ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สาย

Electrical Equipment Controller via Internet

พลวัฒน์ ริ้วทอง

PHOLAWAT RIEWTHONG

ศิรวิทย์ ศรีสมบูรณ์สกุล

SIRAVIT SRISOMBOONSAKUL

สรวิศ สระบุรินทร์

SORAWIT SARABURIN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

Electrical Equipment Controller via Internet

PHOLAWAT RIEWTHONG

SIRAVIT SRISOMBOONSAKUL

SORAWIT SARABURIN

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

หัวข้อปริญญานิพนธ์	ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สาย	
	Electrical Equipment Controller via Internet	
รายชื่อนักศึกษา	นายพลวัฒน์ ริ้วทอง	รหัสนักศึกษา 50011065
	นายศิรวิทย์ ศรีสมบูรณ์สกุล	รหัสนักศึกษา 50011560
	นายสรวิศ สระบุรินทร์	รหัสนักศึกษา 50011641
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	รศ.นิกร สุขุตมตันติ	
	ผศ.ไพศาล สิทธิโยภาสกุล	

ปริญญานิพนธ์ฉบับนี้ ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

.....

(รศ.นิกร สุขุตมตันติ)

.....

(ผศ.ไพศาล สิทธิโยภาสกุล)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	ระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สาย Electrical Equipment Controller via Internet	
รายชื่อนักศึกษา	นายพลวัฒน์ ริ้วทอง	รหัสนักศึกษา 50011065
	นายศิรวิทย์ ศรีสมบูรณ์สกุล	รหัสนักศึกษา 50011560
	นายสรวิศ สระบุรินทร์	รหัสนักศึกษา 50011641
ปริญญา	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมสารสนเทศ	
พ.ศ.	2553	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	รศ.นิกร สุขุตมตันติ	
	ผศ.ไพศาล สิทธิโยภาสกุล	

บทคัดย่อ

ในการใช้ชีวิตประจำวันของมนุษย์ในปัจจุบันนั้น เป็นไปด้วยความเร่งด่วน ต้องการความสะดวกสบายและความปลอดภัยในชีวิต และอินเตอร์เน็ตก็เริ่มเข้ามามีความสำคัญมากขึ้น มีการใช้งานอย่างแพร่หลาย จึงได้จัดทำปริญญานิพนธ์ ระบบควบคุมอุปกรณ์ไฟฟ้าผ่านอินเตอร์เน็ต โดยควบคุมผ่านทางหน้าเว็บบราวเซอร์ และแอพพลิเคชั่นบนโทรศัพท์มือถือ เพื่ออำนวยความสะดวกให้แก่ผู้ใช้งาน การทำงานของระบบที่สร้างขึ้นด้วยบอร์ดควบคุม ซึ่งใช้โมดูลไมโครคอนโทรเลอร์ dsPIC33WEB ในการสั่งการเปิด-ปิด อุปกรณ์ไฟฟ้า และใช้ HTML , JAVA ในการสร้างหน้าเว็บเพจและ Microsoft Visual C# ในการสร้าง แอพพลิเคชั่นสำหรับโทรศัพท์มือถือ ในการสั่งควบคุม

Thesis Title	Electrical Equipment Controller via Internet
Student	Mr. Pholawat Riewthong ID 50011065
	Mr. Siravit Srisomboonsakul ID 50011560
	Mr. Sorawit Saraburin ID 50011641
Graduate Level	Bachelor Degree of Information Engineering
Department	Information Engineering
Academic Year	2010
Advisor	Associate Professor Nikorn Sukutamatunti
	Assistant Professor Paisarn Sitthiyopasakul

ABSTRACT

In the fast pace of modern city life demands more convenience and safety. Internet plays a prevailingly important role. The thesis is "Electrical Equipment Controller via Internet" to use micro-controller to control electrical equipment to turn on – off, And Can be set to open - close to 24 hours. In control section, use module micro-controller dsPic33WEB working with relay 4 channel to control electrical device . In command section, use HTML and Java to create webpage and use Microsoft Visual C# to create windows mobile application.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.นิกร สุขุตมตันติ และ ผศ.ไพศาล สิทธิโยภาสกุล ที่ให้ความช่วยเหลือและให้คำปรึกษาเพื่อเป็นแนวทางในการแก้ไขปัญหาต่างๆจนกระทั่งสำเร็จเป็นโครงงานนี้ ทีมงานผู้จัดทำโครงงานขอกราบขอบพระคุณท่านอาจารย์ที่ให้ความกรุณา ณ. ที่นี้

สุดท้ายนี้ ขอขอบคุณบิดามารดาสำหรับความเข้าใจและกำลังใจในการทำโครงงาน และขอบคุณเพื่อนๆ ร่วมห้อง E12-1109 ทุกคนสำหรับกำลังใจและความช่วยเหลือซึ่งกันและกัน

นายพลวัฒน์ ริ้วทอง
นายศิรวิทย์ ศรีสมบูรณ์สกุล
นายสรวิศ สระบุรินทร์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

ในยุคปัจจุบันที่การสื่อสารมีวิวัฒนาการความก้าวหน้าอย่างต่อเนื่อง จึงไม่อาจเข้าถึงข้อมูลข่าวสารที่มีการกระจายจากที่หนึ่งไปยังที่หนึ่งอย่างรวดเร็วได้สะดวก เพื่อให้ทันกับความต้องการของผู้ใช้ซึ่งสิ่งหนึ่งที่เป็นตัวแปรผลักดันที่สำคัญในการรับส่งและกระจายข้อมูลนั้น คือ อินเตอร์เน็ต จะเห็นได้ว่าอินเตอร์เน็ตได้เข้ามามีบทบาทในชีวิตประจำวัน ไม่ว่าจะเป็นการรับ ส่งอีเมล์ การค้นหาข้อมูลต่างๆ การซื้อขายสินค้าผ่านการบริการทางเว็บไซต์ แต่รูปแบบหนึ่งของการสื่อสารข้อมูลที่ได้รับความนิยมอย่างสูงในขณะนี้คือ การควบคุมระยะไกล (Remote Control) ซึ่งเครือข่ายอินเตอร์เน็ตนั้นได้คลอบคลุมไปทั่วทุกมุมโลก และในขณะนี้จำนวนผู้ใช้ก็มีจำนวนเพิ่มขึ้นอย่างรวดเร็ว เราจึงใช้คุณสมบัติตรงนี้ของอินเตอร์เน็ตมาประยุกต์ใช้ในชีวิตประจำวันของเรา โดยจะนำประโยชน์จากอินเตอร์เน็ตมาใช้ในการควบคุมอุปกรณ์ไฟฟ้าต่างๆ ภายในบ้าน

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากข้อจำกัดในการควบคุมอุปกรณ์ไฟฟ้าภายในที่พักอาศัยหรือออฟฟิศ สามารถที่จะกระทำได้ภายในบริเวณนั้นๆ เท่านั้น ไม่สามารถควบคุมจากระยะไกลได้ ซึ่งเป็นอุปสรรคที่สำคัญ ในกรณีที่ลืมเปิดหรือปิดอุปกรณ์ไฟฟ้าเหล่านั้นเมื่อเราไม่อยู่ในที่พักอาศัยหรือออฟฟิศ จึงได้เกิดแนวคิดที่จะนำระบบอินเตอร์เน็ต และแอพพลิเคชั่นบนมือถือมาใช้ในการควบคุมอุปกรณ์ไฟฟ้าภายในที่พักอาศัยหรือออฟฟิศ ซึ่งทำให้เกิดความสะดวกและความปลอดภัย ในกรณีที่เกิดการลืมเปิด หรือปิดอุปกรณ์ไฟฟ้า ซึ่งเป็นสิ่งที่มีประโยชน์อย่างมากสำหรับยุคปัจจุบันที่ต้องอยู่ในเวลาเร่งรีบ

1.2 จุดประสงค์ของโครงงาน

1. เพื่อนำระบบอินเตอร์เน็ตและแอพพลิเคชั่นบนโทรศัพท์มือถือมาประยุกต์ใช้ในการดำเนินชีวิตประจำวัน
2. เพื่อศึกษาและออกแบบโปรแกรมสำหรับควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า
3. เพื่อศึกษาและสร้างแอพพลิเคชั่นระบบควบคุมผ่านโทรศัพท์มือถือ
4. เพื่อเป็นระบบที่สามารถอำนวยความสะดวกในการดำเนินชีวิตประจำวันได้

1.3 ขอบเขตของโครงงาน

1. ควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าผ่านทางเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือได้ 4 อุปกรณ์
2. สามารถตั้งเวลาเปิด-ปิดอุปกรณ์ไฟฟ้าได้ 24 ชั่วโมง
3. สามารถแสดงสถานะเปิด-ปิดปัจจุบันของอุปกรณ์ไฟฟ้าได้

1.4 ผลที่คาดว่าจะได้รับ

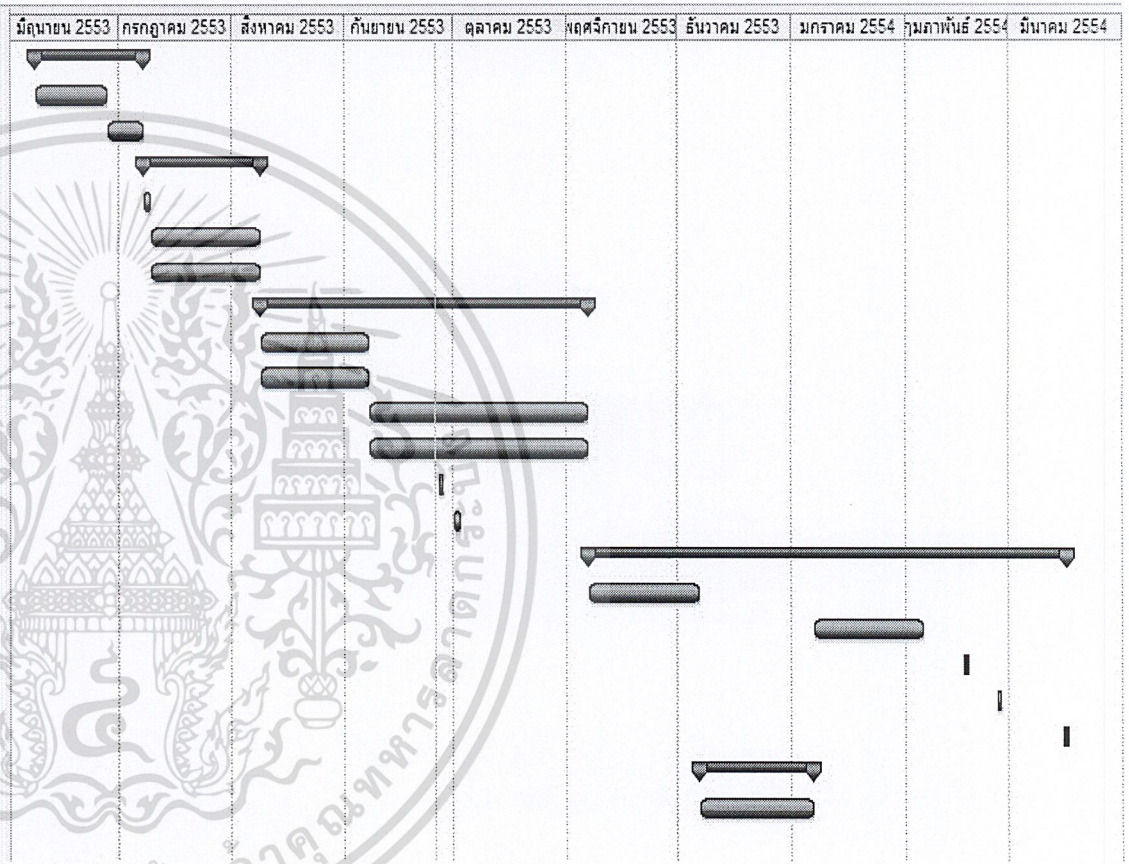
1. สามารถสร้างระบบควบคุมเครื่องใช้ไฟฟ้าได้
2. สามารถนำเทคโนโลยีการสื่อสารทางอินเตอร์เน็ตและโทรศัพท์มือถือมาประยุกต์ใช้ให้เกิดประโยชน์ในชีวิตประจำวันได้

1.5 อุปกรณ์และซอฟแวร์ที่ใช้

1. ไมโครคอนโทรลเลอร์ dsPIC33WEP ใช้เป็นตัวควบคุมและเว็บเซิฟเวอร์
2. Adobe Dreamweaver CS3 ใช้ในการออกแบบเว็บเพจ
3. MPLAB IDE ใช้ในการเขียนโปรแกรมของ dsPIC33 และแปลงเป็นไฟล์ .hex
4. PICkit 2 ใช้ในการนำไฟล์ .hex ลงไมโครคอนโทรลเลอร์
5. Visual Studio 2008,Visual C# ใช้ในการเขียนแอพพลิเคชั่นบนมือถือ
6. Windows Mobile Emulator 6.0 ใช้ในการจำลองการแสดงผลหน้าจอโทรศัพท์
7. Windows Mobile Device Center ใช้ในการ Synchronize โทรศัพท์เข้ากับคอมพิวเตอร์
8. โทรศัพท์มือถือแพลทฟอร์ม Windows Mobile Version 6.1 Professional

1.6 ขั้นตอนการดำเนินงาน

ID	Task Name	Duration	Start	Finish
1	ระบุปัญหาและวิธีแก้ปัญหา	30 days	อ 8/6/53	พ 7/7/53
2	ค้าหาข้อมูล	20 days	อ 8/6/53	อ 27/6/53
3	ระบุปัญหาและวิธีแก้ปัญหา	10 days	จ 28/6/53	พ 7/7/53
4	ศึกษาค้นคว้ามูล	32 days	พ 8/7/53	อ 8/8/53
5	Proposal	2 days	พ 8/7/53	ศ 9/7/53
6	ศึกษาข้อมูลอุปกรณ์ Hardware ที่ใช้งาน	30 days	ส 10/7/53	อ 8/8/53
7	ศึกษาข้อมูล Software ที่ควบคุมสั่งการ	30 days	ส 10/7/53	อ 8/8/53
8	ออกแบบและสร้าง	90 days	จ 9/8/53	ส 6/11/53
9	ออกแบบแผงวงจรอุปกรณ์ Hradware	30 days	จ 9/8/53	อ 7/9/53
10	สร้างแผงวงจรอุปกรณ์ Hardware	30 days	จ 9/8/53	อ 7/9/53
11	ออกแบบ Software ที่ใช้ควบคุมสั่งการ	60 days	พ 8/9/53	ส 6/11/53
12	สร้าง Software ที่ใช้ควบคุมสั่งการ	60 days	พ 8/9/53	ส 6/11/53
13	ส่งเอกสารสอบโปรเจค	1 day	จ 27/9/53	จ 27/9/53
14	สอบโปรเจค 1	2 days	ศ 1/10/53	ส 2/10/53
15	ทดสอบและพัฒนา	130 days	อ 7/11/53	พ 16/3/54
16	ทดสอบการทำงานของอุปกรณ์	30 days	อ 7/11/53	จ 6/12/53
17	พัฒนาและปรับปรุงอุปกรณ์	30 days	ศ 7/1/54	ส 5/2/54
18	ส่งปริญญานิพนธ์	1 day	พ 17/2/54	พ 17/2/54
19	สอบโปรเจค 2	1 day	ส 26/2/54	ส 26/2/54
20	ส่งปริญญานิพนธ์เล่มจริง	1 day	พ 16/3/54	พ 16/3/54
21	สร้างเอกสาร	31 days	อ 7/12/53	พ 6/1/54
22	ทำเอกสารทั้งหมด	31 days	อ 7/12/53	พ 6/1/54



รูปที่ 1.1 แผนผังการดำเนินงาน แสดงขั้นตอนและระยะเวลาในการจัดทำโครงงาน

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้

2.1 ไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708

ไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708 เป็นไมโครคอนโทรลเลอร์ในตระกูล dsPIC ของบริษัท Microchip โดยได้นำเอาไมโครคอนโทรลเลอร์ที่ประมวลผลข้อมูลแบบ 16 บิต เบอร์ dsPIC33FJ128GP708 มาพัฒนาเป็นบอร์ดใช้งาน ซึ่งคุณสมบัติเด่นของบอร์ด dsPIC33 คือ หน่วยประมวลผลสัญญาณดิจิตอล (Digital Signal Processing) และ ทรัพยากรต่างๆ ดังต่อไปนี้

2.1.1 หน่วยประมวลผล (CPU)

- ความเร็วในการประมวลผล 40 MIPS (16 Bit Data / 24 Bit Instruction C
- ฮาร์ดแวร์รองรับการคูณข้อมูล 16 x 16 บิต โดยใช้เวลาเพียง 1 ไซเคิลคำสั่ง
- ฮาร์ดแวร์รองรับการหารข้อมูล 32-bit x 16 บิต
- C Compiler ถูกออกแบบให้มีความกระชับ Optimized Instruction Set
- รองรับการ Interrupt มากถึง 118 Vector Interrupt จาก 63 แหล่ง 7 Priority Level Program
- รองรับการ DMA กับ Peripheral Hardware ได้ 8 ช่อง พร้อม DMA Buffer 2KByte

2.1.1.1 ระบบ (System)

- แหล่งกำเนิดสัญญาณนาฬิกาสามารถเลือกได้ ทั้งจากภายในและภายนอก
- มีวงจร Power-Up Timer และ Oscillator Start-Up
- มีระบบตรวจสอบสัญญาณนาฬิกา (Fail-Safe Clock Monitor)
- ระบบ Watchdog Timer ที่ใช้แหล่งสัญญาณนาฬิกาแบบ RC oscillator ที่แยกจากส่วนอื่นๆ
- ทำงานที่แรงดันระดับ 3.0 ถึง 3.6 โวลต์
- I/O Pin 4mA Sink สามารถเชื่อมต่อกับสัญญาณ 5VTTL ได้ (5V Tolerant)
- รองรับโหมดการทำงานแบบ Run, Idle และ Sleep modes
- สามารถปรับเปลี่ยนโหมดการทำงานของสัญญาณนาฬิกาได้หลากหลายเพื่อประสิทธิภาพและให้สอดคล้องกับการดูแลจัดการในเรื่องของพลังงาน

2.1.1.2 คุณสมบัติทางด้านสัญญาณอนาลอก (Analog Features)

- โมดูลแปลงสัญญาณ Analog to Digital ความละเอียด 10-bit จำนวน 24 ช่อง และสามารถโปรแกรมเป็น 12Bit ได้ 2 ช่อง ความเร็วในการ Sampling สัญญาณสูงสุด 1.1 MSPS

2.1.2 คุณสมบัติโดยทั่วไปของบอร์ดไมโครคอนโทรลเลอร์เบอร์ dsPIC33

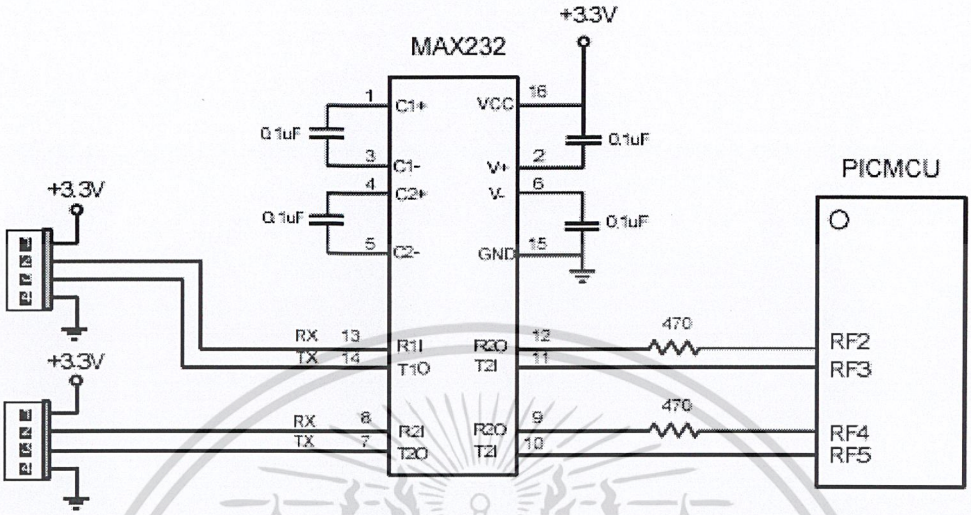
- หน่วยความจำโปรแกรมแบบ Flash Memory ขนาด 128 KByte
- หน่วยความจำข้อมูล SRAM ขนาด 16 K Byte
- I/O Ports ใช้งานจำนวน 69 บิต (รวม Peripheral Function ต่างๆ)
  - โมดูลการสื่อสาร UART จำนวน 2 ช่อง
  - โมดูลการสื่อสารแบบ SPI จำนวน 2 ช่องรองรับทั้ง Master และ Slave Modes
  - โมดูลการสื่อสารแบบ I^2C จำนวน 2 ช่องรองรับทั้ง Master และ Slave Modes
  - โมดูลการสื่อสารแบบ CAN จำนวน 2 ช่อง
  - โมดูล Timer ขนาด 16 บิต จำนวน 9 ช่อง และสามารถจับคู่ใช้งานเป็น Timer ขนาด 32 บิต ได้พร้อมกันจำนวน 4 ช่อง
  - โมดูล Capture , Compare / PWM จำนวน 8 ชุด
  - ระบบฮาร์ดแวร์ RTCC, Real-Time Clock Calendar with Alarms ภายใน
  - โมดูล ADC ขนาด 10 บิต จำนวน 24 ช่อง และสามารถโปรแกรมค่าเป็น 12 บิต ได้ 2 ช่อง
  - ระบบการสื่อสารแบบขนาน DCI(Data Converter Interface) จำนวน 1 ช่อง

2.1.3 คุณสมบัติโดยทั่วไปของบอร์ดไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708

- ใช้ไมโครคอนโทรเลอร์ dsPIC33FJ128GP708 ขนาด 80 PIN
- สัญญาณนาฬิกาคริสตอลออสซิลเลเตอร์ขนาด 8 MHz (สามารถใช้ PLL รันได้ถึง 40 MHz)
- สัญญาณนาฬิกาคริสตอลออสซิลเลเตอร์ขนาด 32.768KHz สำหรับ RTC
- I/O Port ขนาด 10 PIN จำนวน 9 พอร์ต
- ชุดวงจร Line Driver RS232 จำนวน 2 พอร์ต
- พอร์ตสำหรับต่อ LCD (14Pin ET-CLCD) จำนวน 1 พอร์ต
- ขั้วต่อสัญญาณดาวน์โหลดโปรแกรมแบบ ICD2 และ สวิตช์ตัดต่อสัญญาณ Run / Program
- วงจรสวิตช์ Push-Button สำหรับใช้ทดลองอินพุตแบบ Digital จำนวน 4 ช่อง
- วงจรสร้างแรงดัน 0-3.3V จากตัวต้านทานปรับค่าได้ สำหรับทดลองโมดูล A/D จำนวน 1 ช่อง
- พอร์ตสำหรับเชื่อมต่อกับโมดูล Ethernet ENC28J60(ใช้ SPI1)
- พอร์ตเชื่อมต่อกับหน่วยความจำ EEPROM 25LCxxx จำนวน 1 ช่อง(ใช้ SPI2)
- ชุด Regulate แบบ Switching สำหรับแปลงไฟ DC Input ให้เป็น 5V และ 3.3 V
- ขั้วต่อแรงดันไฟ VCC และ GND ใช้ได้กับไฟ 7-12 VDC

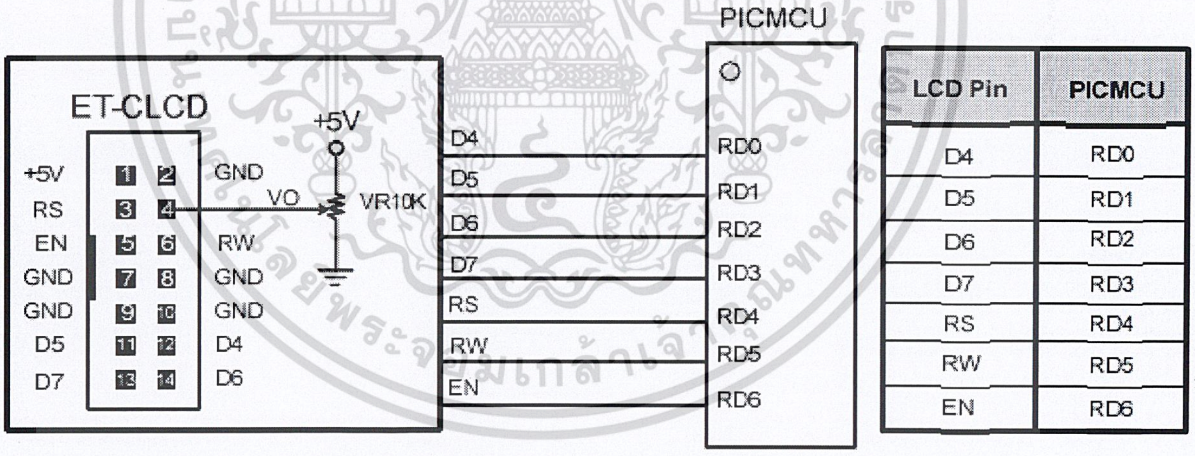
มีรายละเอียดดังต่อไปนี้

- พอร์ตเชื่อมต่อสัญญาณแบบ RS-232 จำนวน 2 พอร์ต มีวงจรการเชื่อมต่อดังต่อไปนี้



รูปที่ 2.1 แสดงวงจรพอร์ตเชื่อมต่อสัญญาณแบบ RS-232

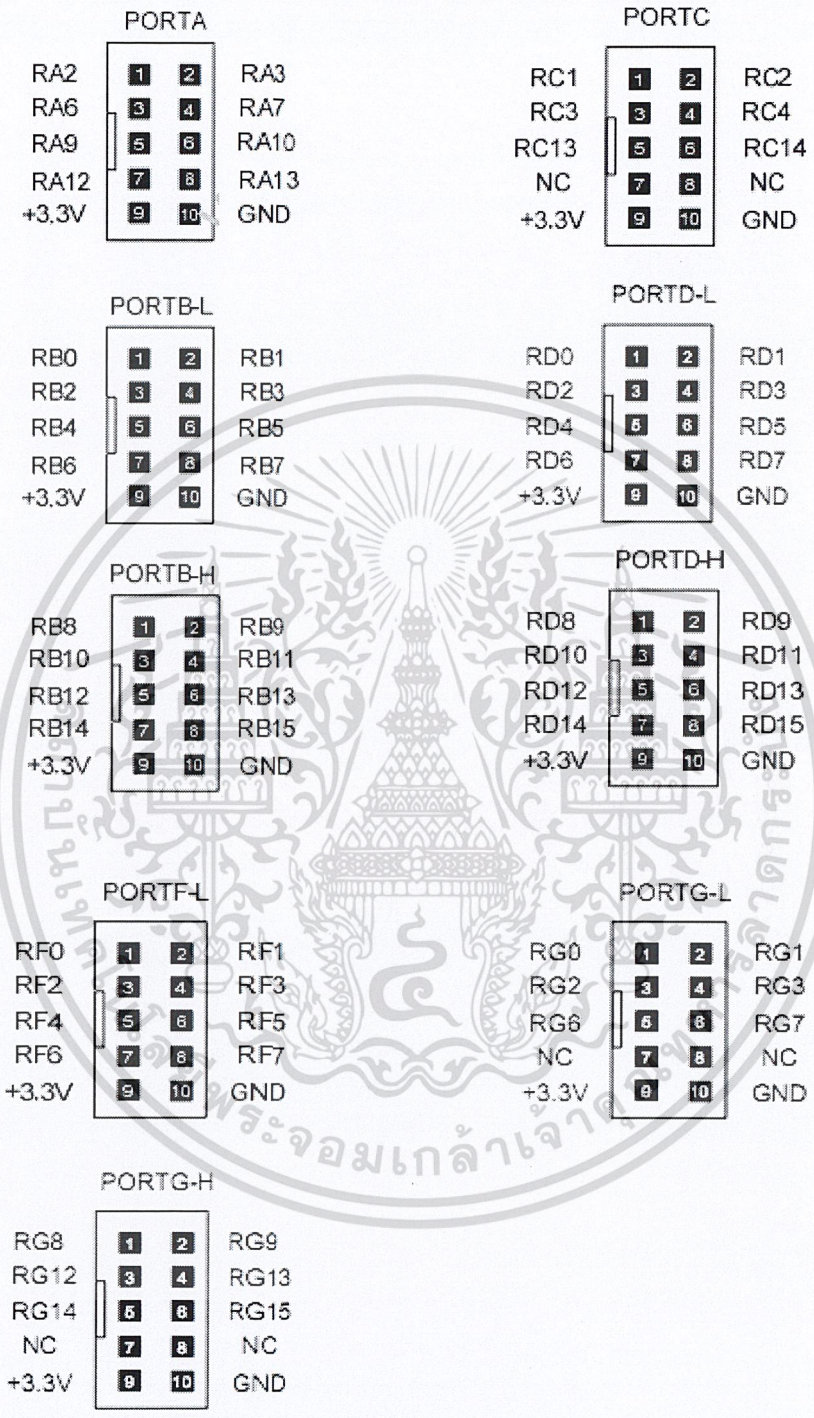
- พอร์ต ET-CLCD สำหรับเชื่อมต่อกับจอแสดงผล LCD แบบตัวอักษร (Character--LCD) โดยมีการจัดวางขาสัญญาณต่างๆ ดังต่อไปนี้



