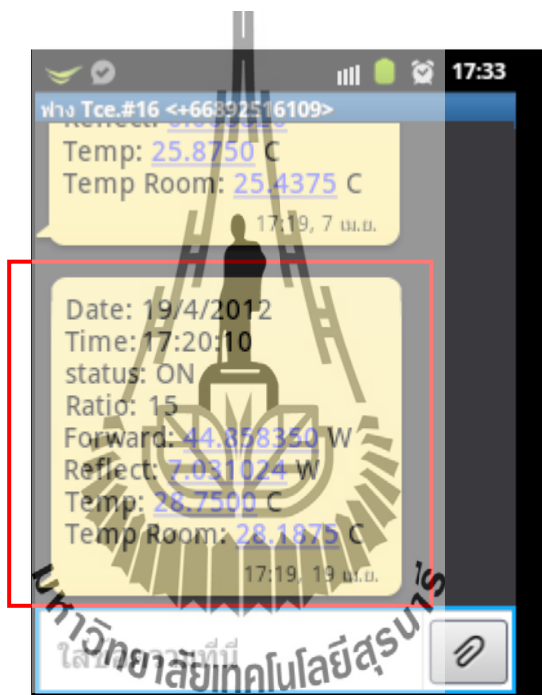
รูปที่ 4.11 แสดงข้อความที่ระบบส่งถึงผู้ใช้ เมื่ออุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียงเกินค่าที่กำหนดไว้

4.3.2 ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง ส่งข้อมูลถึงผู้ใช้ เมื่อค่า Ratio ของเครื่องส่งวิทยุกระจายเสียง เกินที่ผู้ใช้กำหนดไว้ ดังรูปที่ 4.12



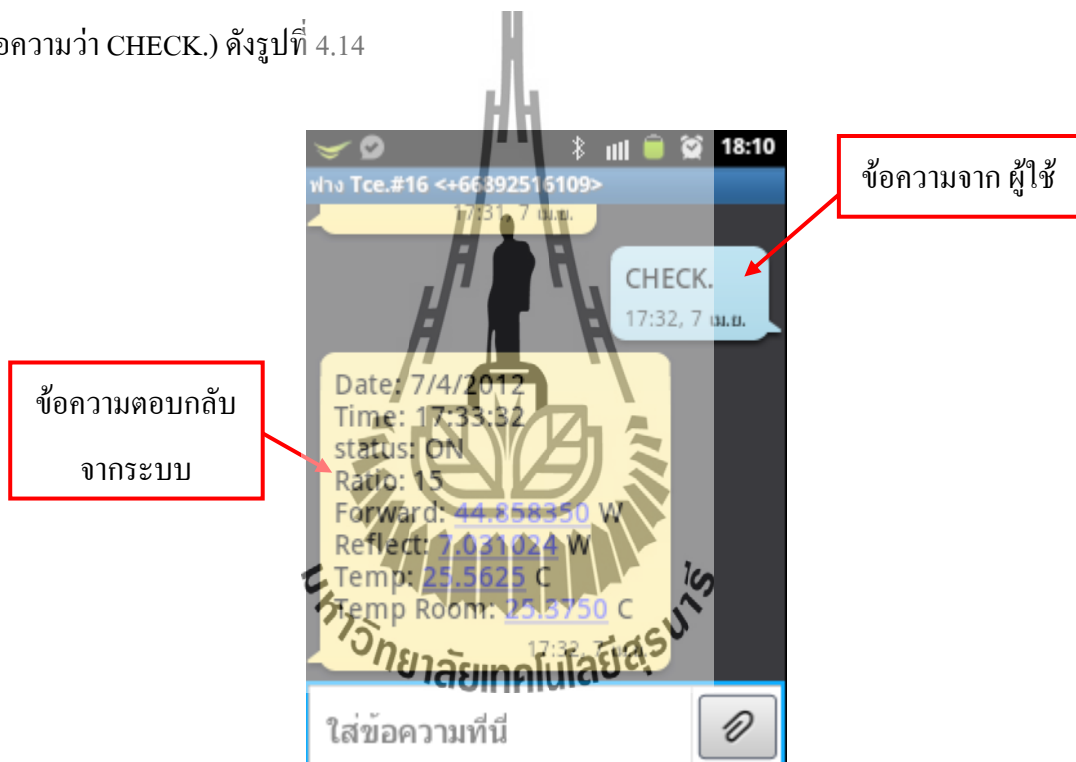
รูปที่ 4.12 แสดงข้อความที่ระบบส่งถึงผู้ใช้ เมื่อค่า Ratio เกินค่าที่กำหนดไว้

4.3.3 ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียงส่งข้อมูลถึงผู้ใช้ เมื่อถึงกำหนดเวลาที่ผู้ใช้ต้องการให้ส่ง ดังรูปที่ 4.13



รูปที่ 4.13 แสดงข้อความที่ระบบส่งถึงผู้ใช้ เมื่อถึงกำหนดเวลาที่ตั้งค่าไว้

นอกจากนี้ ระบบรายงานสถานะเครื่องส่งวิทยุกระจายเสียงผ่านทางระบบ SMS นี้ยังสามารถตรวจสอบค่าพารามิเตอร์ผ่านทางระบบ SMS ได้ เมื่อผู้ใช้ส่งข้อความผ่านทางระบบ SMS (ใช้ข้อความว่า CHECK.) ดังรูปที่ 4.14



รูปที่ 4.14 แสดงการรับ – ส่งข้อความเพื่อตรวจสอบค่าพารามิเตอร์ของผู้ใช้

## สรุปผลการทดลอง

จากการทดลอง จะเห็นว่า ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง สามารถทำการวัดค่าพารามิเตอร์และตรวจสอบค่าพารามิเตอร์ต่างๆ คือ ค่าอณูภูมิภายในเครื่องส่งวิทยุกระจายเสียง, ค่า Ratio (อัตราส่วนระหว่าง Forward และ Reflect) รวมถึงสามารถส่งข้อมูลที่ได้ออกจากการตรวจสอบผ่านทางระบบ SMS ถึงผู้ใช้ได้ในกรณี ดังต่อไปนี้

1. กรณีค่าพารามิเตอร์ (ค่าอณูภูมิภายในเครื่องส่งวิทยุกระจายเสียงและค่า Ratio) เกินจากที่ผู้ใช้ได้ทำการกำหนดไว้ผ่านทางหน้าจอแสดงผล
  2. กรณี ผู้ใช้ระบุเวลาที่ต้องการให้ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง ส่งข้อมูลทั้งหมดให้
  3. กรณี ผู้ใช้ต้องการทราบค่าพารามิเตอร์ ณ เวลาใดเวลาหนึ่ง ผู้ใช้สามารถส่งข้อความผ่านทางระบบ SMS มายังระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง เพื่อให้ระบบทำการส่งข้อมูลกลับไปยังผู้ใช้ได้
- ทั้งนี้ หมายเลขโทรศัพท์ที่จะใช้ในการสื่อสารระหว่างผู้ใช้กับระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง จะต้องทำการ Setting ผ่านทางหน้าจอแสดงผลด้วย