LCD Pin	PICMCU
D4	RD0
D5	RD1
D6	RD2
D7	RD3
RS	RD4
RW	RD5
EN	RD6

รูปที่ 2.2 แสดงพอร์ต ET-CLCD

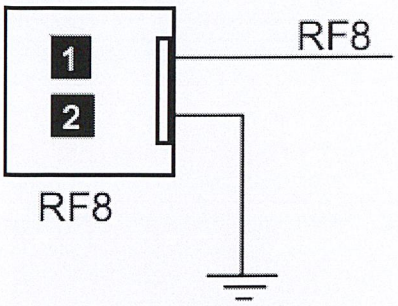
- พอร์ต I/O ของไมโครคอนโทรเลอร์ ที่ถูกออกแบบให้อยู่ในรูปแบบของพอร์ตมาตรฐาน 10-PIN ETT โดยในแต่ละพอร์ตมีการจัดเรียงสัญญาณดังต่อไปนี้



รูปที่ 2.3 แสดงพอร์ต I/O ของไมโครคอนโทรเลอร์

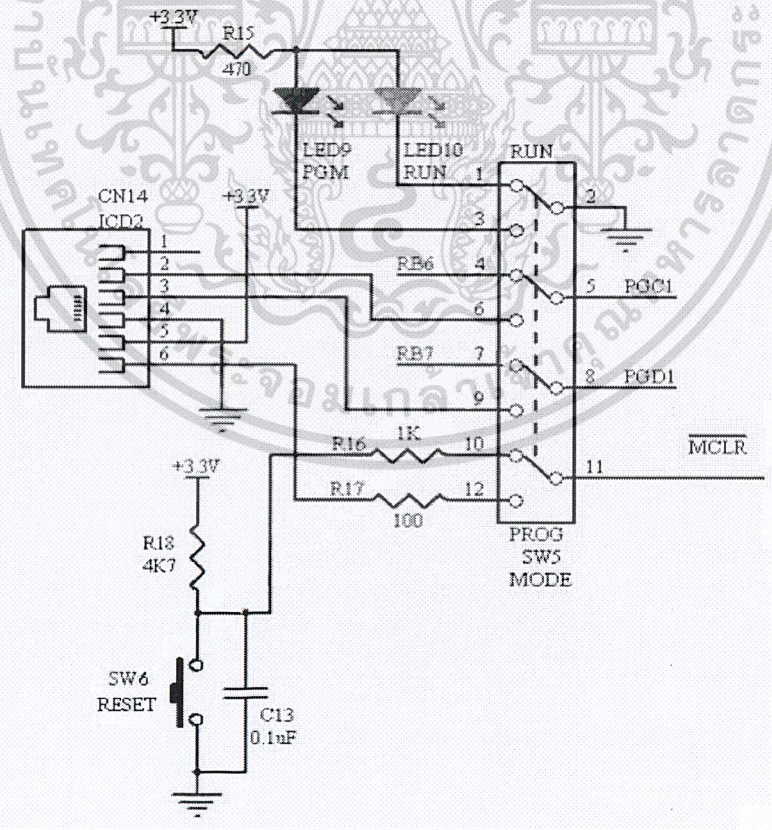
-

- พอร์ตสัญญาณ I/O ขนาด 2 PIN คือ สัญญาณ RF8 และ GND ดังต่อไปนี้



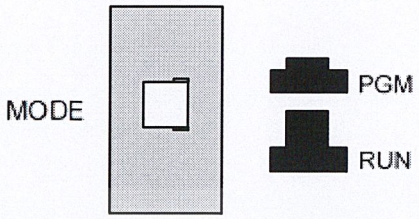
รูปที่ 2.4 แสดงวงจรพอร์ตสัญญาณ I/O ขนาด 2 PIN

- ขั้วต่อสำหรับดาวน์โหลดโปรแกรม เป็นขั้วที่จัดเรียงตามมาตรฐานของ ICD2 รองรับเครื่อง โปรแกรมที่มีการเชื่อมต่อตามมาตรฐานของ ICD2 เช่น PICKit2, ICD2 และ ET-PGMPIC USB โดยก่อนทำการโปรแกรมทุกครั้งต้องกดสวิตช์ MODE ให้มาอยู่ที่ตำแหน่ง PGM ทุกครั้งเพื่อตัดต่อขาสัญญาณมาเข้ากับเครื่องโปรแกรมจากภายนอก



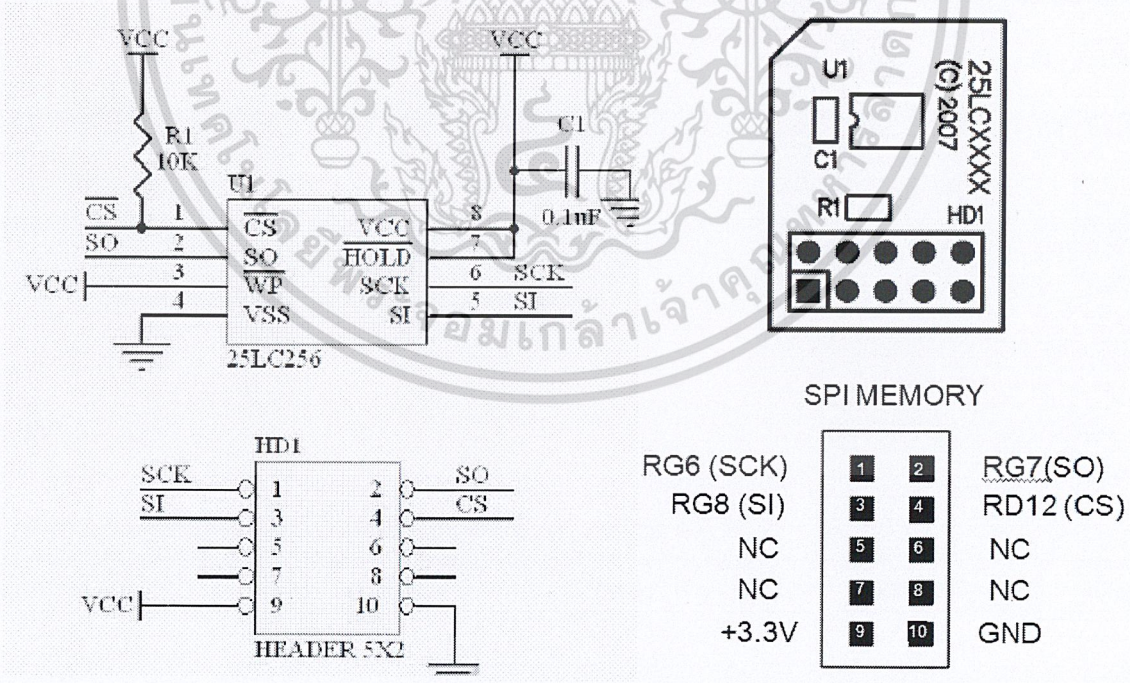
รูปที่ 2.5 แสดงวงจรพอร์ต ICD2

- สวิตช์เลือกโหมด RUN และ PGM สวิตช์ เมื่อกดมาที่ตำแหน่ง PGM จะทำหน้าที่ตัดต่อขาสัญญาณที่ใช้ในการโปรแกรมโค๊ดข้อมูลเข้ากับเครื่องโปรแกรม เพื่อทำการโปรแกรมข้อมูลโปรแกรมที่เราออกแบบ และ เมื่อกดปล่อยกลับมาที่ตำแหน่ง RUN ขาสัญญาณต่างๆ จะกลับมาเป็น I/O ใช้งานได้ตามปรกติ



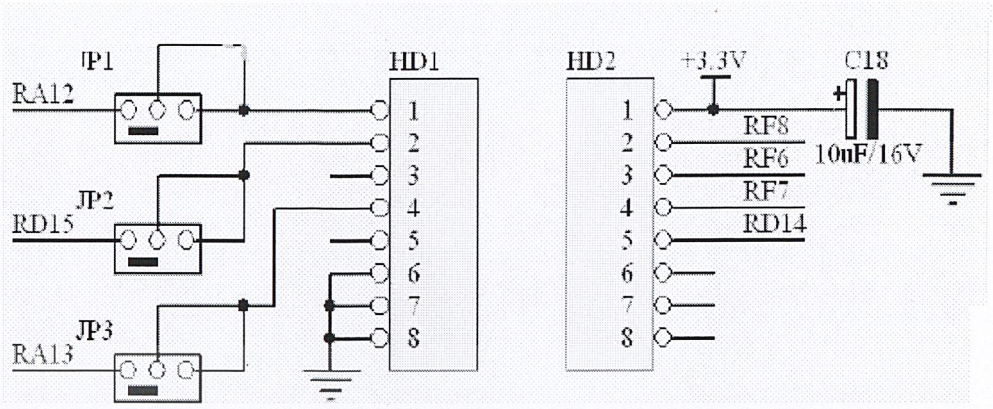
รูปที่ 2.6 แสดงสวิตช์เลือกโหมด RUN และ PGM

- ขั้วต่อ DC-JACK สัญญาณไฟเลี้ยงบอร์ด รองรับแรงดันไฟจากภายนอก 7-12 VDC
- สวิตช์รีเซต (Reset Switch)
- ไอซีไมโครคอนโทรเลอร์ dsPIC33FJ128GP708
- หน่วยความจำ EEPROM เบอร์ 25LCxxx ของ บริษัท Microchip เชื่อมต่อแบบ SPI



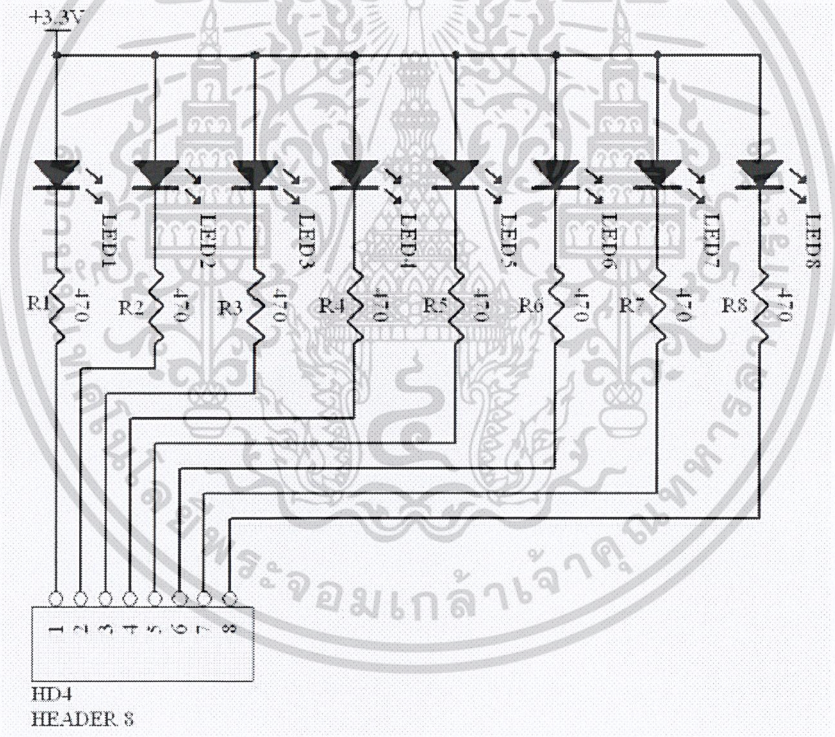
รูปที่ 2.7 แสดงวงจรหน่วยความจำ EEPROM เบอร์ 25LCxxx

- ขั้วสัญญาณเชื่อมต่อกับโมดูลสื่อสาร Ethernet รุ่น ET-MINI ENC28J60



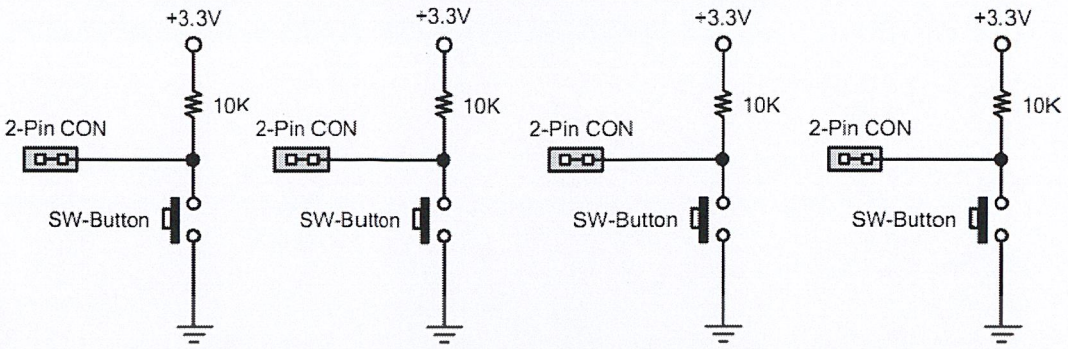
รูปที่ 2.8 แสดงวงจรขั้วสัญญาณเชื่อมต่อกับโมดูล Ethernet รุ่น ET-MINI ENC28J60

- ชุด Test I/O LED ประกอบด้วยหลอดไฟ LED จำนวน 8 ชุด ดังวงจรต่อไปนี้



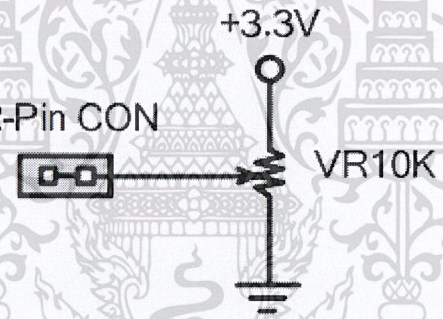
รูปที่ 2.9 แสดงวงจรชุด Test I/O LED ประกอบด้วยหลอดไฟ LED จำนวน 8 ชุด

- ชุดทดลองสัญญาณอินพุท จากสวิตช์ 4 ชุด สามารถสร้างสัญญาณลอจิก 0 (0 โวลต์) และ ลอจิก 1 (+3.3 โวลต์) ดังวงจรต่อไปนี้



รูปที่ 2.10 แสดงวงจรชุดทดลองสัญญาณอินพุท จากสวิตช์ 4 ชุด

- ชุดทดลองแรงดันอนาล็อก 4 ชุด สามารถปรับระดับแรงดันไฟได้ตั้งแต่ 0 – 3.3 โวลต์ โดยมีการต่อวงจรดังต่อไปนี้



รูปที่ 2.11 แสดงวงจรชุดทดลองแรงดันอนาล็อก 4 ชุด

2.1.4 การจัดสรรและใช้งานทรัพยากรของไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708

ตามปรกติแล้วไมโครคอนโทรลเลอร์เบอร์ dsPIC33FJ128GP708 ที่ใช้กับบอร์ด ET-dsPIC33WEB นั้น จะมีขาสัญญาณให้ใช้งานได้โดยอิสระมากถึง 69 I/O แต่จะมีขาสัญญาณบางส่วน ถูกออกแบบและเชื่อมต่อไว้กับอุปกรณ์ I/O เป็นการเฉพาะไว้เรียบร้อยแล้ว ไม่สามารถนำมาใช้เป็น I/O โดยทั่วไปได้ ซึ่งพอสรุปได้ดังนี้

- MCU Oscillator
    - RC12 ใช้เป็น OSC1 ต่อกับ Crystal ค่า 8.00MHz สำหรับใช้เป็นสัญญาณนาฬิกาของ MCU
    - RC15 ใช้เป็น OSC2 ต่อกับ Crystal ค่า 8.00MHz สำหรับใช้เป็นสัญญาณนาฬิกาของ MCU

- RTC Oscillator
    - RC13 ใช้เป็น OSC1 ต่อกับ Crystal ค่า 32.768KHz สำหรับใช้เป็นสัญญาณนาฬิกาของ RTC
    - RC14 ใช้เป็น OSC2 ต่อกับ Crystal ค่า 32.768KHz สำหรับใช้เป็นสัญญาณนาฬิกาของ RTC

- พอร์ตสื่อสารอนุกรม(UART) RS232-CH1
    - RF2 ใช้เป็นขา RXD สำหรับรับข้อมูลจาก RS-232 ช่อง-1
    - RF3 ใช้เป็นขา TXD สำหรับส่งข้อมูลให้ RS-232 ช่อง-1

- พอร์ตสื่อสารอนุกรม(UART) RS232-CH2
    - RF4 ใช้เป็นขา RXD สำหรับรับข้อมูลจาก RS-232 ช่อง-2
    - RF5 ใช้เป็นขา TXD สำหรับส่งข้อมูลให้ RS-232 ช่อง-2

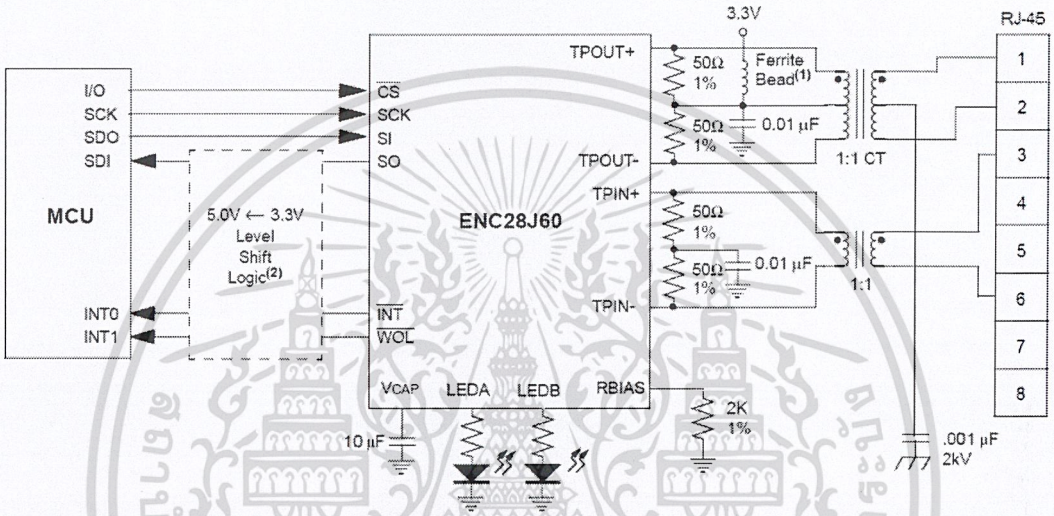
- Ethernet Module (SPI-1)
    - RF6 ใช้เป็น SCK ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60)
    - RF7 ใช้เป็น SDI ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60)
    - RF8 ใช้เป็น SDO ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60)
    - RD14 ใช้เป็น CS ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60)
    - RA12 ใช้เป็น INT1ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60) โดยสามารถเลือกใช้หรือไม่ใช้ได้ โดยการกำหนดที่ Jumper INT(EN/DS) ซึ่งตามปรกติเลือกเป็น DS(Disable:ไม่ใช้งาน)
    - RA13 ใช้เป็น WOLในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60) โดยสามารถเลือกใช้หรือไม่ใช้ได้ โดยการกำหนดที่ Jumper WOL(EN/DS) ซึ่งตามปรกติเลือกเป็น DS(Disable:ไม่ใช้งาน)
    - RD15 ใช้เป็น RST ในการเชื่อมต่อกับ Ethernet Module (ET-MINI ENC28J60) โดยสามารถเลือกใช้หรือไม่ใช้ได้ โดยการกำหนดที่ Jumper RST(EN/DS) ซึ่งตามปรกติเลือกเป็น DS(Disable:ไม่ใช้งาน)

- SPI Memory Module (SPI-2)
    - RG6 ใช้เป็น SCK ในการเชื่อมต่อกับ SPI Memory
    - RG7 ใช้เป็น SDI ในการเชื่อมต่อกับ SPI Memory
    - RG8 ใช้เป็น SDO ในการเชื่อมต่อกับ SPI Memory
    - RD12 ใช้เป็น CS ในการเชื่อมต่อกับ SPI Memory

- Character LCD Display
    - RD0 ใช้เป็น LCD D4 ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD1 ใช้เป็น LCD D5 ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD2 ใช้เป็น LCD D6 ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD3 ใช้เป็น LCD D7 ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD4 ใช้เป็น LCD RS ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD5 ใช้เป็น LCD RW ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต
    - RD6 ใช้เป็น LCD EN ในการเชื่อมต่อกับ Character LCD แบบ 4 บิต

2.1.5 คุณสมบัติของ IC ENC28J60

การเชื่อมต่อกับไมโครคอนโทรลเลอร์ สามารถทำได้โดยง่าย โดยจะใช้การเชื่อมต่อแบบ SPI Bus ซึ่งจะใช้ขาสัญญาณเพียงไม่กี่ขา และ ในส่วนของระบบไฟ เนื่องจาก ENC28J60 เป็นไอซีที่ทำงานที่แรงดัน 3 โวลต์ ดังนั้น เพื่อให้สามารถใช้งานได้กับไมโครคอนโทรลเลอร์ที่ทำงาน 5 โวลต์ จึงไดออกแบบ วงจรบัฟเฟอร์ สำหรับรองรับการเชื่อมต่อระบบไฟ ระหว่าง 3 โวลต์ กับ 5 โวลต์ เอาไวภายในบอร์ด ENC28J60 ซึ่งสามารถเลือกระบบไฟได้โดยการเลือก จัมพ์เปอร์ 5V/3V



รูปที่ 2.12 แสดงวงจรการเชื่อมต่อ ENC28J60 กับไมโครคอนโทลเลอร์

พอร์ต HD1 และ HD2 ออกแบบไว้สำหรับการเชื่อมต่อกับไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708 โดยสามารถเสียบบอร์ด ENC28J60 ซ้อนทับบนไมโครคอนโทรลเลอร์ dsPIC33FJ128GP708 เข้ากับขั้ว Connector ตัวเมียที่จัดเตรียมไว้ให้ได้ทันที ส่วน HD3 ออกแบบไว้สำหรับนำไปใช้เชื่อมต่อกับไมโครคอนโทรเลอร์อื่นๆตามต้องการ

2.2 TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

ชุดโปรโตคอลนี้ได้รับการพัฒนามาตั้งแต่ปี 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

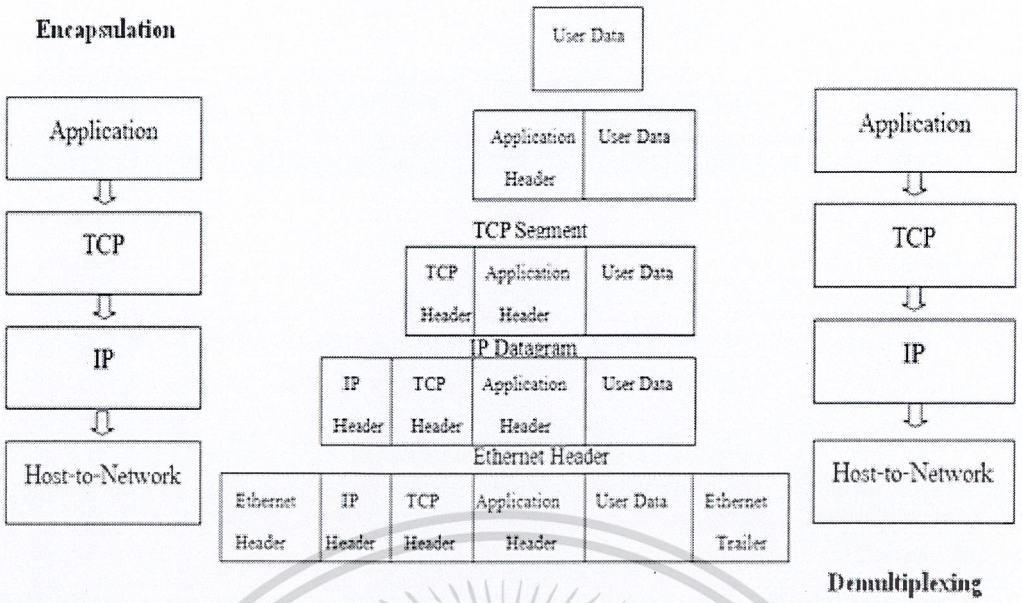
TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน สามประการคือ

1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน

2. ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่นในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ

3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูล และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (Voice) และข้อมูล (data)

การส่งข้อมูลผ่านในแต่ละเลเยอร์จะทำการประกอบข้อมูลที่ได้รับมา กับข้อมูลส่วนควบคุมซึ่งถูกนำมาไว้ในส่วนหัวของข้อมูลเรียกว่า Header ภายใน Header จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการ Encapsulate เมื่อผู้รับได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับ คือ โปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็น Header ก่อนและนำไปประมวลและทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่า Demultiplexing

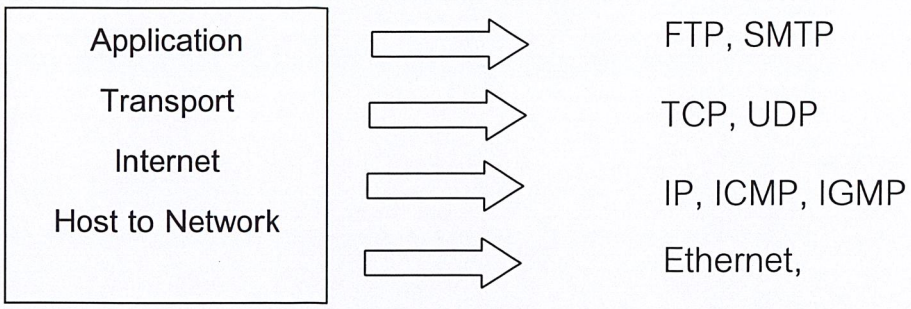


รูปที่ 2.13 ขั้นตอนการ Encapsulation และ Demultiplexing

ข้อมูลที่ผ่านการ Encapsulate ในแต่ละเลเยอร์มีชื่อเรียกแตกต่างกัน ดังนี้

- ข้อมูลจาก User หรือก็คือข้อมูลที่ User เป็นผู้ป้อนให้กับ Application เรียกว่า User Data
- เมื่อแอพพลิเคชั่นได้รับข้อมูลจาก User ก็จะนำมาประกอบกับส่วนหัวของแอพพลิเคชั่น เรียกว่า Application Data และส่งต่อไปยังโปรโตคอล TCP
- เมื่อโปรโตคอล TCP ได้รับ Application Data ก็จะนำมารวมกับ Header ของ โปรโตคอล TCP เรียกว่า TCP Segment และส่งต่อไปยังโปรโตคอล IP
- เมื่อโปรโตคอล IP ได้รับ TCP Segment ก็จะนำมารวมกับ Header ของ โปรโตคอล IP เรียกว่า IP Datagram และส่งต่อไปยังเลเยอร์ Host-to-Network Layer
- ในระดับ Host-to-Network จะนำ IP Datagram มาเพิ่มส่วน Error Correction และ flag เรียกว่า Ethernet Frame ก่อนจะแปลงข้อมูลเป็นสัญญาณไฟฟ้า ส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป

ในแต่ละเลเยอร์ของโครงสร้าง TCP/IP สามารถอธิบายได้ดังนี้



รูปที่ 2.14 ชั้นเลเยอร์ใน TCP/IP

โครงสร้างของ TCP/IP จะมีการทำงานแบ่งเป็นชั้นเลเยอร์ โดยในแต่ละเลเยอร์จะมีโปรโตคอลต่างๆ ที่ช่วยให้สามารถทำงานร่วมกันได้

2.2.1 ชั้นโฮสต์-เครือข่าย (Host-to-Network Layer)

โพรโตคอลสำหรับการควบคุมการสื่อสารในชั้นนี้เป็นสิ่งที่ไม่มีการกำหนดรายละเอียดอย่างเป็นทางการ หน้าที่หลักคือการรับข้อมูลจากชั้นสื่อสาร IP มาแล้วส่งไปยังโหนดที่ระบุไว้ในเส้นทางเดินข้อมูลทางด้านผู้รับก็จะทำงานในทางกลับกัน คือรับข้อมูลจากสายสื่อสารแล้วนำส่งให้กับโปรแกรมในชั้นสื่อสาร

2.2.2 ชั้นสื่อสารอินเทอร์เน็ต (The Internet Layer)

ใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพ็คเกจ (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (Connectionless)

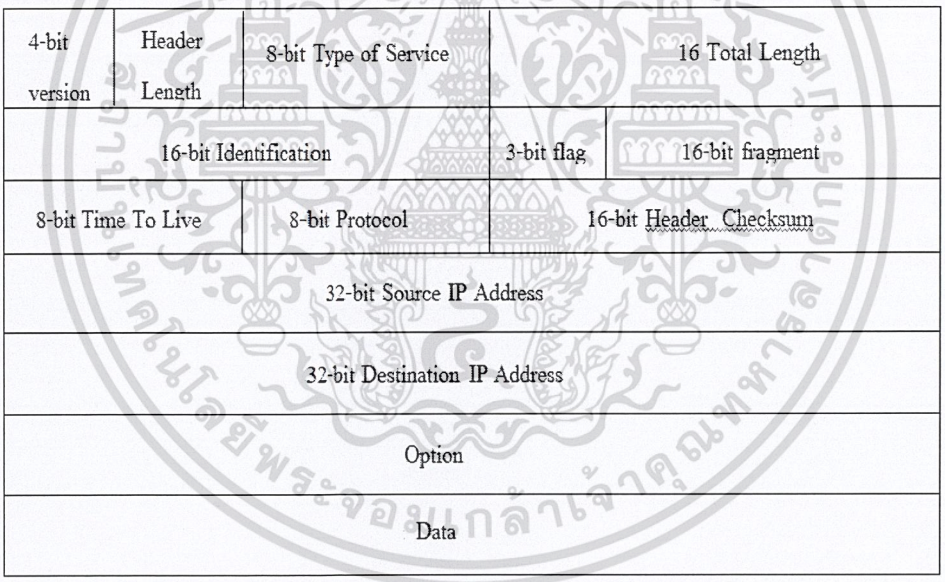
หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่า แพ็คเกจ (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ หากว่ามีการส่งแพ็คเกจออกมาเป็นชุดโดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่ายแพ็กเกจแต่ละตัวในชุดนี้ก็จะเป็นอิสระแก่กันและกัน ดังนั้นแพ็คเกจที่ส่งไปถึงปลายทางอาจจะไม่เป็นไปตามลำดับก็ได้

2.2.2.1 IP (Internet Protocol)

เป็นโปรโตคอลในระดับเน็ตเวิร์คเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็คเกจ ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล

IP ยังมีระบบการแยกและประกอบดาต้าแกรม (datagram) เพื่อรองรับการส่งข้อมูลระดับ data link ที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำ IP ไปใช้บนโปรโตคอลอื่นได้หลากหลาย เช่น Ethernet, Token Ring หรือ Apple Talk

การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบ Connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆ ครั้งของการส่งข้อมูล 1 ดาต้าแกรม โดยจะไม่ทราบถึงข้อมูลดาต้าแกรมที่ส่งก่อนหน้าหรือส่งตามมาแต่การส่งข้อมูลใน 1 ดาต้าแกรม อาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นดาต้าแกรมเดิมเมื่อถึงปลายทาง



รูปที่ 2.15 IP Header

Header ของ IP โดยปกติจะมีขนาด 20 ไบต์ ยกเว้นในกรณีที่มีการเพิ่ม option บางอย่าง ฟิลด์ของ IP Header จะมีความหมายดังนี้

- Version: หมายเลขเวอร์ชันของโปรโตคอล ที่ใช้งานในปัจจุบันคือ เวอร์ชัน 4 (IPv4) และเวอร์ชัน 6 (IPv6)

- Header Length: ความยาวของ Header โดยทั่วไปถ้าไม่มีส่วน option จะมีค่าเป็น 5 (5*32 bit)

- Type of Service (TOS): ใช้เป็นข้อมูลสำหรับเราเตอร์ในการตัดสินใจเลือกการเราต์ข้อมูลในแต่ละดาต้าแกรม แต่ในปัจจุบันไม่ได้มีการนำไปใช้งานแล้ว

- Length: ความยาวทั้งหมดเป็นจำนวนไบต์ของดาต้าแกรม ซึ่งด้วยขนาด 16 บิตของฟิลด์จะหมายถึงความยาวสูงสุดของดาต้าแกรม คือ 65535 ไบต์ (64 Kbyte) แต่ในการส่งข้อมูลจริงข้อมูลจะถูกแยกเป็นส่วนๆ ตามขนาดของ MTU ที่กำหนดในลิงค์เลเยอร์ และนำมารวมกันอีกครั้งเมื่อส่งถึงปลายทาง แอพพลิเคชั่นส่วนใหญ่จะมีขนาดของดาต้าแกรมไม่เกิน 512 ไบต์

- Identification: เป็นหมายเลขของดาต้าแกรมในกรณีที่มีการแยกดาต้าแกรมเมื่อข้อมูลส่งถึงปลายทางจะนำข้อมูลที่มี identification เดียวกันมารวมกัน

- Flag: ใช้ในกรณีที่มีการแยกดาต้าแกรม

- Fragment offset: ใช้ในการกำหนดตำแหน่งข้อมูลในดาต้าแกรมที่มีการแยกส่วน เพื่อให้สามารถนำกลับมาเรียงต่อกันได้อย่างถูกต้อง

- Time to live (TTL): กำหนดจำนวนครั้งที่มากที่สุดที่ดาต้าแกรมจะถูกส่งระหว่าง hop (การส่งผ่านข้อมูลระหว่างเน็ตเวิร์ค) เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 hop จะทำการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และข้อมูลยังไม่ถึงปลายทาง ข้อมูลนั้นจะถูกยกเลิก และเราเตอร์สุดท้ายจะส่งข้อมูล ICMP แจ้งกลับมายังต้นทางว่าเกิด time out ในระหว่างการส่งข้อมูล

- Protocol: ระบุโปรโตคอลที่ส่งในดาต้าแกรม เช่น TCP, UDP หรือ ICMP

- Header checksum: ใช้ในการตรวจสอบความถูกต้องของข้อมูลใน Header

- Source IP address: หมายเลข IP ของผู้ส่งข้อมูล

- Destination IP address: หมายเลข IP ของผู้รับข้อมูล

- Data: ข้อมูลจากโปรโตคอลระดับบน

2.2.2.2 ICMP (Internet Control Message Protocol)

เป็นโปรโตคอลที่ใช้ในการตรวจสอบและรายงานสถานภาพของดาต้าแกรม (Datagram) ในกรณีที่เกิดปัญหากับดาต้าแกรม เช่น เราเตอร์ไม่สามารถส่งดาต้าแกรมไปถึงปลายทางได้ ICMP จะถูกส่งออกไปยังโฮสต้นทางเพื่อรายงานข้อผิดพลาดที่เกิดขึ้น

อย่างไรก็ดี ไม่มีอะไรรับประกันได้ว่า ICMP Message ที่ส่งไปจะถึงผู้รับจริงหรือไม่ หากมีการส่งดาต้าแกรมออกไปแล้วไม่มี ICMP Message ฟ้อง Error กลับมา ก็แปลความหมายได้สองกรณีคือ ข้อมูลถูกส่งไปถึงปลายทางอย่างเรียบร้อย หรืออาจจะมีปัญหา ในการสื่อสารทั้งการส่งดาต้าแกรม และ ICMP Message ที่ส่งกลับมาก็มีปัญหาระหว่างทางก็ได้

ICMP จึงเป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (unreliable) ซึ่งจะเป็นหน้าที่ของโปรโตคอลในระดับสูงกว่า Network Layer ในการจัดการให้การสื่อสารนั้นๆ มีความน่าเชื่อถือใน

ส่วนของ ICMP Message จะประกอบด้วย Type ขนาด 8 บิต Checksum ขนาด 16 บิต และส่วนของ Content ซึ่งจะมีขนาดแตกต่างกันไปตาม Type และ Code ดังรูป 2.16

8-bit Type	8-bit Code	16-bit Checksum
ICMP Content		

รูปที่ 2.16 ICMP Header

2.2.3 ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

แบ่งเป็นโปรโตคอล 2 ชนิดตามลักษณะ โดยลักษณะแรกเรียกว่า Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบ Byte stream ที่ไว้ใจได้โดยไม่มีข้อผิดพลาด

ข้อมูลที่มีปริมาณมากจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า Message ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต ทางฝ่ายผู้รับจะนำ message มาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิม TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่ง ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย

โปรโตคอลการนำส่งข้อมูลแบบที่สองเรียกว่า UDP (User Datagram Protocol) เป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) มีการตรวจสอบความถูกต้องของข้อมูลแต่จะไม่มีการแจ้งกลับไปยังผู้ส่ง จึงถือได้ว่าไม่มีการตรวจสอบความถูกต้องของข้อมูล อย่างไรก็ตามวิธีการนี้มีข้อดีในด้านความรวดเร็วการส่งข้อมูลจึงนิยมใช้ในระบบผู้ให้และผู้ใช้บริการ (client/server system) ซึ่งมีการสื่อสารแบบถาม/ตอบ (request/reply) นอกจากนั้นยังใช้ในการส่งข้อมูลประเภทภาพเคลื่อนไหวหรือการส่งเสียง (voice) ทางอินเทอร์เน็ต

2.2.3.1 UDP (User Datagram Protocol)

เป็นโปรโตคอลที่อยู่ใน Transport Layer เมื่อเทียบกับโมเดล OSI โดยการส่งข้อมูลของ UDP นั้นจะเป็นการส่งครั้งละ 1 ชุดข้อมูล เรียกว่า UDP datagram ซึ่งจะไม่มีความสัมพันธ์กันระหว่างดาต้าแกรมและจะไม่มีกลไกการตรวจสอบความสำเร็จในการรับส่งข้อมูล

กลไกการตรวจสอบโดย Checksum ของ UDP นั้นเพื่อเป็นการป้องกันข้อมูลที่อาจจะถูกแก้ไข หรือมีความผิดพลาดระหว่างการส่ง และหากเกิดเหตุการณ์ดังกล่าว ปลายทางจะได้รู้ว่ามีข้อผิดพลาดเกิดขึ้น แต่มันจะเป็นการตรวจสอบเพียงฝ่ายเดียวเท่านั้น โดยในข้อกำหนดของ UDP หากพบว่า Checksum Error ก็ให้ผู้รับปลายทางทำการทิ้งข้อมูลนั้น แต่จะไม่มีการแจ้งกลับไปยังผู้ส่งแต่อย่างใด

การรับส่งข้อมูลแต่ละครั้งหากเกิดข้อผิดพลาดในระดับ IP เช่น ส่งไม่ถึง, หมดเวลา ผู้ส่งจะได้รับ Error Message จากระดับ IP เป็น ICMP Error Message เมื่อข้อมูลส่งถึงปลายทางถูกต้องแต่เกิดข้อผิดพลาด ในส่วนของ UDP เองจะไม่มีการยืนยันหรือแจ้งให้ผู้ส่งทราบแต่อย่างใด

16-bit Source Port	16-bit Destination Port
Length	Checksum
Data	

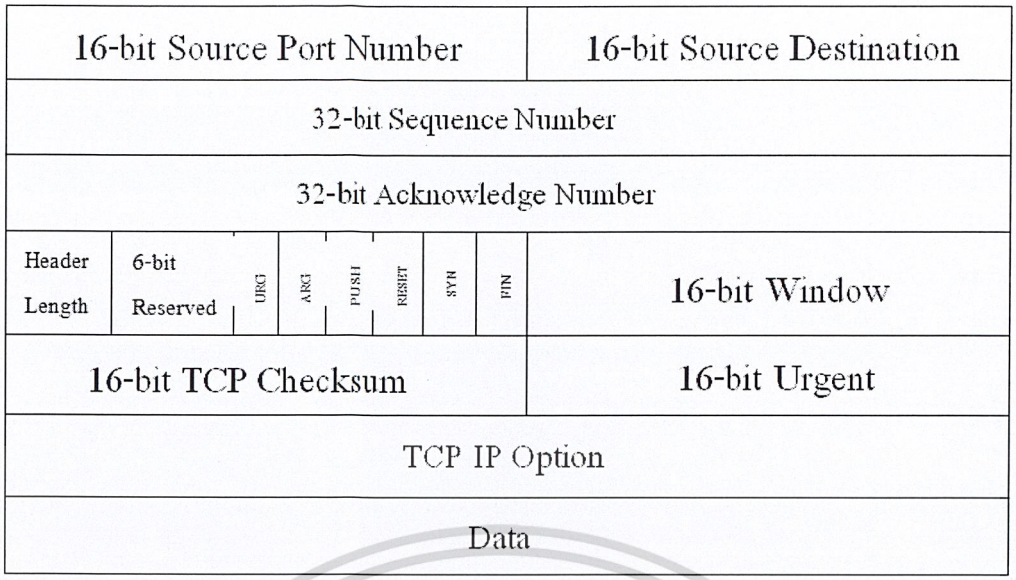
รูปที่ 2.17 UDP Header

จากรูปที่ 2.17 มีรายละเอียด ดังนี้

- Source Port Number : หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้
- Destination Port Number : หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม
- UDP Length : ความยาวของดาต้าแกรม ทั้งส่วน Header และ data นั่นหมายความว่า ค่าที่น้อยที่สุดในฟิลด์นี้คือ 8 ซึ่งเป็นขนาดของ Header
- Checksum : เป็นตัวตรวจสอบความถูกต้องของ UDP datagram และจะนำข้อมูลบางส่วนใน IP Header มาคำนวณด้วย

2.2.3.2 TCP (Transmission Control Protocol)

อยู่ใน Transport Layer เช่นเดียวกับ UDP ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูล ซึ่งมีความสามารถและรายละเอียดมากกว่า UDP โดยดาต้าแกรมของ TCP จะมีความสัมพันธ์ต่อเนื่องกัน และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (reliable) และมีการสื่อสารอย่างเป็นกระบวนการ (connection oriented)



รูปที่ 2.18 TCP Header

จากรูปที่ 2.18 มีรายละเอียด ดังนี้

- Source Port Number : หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้
- Destination Port Number : หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม
- Sequence Number : ฟิลด์ที่ระบุหมายเลขลำดับอ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกแยะว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง
- Acknowledgment Number : ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับ
- Header Length : โดยปกติความยาวของเฮดเดอร์ TCP จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ option แต่ต้องไม่เกิน 60 ไบต์
- Flag : เป็นข้อมูลระดับบิตที่อยู่ในเฮดเดอร์ TCP โดยใช้เป็นตัวบอกคุณสมบัติของแพ็คเกจ TCP ขณะนั้นๆ และใช้เป็นตัวควบคุมจังหวะการรับส่งข้อมูลด้วย ซึ่ง Flag มีอยู่ทั้งหมด 6 บิต แบ่งได้ดังนี้

ตารางที่ 2.1 ประเภทของ flag

TYPE	**Description**
URG	ใช้บอกความหมายว่าเป็นข้อมูลด่วนและมีข้อมูลพิเศษมาด้วย
ACK	แสดงว่าข้อมูลในฟิลด์Acknowledge Number นำมาใช้งานได้
DSH	เป็นการแจ้งให้ผู้รับข้อมูลทราบว่าควรจะส่งข้อมูลSegmentนี้ไปยัง Application ที่กำลังรออยู่โดยเร็ว
RST	ยกเลิกการติดต่อ เนื่องจากกรณีที่เกิดการสับสนขึ้นด้วยเหตุผลต่างๆ
SYN	ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง

Flag ใน Header ของ TCP มีความสำคัญในการกำหนดการทำงานของ TCP segment เนื่องจากข้อมูลใน Header ของTCP จะมีข้อมูลครบถ้วนทั้งการรับและการส่งข้อมูล ซึ่งในการทำงานแต่ละอย่างจะมีการใช้งานฟิลด์ไม่เหมือนกัน flag จะเป็นตัวกำหนดว่าให้ใช้งานฟิลด์ไหน เช่น ฟิลด์ Acknowledgment number จะไม่ถูกใช้ในขั้นตอนการเริ่มต้นการเชื่อมต่อแต่จะมีข้อมูลในฟิลด์ ซึ่งเป็นข้อมูลที่ไม่มีความหมายใดๆ ซึ่งถ้าไม่มี flag เป็นตัวกำหนดก็อาจจะมีการนำข้อมูลมาใช้และก่อให้เกิดความผิดพลาดได้

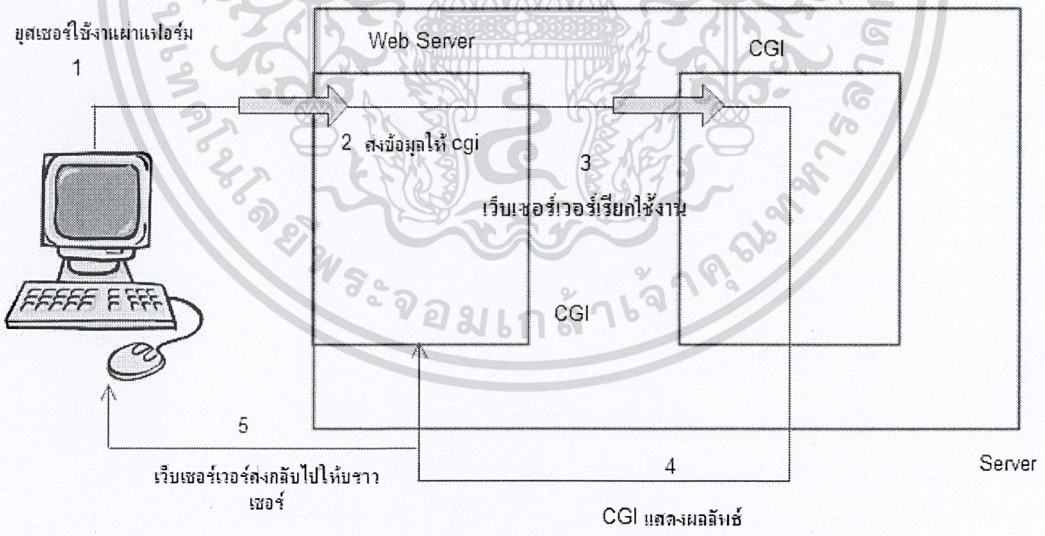
2.2.4 ชั้นสื่อสารการประยุกต์ (Application Layer)

มีโปรโตคอลสำหรับสร้างจอเทอร์มินัลเสมือน เรียกว่า TELNET และโปรโตคอลสำหรับการจัดการแฟ้มข้อมูล เรียกว่า FTP และโปรโตคอลสำหรับการให้บริการจดหมายอิเล็กทรอนิกส์ เรียกว่า SMTP

โดยโปรโตคอลสำหรับสร้างจอเทอร์มินัลเสมือนช่วยให้ผู้ใช้สามารถติดต่อกับเครื่องโฮสต์ที่อยู่ไกลออกไปโดยผ่านอินเทอร์เน็ต และสามารถทำงานได้เสมือนกับว่ากำลังนั่งทำงานอยู่ที่เครื่องโฮสต์นั้น โปรโตคอลสำหรับการจัดการแฟ้มข้อมูลช่วยในการคัดลอกแฟ้มข้อมูลมาจากเครื่องอื่นที่อยู่ในระบบเครือข่ายหรือส่งสำเนาแฟ้มข้อมูลไปยังเครื่องใดๆ ก็ได้ โปรโตคอลสำหรับให้บริการจดหมายอิเล็กทรอนิกส์ช่วยในการจัดส่งข้อความไปยังผู้ใช้ในระบบ หรือรับข้อความที่มีผู้ส่งเข้ามา

2.3 Common Gateway Interface (CGI)

เมื่อเกิดระบบเครือข่ายเวิลด์ไวด์เว็บใช้งานจนเป็นที่นิยมดังปัจจุบัน หลายๆไซต์ (Web Site) เริ่มต้องการนำเสนอข้อมูลภายในองค์กร ที่เคยใช้งานกับโปรแกรมประยุกต์ของตนภายในองค์กรผ่านเว็บเพจ หรือ โฮมเพจ (Homepage) ของตน จึงเกิดปัญหาว่าจะสามารถทำอย่างไร ทั้งนี้เพราะทั้งสองแอพพลิเคชั่น อยู่คนละส่วนกัน และวิธีทำงานก็แตกต่างกันอย่างสิ้นเชิง ทางออกคือการพัฒนาแอพพลิเคชั่น ในลักษณะเหมือนโปรแกรมประยุกต์ที่องค์กรใช้งานอยู่ โดยอาศัยหลักการของ CGI ในการพัฒนา แต่นี้ยังเป็นเพียงแค่จุดเริ่มต้น ของความต้องการเท่านั้น เพราะปัจจุบันเราจะเห็นว่า มีแอพพลิเคชั่นหลากหลายรูปแบบบนระบบเว็บ เช่น การให้บริการส่งเพจ (Page), การให้บริการค้นหางาน, การให้บริการความช่วยเหลือแบบออนไลน์ เป็นต้น ซึ่งแอพพลิเคชั่นเหล่านี้เกิดจากความต้องการที่หลากหลาย และต่างความคิด รวมไปถึงวิสัยทัศน์ของแต่ละคนในการที่คิดประยุกต์ และร่วมสร้างกิจกรรมต่างๆ บนระบบเว็บ จนทำให้การใช้งานบนระบบเว็บนี้กลายเป็นส่วนสำคัญหลักของเครือข่ายอินเตอร์เน็ตในปัจจุบันไปเสียแล้วและด้วยความสามารถของหลักการ CGI นี้เองทำให้หลายๆองค์กรต้องนำมาประยุกต์ใช้ในองค์กรจนเกิดคำที่ว่า “แอพพลิเคชั่นในอนาคต คือ แอพพลิเคชั่นที่ใช้งานผ่านบราวเซอร์ หรือใช้งานภายใต้พื้นฐานเว็บ (Web – based หรือเรียกว่า Web – based Application)”



รูปที่ 2.19 แสดงระบบการทำงานของ CGI

จากรูปที่ 2.19 เราจะมาให้ความหมายว่าอะไรคือ CGI จริงๆแล้ว CGI ก็คือหลักการหรือวิธีการของการการพัฒนาแอพพลิเคชั่น ที่ทำหน้าที่เสมือนประตู (Gateway) เชื่อมโยงการติดต่อกับการทำงานอื่นๆ เพื่อให้เกิดการทำงานที่หลากหลายในการใช้งาน โดยอาศัยพื้นฐานของระบบเว็บหรือจะกล่าวได้ว่าทำงานควบคู่กับเว็บเซิร์ฟเวอร์ เพราะบราวเซอร์ไม่สามารถติดต่อส่วนอื่นๆ โดยตรงได้ เช่น จะติดต่อกับฐานข้อมูล เป็นต้น จำเป็นต้องติดต่อผ่านเว็บเซิร์ฟเวอร์ ไปยังส่วนของ CGI โดยเรามักเรียกว่า “CGI โปรแกรม” หรือ “CGI แอพพลิเคชั่น” หรือ “เว็บแอพพลิเคชั่น” ก็ได้ ด้วยเหตุนี้เองเราจึงเห็นว่าจริงๆแล้ว CGI แอพพลิเคชั่น หรือ แอพพลิเคชั่น ที่พัฒนาตามแนวทาง CGI เป็นแอพพลิเคชั่นประเภท เซิร์ฟเวอร์ แอพพลิเคชั่น (Sever Application) หรือแอพพลิเคชั่นที่ทำงานอยู่ที่ฝั่งเซิร์ฟเวอร์โดยมี ส่วนที่ทำหน้าที่ติดต่อกับผู้ขอใช้บริการ หรือไคลเอ็นต์ (Client) คือเว็บเซิร์ฟเวอร์ และไคลเอ็นต์ใช้งานผ่านเว็บบราวเซอร์ ข้อดีของเซิร์ฟเวอร์แอพพลิเคชั่นที่เห็นได้ชัดคือ การปรับปรุงหรือเปลี่ยนเวอร์ชั่นจะทำได้ง่ายโดยไม่ต้องแจกจ่ายให้ผู้ใช้งานทุกครั้งแต่สามารถดูแลปรับปรุงได้ที่เซิร์ฟเวอร์โดยตรงพอมีวิธีการของ CGI เกิดขึ้นปัจจุบันเราจึงได้เห็นรูปแบบของโฮมเพจที่เปลี่ยนไปจากเดิมที่เคยเป็นแค่ “Static Hypermedia Document” คือเอกสารที่แสดงโดยไม่มีการเปลี่ยนแปลงไปเป็นเอกสาร ที่สามารถเปลี่ยนแปลงรูปแบบได้ ตลอดจนเห็นเป็นโฮมเพจ ที่สามารถโต้ตอบ หรือ เป็น อินเตอร์แอกทีฟ (Interactive) เหมือนส่วนของอินเตอร์เฟส (Interface) ของ CGI แอพพลิเคชั่นที่แปรเปลี่ยนตลอดเหมือนกับการใช้งานโปรแกรมประยุกต์นั้นเอง

2.3.1 การทำงานของ CGI

การทำงานของ CGI ก็ยังคงอาศัยหลักการพื้นฐานของ ไคลเอ็นต์-เซิร์ฟเวอร์ โดยเว็บเซิร์ฟเวอร์จะเป็นผู้ติดต่อขอใช้บริการและรอรับผลลัพธ์ของ CGI กลับมา แล้วส่งต่อให้กับยูสเซอร์ที่ใช้งานผ่านบราวเซอร์ที่ไคลเอ็นต์ลองศึกษาตัวอย่างต่อไปนี้ จะทำให้เห็นภาพการทำงานทั้งระบบได้อย่างชัดเจนยิ่งขึ้น

- สมมุติยูสเซอร์ กำหนด URL สำหรับเรียกใช้ CGI โปรแกรม เป็น http://dream.world.net/cgibin/helloworld.cgi เมื่อเว็บเซิร์ฟเวอร์รับรายละเอียดมาตรวจสอบพบว่ามีการเรียก CGI หรือ Script Alias ที่กำหนดในคอนฟิกไฟล์ ก็จะทราบทันทีว่าจะต้องอ่านโปรแกรมหรือสคริปต์ที่ไดเรกทอรี่ใด ตามที่กำหนดในคอนฟิก ซึ่งไฟล์ที่ใช้งานนั้น คือไฟล์ CGI โปรแกรม
- หรือไม่เช่นนั้นอาจทำการตรวจสอบชนิดของไฟล์ ก็พบว่านามสกุลดังกล่าวเป็น “CGI” และพบในรายละเอียด คอนฟิกว่าเป็น CGI แอปพลิเคชัน แสดงว่า การเรียกหรือขอใช้บริการครั้งนี้เป็นการเรียกรัน CGI แอปพลิเคชัน เว็บเซิร์ฟจะทำการเรียก CGI ให้ทำงานและจะรอรับผลลัพธ์ของการทำงานด้วย