## บทที่ 5

### สรุปผลการทดสอบและข้อเสนอแนะ

เนื่องจากสถานีวิทยุกระจายเสียง จะต้องมีการกระจายเสียงอย่างต่อเนื่องตลอดเวลา จึงอาจเกิด ความผิดปกติอันเกิดจากเครื่องส่งวิทยุกระจายเสียง ซึ่งมีค่าพารามิเตอร์ต่างๆ เกินขีดจำกัด ได้ ทำให้ จำเป็นต้องมีบุคลากรคอยดูแลเครื่องส่งวิทยุกระจายเสียงอยู่ด้วย และในการตรวจสอบค่าพารามิเตอร์ ต่างๆ นั้น จำเป็นที่จะต้องใช้เครื่องมือตรวจสอบอีกหลายชิ้น ที่เป็นไปตามชนิดของพารามิเตอร์ นั้นๆ เราจึงนำปัญหาดังกล่าวมาคิดพัฒนาเป็น ระบบระบบ ตรวจสอบ สถานะเครื่องวิทยุกระจายเสียง ขึ้น เพื่อให้ผู้ดูแลเครื่องส่งวิทยุกระจายเสียงนั้นทราบถึงค่าพารามิเตอร์ต่างๆ ได้โดย ไม่จำเป็นต้องเดินทาง ไปที่สถานีวิทยุด้วยตนเอง โดยผู้ใช้สามารถตั้งค่าพารามิเตอร์แล้วบันทึกค่านั้นได้ตามความต้องการ ของผู้ใช้และจากการทดสอบการใช้งาน โครงการเรื่องระบบตรวจสอบสถานะเครื่องวิทยุกระจายเสียง สามารถสรุปได้ดังนี้

1. สามารถวัดกำลังส่งของเครื่องส่งวิทยุกระจายเสียง และค่าการสะท้อนกลับ เพื่อ นำมาคำนวณเป็นค่า Ratio (อัตราส่วนระหว่าง Forward กับ Reflect) รวมถึงค่า อุณหภูมิภายในเครื่องส่งวิทยุกระจายเสียง และให้แสดงผลผ่านทางจอ LCD ได้
2. สามารถส่งค่าที่ตรวจสอบ ผ่านทางระบบ SMS เป็นรายวันตามต้องการได้
3. ในกรณีที่ค่าพารามิเตอร์ภายในเครื่องส่งวิทยุกระจายเสียงเกินจากค่าที่ผู้ใช้กำหนด ไว้ ระบบตรวจสอบสถานะเครื่องส่งวิทยุกระจายเสียง จะส่งข้อมูลผ่านทางระบบ SMS ไปยังผู้ดูแลได้ รวมถึงจะทำการปิดเครื่องส่ง วิทยุกระจายเสียงได้โดย อัตโนมัติ

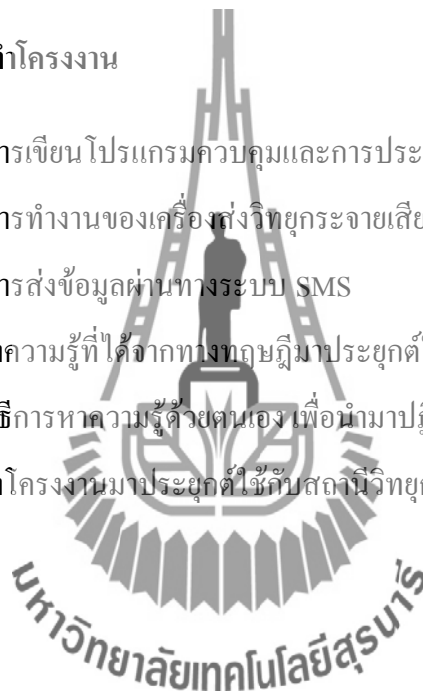
## 5.1 ปัญหาและอุปสรรค

1. ยังไม่มีความรู้ด้านทฤษฎีและปฏิบัติต้องใช้เวลาในการศึกษา
2. หากเกิดไฟดับขึ้นมาจะไม่สามารถส่งข้อมูลผ่านทางระบบ SMS
3. เครื่องจะไม่สามารถส่งข้อมูลผ่านทางระบบ SMS ได้หากที่ตรงนั้นไม่มีสัญญาณ

โทรศัพท์

## 5.2 สิ่งที่ได้รับจากการทำโครงการ

1. ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์
2. ได้เรียนรู้การทำงานของเครื่องส่งวิทยุกระจายเสียง
3. ได้เรียนรู้การส่งข้อมูลผ่านทางระบบ SMS
4. สามารถนำความรู้ที่ได้จากทฤษฎีมาประยุกต์ใช้ในทางปฏิบัติ
5. ได้เรียนรู้วิธีการหาความรู้ด้วยตนเอง เพื่อนำมาปฏิบัติและใช้งานจริง
6. สามารถนำโครงการมาประยุกต์ใช้กับสถานีวิทยุกระจายเสียงได้



## ประวัติผู้จัดทำ



**นายพิพัฒน์ พงษ์ศรีเพชร**

เกิดเมื่อวันที่ 21 พฤศจิกายน พ.ศ. 2532

ภูมิลำเนาอยู่ที่ ตำบลสุรนารี อำเภอเมืองนครราชสีมา

จังหวัดนครราชสีมา

สำเร็จการศึกษาระดับมัธยมปลายจาก โรงเรียนราชสีมาวิทยาลัย

อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา เมื่อปี 2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี



**นางสาวศิริกัญญา ประไพทรัพย์**

เกิดเมื่อวันที่ 16 กันยายน พ.ศ. 2532

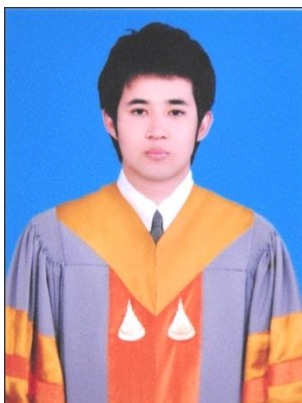
ภูมิลำเนาอยู่ที่ ตำบลบางพระ อำเภอเมืองตราด จังหวัดตราด

สำเร็จการศึกษาระดับมัธยมปลายจาก โรงเรียนสตรีประเสริฐศิลป์

อำเภอเมืองตราด จังหวัดตราด เมื่อปี 2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี



**นายเสรี แก้วตา**

เกิดเมื่อวันที่ 7 ธันวาคม พ.ศ. 2532

ภูมิลำเนาอยู่ที่ ตำบลท่าเสา อำเภอท่ามะกา จังหวัดกาญจนบุรี

สำเร็จการศึกษาระดับมัธยมปลายจาก โรงเรียนสารสิทธิ์พิทยาลัย

อำเภอบ้านโป่ง จังหวัดราชบุรี เมื่อปี 2551

ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี



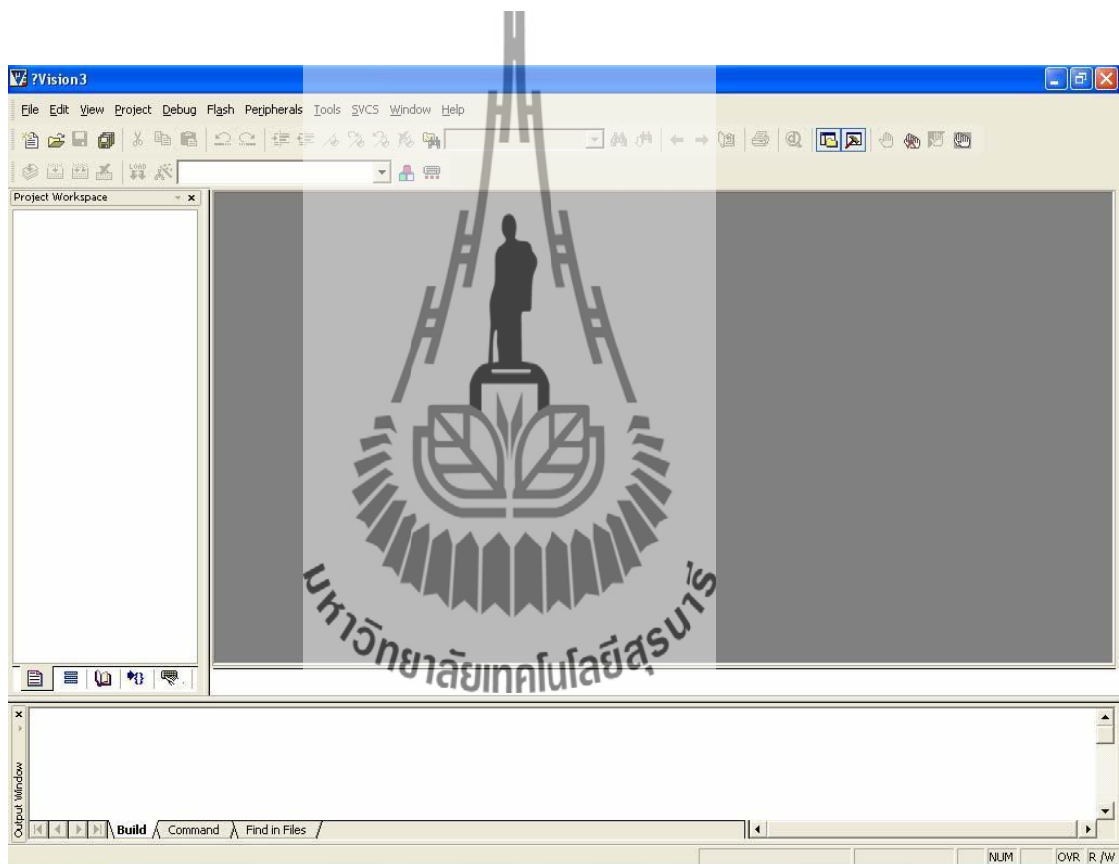
## บรรณานุกรม

- [1] นคร ภัคดีชาติ, อรรถพล บุญยะโกคาม, โอภาส ศิริธรรมชิตถาวร และชัยวัฒน์ ลิ้มพรวิไล (ม.ป.ป.) คู่มือทดลองไมโครคอนโทรลเลอร์ 32 บิต ตระกูล ARM7 เบื้องต้น ฉบับ LPC2148. บริษัท อินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด .
- [2] บริษัทอีทีที 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](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/Reference/LCD/LCD-Module1.htm>
- [4] ตรวจวัดอุณหภูมิ ด้วยดิจิตอลเทอร์โมมิเตอร์ DS18B20 [On line] จาก: <http://www.mindtek.net/ds18b20.php>
- [5] SWR CN-1011 คู่มือการใช้งาน [On line] จาก: [http://www.bellscb.com/products/testequipment/Daiwa/Daiwa\\_CN-1011.htm](http://www.bellscb.com/products/testequipment/Daiwa/Daiwa_CN-1011.htm)



## การใช้ Keil uVision3 ในการสร้าง Project File ของ Keil ARM

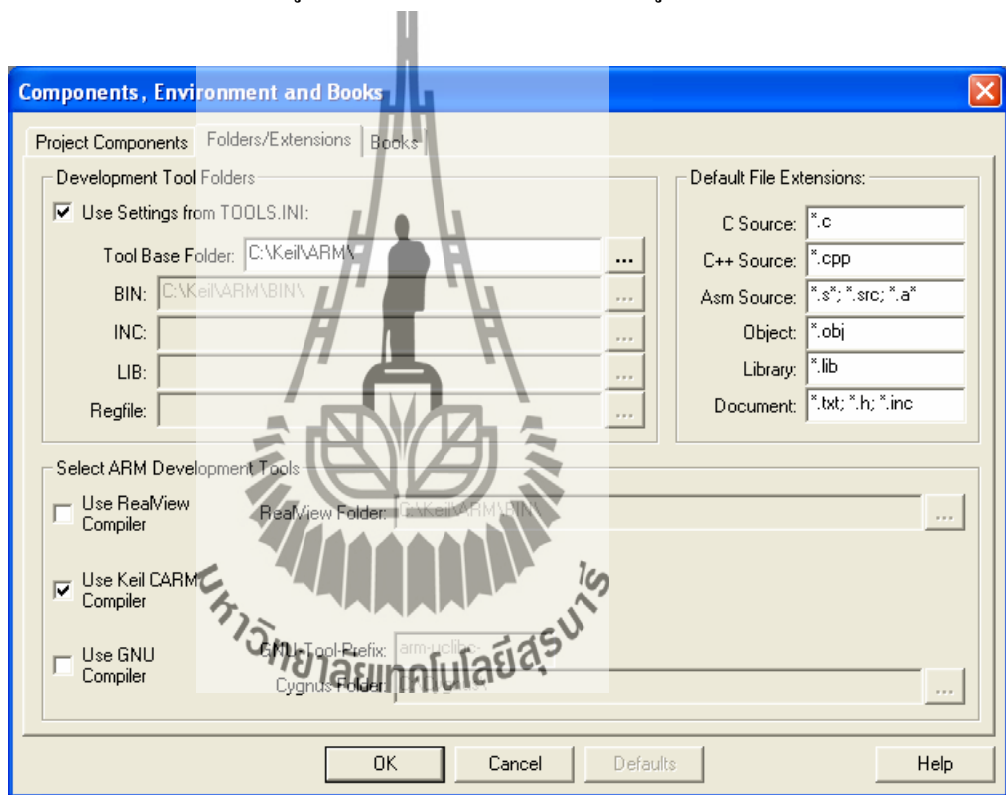
ในที่นี้จะขอแสดงแนวทางการเขียนโปรแกรมภาษาซีโดยใช้ Keil-CARM ในการแปลคำสั่ง ภายใต้โปรแกรม Text Editor ของ Keil (Keil uVision3) โดยจะขออธิบายถึงเฉพาะวิธีการกำหนดค่า Option สำหรับเชื่อมโยงคำสั่งในการสั่งแปลโปรแกรมด้วย Keil-CARM ผ่านทาง Keil uVision3 เท่านั้น ส่วนรายละเอียดคำสั่งและการใช้งานฟังก์ชันต่างๆในการเขียนโปรแกรมของ Keil-CARM นั้นขอให้ผู้ศึกษาจากคู่มือคำสั่งของ Keil-CARM เอง โดยวิธีการกำหนดค่าตัวเลือกของ Keil uVision3 ให้ใช้งานกับ Keil-CARM นั้นมีขั้นตอนพอสังเขปดังนี้คือ