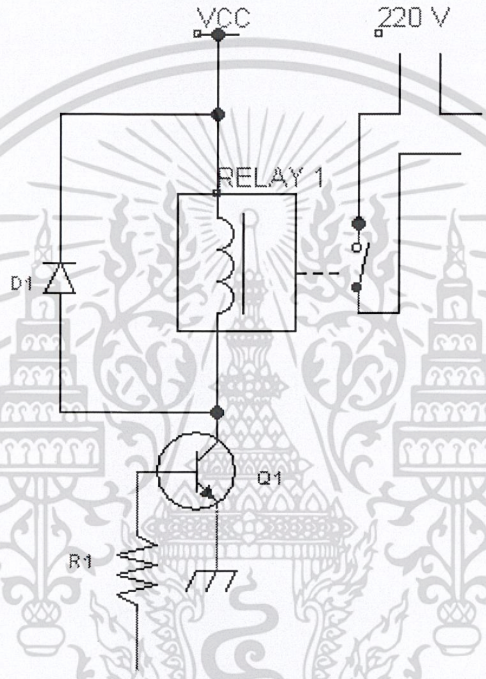
- เมื่อโปรแกรมหรือ CGI สคริปต์ ทำงานเสร็จสิ้นก็จะส่งผลลัพธ์กลับมาให้เว็บเซิร์ฟเวอร์ จากนั้นเว็บเซิร์ฟเวอร์ก็จะทำการรับข้อมูลส่งกลับไปให้ไคลเอ็นต์ การตอบหรือส่งผลลัพธ์ของ CGI ให้กับเว็บเซิร์ฟเวอร์เพื่อคืนให้ไคลเอ็นต์นั้น จะมีรูปแบบของการส่ง โดยทุกครั้งที่มีการส่งกลับจะมีการระบุส่วนหัว (Header) ในการติดต่อสื่อสาร ว่าข้อมูลหรือรูปแบบข้อมูลที่ส่งตามมานั้นจะเป็นอย่างไร

2.3.2 CGI กับการพัฒนาแอพพลิเคชั่นใน Internet, Intranet และ Extranet

สำหรับวิธีการของ CGI ที่เรานำมาพัฒนาแอพพลิเคชั่นนั้น เราทราบอยู่แล้วว่าเป็นแอพพลิเคชั่นที่ใช้งานภายใต้ระบบเครือข่ายเวิลด์ไวด์เว็บ และแน่นอนว่าเครือข่ายเว็บเป็นส่วนหนึ่งของการอาศัยโครงสร้างของเครือข่ายอินเตอร์เน็ตเป็นพื้นฐานในการใช้งาน แต่ไม่ใช่ว่าทุกแอพพลิเคชั่นที่ประยุกต์ใช้งานบนเครือข่ายอินเตอร์เน็ต จะเป็นการทำงานภายใต้ระบบเว็บเพียงแต่ว่าระบบเว็บจะได้รับความนิยมมากกว่าเพราะความสามารถของตัวบราวเซอร์โปรแกรมและอีกหลายๆเหตุผลที่สนับสนุน ดังนั้นไซต์ต่างๆจึงพยายามพัฒนา CGI ในการติดตั้งกับแอพพลิเคชั่นอื่นๆให้สามารถใช้งานผ่านระบบเว็บได้ และจุดนี้เองคือจุดเริ่มต้นสำหรับที่มาของ CGI แต่ในปัจจุบันวิธีการนี้ถูกนำไปพัฒนาแอพพลิเคชั่นอื่นๆ จนแอพพลิเคชั่นหลายๆแบบ ไม่ว่าจะเป็นระบบจัดการงานขององค์กร, การบริหารองค์กร และอื่นๆต่างก็ถูกพัฒนาบนระบบเว็บไปเสียทุกแขนงจนเกิดการใช้งานในรูปแบบต่างๆมากมายดังที่เห็นในปัจจุบัน ไม่ว่าจะเป็นการทำการประชุมทางไกล, การทำกระดานข่าว, การทำระบบจัดการเอกสาร, การติดต่อสื่อสารในรูปแบบต่างๆเป็นต้น ทั้งนี้ก็เพราะ อาศัยข้อดีของระบบเครือข่ายในการช่วย เพิ่มประสิทธิภาพและเพื่อลดค่าใช้จ่ายนั้นเองและด้วยเหตุผลที่เห็นภาพๆได้ชัดเจนจากเครือข่ายอินเตอร์เน็ตนี่เองที่ทำให้หลายๆองค์กร พยายามนำมาประยุกต์ใช้งานในองค์กรตนเองบ้างจึงเกิดระบบอินทราเน็ต (Intranet) ซึ่งระบบอินทราเน็ตนี้เกิดขึ้นโดยองค์กรเหล่านี้ นำระบบเครือข่ายอินเตอร์เน็ตมาประยุกต์เพื่อใช้งาน และพัฒนาแอพพลิเคชั่นใช้งานจนแอพพลิเคชั่นแบบเดิมหลายๆอย่างเริ่มโยกย้ายเข้าสู่การใช้งานในระบบอินทราเน็ตนี้ด้วย บวกกับยังมีระบบแอพพลิเคชั่นแบบเดิมที่เป็นพื้นฐานเดิมอยู่แล้วของอินเตอร์เน็ตอีกเช่น ระบบอีเมล์, ระบบเว็บบอร์ดหรือกระดานข่าวสาร เป็นต้น ทำให้อินทราเน็ตยิ่งเป็นทางออก สำหรับการจัดองค์กรด้านไอทีที่เปลี่ยนแปลงเร็วอีกทั้งอินทราเน็ตมีทิศทางและแนวโน้มที่ชัดเจน และดีกว่าในทุกด้าน จากแนวทางเดิม

2.4 รีเลย์ (Relay)

เป็นอุปกรณ์ที่ใช้กระแสต่ำๆ เพื่อควบคุมสวิตช์ให้ตัดต่อโหลดที่มีกระแสสูงๆ โครงสร้างส่วนประกอบของรีเลย์ การทำงานของรีเลย์คือ เมื่อมีแรงดันกดคร่อมขดลวดจะทำให้เกิดกระแสไหลผ่านขดลวด ซึ่งจะทำให้หน้าสัมผัสเคลื่อนที่ (moving contact) และเกิดสนามแม่เหล็กไฟฟ้าดูดหน้าสัมผัส NO (ปกติเปิดวงจร : normally open) ให้ต่อวงจร และเมื่อปลดแรงดันออก สนามแม่เหล็กก็จะหมดลงหน้าสัมผัสเคลื่อนที่ก็จะดีดกลับมาต่อยังหน้าสัมผัส NC (ปกติต่อวงจร : normally close)



รูปที่ 2.20 วงจรรีเลย์

2.5 ระบบปฏิบัติการ Windows Mobile

2.5.1 Windows mobile

Windows mobile คือ ระบบปฏิบัติการที่ประกอบด้วยชุดแอปพลิเคชั่นพื้นฐาน สำหรับอุปกรณ์เคลื่อนที่บน Microsoft Win32 API อุปกรณ์ที่ใช้ระบบWindows mobile มี Pocket PC, PDA Phone และ Smart phone ซึ่งเป็นอุปกรณ์เคลื่อนที่ที่ทำงานอัตโนมัติ โดยถูกออกแบบให้มีระบบปฏิบัติการคล้ายวินโดวส์บนเครื่อง PC ทั่วไป เช่น รูปแบบ จุดเด่น เป็นต้น ต้นกำเนิดของระบบปฏิบัติการ Windows mobile คือ ระบบปฏิบัติการ Pocket PC 2000

2.5.2 Windows mobile 6 Professional

Windows Mobile 6: WM6 กำเนิดภายใต้ชื่อ Crossbow เข้ามาเมื่อวันที่ 12 กุมภาพันธ์ 2007 ที่งาน 3GSM Word Congress 2007 ใน Windows mobile 6 ได้พัฒนาเป็น Windows mobile 6.1, Windows mobile 6.5 และ Windows mobile 6.5.3 ซึ่งมีการพัฒนาฟังก์ชันการทำงานในบางส่วน โดยที่ WM6 ต้องการหน่วยความจำเท่ากับ WM5 คือ Flash ROM ขนาด 32 MB และ RAM ขนาดเท่ากัน ส่วนซีพียูจะต้องมีความเร็วไม่ต่ำกว่า 200 MHz อุปกรณ์ที่จะมีขายมักมีหน่วยความจำแต่ละแบบไม่ต่ำกว่า 64 MB และมีซีพียูความเร็ว 400 MHz ซึ่ง Windows Mobile 6 ถูกสร้างขึ้นมาให้คล้ายกับระบบปฏิบัติการ Windows Vista

HPC ที่ใช้งานกับ WM6 ได้มี 3 แบบคือ พ็อกเก็ตพีซี พีดีเอโฟน และสมาร์ทโฟน

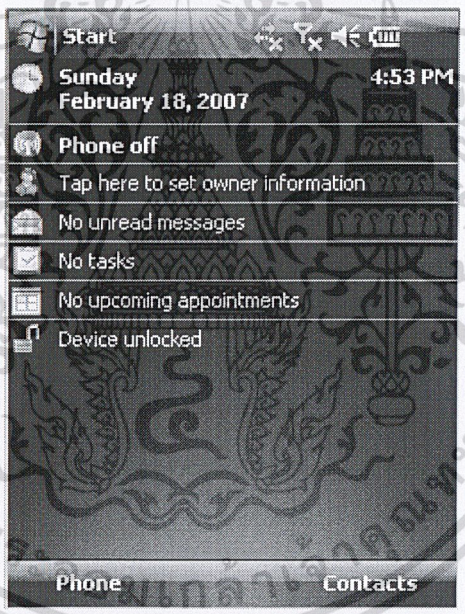
- **Pocket PC** คือ คอมพิวเตอร์ขนาดกระเป๋า (เสื้อ) ธรรมดาที่ไม่มีโทรศัพท์ในตัว มักเรียกว่า PDA หรือ Personal Digital Assistance มีจุดเด่น คือ หน้าจอใหญ่และเป็นระบบสัมผัสใช้ Stylus ซึ่งทำหน้าที่เป็นทั้งปากกาและเมาส์ ช่วยให้ผู้ใช้ป้อนข้อมูลได้สะดวกเหมือนใช้ปากกาจดบันทึกในกระดาษ
- **PDA Phone** บางที่เรียกว่า “พ็อกเก็ตพีซีโฟน” หมายถึง Pocket PC ที่มีโทรศัพท์ในตัว รูปร่างหน้าตาและคุณสมบัติเหมือน Pocket PC ทุกอย่าง แต่จะมีคุณสมบัติด้านโทรศัพท์เพิ่มขึ้น ขนาดเครื่องจะหนากว่า และมักมีเสาอากาศสั้น
- **SmartPhone** หมายถึง โทรศัพท์เคลื่อนที่ซึ่งมีคุณสมบัติบางอย่างคล้าย Pocket PC แต่ไม่มีจอระบบสัมผัส (คือใช้ Stylus ไม่ได้) การป้อนข้อมูลต้องใช้เพียงปุ่มหน้าเครื่อง ขนาดของเครื่องมักจะเล็กกว่าอุปกรณ์สองแบบแรก บางรุ่นอาจมีแป้นพิมพ์ครบชุด (แป้นพิมพ์แบบ QWERTY) เพื่ออำนวยความสะดวกในการป้อนพิมพ์ข้อความยาวๆ

2.5.2.1 การป้อนข้อมูล

WM6 สนับสนุนการกดปุ่มทางลัดในแป้นพิมพ์ QWERTY และใน WM5 มีคุณสมบัติทำนายข้อความ (Text-prediction หรือ CompIME) ซึ่งเป็นสิ่งอำนวยความสะดวกที่ช่วยให้ป้อนข้อมูลได้รวดเร็วขึ้นเพราะไม่ต้องกดแป้นพิมพ์มากครั้ง คุณสมบัตินี้ได้รับการปรับปรุงขึ้นมากใน WM6 เมื่อกดปุ่มพิมพ์ข้อความเพียงอักษรเดียวจะมีคำปรากฏให้เลือกถึงสี่คำ (ใน WM5 จะแสดงเพียงคำเดียว) ซึ่งคำเหล่านี้ถูกนำมาจากพจนานุกรมและจากคำที่เคยพิมพ์ไปแล้ว นั่นคือ ระบบจะรู้จักคำมากขึ้นเมื่อเราใช้งานไปสักพัก และจะทำนายการป้อนพิมพ์ของเราได้แม่นยำมากขึ้น คุณสมบัติใหม่อีกอย่าง คือ หากต้องการล้างคำที่ระบบจำไว้ก็สามารถทำได้ ซึ่งจะมีประโยชน์เมื่อต้องการเปลี่ยนผู้ใช้อุปกรณ์

2.5.2.2 การใช้งานอินเตอร์เน็ต

การเปิดเว็บด้วยโปรแกรม Internet Explorer สามารถเปิดเว็บได้เพียงหน้าเดียว ไม่มีคุณสมบัติที่ช่วยให้เปิดเว็บได้หลายๆ หน้าพร้อมกันแบบใน FireFox หรือ IE7



รูปที่ 2.21 แสดง หน้าจอของโทรศัพท์ Windows Mobile 6.0

2.5.3 Microsoft ActiveSync

ใช้เชื่อมต่อเครื่อง Pocket PC กับเครื่องคอมพิวเตอร์ ให้สามารถโอน ย้าย ถ่าย ข้อมูลระหว่างกันได้ การติดตั้งโปรแกรมลงในเครื่อง Pocket PC ผ่าน desktop computer (เครื่องคอมพิวเตอร์ อาจเป็น PC หรือ Notebook) จะต้องใช้โปรแกรม Microsoft ActiveSync ในการเชื่อมต่อ

2.5.4 Windows Mobile Application

แอพพลิเคชั่น ระบบ Windows Mobile จะใช้ Microsoft Visual Studio, Visual C# ในการสร้าง ซึ่ง Microsoft Visual C# เหมาะสำหรับนักพัฒนาซอฟต์แวร์ เพราะมีการใช้งานง่ายและสะดวก เนื่องจากมี tools ต่างๆให้เลือกใช้งาน และมีฟังก์ชั่นที่สามารถ แปลง Code ที่เขียนขึ้นให้เป็นไฟล์ที่สามารถติดตั้งลงบนโทรศัพท์มือถือได้ (.CAB file)

2.6 Microsoft Visual Studio 2008, Visual C# 2008

2.6.1 Microsoft Visual Studio 2008

Windows mobile ถูกพัฒนาใน Visual Studio ซึ่งสามารถสร้างแบบจำลองภาพเพื่อให้ผู้พัฒนาระบบได้ทำการทดสอบ ค้นหา Bug และช่วยเขียนแอพพลิเคชั่นขึ้นมาก่อนที่จะนำเข้า Windows Mobile 2003 ซอฟแวร์ถูกพัฒนาโดยใช้ Microsoft's embedded Visual Tools

2.6.2 Microsoft Visual C# 2008

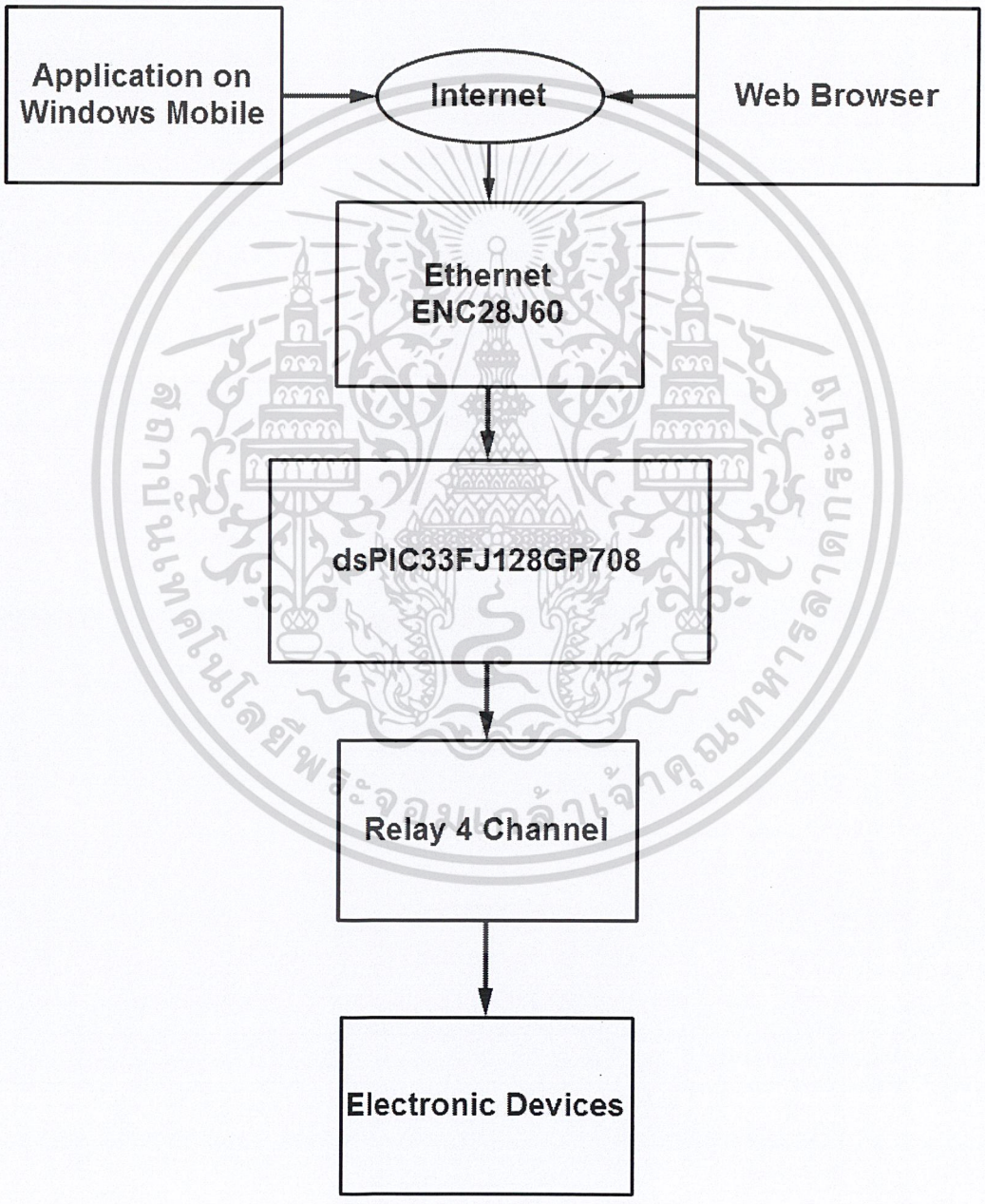
Microsoft Visual Studio .NET (C#) ในการออกแบบเนื่องจาก ภาษา C# เป็นภาษาที่ ถูกออกแบบมา เพื่อรองรับการทำงานในยุค .NET โดยมีแนวของภาษาเป็นแบบการเขียนโปรแกรมเชิงวัตถุสมัยใหม่ (Modern Object Oriented Programming) จุดเด่นสำคัญของภาษา Visual C# คือการรวมเอาความสามารถของภาษา C++ มารวมกับความใช้งานง่ายของภาษา Visual Basic ทำให้ภาษา Visual C# เป็นภาษาคอมพิวเตอร์ที่ความน่าเชื่อถือสูง ทำให้การพัฒนาโปรแกรมทำได้อย่างมีประสิทธิภาพ สะดวก ง่ายดาย รวดเร็ว และการใช้ภาษาคอมพิวเตอร์มาตรฐานกลาง (Common Language Specification) ทำให้สะดวกต่อการศึกษาและพัฒนาต่อได้

ใช้เทคโนโลยี Microsoft Visual Studio 2005 ร่วมกับ Emulator PDA เพื่อทำการแสดงผลออกทางหน้าจอของ Pocket PC ซึ่งในขั้นตอนนี้จะทำให้ได้เห็นผลลัพธ์เหมือนใช้เครื่องพีดีเอจริงทุกประการ

บทที่ 3
การออกแบบโครงงาน

3.1 ภาพรวมของระบบ

การทำงานของระบบทั้งการสั่งการควบคุมจากหน้าเว็บเพจและหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือ จะใช้อินเตอร์เน็ตในการสั่งการและใช้ไฟล์ CGI ในการติดต่อกันดังรูปที่ 3.1

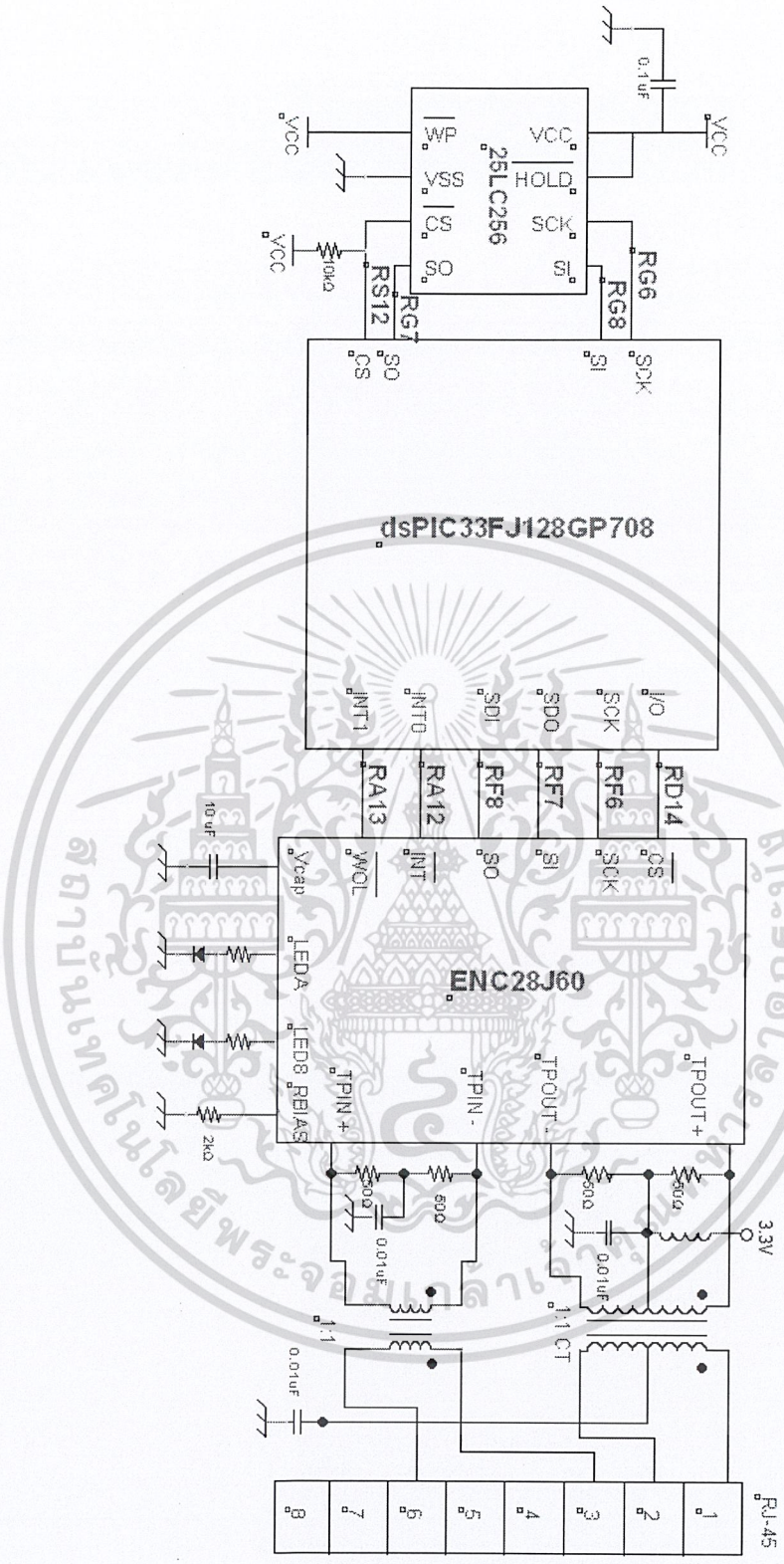


รูปที่ 3.1 ภาพรวมของระบบ

จากรูปที่ 3.1 การทำงานของระบบเริ่มจากการสั่งการจากผู้ใช้งานผ่านเครือข่ายอินเตอร์เน็ตหรือแอพพลิเคชั่นบนโทรศัพท์มือถือระบบปฏิบัติการ Windows Mobile ซึ่งอาจจะเป็นการสั่งการเปิด-ปิดอุปกรณ์ไฟฟ้า หรือสั่งการเปิด-ปิดโดยการตั้งเวลาว่าอีกเท่าไหร่จะให้อุปกรณ์ตัวไหนทำงานหรือหยุดทำงาน โดยการสั่งการจะส่งสัญญาณการควบคุมจากหน้าเว็บผ่าน Ethernet ENC28J60 ไปที่ตัวอุปกรณ์ไมโครคอนโทรลเลอร์ dsPIC33JF128GP708 ซึ่งไมโครคอนโทรลเลอร์ตัวนี้จะเป็น Web Sever ภายในตัวด้วย ในตัวไมโครคอนโทรลเลอร์ dsPIC33JF128GP708 จะมีพอร์ตเชื่อมต่อกับหน่วยความจำ EEPROM เบอร์25LC256 จะเป็นตัวเก็บหน้าเว็บเพจและ CGI Script ซึ่งไฟล์ในหน่วยความจำ EEPROM นี้ทั้งหมดจะเป็นไฟล์ Binary (.bin) เมื่อมีการสั่งการจากหน้าเว็บแต่ละครั้งก็จะส่งไฟล์ CGI มาที่ไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ก็จะดำเนินงานตามคำสั่ง และส่งสัญญาณควบคุมอุปกรณ์ไฟฟ้าเป็นสัญญาณไฟ Output ขนาด 3.3 โวลต์เข้าสู่ชุดควบคุม Relay เพื่อเป็นส่วนในการเชื่อมต่อกับอุปกรณ์ไฟฟ้า โดยระบบจะทำการตรวจสอบค่าสถานะของอุปกรณ์ไฟฟ้า และแสดงผลออกทางหน้าเว็บให้ผู้ใช้งานทราบ

3.2 การออกแบบวงจรของระบบ

ภายในระบบจะมีวงจรที่เชื่อมต่อกับไมโครคอนโทรลเลอร์ dsPIC33WEB ซึ่งใช้ไมโครชิพ รุ่น dsPIC33FJ128GP708 โดยจะมีอยู่ 3 ส่วนคือ ส่วน Ethernet ENC28J60 ส่วนหน่วยความจำ EEPROM 25LC256 และส่วนชุดควบคุม Relay จำนวน 4 ช่อง ซึ่งมีรายละเอียดดังนี้



รูปที่ 3.2 แสดงรูปวงจรการเชื่อมต่อ EEPROM 25LC256 และ ENC28J60 เข้ากับ dsPIC33FJ128GP708

จากรูปที่ 3.2 Ethernet ENC28J60 มีการเชื่อมต่อกับ ไมโครชิพรุ่น dsPIC33FJ128GP708 แบบ SPI BUS โดยจะใช้ขาสัญญาณในการรับส่งข้อมูลจำนวน 7 ขา โดยเรียงตามขาสัญญาณ ดังนี้