รูปที่ 5.1 หน้าต่างของโปรแกรม Keil uVision3

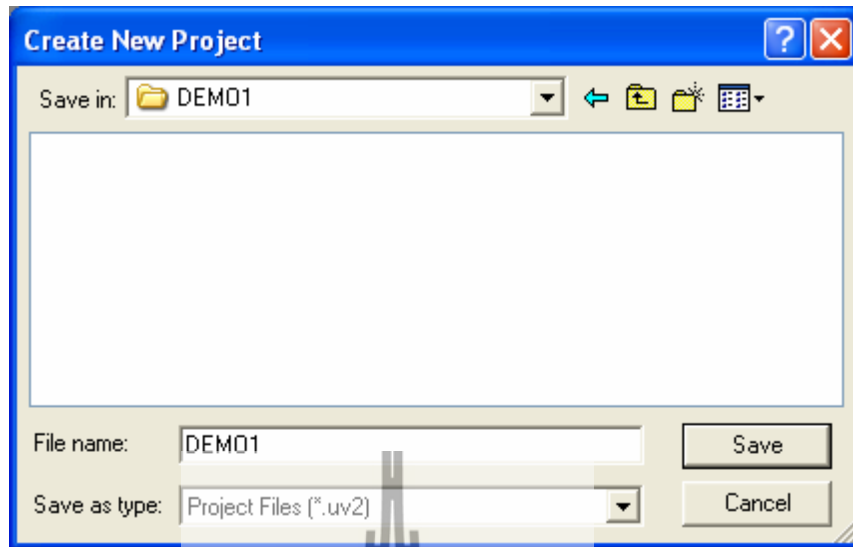
1. เปิดโปรแกรม Keil uVision3 ซึ่งเป็นโปรแกรม Text Editor ของ Keil-CARM ใช้สำหรับใช้ในการเขียนโปรแกรมที่เป็น Source Code ภาษาซี โดยจะมีลักษณะดังรูป

2. ทำการกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3 ให้ใช้งานกับโปรแกรม Keil uVision3 และ Keil-CARM โดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง Project → Components ,Environment, Books... จากนั้นให้เลือกค่าตัวเลือกสำหรับกำหนดการใช้งาน Compiler จากหัวข้อ Select ARM Development Tools ซึ่งจะมีค่าตัวเลือกอยู่ 3 แบบ คือ Use Keil-CARM Tools ,Use GNU Tools และ Use ARM Tools โดยให้เลือกเป็น “Use Keil Tools” จากนั้นให้ทำการกำหนดตำแหน่ง Folder สำหรับเก็บค่าตัวเลือกการทำงานของ โปรแกรม Keil ARM ซึ่งตามปรกติ แล้วจะเป็น “C:\Keil\ARM\” แต่ถ้าติดตั้ง Keil ไว้ที่อื่นก็ต้องปรับเปลี่ยนให้ถูกต้องตามความเป็นจริงด้วย ดังรูป



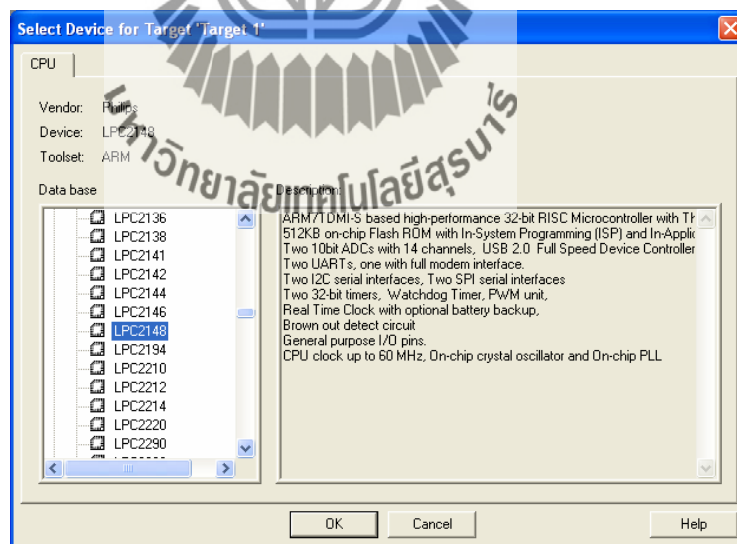
รูปที่ 5.2 การกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3

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



รูปที่ 5.3 หน้าต่างของการสร้าง Project ใหม่

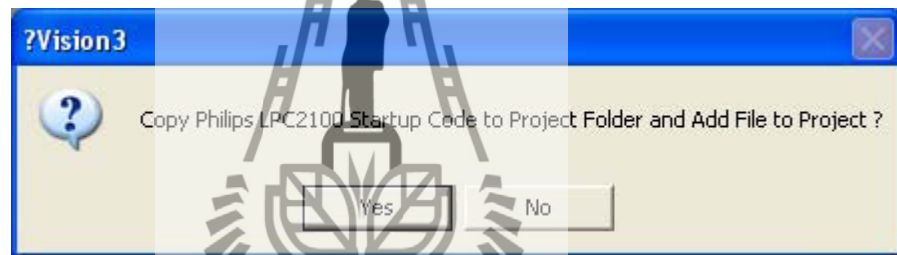
หลังจากกำหนดชื่อและสั่ง Save Project File แล้ว โปรแกรมจะรอให้ผู้ใช้ทำการกำหนดเบอร์ MCU ที่จะใช้งานใน Project ที่ตั้ง Save นั้น ซึ่งในกรณีที่ใช้งานกับบอร์ด CP-JR ARM7 USB-LPC2148 นั้น ให้เลือกกำหนดเบอร์ของ MCU เป็น LPC2148 ของ Philips แล้วเลือก OK ดังรูป



รูปที่ 5.4 การกำหนดเบอร์ MCU ที่จะใช้งาน

หลังจากเลือกกำหนดเบอร์ของ MCU เป็นที่เรียบร้อยแล้ว ในขั้นตอนนี้โปรแกรมจะรอให้ผู้ใช้ยืนยันว่าต้องการจะทำการ Copy ไฟล์ Startup ของ Keil เพื่อใช้งานกับ MCU ของ Philips มาใช้

ใน Project ด้วยหรือไม่ โดย Startup ไฟล์จะเป็นส่วนของการกำหนดค่าเริ่มต้นการทำงานให้กับ MCU เช่น การกำหนดค่า Stack และการกำหนดค่าการทำงานให้กับ Phase-Lock-Loop ต่างๆ ก่อนที่จะเริ่มต้นทำงานตามโปรแกรมที่เราเขียนขึ้น ไม่เช่นนั้นแล้วโปรแกรมที่เราเขียนขึ้นมานั้นจะต้องเพิ่มคำสั่งในการเตรียมการทำงานส่วนเหล่านี้ให้ MCU เองทั้งหมดแต่เนื่องจากไฟล์ Startup ของ Keil-ARM นั้น เป็นไฟล์ภาษาแอสเซมบลี (Assembly) ซึ่งกำหนดค่าการทำงานไว้กับชุดพัฒนาของ Keil เอง ดังนั้นข้อกำหนดและการกำหนดค่าบางอย่างจะมีความแตกต่างกันอยู่กับค่าที่ต้องการสำหรับบอร์ด “CP-JR ARM7 USB-LPC2148” ไม่สามารถใช้งานไฟล์ Startup ได้ทันที ต้องมีการแก้ไขค่าตัวเลือกใหม่ดังนั้นก่อนที่จะใช้โปรแกรม Keil-CARM ในการแปลคำสั่งให้มัน ผู้ใช้จะต้องเข้าไปแก้ไขไฟล์ Startup ใหม่โดยต้องกำหนดรูปแบบให้ถูกต้องตรงกับความต้องการของบอร์ดด้วย ดังนั้นในที่นี้ขอแนะนำให้เลือก “No” เพื่อไม่ให้ Keil uVision3 ทำการ Copy ไฟล์ Startup ของ Keil-CARM มาใช้ใน Project นี้ด้วย