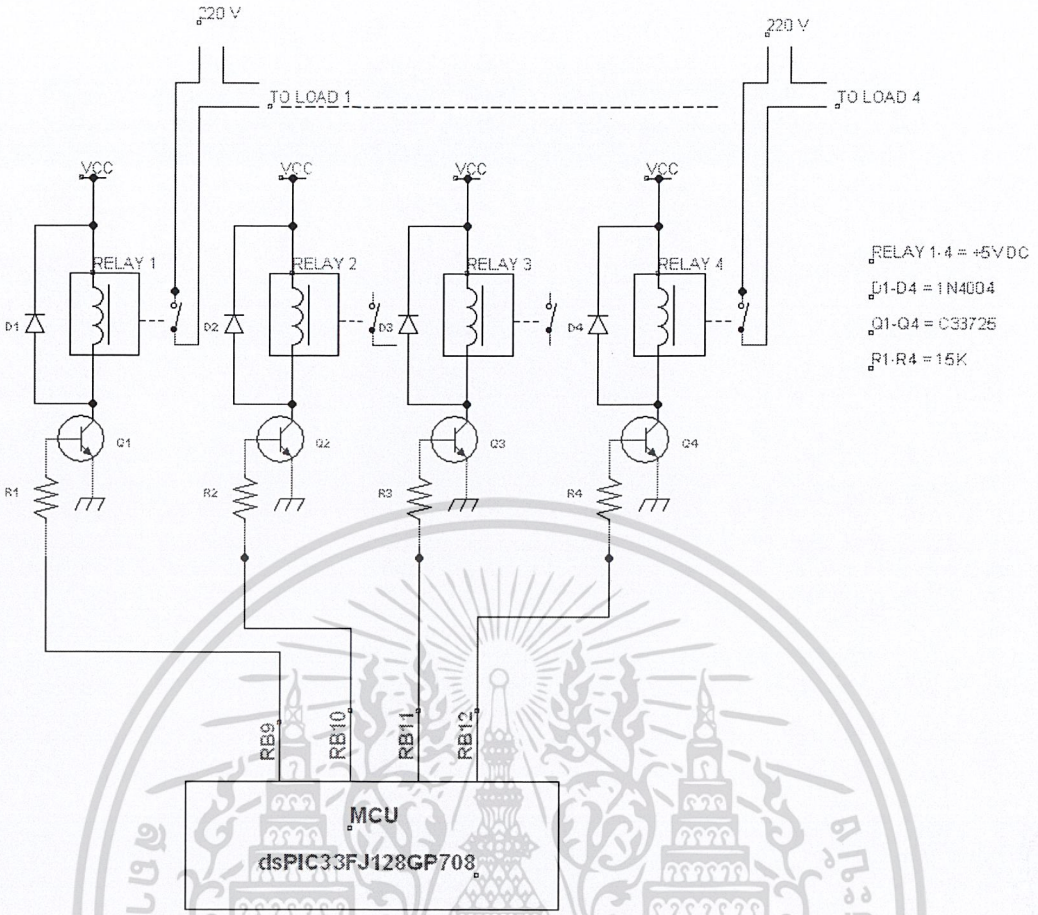
- RF6 ใช้เป็น Clock in pin for SPI interface (SCK)
- RF7 ใช้เป็น Data in pin for SPI interface (SDI)
- RF8 ใช้เป็น Data out pin for SPI interface (SDO)
- RD14 ใช้เป็น Data out pin for SPI interface (CS)
- RA12 ใช้เป็น interrupt output pin INT1 โดยสามารถเลือกใช้หรือไม่ใช้ได้ โดยการกำหนดที่ Jumper INT(EN/DS)
- RA13 ใช้เป็น Wake on LAN (WOL) โดยสามารถเลือกใช้หรือไมใช้ได้ โดยการกำหนดที่ Jumper WOL(EN/DS)
- RD15 ใช้เป็น Active-low device Reset input (RST) โดยสามารถเลือกใช้หรือไม่ใช้ได้ โดยการกำหนดที่ Jumper RST(EN/DS)

และมีการเชื่อมต่อกับเครือข่ายอินเตอร์เน็ตโดยใช้พอร์ต RJ-45 ซึ่งมีการเชื่อมต่อขาสัญญาณจาก ENC28J60 โดยใช้ขาสัญญาณจำนวน 4 ขา ดังต่อไปนี้

- TPOUT+ ใช้เป็น Differential Signal Output
- TPOUT- ใช้เป็น Differential Signal Output
- TPIN+ ใช้เป็น Differential Signal Input
- TPIN- ใช้เป็น Differential Signal Input

ในส่วน SPI Memory Module EEPROM25LC256 ซึ่งใช้ในการเก็บข้อมูล เว็บเพจ มีขาสัญญาณที่ใช้ในการเชื่อมต่อกับไมโครชิพรุ่น dsPIC33FJ128GP708

- RG6 ใช้เป็น Chip Select Input (SCK)
- RG7 ใช้เป็น Serial Data Input (SDI)
- RG8 ใช้เป็น Serial Data Output (SDO)
- RD12 ใช้เป็น Chip Select Input (CS)



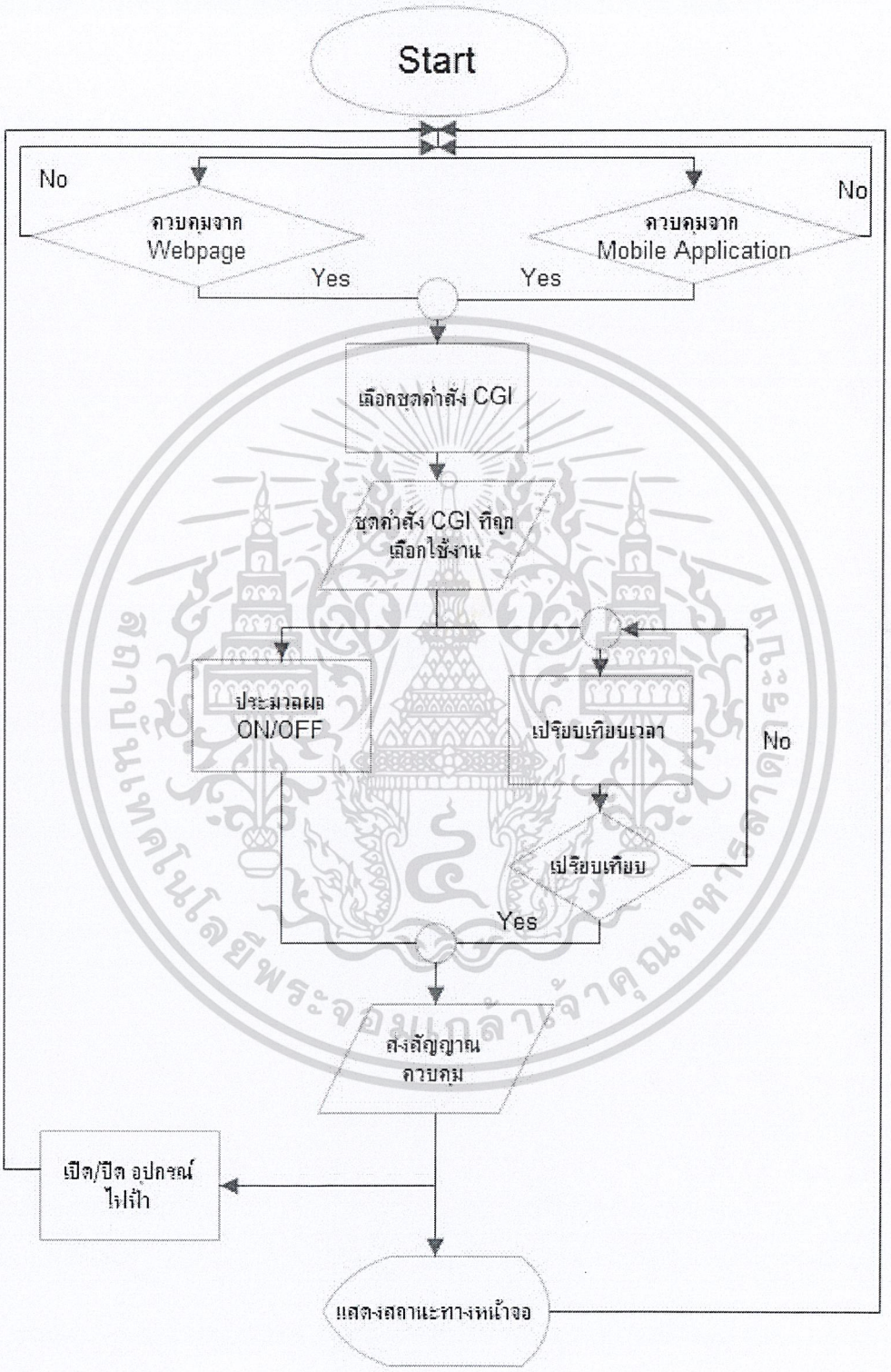
รูปที่ 3.3 แสดงรูปวงจรการเชื่อมต่อ Relay เข้ากับ dsPIC33FJ128GP708

จากรูปที่ 3.3 สัญญาณที่ออกจากไมโครคอนโทรลเลอร์เป็นสัญญาณไฟขนาด 3.3V ซึ่งพอร์ต I/O ของไมโครคอนโทรเลอร์ถูกออกแบบให้อยู่ในรูปแบบของพอร์ตมาตรฐาน 10-PIN โดยจะรับสัญญาณจากขา RB9-RB12 ของไมโครชิพ และ 1 พอร์ตจะสามารถควบคุมอุปกรณ์ไฟฟ้าได้ 1 อุปกรณ์ ซึ่งจะสามารถใช้ได้ทั้งหมด 4 อุปกรณ์ โดยวงจรรีเลย์ที่สร้างจะสร้างตามแบบมาตรฐาน ซึ่งมีส่วนประกอบดังนี้

1. Relay แบบหน้าสัมผัส ขนาด 5 โวลต์ ทำหน้าที่ ตัดต่อวงจรไฟฟ้า โดยอาศัยหลักการจ่ายไฟ 3.3 โวลต์ หรือ 0 โวลต์ เพื่อทำการตัดหรือต่อวงจร
2. ตัวต้านทานขนาด 15 กิโลโอห์ม ทำหน้าที่จำกัดกระแสไฟฟ้าให้ตรงตามต้องการ
3. ทรานซิสเตอร์ เบอร์ C33725 ทำหน้าที่ขยายสัญญาณไฟฟ้า
4. ไดโอด เบอร์ 1N4004 ทำหน้าที่ บังคับให้กระแสไฟฟ้าไหลทางเดียว เพื่อป้องกันกระแสไฟฟ้าไหลย้อนกลับ ซึ่งอาจทำให้บอร์ดเกิดความเสียหายได้

3.3 การออกแบบโปรแกรมการทำงานของระบบ

3.3.1 การออกแบบโปรแกรมสั่งการควบคุม

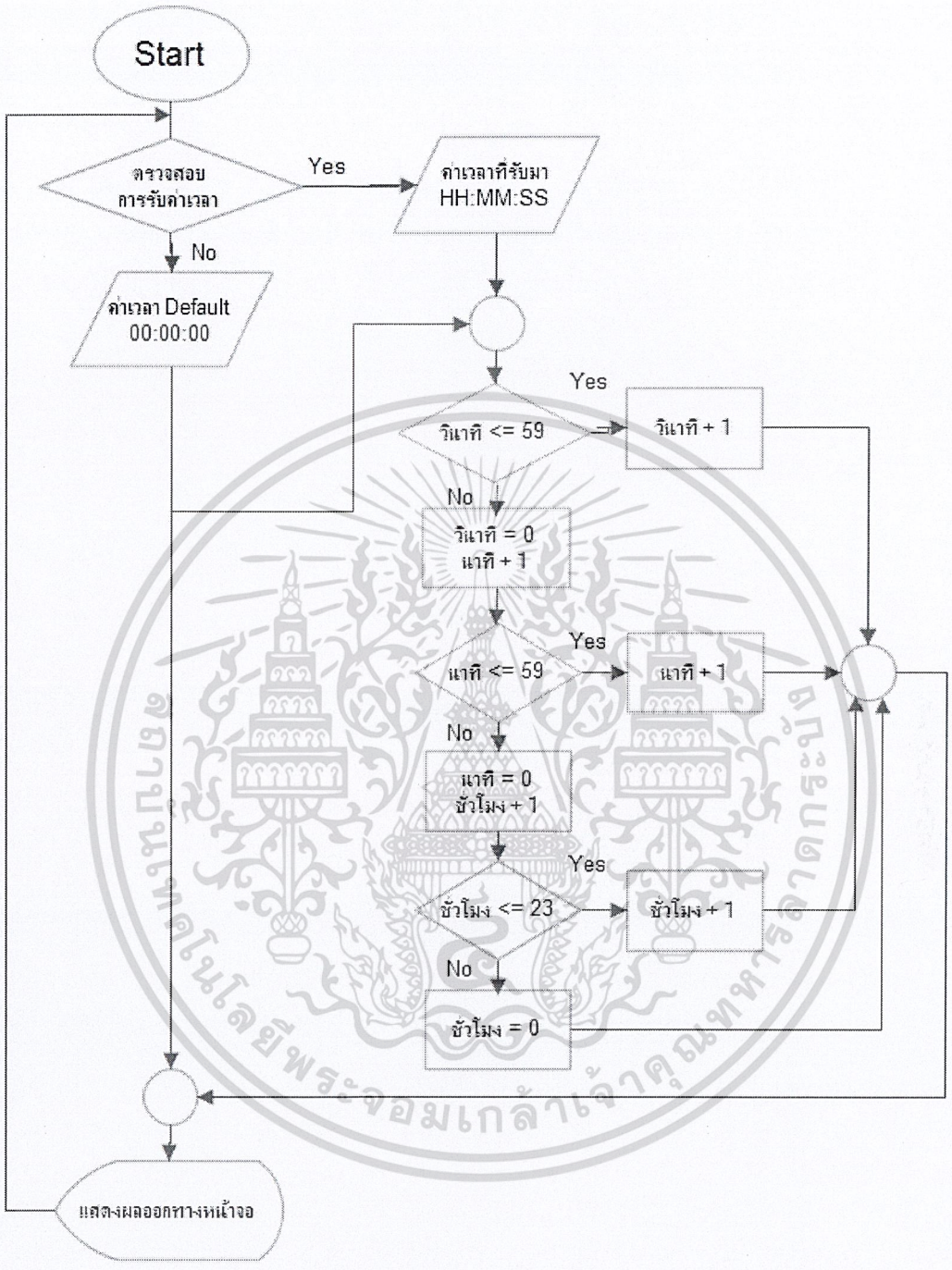


รูปที่ 3.4 แสดงผังโปรแกรมการทำงานของระบบโดยภาพรวม

การออกแบบโปรแกรมในส่วนของการเชื่อมต่อการสื่อสารกับเครือข่ายอินเตอร์เน็ตจะใช้โค้ดที่ทางผู้พัฒนาบอร์ดได้จัดทำขึ้น และได้ศึกษาเพิ่มเติมในส่วนของการตั้งเวลาเพื่อให้เข้ากับระบบที่ได้ทำการออกแบบไว้ ซึ่งเป็นโค้ดการเชื่อมต่อเครือข่ายอินเตอร์เน็ตตามมาตรฐาน TCP/IP ของบริษัทไมโครชิพ

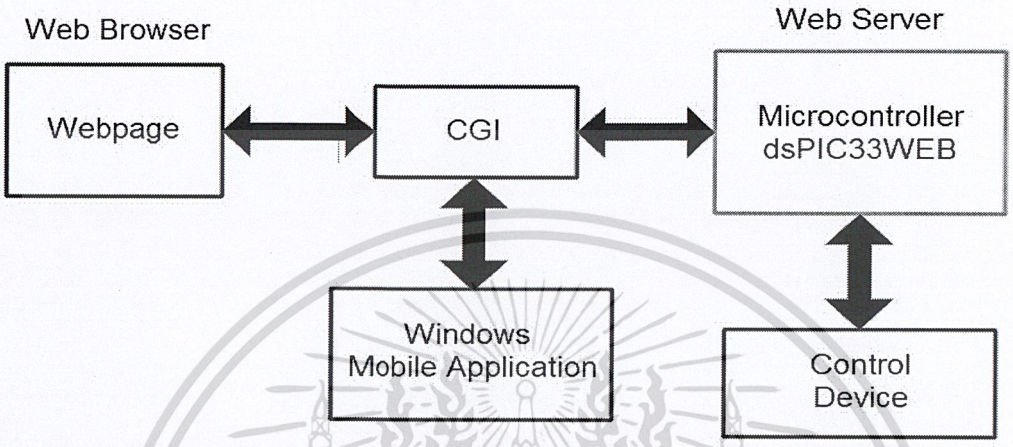
จากรูปที่ 3.4 การทำงานของโปรแกรมจะเริ่มจากตรวจสอบว่าทางหน้าเว็บหรือหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือได้มีการสั่งการมาหรือไม่ ถ้ามีการสั่งการมาได้ออกแบบโปรแกรมในส่วนนี้ไว้ว่าในการสั่งการแต่ละครั้งระบบจะทำการส่งไฟล์ CGI Script จากหน้าเว็บหรือหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือไปยังไมโครคอนโทรลเลอร์ จากนั้นโปรแกรมจะตรวจสอบว่าไฟล์ CGI Script ที่ได้รับมานั้นตรงกับชุดคำสั่งใด เมื่อได้ชุดคำสั่งแล้วโปรแกรมจะทำการประมวลผลชุดคำสั่งไฟล์ CGI Script ว่าทางหน้าเว็บหรือหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือต้องการตั้งเวลาบอร์ดหรือสั่งการควบคุมอุปกรณ์ไฟฟ้าชิ้นไหนอย่างไร แล้วจึงส่งสัญญาณไปตั้งเวลาบอร์ดหรือไปสั่งการควบคุมอุปกรณ์ไฟฟ้า จากนั้นจะส่งค่าเวลาของบอร์ดและสถานะของอุปกรณ์ไฟฟ้าผ่านไฟล์ CGI Script เพื่อไปแสดงผลผ่านทางหน้าเว็บหรือหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือให้ผู้ใช้งานทราบ

3.3.2 การออกแบบโปรแกรมการตั้งเวลาบอร์ด



รูปที่ 3.5 แสดงผังโปรแกรมการตั้งเวลาบอร์ด

จากรูปที่ 3.5 จะแสดงถึงระบบการทำงานของการตั้งเวลาบอร์ด ซึ่งจะอยู่ในส่วนของ CGI ที่ถูกเลือกใช้งานในรูปที่ 3.4 เริ่มแรกเมื่อบอร์ดเริ่มทำงานเวลาของบอร์ดจะเริ่มที่ 00:00:00 และเพิ่มขึ้นทีละ 1 วินาทีตามเวลาปกติทั่วไป ซึ่งในส่วนของเวลานี้ผู้ใช้งานทำการตั้งเวลาเองได้ตามที่ผู้ใช้งานต้องการ โดยที่ในการใส่ค่าเวลานั้นจะอยู่ในรูปแบบ HH:MM:SS เช่น 23:59:59



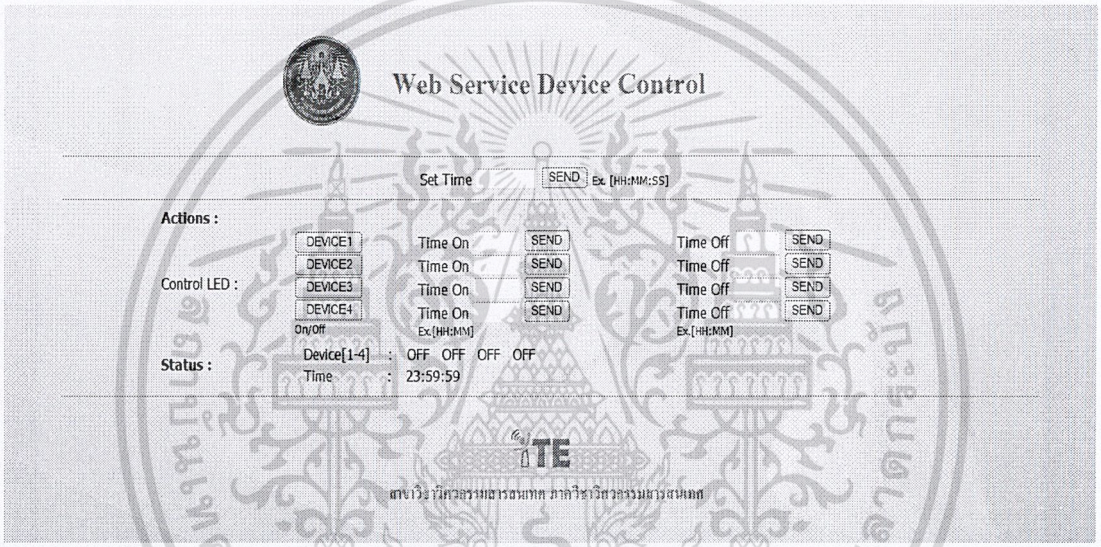
รูปที่ 3.6 แสดงผังการทำงานระหว่าง Web Browser และ Web Server

จากรูปที่ 3.6 จะเห็นได้ว่าไฟล์ CGI จะเป็นส่วนที่เชื่อมระหว่างไมโครคอนโทรลเลอร์และเว็บเพจในรูปที่ 3.4 โดยการติดต่อกันจะส่งผ่านไฟล์ CGI เท่านั้น โดยเมื่อมีการสั่งการจากเว็บเพจแต่ละครั้งระบบจะทำการกำหนดไฟล์ CGI ที่ได้สั่งการ (CGI Script) เพื่อใช้ในการเลือกชุดคำสั่งการทำงานของไฟล์ CGI ในไมโครคอนโทรลเลอร์ จากนั้นเมื่อไมโครคอนโทรลเลอร์ได้รับไฟล์ CGI script จะทำการประมวลผลคำสั่งการในการเปิด-ปิด หรือ เลือกเวลาในการเปิด-ปิด แล้วทำการส่งสัญญาณให้กับส่วนควบคุมอุปกรณ์ไฟฟ้าเพื่อควบคุม และส่งค่าสถานะของอุปกรณ์ไฟฟ้าโดยการผ่านไฟล์ CGI Script ในส่วนของ Status เพื่อแสดงสถานะออกมาบนหน้าเว็บเพจ

3.4 การออกแบบส่วนติดต่อผู้ใช้งาน

3.4.1 ส่วนติดต่อผู้ใช้งานทางหน้าเว็บเพจ

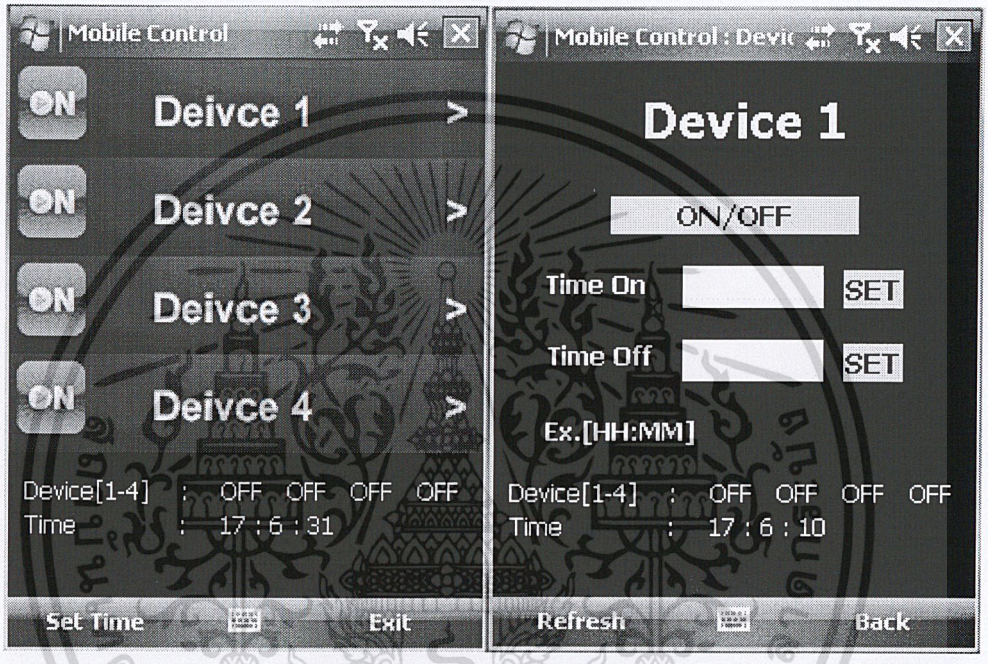
ในส่วนนี้จะเป็นส่วนของหน้าเว็บเพจ ที่ใช้ติดต่อระหว่างผู้ใช้งานกับระบบควบคุมผ่านเครือข่ายอินเตอร์เน็ตซึ่งจะอยู่ในส่วนของหน้าจอแสดงผลในรูปที่ 3.5 โดยมีส่วนสำคัญอยู่ 4 ส่วน คือ ส่วนของการตั้งเวลาบอร์ดโดยค่าเวลาในการตั้งลงไปนั้นจะอยู่ในลักษณะ HH:MM:SS เช่น 12:00:00 ส่วนของปุ่มกดแบบ Toggle Switch สำหรับสั่งเปิด-ปิดอุปกรณ์ไฟฟ้า ส่วนของการตั้งเวลาในการเปิดปิดอุปกรณ์ไฟฟ้าโดยการตั้งเวลานั้นจะอยู่ในลักษณะ HH:MM เช่น 12:00 และส่วนของการแสดงสถานะเปิด-ปิดของอุปกรณ์ไฟฟ้า รวมไปถึงสถานะเวลาบอร์ดด้วย ดังรูปที่ 3.7



รูปที่ 3.7 แสดงหน้าเว็บเพจที่ใช้ในการสั่งการเปิดปิด อุปกรณ์ไฟฟ้า

3.4.2 ส่วนติดต่อผู้ใช้งานทางแอพพลิเคชั่นบนโทรศัพท์มือถือ

ในส่วนนี้จะอยู่ในส่วนของหน้าจอแสดงผลในรูปที่ 3.4 โดยที่แอพพลิเคชั่นจะมีการทำงานเช่นเดียวกับหน้าเว็บเพจ แต่สามารถใช้งานได้สะดวกกว่าเพราะไม่ต้องทำการระบุหมายเลขไอพีหรือชื่อเว็บไซต์ที่ใช้ในการสั่งการควบคุม และยังแยกส่วนการควบคุมออกเป็นส่วนๆ โดยสามารถสั่งการเปิด-ปิดอุปกรณ์ไฟฟ้าในทันที ตั้งเวลาการเปิด-ปิดอุปกรณ์ไฟฟ้า แสดงสถานะของอุปกรณ์ไฟฟ้าแต่ละตัว และยังสามารถตั้งเวลาของบอร์ด และแสดงเวลาของบอร์ดที่หน้าแอพพลิเคชั่นได้ ดังรูปที่ 3.8



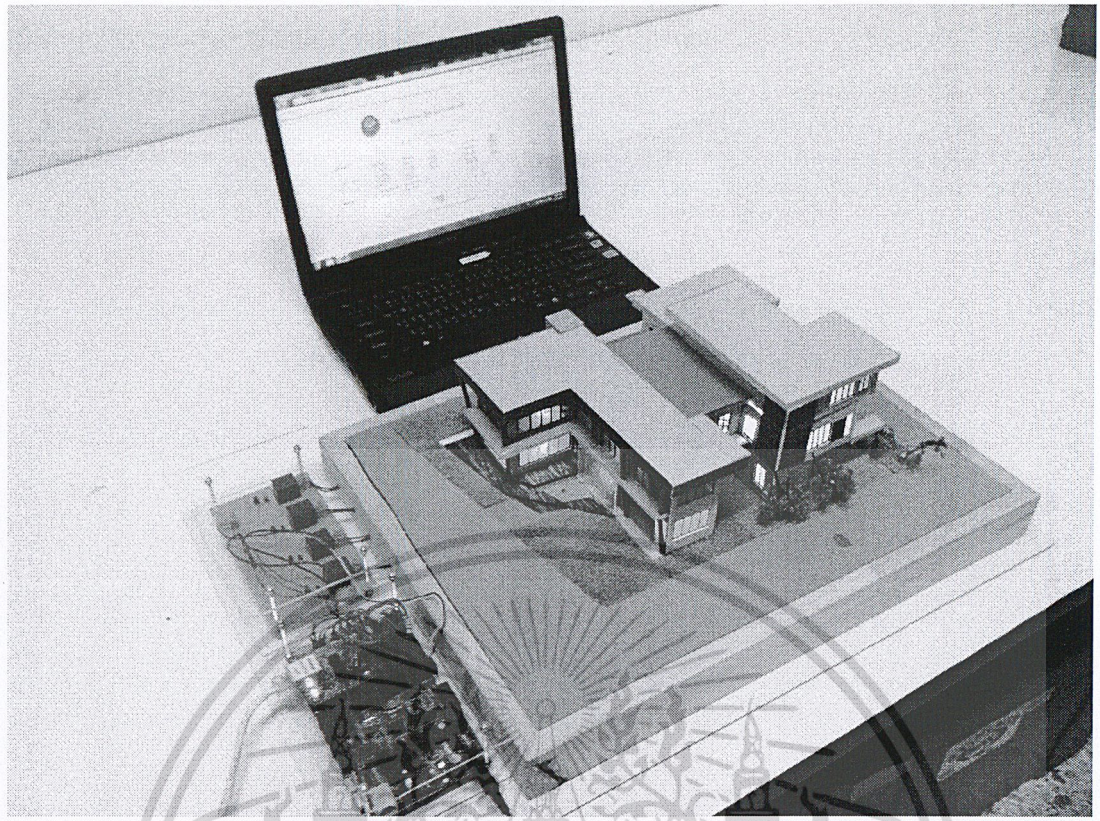
รูปที่ 3.8 แสดงหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือ

บทที่ 4

การทดลอง

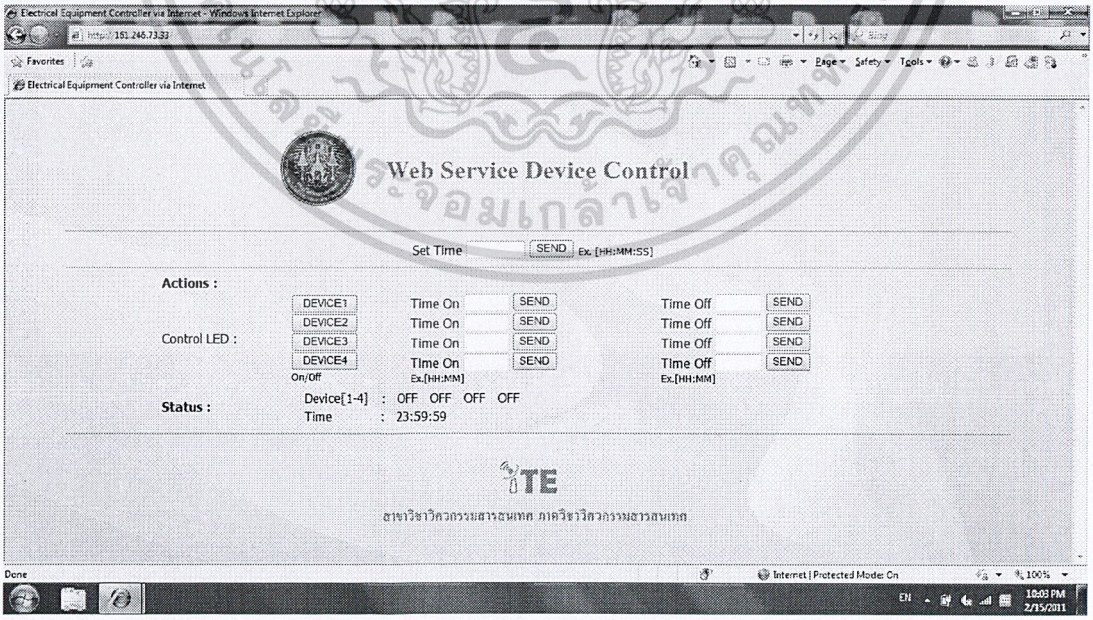
4.1 ขั้นตอนการทดลองสั่งการควบคุมจากหน้าเว็บบราวเซอร์

1. ทำการดาวน์โหลดโปรแกรมลงในไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708
2. สร้างหน้าเว็บบราวเซอร์จากโปรแกรม Dream weaver และกำหนดไฟล์ CGI Script
3. ทำการดาวน์โหลดไฟล์ Binary ที่ได้จากการแปลงหน้าเว็บและ CGI Script ลงในหน่วยความจำ EEPROM
4. จำลองไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708 เป็น Web Server โดยกำหนดค่า IP Address เป็น 161.246.73.33
5. ทำการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับบอร์ดโดยใช้สาย LAN แบบ Cross และกำหนดค่า IP Address ของเครื่องคอมพิวเตอร์เป็น 161.246.73.200
6. ทำการทดลองสั่งการควบคุม ตั้งเวลาบอร์ด เปิด-ปิดอุปกรณ์ไฟฟ้า ตั้งเวลาเปิด-ปิดอุปกรณ์ไฟฟ้าได้
7. ทำการเชื่อมต่อระหว่าง Router ของสถาบันกับบอร์ดโดยใช้สาย LAN แบบธรรมดา และกำหนดค่า IP Address ของบอร์ดเป็น 161.246.73.33
8. ทำการทดลองสั่งการควบคุม ตั้งเวลาบอร์ด เปิด-ปิดอุปกรณ์ไฟฟ้า ตั้งเวลาเปิด-ปิดอุปกรณ์ไฟฟ้า จากที่อื่นๆระยะไกลผ่านทางอินเตอร์เน็ตได้



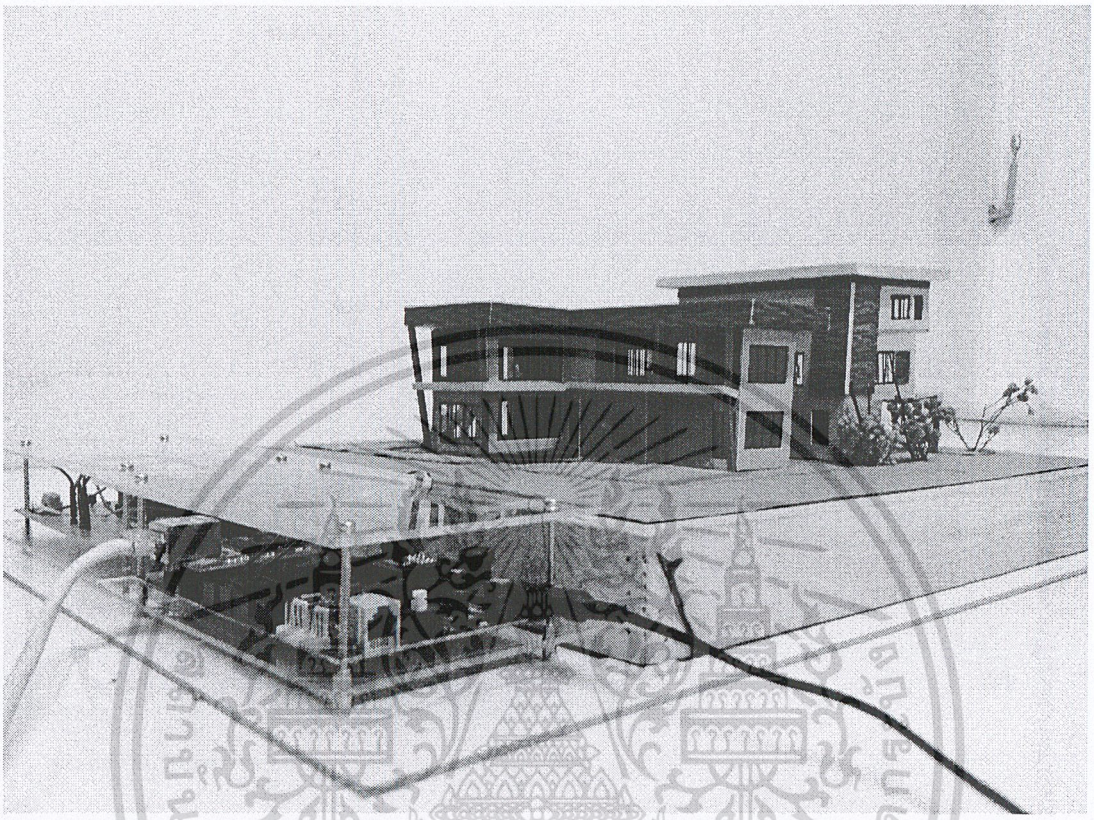
รูปที่ 4.1 การเชื่อมต่อระหว่างบอร์ดและอุปกรณ์ไฟฟ้า

เมื่อทดลองเปิดหน้าเว็บโดยใช้ Internet Explorer และกำหนดให้ URL เป็น 161.246.73.33 จะปรากฏหน้าเว็บเพจ ดังรูปที่ 4.2



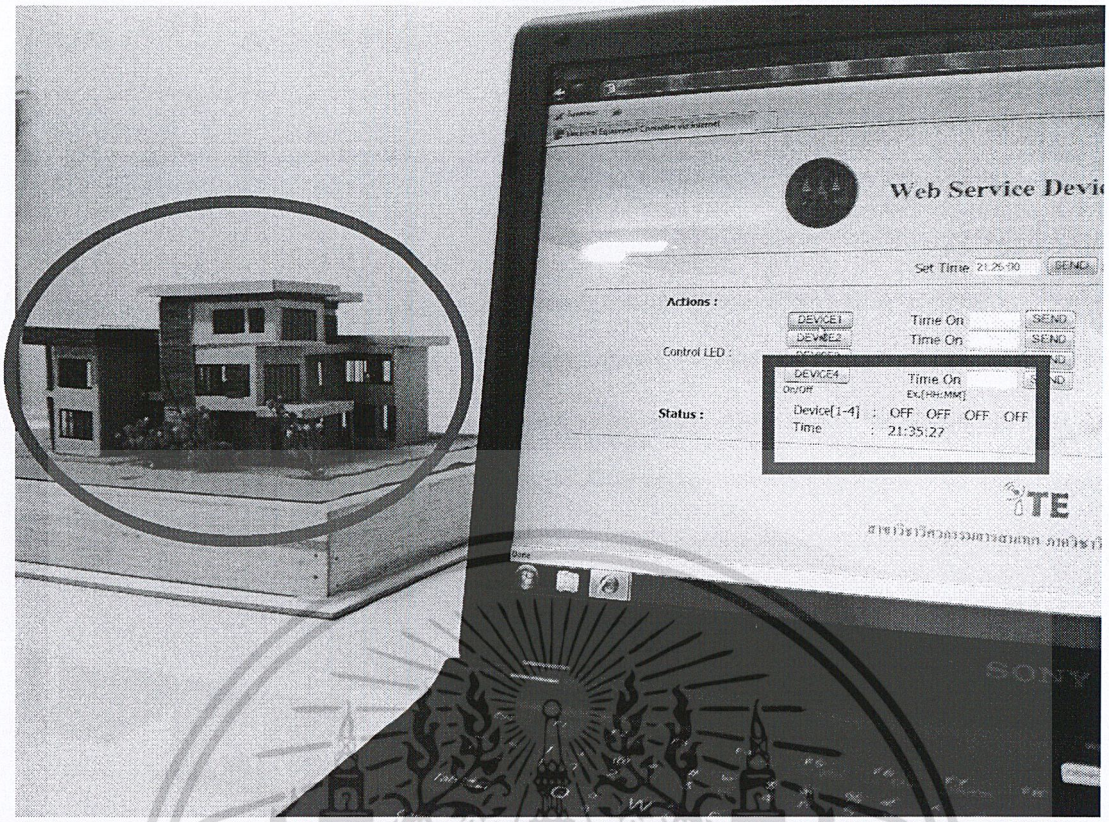
รูปที่ 4.2 แสดงหน้าเว็บเพจ

จากรูปที่ 4.2 ที่หน้าเว็บเพจจะมีส่วนควบคุมอุปกรณ์ไฟฟ้าและแสดงสถานะที่อุปกรณ์ไฟฟ้านั้นเป็นอยู่ เช่น ถ้าอุปกรณ์ไฟฟ้านั้นทำงานอยู่เมื่อเปิดหน้าเว็บเพจขึ้นมา ค่าสถานะของอุปกรณ์นั้นก็จะเป็น ON ซึ่งในการทดลองได้กำหนดให้อุปกรณ์ไฟฟ้าไม่ได้ทำงานอยู่ทั้งหมด



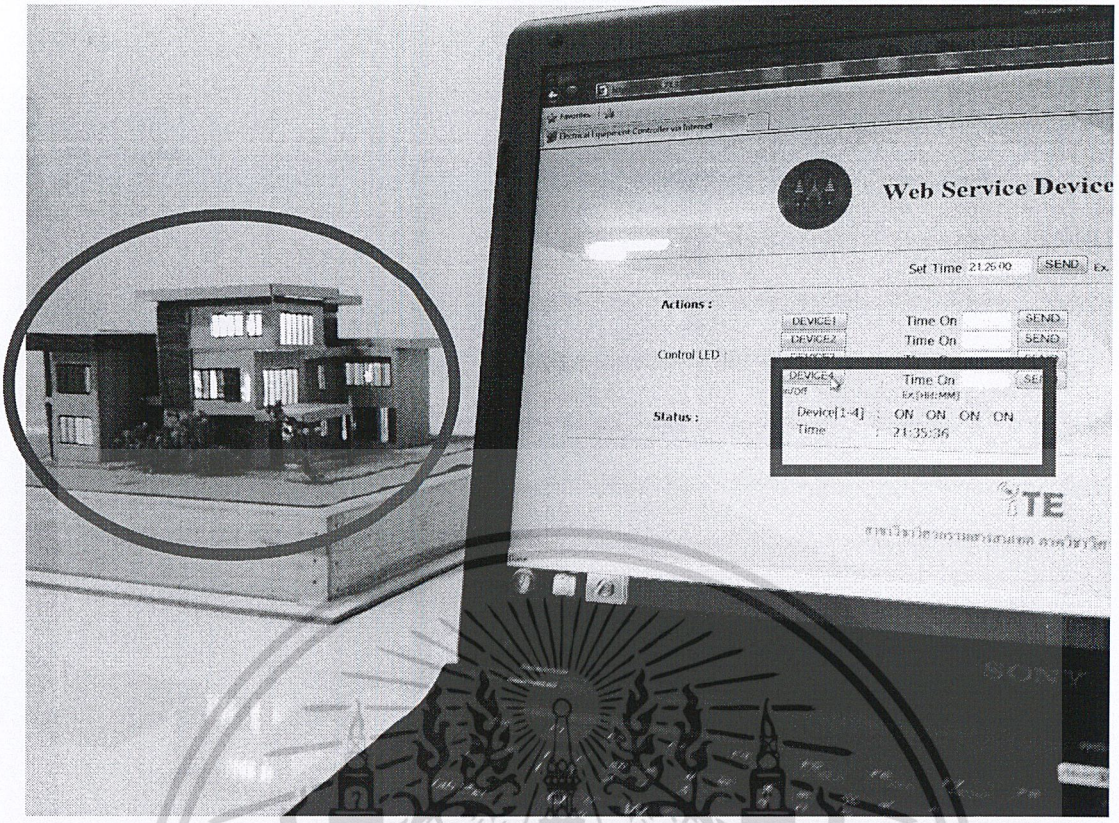
รูปที่ 4.3 แสดงอุปกรณ์ไฟฟ้าเมื่อไม่ได้เปิดใช้งาน

จากรูปที่ 4.3 อุปกรณ์ไฟฟ้าทุกตัวนั้นไม่ได้ทำงานอยู่เมื่อทำการเปิดหน้าเว็บเพจขึ้นมา ช่อง Status จะแสดงสถานะปัจจุบันของอุปกรณ์ฟ้า ดังจากรูปที่ 4.4

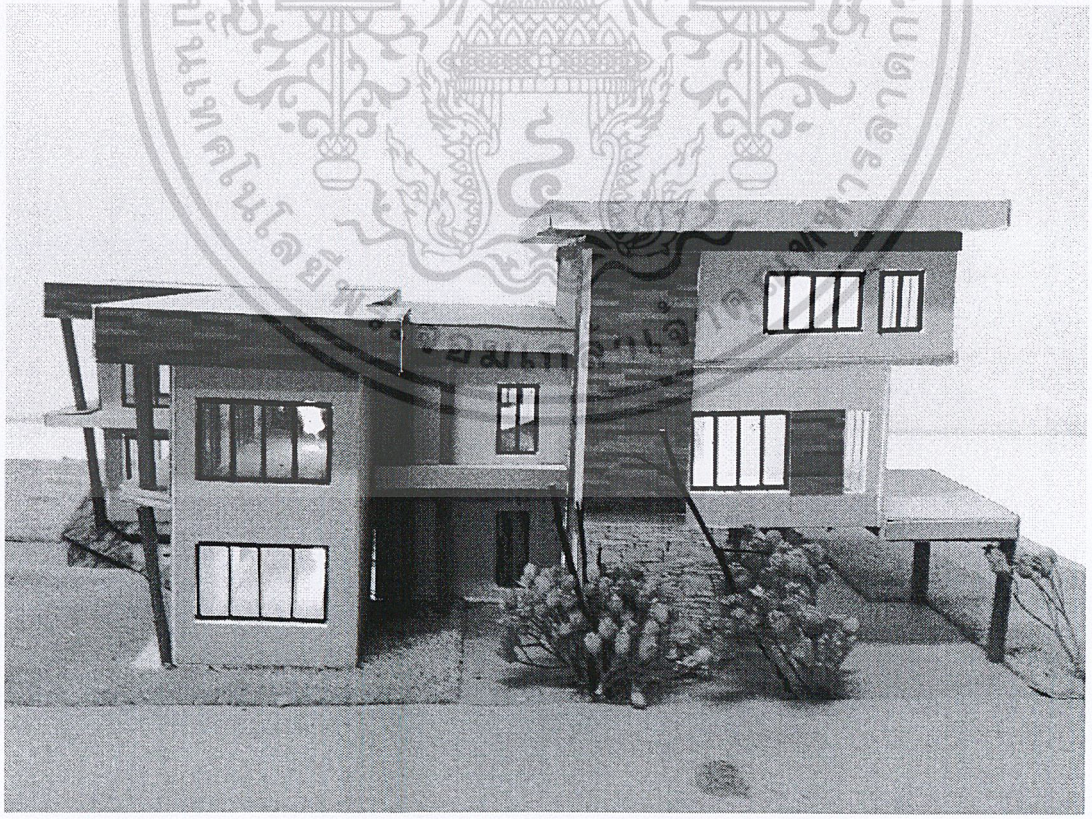


รูปที่ 4.4 แสดงสถานะของอุปกรณ์ไฟฟ้าเมื่อไม่ได้เปิดใช้งาน

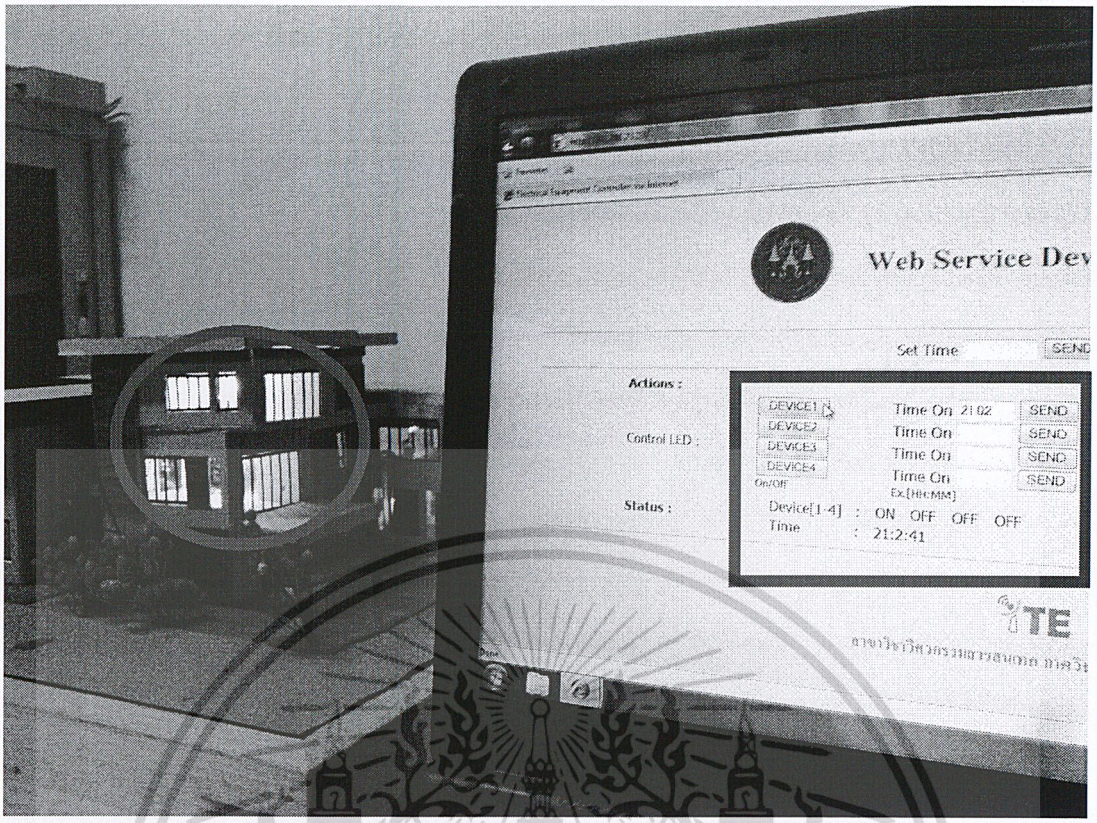
เมื่อได้ทำการทดลองเปิดใช้งานอุปกรณ์ไฟฟ้าเพื่อควบคุมอุปกรณ์ไฟฟ้าทั้งหมด ที่ช่อง Status ค่าสถานะของอุปกรณ์ไฟฟ้าทั้งหมดจะเป็น ON และอุปกรณ์ไฟฟ้าทั้งหมดทำงาน สังเกตได้จากรูปที่ 4.5



รูปที่ 4.5 แสดงสถานะของอุปกรณ์ไฟฟ้าเมื่อมีการเปิดใช้งาน

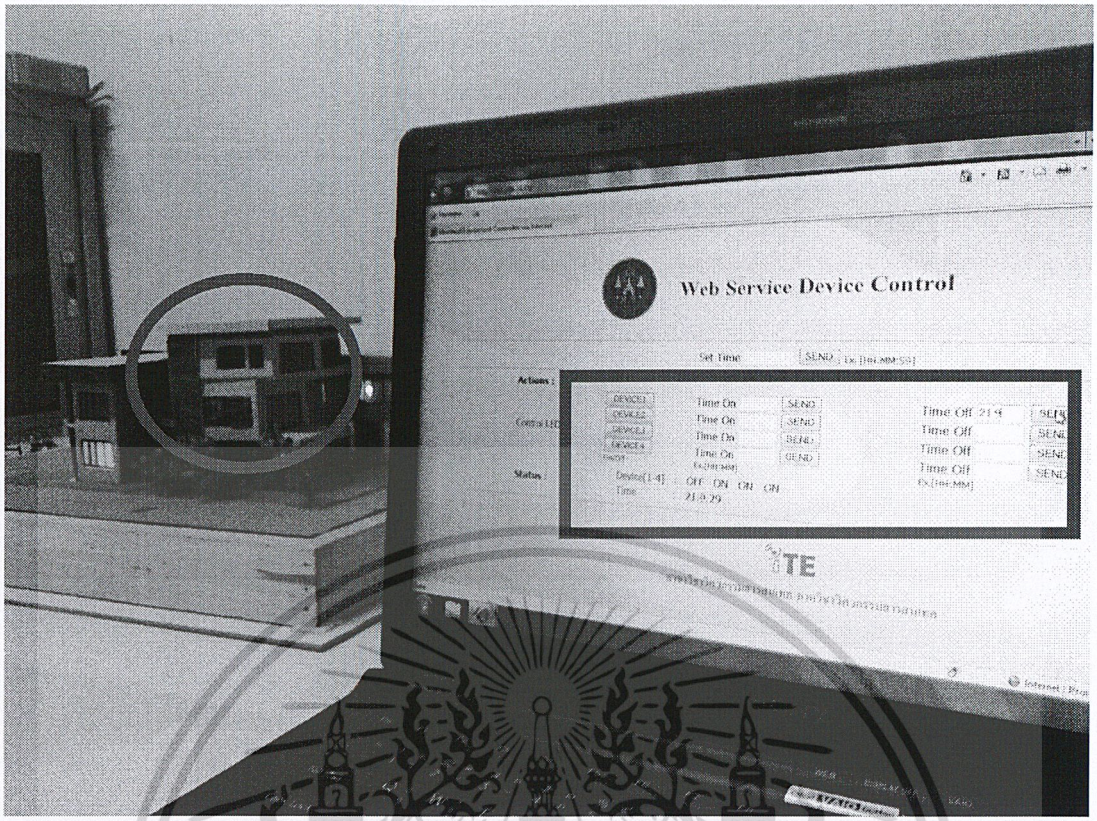


รูปที่ 4.6 แสดงอุปกรณ์ไฟฟ้าเมื่อเปิดใช้งาน



รูปที่ 4.7 แสดงการตั้งเวลาเปิดใช้งานอุปกรณ์ไฟฟ้าตัวที่ 1

จากรูปที่ 4.7 ได้ทำการทดลองตั้งเวลาเปิดใช้งานอุปกรณ์ไฟฟ้าตัวที่ 1 เมื่อถึงเวลาตามที่ตั้งไว้จะเห็นได้ว่าอุปกรณ์ไฟฟ้าตัวที่ 1 ทำงาน และสถานะของอุปกรณ์ไฟฟ้าที่ช่อง Status จะเปลี่ยนจาก OFF กลายเป็น ON ทันที

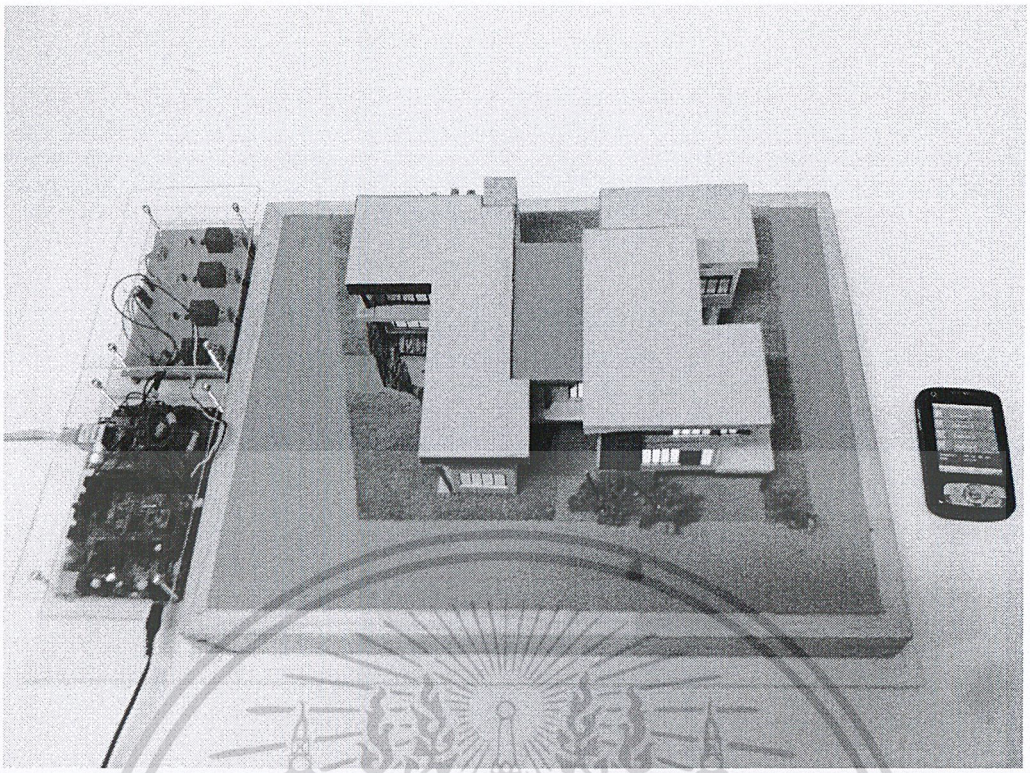


รูปที่ 4.8 แสดงการตั้งเวลาปิดใช้งานอุปกรณ์ไฟฟ้าตัวที่ 1

จากรูปที่ 4.8 ได้ทำการทดลองตั้งเวลาปิดการใช้งานอุปกรณ์ไฟฟ้าตัวที่ 1 เมื่อถึงเวลาตามที่ตั้งไว้ อุปกรณ์ไฟฟ้าตัวที่ 1 จะหยุดทำงานและสถานะที่ช่อง Status จะเปลี่ยนเป็น OFF ทันที