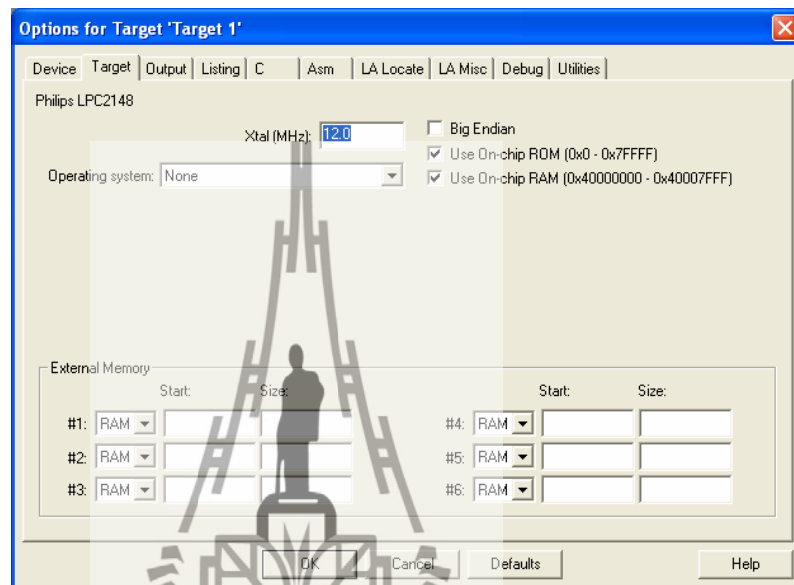


รูปที่ 5.5 แสดงข้อความการกำหนด Startup File

4. ให้ทำการ Copy File ชื่อ Startup.s ที่ทาง ผู้จัดทำจัดเตรียมไว้ใน CD-ROM ชื่อ “Startup.s” มาไว้ในตำแหน่ง Folder เดียวกันกับ Project File ที่สร้างขึ้นมานี้โดยไฟล์ “Startup.s” จะเป็นไฟล์ซึ่งบรรจุคำสั่งภาษาแอสเซมบลีของ 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 ซึ่งจะส่งผลกระทบต่อการทำงานของโปรแกรมด้วย

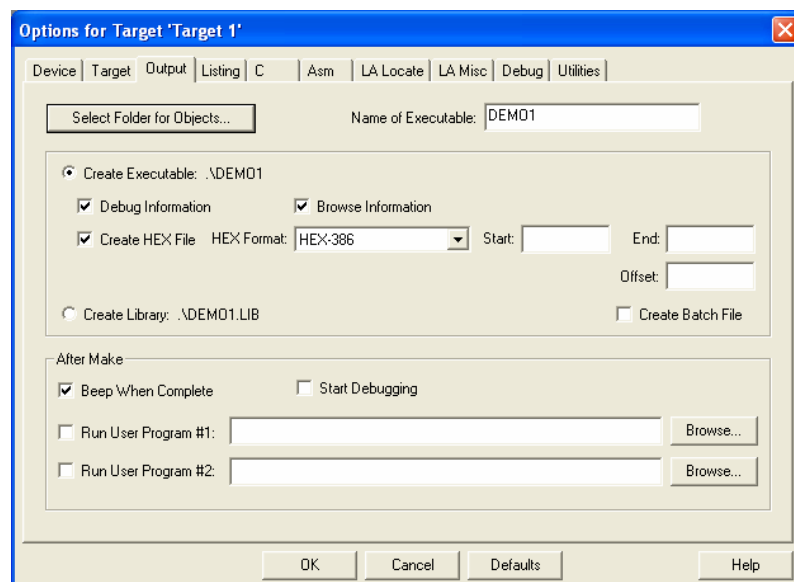
5. ให้ทำการกำหนดค่า Option ของ Project File โดยเลือกเมนูคำสั่ง Project → Option for Target 'Target 1' จากนั้นเลือกที่ Tab ของ Target เพื่อกำหนดค่าของ MCU Target โดยให้กำหนดดังนี้

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



รูปที่ 5.6 การกำหนดค่า XTAL ของ MCU ที่ใช้งาน

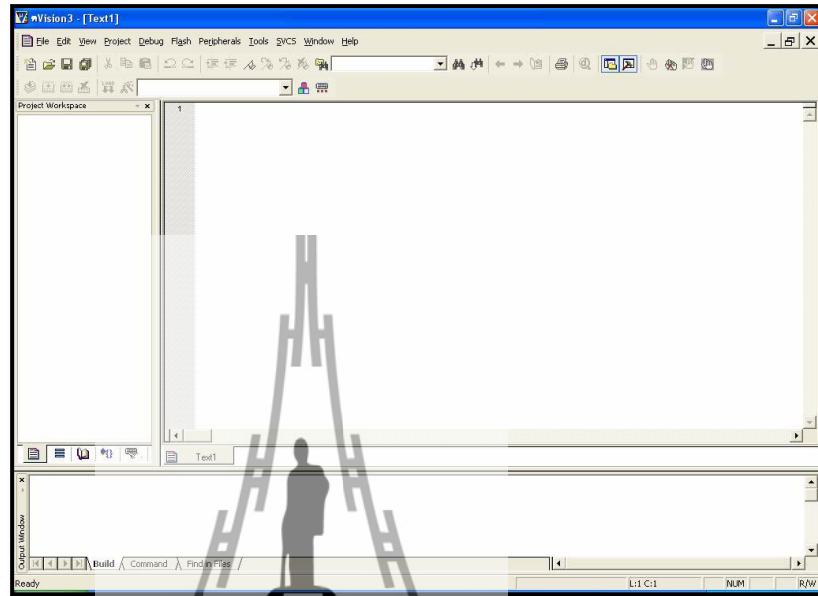
5.2 เลือกที่ Output ให้เลือกคลิกเมนูคำสั่งเลือก Create HEX File พร้อมกับเลือกกำหนดรูปแบบของ Hex ให้เป็นแบบ HEX-386 แล้วเลือก OK ดังรูป



รูปที่ 5.7 การกำหนดค่าตัวเลือก Create HEX File



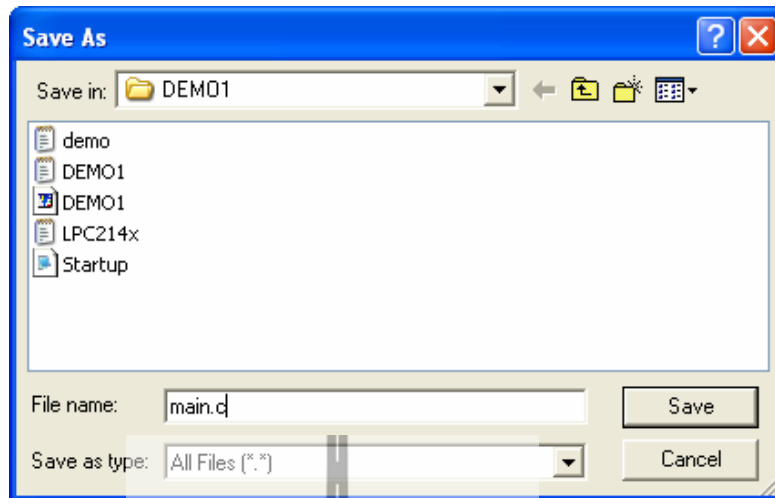
6. เริ่มต้นเขียน Source Code ภาษาซี โดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง File → New... ซึ่งจะได้พื้นที่ในการเขียน Text File เกิดขึ้นมา โดยในครั้งแรกจะกำหนดชื่อตามค่า Default เป็น “Text1” ดังรูป