4.2 ขั้นตอนการทดลองสั่งการควบคุมจากแอปพลิเคชั่นบนโทรศัพท์มือถือ

1. ทำการดาวน์โหลดโปรแกรมลงในไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708
2. สร้างหน้า Application โทรศัพท์มือถือจากโปรแกรม Visual C# และกำหนดไฟล์ CGI Script ให้ตรงกับไฟล์ CGI Script ของหน้าเว็บเพจ
3. ทำการแปลงไฟล์แอพพลิเคชั่นที่เขียนขึ้นมาเป็นไฟล์ .CAB สำหรับติดตั้งบนโทรศัพท์มือถือ และติดตั้งลงบนเครื่องโทรศัพท์มือถือ
4. ทดลองตั้งเวลาบอร์ดโดย ใส่รูปแบบเวลาเป็น HH:MM:SS และ ตรวจสอบความถูกต้องของเวลา
5. ทดลองเปิด-ปิด โดยกดปุ่ม OF/OFF ของอุปกรณ์ไฟฟ้าทั้ง 4 อุปกรณ์ และตรวจสอบความถูกต้องของสถานะ
6. ทดลองตั้งเวลาเปิด-ปิด อุปกรณ์ไฟฟ้าทั้ง 4 อุปกรณ์ โดยใส่รูปแบบเวลาเป็น HH:MM และตรวจสอบความถูกต้องของสถานะ



รูปที่ 4.9 แสดงภาพรวมระหว่างบอร์ด อุปกรณ์ไฟฟ้า และโทรศัพท์มือถือ

เมื่อทำการเปิดแอพพลิเคชั่นควบคุมอุปกรณ์ไฟฟ้าที่ได้ทำการ Install ลงเครื่องโทรศัพท์มือถือไว้แล้ว ซึ่งในโปรแกรมแอพพลิเคชั่นได้ใช้ไฟล์ CGI ที่ใช้ในการสั่งการควบคุมตัวเดียวกันกับไฟล์ CGI ที่หน้าเว็บเพจ จึงทำให้ไม่ว่าจะทำการสั่งการควบคุมจากหน้าเว็บเพจหรือหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือ ค่าเวลาและสถานะของอุปกรณ์ไฟฟ้าก็จะเปลี่ยนตามไปด้วย

รูปที่ 4.10 แสดงโปรแกรมที่ใช้ในการเปิดแอพพลิเคชั่น

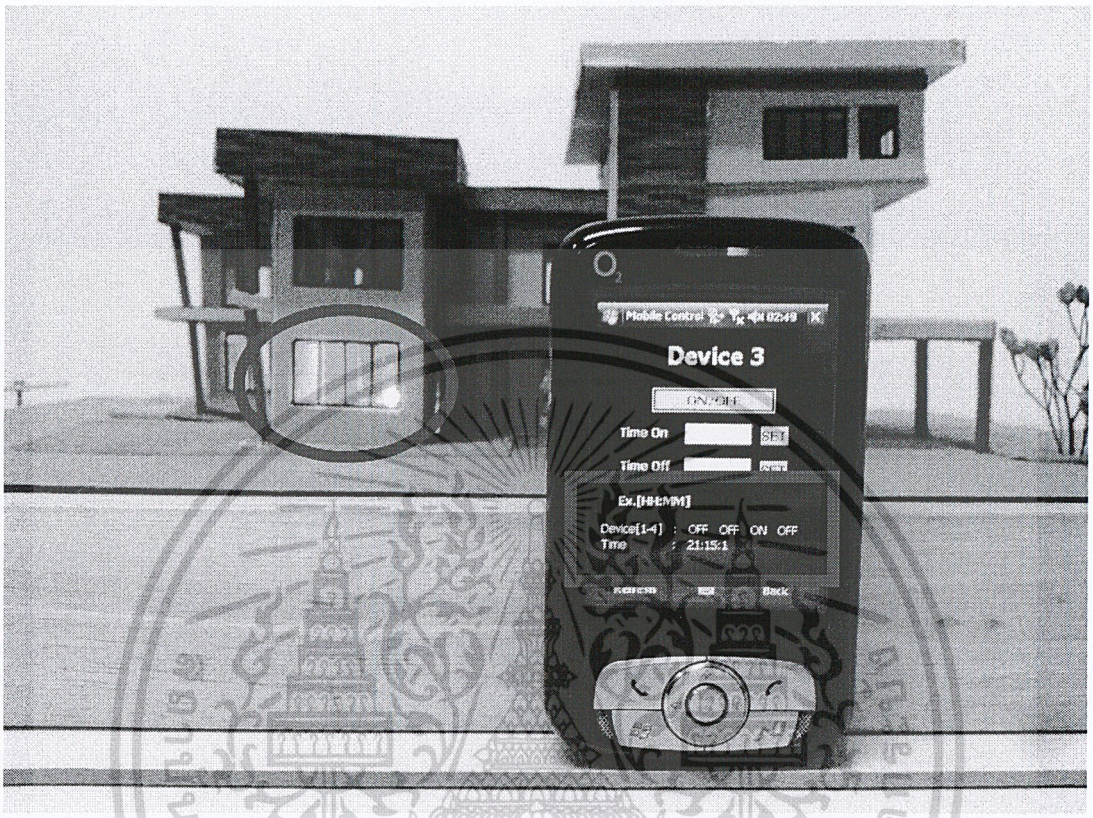
จากรูปที่ 4.10 ทำการทดลองเปิดโปรแกรม mobileCAB เพื่อเข้าไปหน้าควบคุมอุปกรณ์ไฟฟ้า จะปรากฏหน้าแอพพลิเคชั่นดังรูปที่ 4.11

รูปที่ 4.11 แสดงหน้าแรกของแอพพลิเคชั่นควบคุมอุปกรณ์ไฟฟ้า

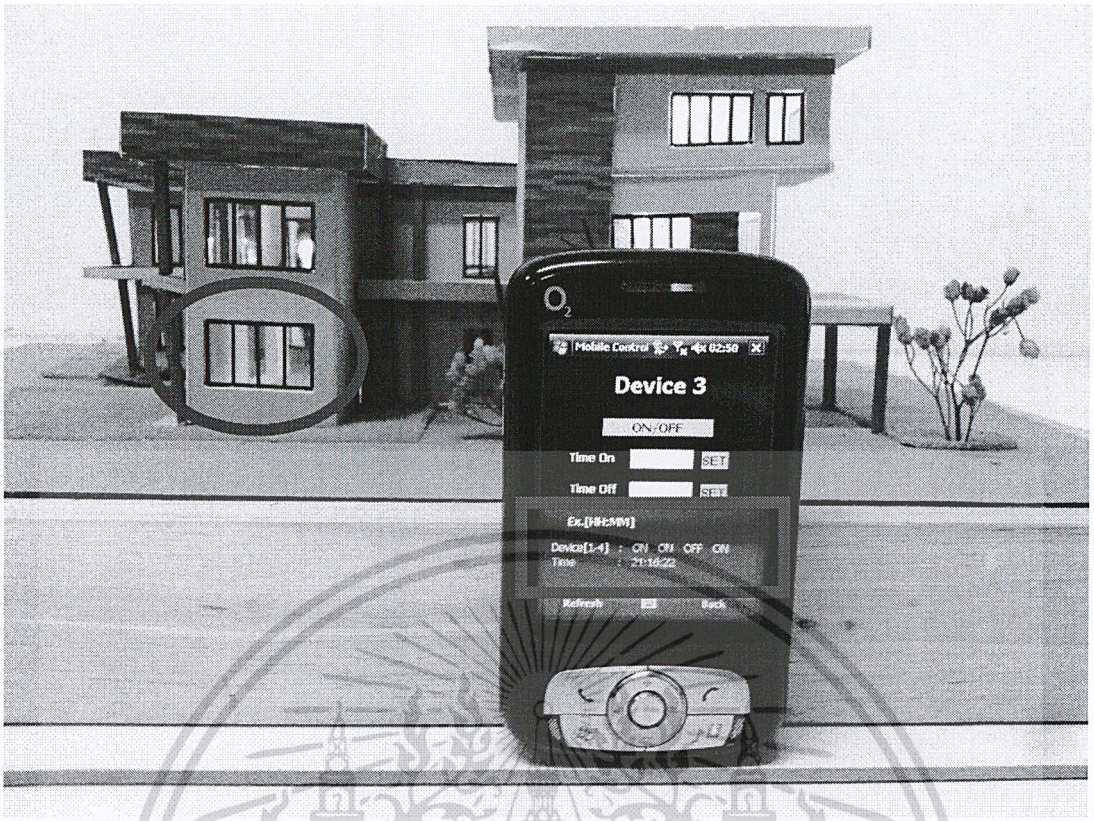
จากรูปที่ 4.11 จะเห็นได้ว่าปุ่มที่ใช้ในการสั่งการจะแยกจากกันเป็นส่วนๆ ซึ่งทำให้ง่ายต่อการใช้งาน และปุ่ม Refresh สำหรับกรณีที่เวลาหรือสถานะของอุปกรณ์ไฟฟ้าค้าง

รูปที่ 4.12 แสดงหน้าของการตั้งเวลาบอร์ด

จากรูปที่ 4.12 ทำการทดลองตั้งเวลาบอร์ดซึ่งในการใส่ค่าเวลานั้นจะใส่รูปลักษณะ HH:MM:SS เช่น 00:09:47

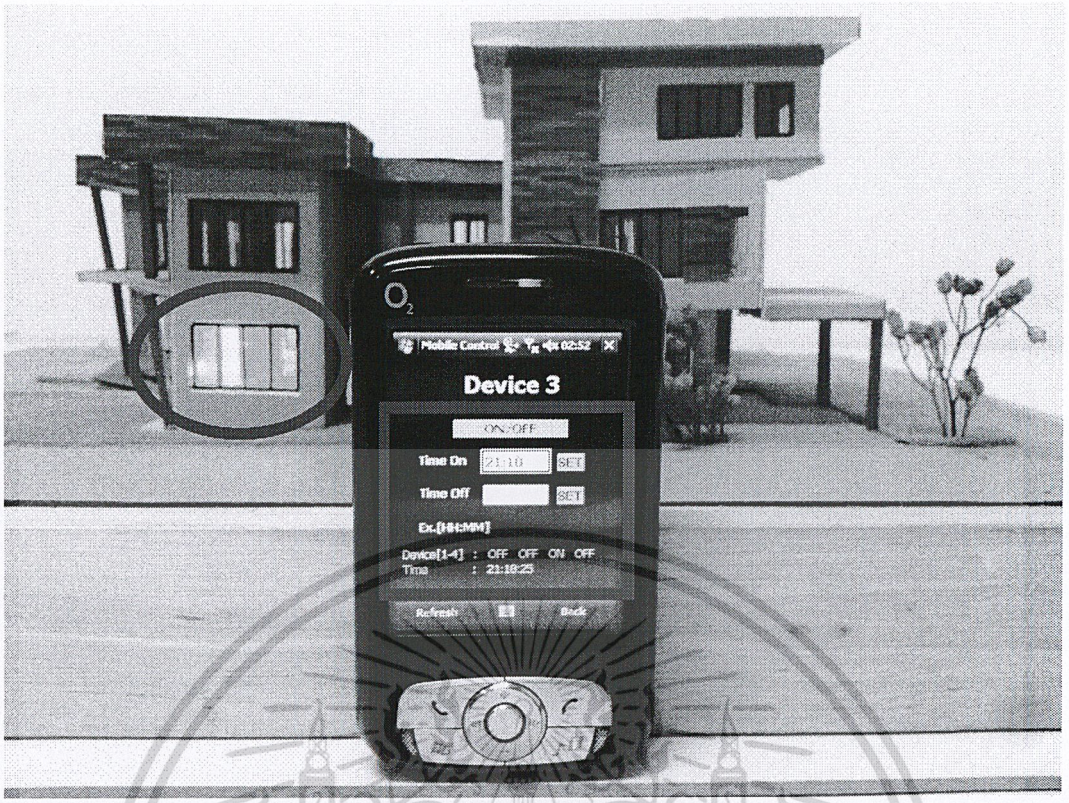


4.13 แสดงสถานะเมื่อเปิดใช้งานอุปกรณ์ไฟฟ้าตัวที่ 3

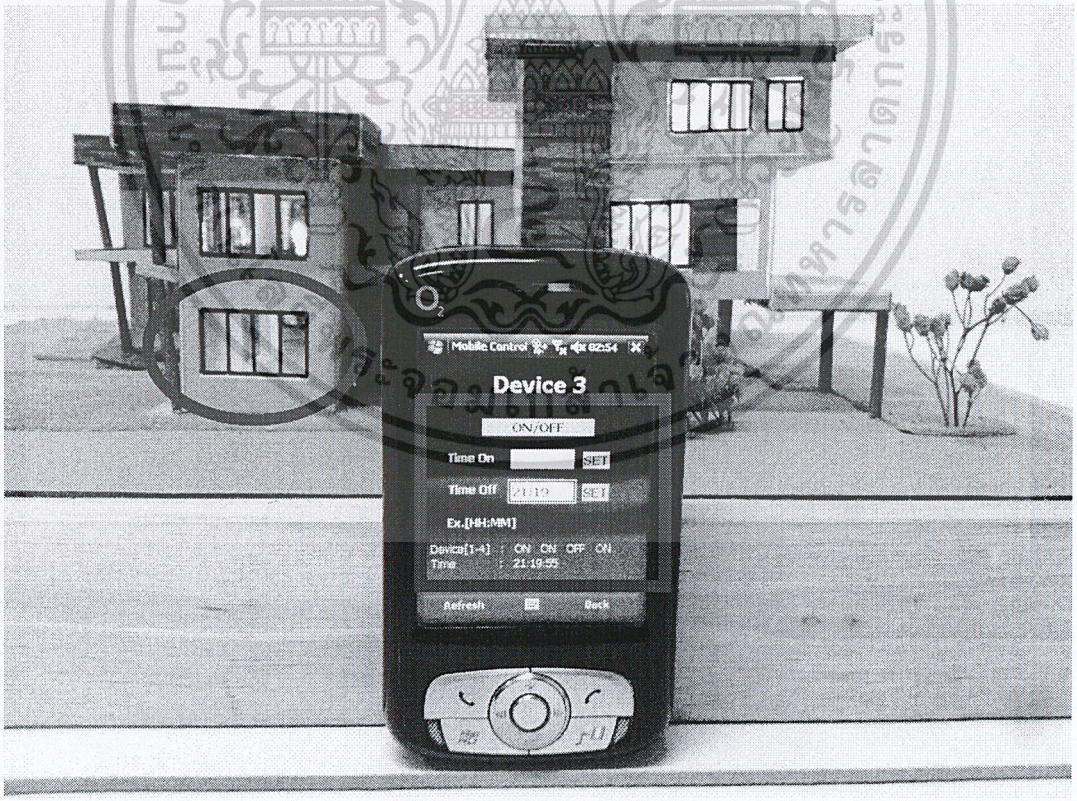


รูปที่ 4.14 แสดงสถานะเมื่อปิดใช้งานอุปกรณ์ไฟฟ้าตัวที่ 3

จากรูปที่ 4.12 และรูปที่ 4.13 ทำการทดลองเปิด-ปิดอุปกรณ์ไฟฟ้าตัวที่ 3 จะเห็นได้ว่าเมื่อทำการเปิด-ปิดอุปกรณ์ไฟฟ้าที่ช่อง Status จะเปลี่ยนค่าสถานะตามอุปกรณ์ไฟฟ้าทันที



รูปที่ 4.15 แสดงการตั้งเวลาเปิดอุปกรณ์ไฟฟ้าตัวที่ 3



รูปที่ 4.16 แสดงการตั้งเวลาปิดอุปกรณ์ไฟฟ้าตัวที่ 3

จากรูปที่ 4.14 และรูปที่ 4.15 ทำการทดลองตั้งเวลาเปิด-ปิดอุปกรณ์ฟ้าตัวที่ 3 เมื่อถึงเวลาตามที่ตั้งไว้อุปกรณ์ไฟฟ้าตัวที่ 3 ได้ทำการเปิดและปิดตามเวลาที่กำหนดและช่อง Status สถานะของอุปกรณ์ไฟฟ้าตัวที่ 3 ก็เปลี่ยนไปตามสถานะของอุปกรณ์ไฟฟ้า

ในการทดลองได้ทำการตั้งให้ทางหน้าเว็บเพจทำการ Refresh ทุกๆ 1 นาทีเมื่อเปิดหน้าเว็บเพจค้างไว้ เพื่อให้ในกรณีที่มีผู้ใช้งานหลายคนทำการสั่งควบคุมอุปกรณ์ไฟฟ้าผ่านทางหน้าเว็บเพจ ในบางครั้งค่า Status ไม่แสดงผลตามสถานะของอุปกรณ์ไฟฟ้า รวมไปถึงเวลาของบอร์ดที่ผู้ใช้งานได้ตั้งไว้ด้วย แต่เมื่อทำการ Refresh แล้วค่า Status ที่หน้าเว็บเพจจะแสดงตามสถานะของอุปกรณ์ไฟฟ้าตามปกติ และในกรณีที่การเชื่อมต่อมีปัญหาจะทำให้ไม่แสดงค่า Status ของอุปกรณ์ไฟฟ้าทั้งหมดซึ่งจะทำให้ตรงส่วนของ Status ขึ้นเป็นคำว่า Loading แทน เมื่อการเชื่อมต่อเป็นปกติแล้วหน้าเว็บทำการ Refresh ตัวเอง Status ก็จะแสดงตามสถานะของอุปกรณ์ไฟฟ้าตามปกติ ซึ่งจะสะดวกในการดูสถานะเมื่อผู้ใช้งานไม่ได้อยู่หน้าจอแล้วกลับมาดูเมื่อการเชื่อมต่อเกิดมีปัญหาแล้วกลับเป็นปกติก่อนที่ผู้ใช้งานจะกลับมา ส่วนในหน้าของแอพพลิเคชั่นบนโทรศัพท์มือถือนั้นจะมีปุ่ม Refresh อยู่ด้านล่างซ้ายเพื่อใช้ในการ Refresh หน้าแอพพลิเคชั่นเพื่อให้ผู้ใช้งานสะดวกในการใช้งาน ซึ่งไม่ต้องออกแล้วเข้าแอพพลิเคชั่นใหม่ ในกรณีที่ Status ไม่แสดงตามสถานะของอุปกรณ์ไฟฟ้าตามปกติ

ทั้งนี้ การสั่งการควบคุมทั้งจากหน้าเว็บเพจและหน้าแอพพลิเคชั่นบนโทรศัพท์มือถือ อาจเกิดการล่าช้าในการแสดงผลเมื่อสั่งการควบคุมและค่า Status ที่จะแสดงออกมา ขึ้นอยู่กับการเชื่อมต่อของอินเตอร์เน็ตและจำนวนผู้ใช้งานใน ถ้าในขณะนั้นการเชื่อมต่อมีความเร็วมากและจำนวนผู้ใช้งานมีน้อยความล่าช้าในการแสดงผลก็จะน้อยลงหรืออาจไม่มีเลย

บทที่ 5
บทสรุปและวิจารณ์

5.1 บทสรุป

จากการทดลองการทำงานของระบบที่ได้สร้างขึ้นสามารถนำไปใช้สั่งการควบคุมการเปิด-ปิด รวมไปถึงการตั้งเวลาเปิด-ปิดอุปกรณ์ไฟฟ้าจากระยะไกล โดยผ่านระบบอินเตอร์เน็ต นอกจากนี้ยังสามารถสั่งการควบคุมอุปกรณ์ไฟฟ้าผ่านทางแอพพลิเคชั่นบนโทรศัพท์มือถือระบบปฏิบัติการ Windows mobile เพื่อสะดวกและง่ายต่อการใช้งาน ซึ่งสามารถนำไปประยุกต์ใช้ภายในบ้านหรืออาคารต่างๆ และยังสามารถแสดงสถานะปัจจุบันของอุปกรณ์ไฟฟ้าเมื่อทำการสั่งการควบคุมให้ผู้ใช้งานทราบ

จากการดำเนินงานที่ทางกลุ่มได้จัดทำโครงงานนี้ขึ้นมา ทำให้ได้รับความรู้และประสบการณ์ในส่วนของการศึกษาไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708 รวมไปถึงการใช้โปรแกรม Visual Studio 2008 เขียนแอพพลิเคชั่นโทรศัพท์มือถือให้แสดงผลที่ Windows Mobile Emulator และภาษาทางคอมพิวเตอร์ที่ใช้ในการศึกษาโครงงานนี้

5.2 บทวิจารณ์

จากการดำเนินงานที่ทางกลุ่มได้จัดทำโครงงานนี้ขึ้นมา การหาข้อมูลของบอร์ดไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708 มีให้ศึกษาน้อยมาก รวมทั้งการที่ทางกลุ่มไม่ชำนาญในการเขียนโปรแกรมที่ใช้ควบคุมไมโครคอนโทรลเลอร์ dsPIC33FJ128JP708 ทำให้มีการดำเนินการทำโครงงานเกิดความล่าช้า รวมไปถึงในส่วนของโทรศัพท์มือถือได้ใช้ยี่ห้อ O2 รุ่น atom เป็นเครื่องทดลองในการทดสอบแอพพลิเคชั่น ซึ่งในโทรศัพท์มือถือรุ่นนี้เป็นรุ่นที่มีชิปเซ็ตในการรับสัญญาณ Wireless ได้ไม่ดีเท่าที่ควร ทำให้การทำงานในการสั่งเปิด-ปิดอุปกรณ์ไฟฟ้ารวมไปถึงการแสดงค่าสถานะของอุปกรณ์ไฟฟ้ามีความล่าช้าในการทำงาน แต่อย่างไรก็ตามทางกลุ่มได้พยายามศึกษา สอบถามผู้รู้และแก้ไขปัญหาที่เกิดขึ้น เพื่อให้เป็นแนวทางในการพัฒนาต่อไปในอนาคต

5.3 แนวทางการพัฒนา

ในโครงงานนี้ได้ทำระบบควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าเป็นอุปกรณ์ต้นแบบ แต่ในการนำไปใช้จริงนั้นสามารถเพิ่มในส่วนของการรับค่าอินพุต แล้วส่งค่าที่รับได้นั้นเข้าไปที่ตัวบอร์ดไมโครคอนโทรลเลอร์แล้วแสดงผลออกมาที่หน้าเว็บหรือแอพพลิเคชั่นบนมือถือ เช่น ทำเซนเซอร์ตรวจจับความร้อนไว้ที่หม้อต้มน้ำ เมื่อน้ำเดือดแล้วจะแสดงสถานะให้ผู้ใช้งานทราบได้ และในการนำไปใช้งานจริงนั้นผู้ใช้งานต้องไปสมัครเพื่อขอใช้ IP Address เป็นของตัวเอง และกำหนดพอร์ตให้กับเราเตอร์ เพื่อให้สามารถสั่งการใช้งานบนเครือข่ายอินเตอร์เน็ตทั่วไปได้

บรรณานุกรม

ประจิน พลังสันติกุล, 2551. dsPIC30F Programming กับ MPLAB C คอมไพลเลอร์.กรุงเทพฯ : แอพซอฟต์เทค จำกัด.

บริษัทอีทีทีจำกัด. คู่มือการใช้งาน ET-dsPIC33WEB V1.0 Box.กรุงเทพ ฯ.

สัจจะ จรัสรุ่งรวี, 2550. เริ่มต้น Visual C# 2008 ฉบับสมบูรณ์. กรุงเทพ ฯ : ไอซีดี จำกัด.

ภาคผนวก

ภาคผนวก ก.