รูปที่ 5.8 หน้าตาของโปรแกรม Keil uVision3 หลังจากตั้งค่าต่างๆ แล้ว

ในขั้นตอนนี้ให้ทำการพิมพ์ Source Code ภาษาซี ตามข้อกำหนดของ Keil-CARM ในพื้นที่เขียนโปรแกรมตามต้องการ หลังจากพิมพ์คำสั่งภาษาซีเสร็จเรียบร้อยแล้ว ให้สั่ง Save ไฟล์ดังกล่าว โดยต้องกำหนดเป็นไฟล์ที่นามสกุลเป็น “.C” ในที่นี้ขอแนะนำให้สั่ง Save โดยใช้คำสั่ง File → Save As... แล้วกำหนดชื่อและนามสกุลของไฟล์เป็น .main.c” ดังรูป

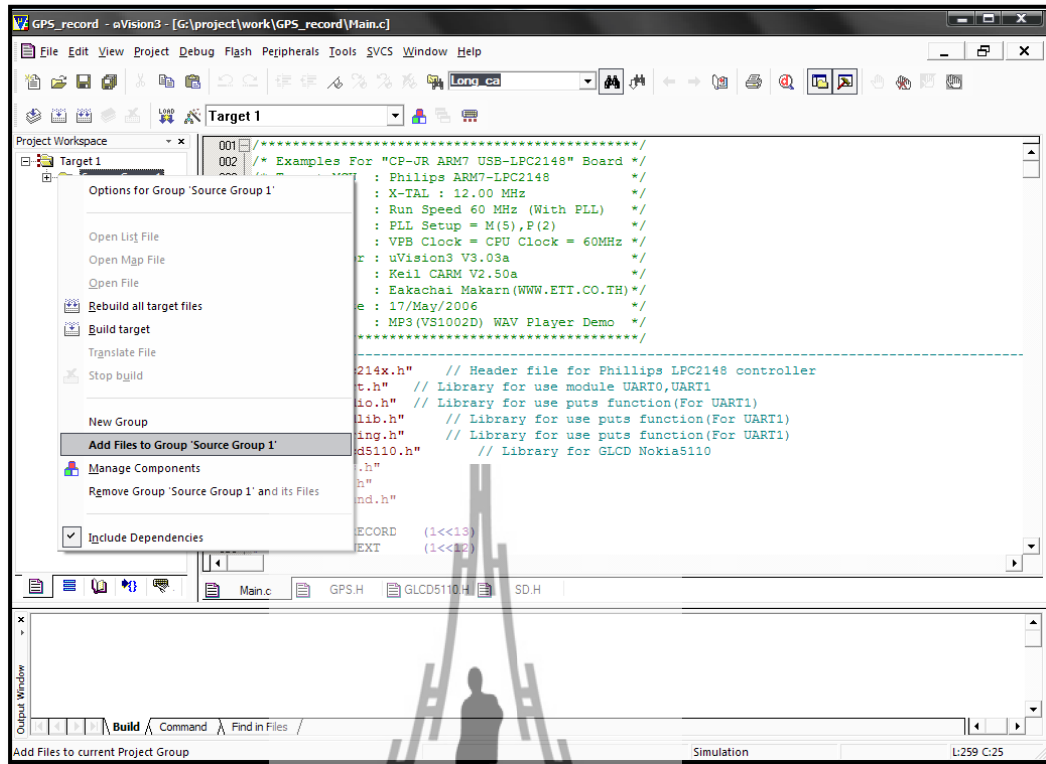




รูปที่ 5.9 หน้าต่างของการ save ไฟล์

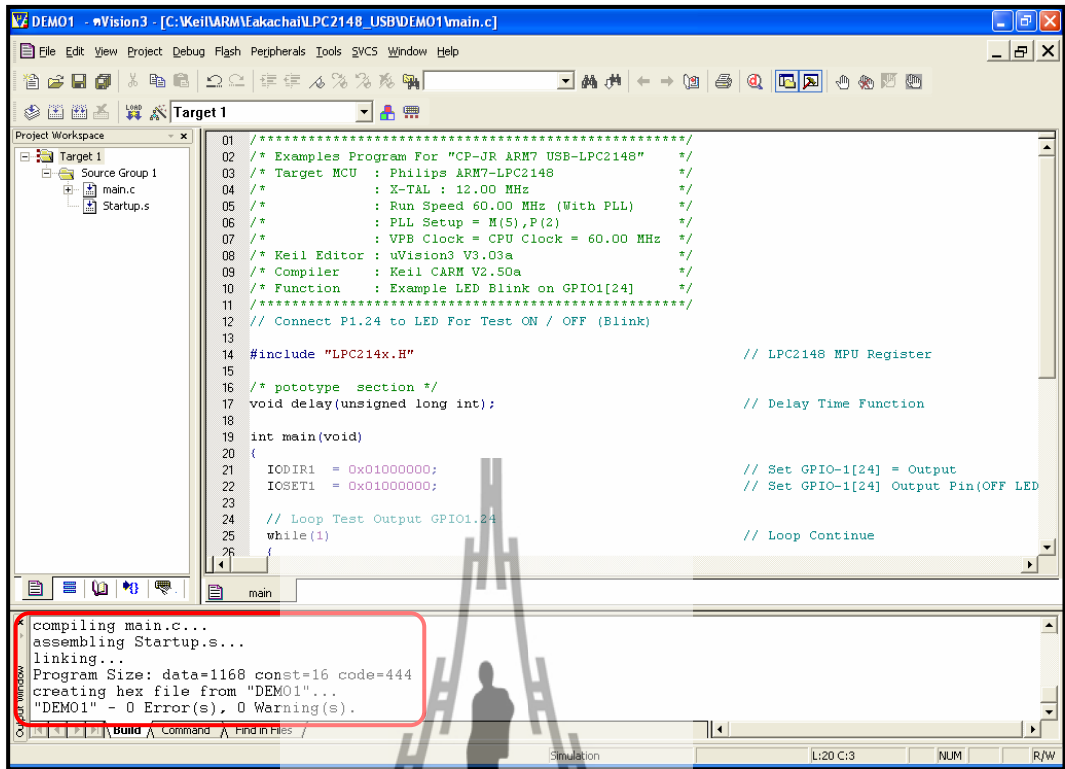
ซึ่งหลังจากที่สั่ง Save ไฟล์เป็น “main.c” แล้วจะเห็นว่าลักษณะสีของตัวอักขระต่างๆ ในโปรแกรมจะเกิดการเปลี่ยนแปลงไปตามหน้าที่ เช่น Comment, ตัวแปร และ คำสั่ง เป็นต้น ซึ่งส่วนนี้เป็นข้อดีของ Keil uVision3 ซึ่งสามารถแยกและแสดงตัวอักขระได้อย่างเป็นหมวดหมู่ ทำให้ง่ายต่อการอ่านโปรแกรมด้วย

7. ทำการสั่ง Add File ต่างๆ เข้ากับ Project File โดยให้เลือกคลิกเมาส์ที่ด้านซ้ายของหน้าต่าง จากนั้นให้คลิกที่ Add Files to Group “Source Group 1” แล้วเลือกที่ Add File ที่ต้องการจะเพิ่มเข้าไปซึ่งรวมกับ Project File โดยในครั้งแรกให้เลือก Files of type เป็น “C Source files(\*.c)” ซึ่งจะปรากฏชื่อไฟล์ต่างๆ ที่เป็น 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-CADDM ทำการแปลคำสั่งให้ทันที ซึ่งหลังจากสั่งแปลโปรแกรมแล้ว ได้ผลถูกต้องและไม่เกิดข้อผิดพลาดใดๆขึ้น (0 Error และ 0 Warning) ก็ได้ Hex File ซึ่งมีชื่อเหมือนกันกับชื่อของ Project File ที่สร้างไว้ ซึ่งผู้ใช้สามารถนำ Hex File ดังกล่าวไปทำการ Download ให้กับ MCU ได้ทันที

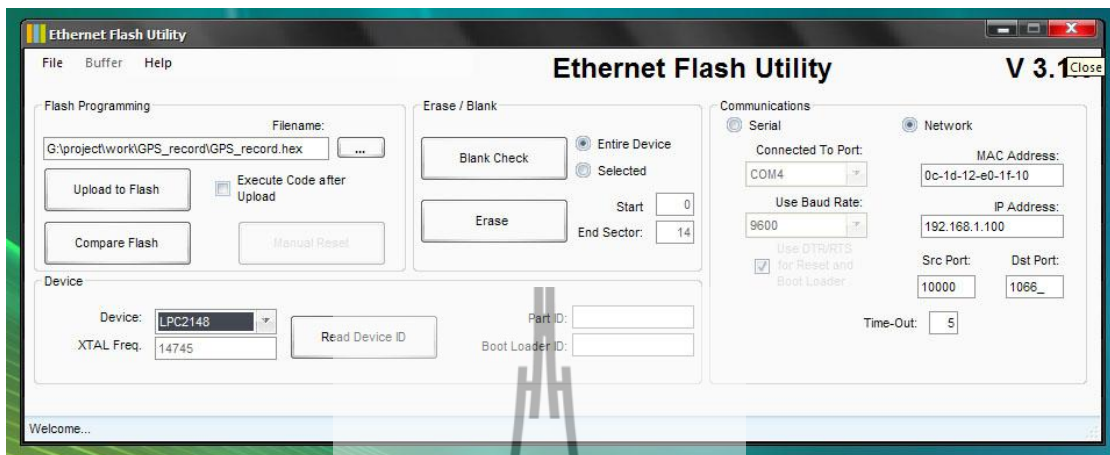


รูปที่ 5.11 การแปลงเป็น Hex File



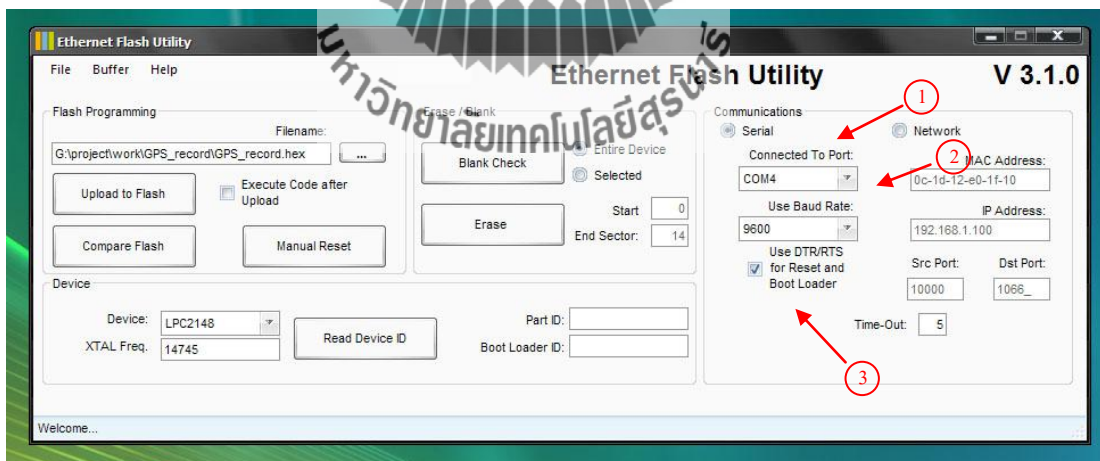
## การโหลดโปรแกรมทดสอบ

เปิดโปรแกรม Ethernet Flash Utility จะได้หน้าต่างดังรูป



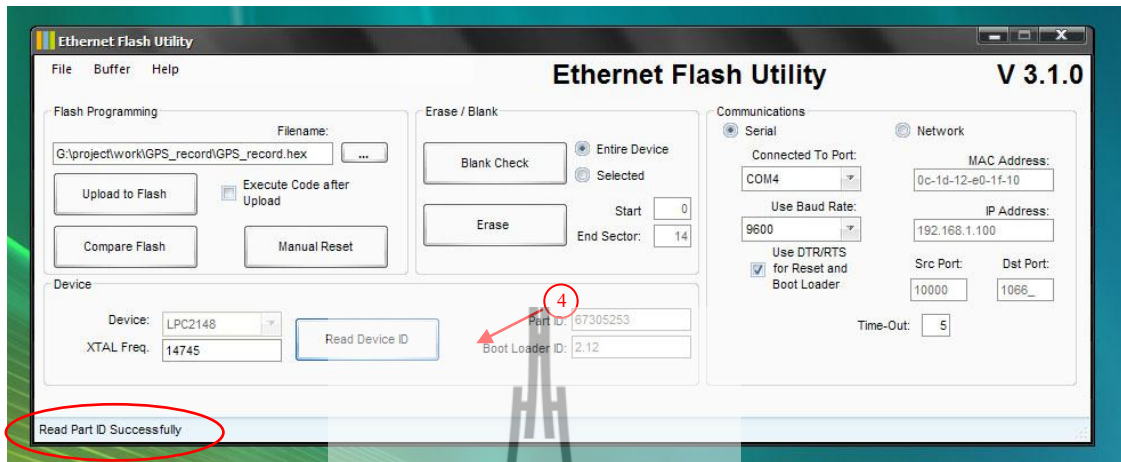
รูปที่ 5.12 หน้าต่างโปรแกรม Ethernet Flash Utility

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



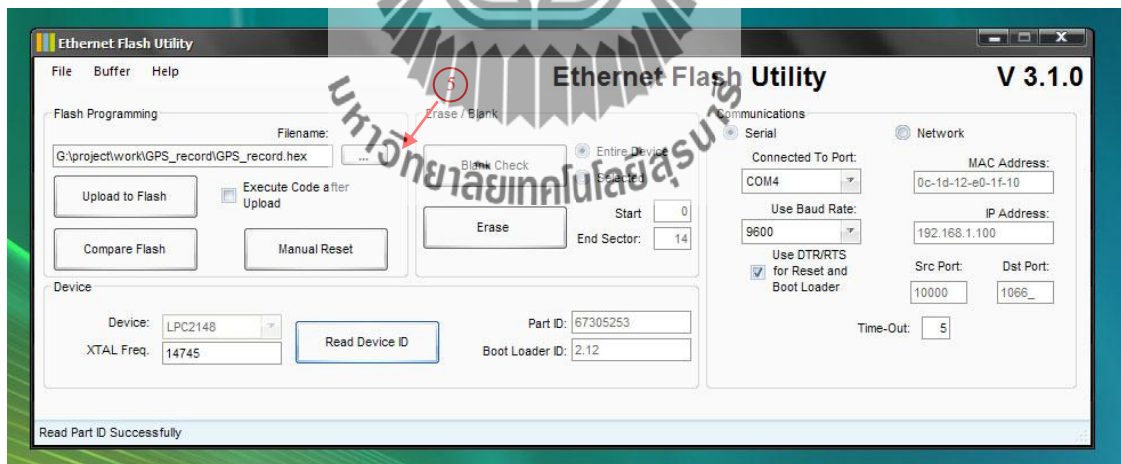
รูปที่ 5.13 หน้าต่างรูปแบบการเชื่อมต่อ

กด Switch Reset ที่บอร์ดไมโครคอนโทรลเลอร์ แล้วเลือกที่ Read Device ID หากเชื่อมต่อกับไมโครคอนโทรลเลอร์สำเร็จจะแสดงข้อความ Read Part ID Successfully

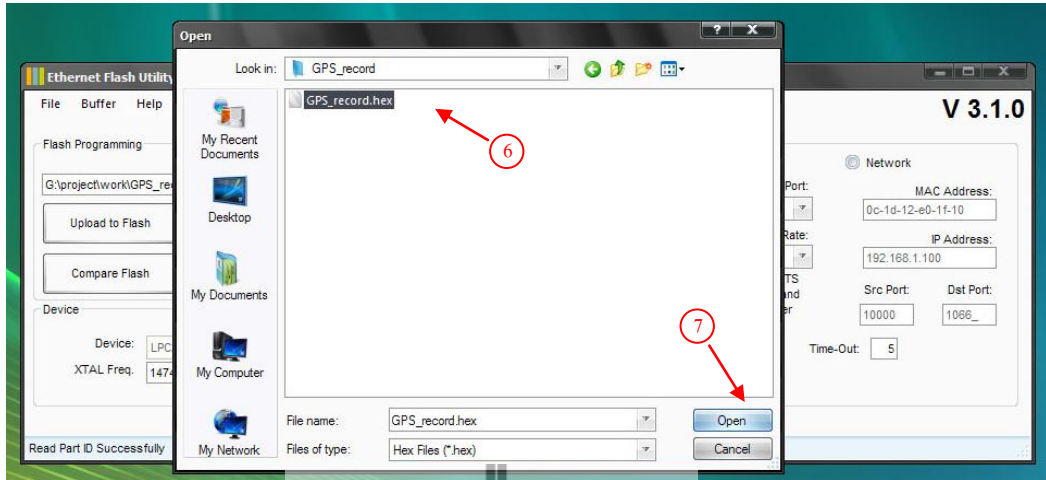


รูปที่ 5.14 หน้าตั้งการเชื่อมต่อสำเร็จ

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

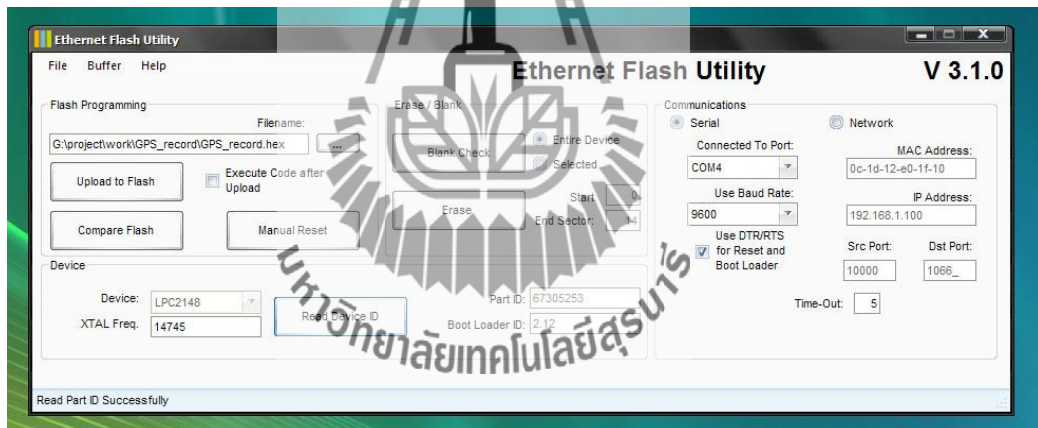


รูปที่ 5.15 หน้าตั้งการโหลดไฟล์ .HEX (1)

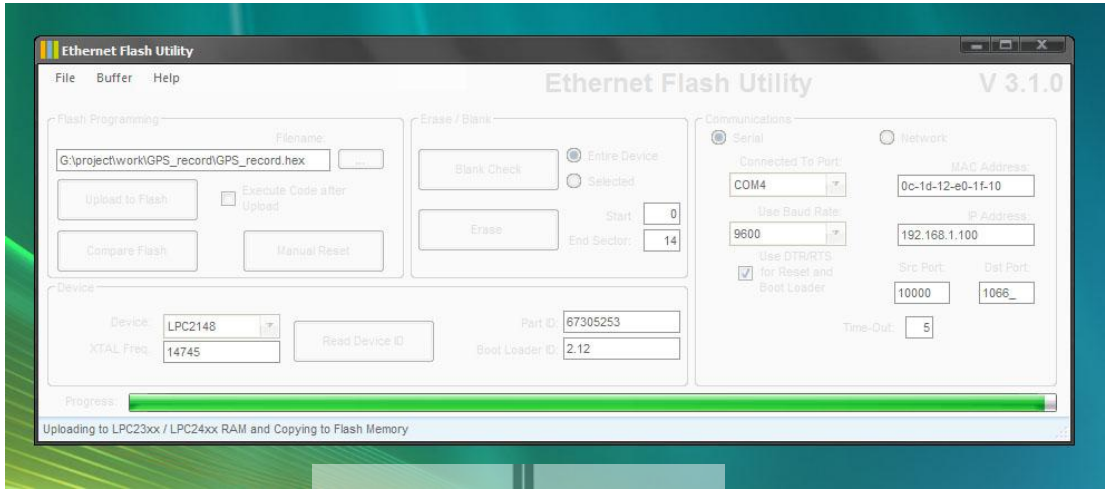


รูปที่ 5.16 หน้าต่างการโหลดไฟล์ .HEX (2)

เลือกที่ Upload to flash และรอนจนกว่าโปรแกรมจะทำการโหลดเสร็จตามรูปที่ 5.17 และรูปที่ 5.18



รูปที่ 5.17 หน้าต่างการโหลดไฟล์ .HEX (3)



รูปที่ 5.18 หน้าต่างการโหลดไฟล์ .HEX (4)