การ Download Code ให้กับตัวอุปกรณ์

การ Download Code ให้กับบอร์ด

หลังจากทำการเขียนโปรแกรมและส่งคอมไพล์จนได้ HEX เรียบร้อยแล้ว ขั้นตอนการพัฒนาโปรแกรมของบอร์ดก็จะเหลือเพียงการ Download Code ให้กับ MCU ซึ่งในขั้นตอนนี้จะต้องใช้โปรแกรม Pickit2ในการ Download Code ร่วมด้วย

การ Download Code ด้วย "ET-PGMPIC USB"

1. ต่อสายสัญญาณ ICD2 จาก ET-PGMPIC USB เข้ากับบอร์ด ET-dsPIC33WEB V1.0 โดยให้เลือกกำหนด Jumper "B/T" ของ ET-PGMPIC USB ไว้ในตำแหน่ง "B" (Target Board) พร้อมกับกำหนดสวิตช์เลือก Mode ของบอร์ด "ET-dsPIC33WEB V1.0" ไว้ในตำแหน่ง PGM (LED PGM สีแดงติดสว่าง)

2. สั่ง Run โปรแกรม Pickit2

3. ทำการคลิกเมาส์ที่เมนูคำสั่ง "Device Family ⟶ dsPIC33" ซึ่งเครื่องโปรแกรมจะทำการเชื่อมต่อกับ MCU พร้อมกับอ่านค่า Configuration ต่างๆของ MCU ขึ้นมาแสดงผลให้เห็นที่หน้าจอของโปรแกรม ให้เห็นในทันที ซึ่งในกรณีของบอร์ด "ET-dsPIC33WEB V1.0" นั้น ถ้าทุกอย่างถูกต้องที่หน้าจอของโปรแกรมจะแสดงข้อความ "dsPIC33 device found" พร้อมกับมีการแสดงเบอร์ของ MCU (Device) เป็น "dsPIC33FJ128GP708"

4. ให้ทำการส่ง Load HEX File ที่ได้จากการแปลโปรแกรมของ C30 โดยให้ทำการคลิกเมาส์ที่เมนูคำสั่ง "File ⟶ Import Hex" แล้วเลือก HEX File ที่ได้จาก การแปลคำสั่งของ C30 จะปรากฏข้อความ "Hex file successfully imported" ซึ่งหมายถึงการส่ง Load Hex ไฟล์เรียบร้อยแล้ว ซึ่งผู้ใช้สามารถส่ง Download Code จาก Hex File ให้กับหน่วยความจำของ MCUได้ทันที โดยการคลิกเมาส์ที่ปุ่มคำสั่ง "Write"

5. เครื่อง "ET-PGMPIC USB" จะเริ่มต้นทำการ Download Code ให้กับ MCU ในทันทีเมื่อการ Download Code เสร็จเรียบร้อยแล้ว จะมีข้อความ "Programming Successful"

6. หลังจากทำการ Download Code เรียบร้อยแล้ว ให้ทำการเลือกกำหนดการทางานของสวิตช์ Mode ของบอร์ด ET-dsPIC33WEB V1.0 ไปไว้ในตำแหน่ง Run โดยให้ตำแหน่งของสวิตช์ Mode อยู่ในตำแหน่งปล่อย และ LED Run สีเขียวติดสว่าง จากนั้นให้กดสวิตช์ Reset 1 ครั้ง บอร์ดก็จะเริ่มต้นทำงานตามคำสั่งที่ Download ให้แล้วในทันที

การดาวน์โหลด Webpage File ให้บอร์ด ET-dsPIC33WEB V1.0

ในตัวอย่างการทดลอง Web Server Control นี้ ไฟล์ต่างๆที่เป็นของ Webpage นั้น จะต้องนำไปเก็บไว้ในหน่วยความจำแบบภายนอก ซึ่งก็คือ EEPROM ที่เป็น SPI Memory Module เบอร์ 25LC256 ซึ่งได้ทำการติดตั้งไว้ในบอร์ดเป็นที่เรียบร้อยแล้ว โดยใน Code Program ที่สร้างขึ้นตามตัวอย่างนี้ จะทำให้เราสามารถใช้ Protocol ย่อยของ TCP/IP ในการส่งไฟล์ไปยังบอร์ดผ่านทางสายสัญญาณ Ethernet LAN ได้ทันที โดยให้ FTP Protocol โดยวิธีการนี้เราสามารถเรียกใช้ คำสั่ง FTP ใน Command Line เพื่อทำการ Login และส่งไฟล์ไปให้กับบอร์ดได้โดยตรง โดยมีลำดับขั้นตอนดังนี้

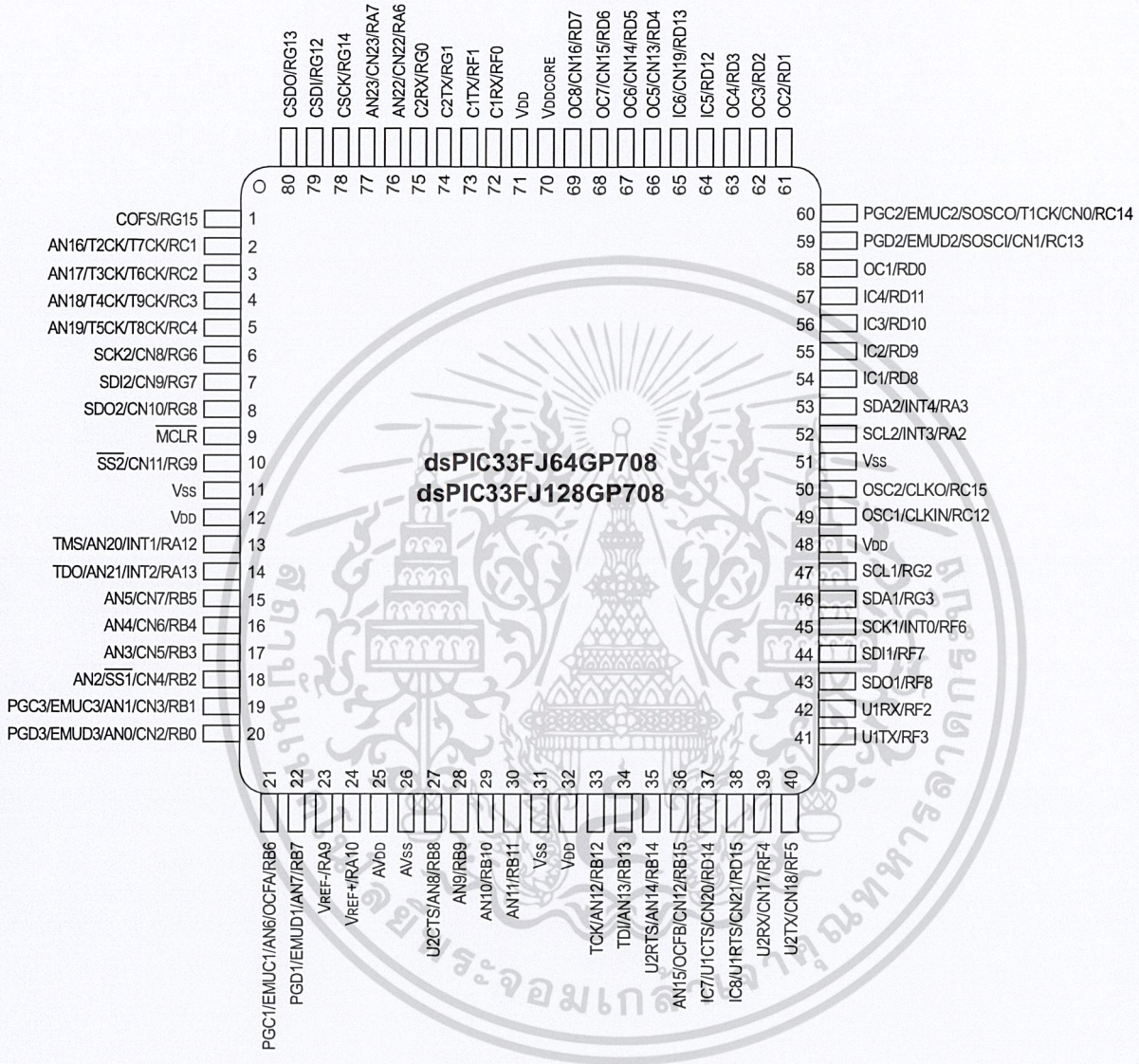
1. ปิดหน้าต่าง Command Prompt โดยคลิกเมาส์ที่ Shortcut ของ Command Prompt
2. ใช้คำสั่ง FTP เพื่อเชื่อมต่อกับ IP Address ของบอร์ด ET-dsPIC33WEB
3. เมื่อผลการ FTP ได้รับการเชื่อมต่อ Connected เป็นที่เรียบร้อยแล้วให้ทำการ Login โดยใช้ชื่อ “ftp” และใช้รหัสผ่าน Password เป็น “ett”
4. เมื่อสามารถทำการ Login ได้สำเร็จให้ใช้คำสั่ง PUT ในการส่งไฟล์ชื่อ “dspic33web.bin” โดยพิมพ์คำสั่งเป็น put dspic33web.bin แล้ว Enter
5. เมื่อการส่งไฟล์เสร็จเรียบร้อยแล้ว ให้ทำการพิมพ์คำสั่ง quit เพื่อออกจากการเชื่อมต่อกับ FTP Protocol ซึ่งหลังจากนี้ก็สามารถใช้งาน Web Server Control ของบอร์ด ET-dsPIC33WEB V1.0 ได้แล้ว
6. เปิดโปรแกรม Internet Explorer แล้ว พิมพ์หมายเลข IP Address ของบอร์ด หรือชื่อ DHCP ของบอร์ด ซึ่งก็คือ dspic33web ลงในช่อง Address แล้ว Enter เพื่อทดสอบ

ภาคผนวก ข.

Datasheet ของอุปกรณ์ที่นำมาใช้

Pin Diagrams (Continued)

80-Pin TQFP



2.0 CPU

Note:	This data sheet summarizes the features of this group of dsPIC33F devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to the *"dsPIC30F Family Reference Manual"* (DS70046).

The dsPIC33F CPU module has a 16-bit (data) modified Harvard architecture with an enhanced instruction set, including significant support for DSP. The CPU has a 24-bit instruction word with a variable length opcode field. The Program Counter (PC) is 23 bits wide and addresses up to 4M x 24 bits of user program memory space. The actual amount of program memory implemented varies by device. A single-cycle instruction prefetch mechanism is used to help maintain throughput and provides predictable execution. All instructions execute in a single cycle, with the exception of instructions that change the program flow, the double word move (`MOV.D`) instruction and the table instructions. Overhead-free program loop constructs are supported using the `DO` and `REPEAT` instructions, both of which are interruptible at any point.

The dsPIC33F devices have sixteen, 16-bit working registers in the programmer's model. Each of the working registers can serve as a data, address or address offset register. The 16th working register (W15) operates as a software Stack Pointer (SP) for interrupts and calls.

The dsPIC33F instruction set has two classes of instructions: MCU and DSP. These two instruction classes are seamlessly integrated into a single CPU. The instruction set includes many addressing modes and is designed for optimum C compiler efficiency. For most instructions, the dsPIC33F is capable of executing a data (or program data) memory read, a working register (data) read, a data memory write and a program (instruction) memory read per instruction cycle. As a result, three parameter instructions can be supported, allowing A + B = C operations to be executed in a single cycle.

A block diagram of the CPU is shown in Figure 2-1, and the programmer's model for the dsPIC33F is shown in Figure 2-2.

2.1 Data Addressing Overview

The data space can be addressed as 32K words or 64 Kbytes and is split into two blocks, referred to as X and Y data memory. Each memory block has its own independent Address Generation Unit (AGU). The MCU class of instructions operates solely through the X memory AGU, which accesses the entire memory map as one linear data space. Certain DSP instructions operate through the X and Y AGUs to support dual operand reads, which splits the data address space into two parts. The X and Y data space boundary is device-specific.

Overhead-free circular buffers (Modulo Addressing mode) are supported in both X and Y address spaces. The Modulo Addressing removes the software boundary checking overhead for DSP algorithms. Furthermore, the X AGU circular addressing can be used with any of the MCU class of instructions. The X AGU also supports Bit-Reversed Addressing to greatly simplify input or output data reordering for radix-2 FFT algorithms.

The upper 32 Kbytes of the data space memory map can optionally be mapped into program space at any 16K program word boundary defined by the 8-bit Program Space Visibility Page (PSVPAG) register. The program to data space mapping feature lets any instruction access program space as if it were data space.

The data space also includes 2 Kbytes of DMA RAM, which is primarily used for DMA data transfers, but may be used as general purpose RAM.

2.2 DSP Engine Overview

The DSP engine features a high-speed, 17-bit by 17-bit multiplier, a 40-bit ALU, two 40-bit saturating accumulators and a 40-bit bidirectional barrel shifter. The barrel shifter is capable of shifting a 40-bit value, up to 16 bits right or left, in a single cycle. The DSP instructions operate seamlessly with all other instructions and have been designed for optimal real-time performance. The `MAC` instruction and other associated instructions can concurrently fetch two data operands from memory while multiplying two W registers and accumulating and optionally saturating the result in the same cycle. This instruction functionality requires that the RAM memory data space be split for these instructions and linear for all others. Data space partitioning is achieved in a transparent and flexible manner through dedicating certain working registers to each address space.

2.3 Special MCU Features

The dsPIC33F features a 17-bit by 17-bit, single-cycle multiplier that is shared by both the MCU ALU and DSP engine. The multiplier can perform signed, unsigned and mixed-sign multiplication. Using a 17-bit by 17-bit multiplier for 16-bit by 16-bit multiplication not only allows you to perform mixed-sign multiplication, it also achieves accurate results for special operations, such as (-1.0) x (-1.0).

The dsPIC33F supports 16/16 and 32/16 divide operations, both fractional and integer. All divide instructions are iterative operations. They must be executed within a REPEAT loop, resulting in a total execution time of 19 instruction cycles. The divide operation can be interrupted during any of those 19 cycles without loss of data.

A 40-bit barrel shifter is used to perform up to a 16-bit, left or right shift in a single cycle. The barrel shifter can be used by both MCU and DSP instructions.

FIGURE 2-1: dsPIC33F CPU CORE BLOCK DIAGRAM

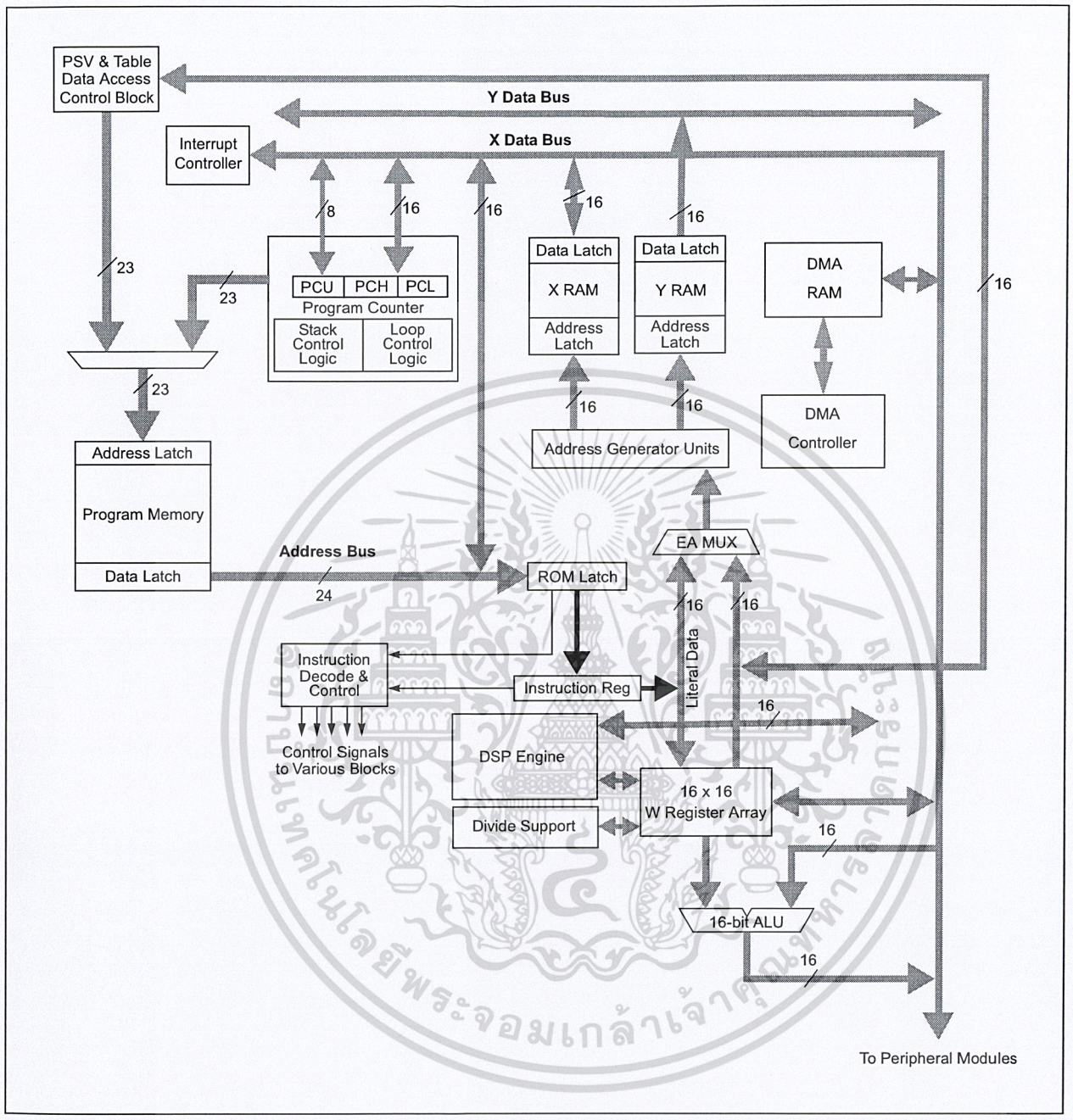
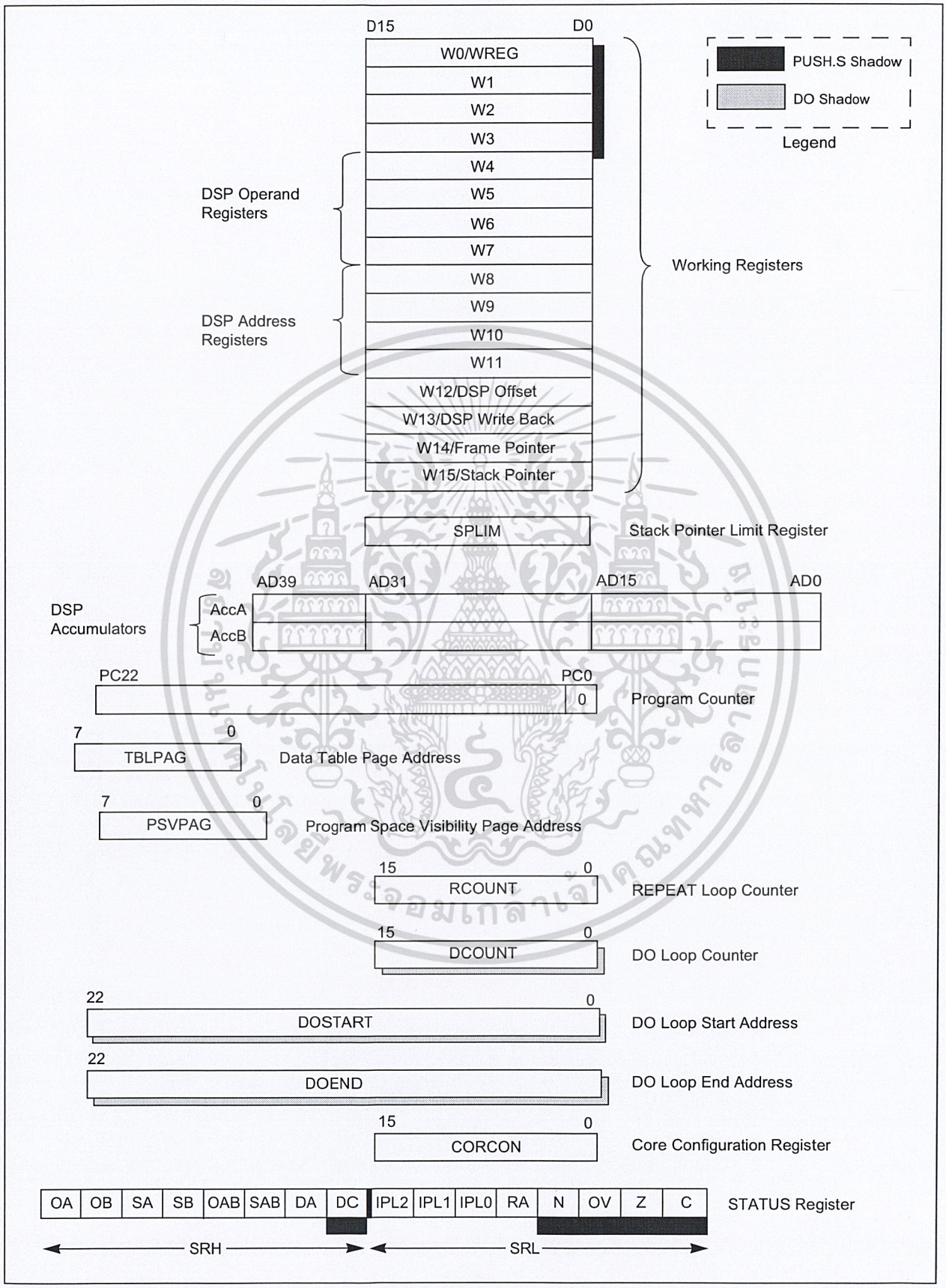


FIGURE 2-2: dsPIC33F PROGRAMMER'S MODEL



2.4 CPU Control Registers

REGISTER 2-1: SR: CPU STATUS REGISTER

R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R -0	R/W-0
OA	OB	SA(1)	SB(1)	OAB	SAB	DA	DC
bit 15							bit 8

R/W-0(2)	R/W-0(3)	R/W-0(3)	R-0	R/W-0	R/W-0	R/W-0	R/W-0
	IPL<2:0>(2)		RA	N	OV	Z	C
bit 7							bit 0

Legend:		
C = Clear only bit	R = Readable bit	U = Unimplemented bit, read as '0'
S = Set only bit	W = Writable bit	-n = Value at POR
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **OA:** Accumulator A Overflow Status bit
1 = Accumulator A overflowed
0 = Accumulator A has not overflowed

bit 14 **OB:** Accumulator B Overflow Status bit
1 = Accumulator B overflowed
0 = Accumulator B has not overflowed

bit 13 **SA:** Accumulator A Saturation 'Sticky' Status bit(1)
1 = Accumulator A is saturated or has been saturated at some time
0 = Accumulator A is not saturated

bit 12 **SB:** Accumulator B Saturation 'Sticky' Status bit(1)
1 = Accumulator B is saturated or has been saturated at some time
0 = Accumulator B is not saturated

bit 11 **OAB:** OA || OB Combined Accumulator Overflow Status bit
1 = Accumulators A or B have overflowed
0 = Neither Accumulators A or B have overflowed

bit 10 **SAB:** SA || SB Combined Accumulator 'Sticky' Status bit
1 = Accumulators A or B are saturated or have been saturated at some time in the past
0 = Neither Accumulator A or B are saturated

Note: This bit may be read or cleared (not set). Clearing this bit will clear SA and SB.

bit 9 **DA:** DO Loop Active bit
1 = DO loop in progress
0 = DO loop not in progress

bit 8 **DC:** MCU ALU Half Carry/$\overline{\text{Borrow}}$ bit
1 = A carry-out from the 4th low-order bit (for byte sized data) or 8th low-order bit (for word sized data) of the result occurred
0 = No carry-out from the 4th low-order bit (for byte sized data) or 8th low-order bit (for word sized data) of the result occurred

Note 1: This bit may be read or cleared (not set).

2: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the IPL if IPL<3> = 1. User interrupts are disabled when IPL<3> = 1.

3: The IPL<2:0> Status bits are read only when NSTDIS = 1 (INTCON1<15>).

REGISTER 2-1: SR: CPU STATUS REGISTER (CONTINUED)

bit 7-5 **IPL<2:0>:** CPU Interrupt Priority Level Status bits(2)
111 = CPU Interrupt Priority Level is 7 (15), user interrupts disabled
110 = CPU Interrupt Priority Level is 6 (14)
101 = CPU Interrupt Priority Level is 5 (13)
100 = CPU Interrupt Priority Level is 4 (12)
011 = CPU Interrupt Priority Level is 3 (11)
010 = CPU Interrupt Priority Level is 2 (10)
001 = CPU Interrupt Priority Level is 1 (9)
000 = CPU Interrupt Priority Level is 0 (8)

bit 4 **RA:** REPEAT Loop Active bit
1 = REPEAT loop in progress
0 = REPEAT loop not in progress

bit 3 **N:** MCU ALU Negative bit
1 = Result was negative
0 = Result was non-negative (zero or positive)

bit 2 **OV:** MCU ALU Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the magnitude which causes the sign bit to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 1 **Z:** MCU ALU Zero bit
1 = An operation which affects the Z bit has set it at some time in the past
0 = The most recent operation which affects the Z bit has cleared it (i.e., a non-zero result)

bit 0 **C:** MCU ALU Carry/$\overline{\text{Borrow}}$ bit
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note 1: This bit may be read or cleared (not set).

2: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the IPL if IPL<3> = 1. User interrupts are disabled when IPL<3> = 1.

3: The IPL<2:0> Status bits are read only when NSTDIS = 1 (INTCON1<15>).

REGISTER 2-2: CORCON: CORE CONTROL REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT(1)	DL<2:0>		
bit 15							bit 8

R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3(2)	PSV	RND	IF
bit 7							bit 0

Legend:	C = Clear only bit		
R = Readable bit	W = Writable bit	-n = Value at POR	'1' = Bit is set
0' = Bit is cleared	'x = Bit is unknown	U = Unimplemented bit, read as '0'	

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **US:** DSP Multiply Unsigned/Signed Control bit
1 = DSP engine multiplies are unsigned
0 = DSP engine multiplies are signed

bit 11 **EDT:** Early DO Loop Termination Control bit(1)
1 = Terminate executing DO loop at end of current loop iteration
0 = No effect

bit 10-8 **DL<2:0>:** DO Loop Nesting Level Status bits
111 = 7 DO loops active
•
•
001 = 1 DO loop active
000 = 0 DO loops active

bit 7 **SATA:** AccA Saturation Enable bit
1 = Accumulator A saturation enabled
0 = Accumulator A saturation disabled

bit 6 **SATB:** AccB Saturation Enable bit
1 = Accumulator B saturation enabled
0 = Accumulator B saturation disabled

bit 5 **SATDW:** Data Space Write from DSP Engine Saturation Enable bit
1 = Data space write saturation enabled
0 = Data space write saturation disabled

bit 4 **ACCSAT:** Accumulator Saturation Mode Select bit
1 = 9.31 saturation (super saturation)
0 = 1.31 saturation (normal saturation)

bit 3 **IPL3:** CPU Interrupt Priority Level Status bit 3(2)
1 = CPU interrupt priority level is greater than 7
0 = CPU interrupt priority level is 7 or less

bit 2 **PSV:** Program Space Visibility in Data Space Enable bit
1 = Program space visible in data space
0 = Program space not visible in data space

bit 1 **RND:** Rounding Mode Select bit
1 = Biased (conventional) rounding enabled
0 = Unbiased (convergent) rounding enabled

bit 0 **IF:** Integer or Fractional Multiplier Mode Select bit
1 = Integer mode enabled for DSP multiply ops
0 = Fractional mode enabled for DSP multiply ops

Note 1: This bit will always read as '0'.
2: The IPL3 bit is concatenated with the IPL<2:0> bits (SR<7:5>) to form the CPU interrupt priority level.

2.5 Arithmetic Logic Unit (ALU)

The dsPIC33F ALU is 16 bits wide and is capable of addition, subtraction, bit shifts and logic operations. Unless otherwise mentioned, arithmetic operations are 2's complement in nature. Depending on the operation, the ALU may affect the values of the Carry (C), Zero (Z), Negative (N), Overflow (OV) and Digit Carry (DC) Status bits in the SR register. The C and DC Status bits operate as Borrow and Digit Borrow bits, respectively, for subtraction operations.

The ALU can perform 8-bit or 16-bit operations, depending on the mode of the instruction that is used. Data for the ALU operation can come from the W register array, or data memory, depending on the addressing mode of the instruction. Likewise, output data from the ALU can be written to the W register array or a data memory location.

Refer to the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for information on the SR bits affected by each instruction.

The dsPIC33F CPU incorporates hardware support for both multiplication and division. This includes a dedicated hardware multiplier and support hardware for 16-bit-divisor division.

2.5.1 MULTIPLIER

Using the high-speed 17-bit x 17-bit multiplier of the DSP engine, the ALU supports unsigned, signed or mixed-sign operation in several MCU multiplication modes:

1. 16-bit x 16-bit signed
2. 16-bit x 16-bit unsigned
3. 16-bit signed x 5-bit (literal) unsigned
4. 16-bit unsigned x 16-bit unsigned
5. 16-bit unsigned x 5-bit (literal) unsigned
6. 16-bit unsigned x 16-bit signed
7. 8-bit unsigned x 8-bit unsigned

2.5.2 DIVIDER

The divide block supports 32-bit/16-bit and 16-bit/16-bit signed and unsigned integer divide operations with the following data sizes:

1. 32-bit signed/16-bit signed divide
2. 32-bit unsigned/16-bit unsigned divide
3. 16-bit signed/16-bit signed divide
4. 16-bit unsigned/16-bit unsigned divide

The quotient for all divide instructions ends up in W0 and the remainder in W1. 16-bit signed and unsigned `DIV` instructions can specify any W register for both the 16-bit divisor (Wn) and any W register (aligned) pair (W(m + 1):Wm) for the 32-bit dividend. The divide algorithm takes one cycle per bit of divisor, so both 32-bit/16-bit and 16-bit/16-bit instructions take the same number of cycles to execute.

2.6 DSP Engine

The DSP engine consists of a high-speed, 17-bit x 17-bit multiplier, a barrel shifter and a 40-bit adder/subtracter (with two target accumulators, round and saturation logic).

The dsPIC33F is a single-cycle, instruction flow architecture; therefore, concurrent operation of the DSP engine with MCU instruction flow is not possible. However, some MCU ALU and DSP engine resources may be used concurrently by the same instruction (e.g., ED, EDAC).

The DSP engine also has the capability to perform inherent accumulator-to-accumulator operations which require no additional data. These instructions are `ADD`, `SUB` and `NEG`.

The DSP engine has various options selected through various bits in the CPU Core Control register (CORCON), as listed below:

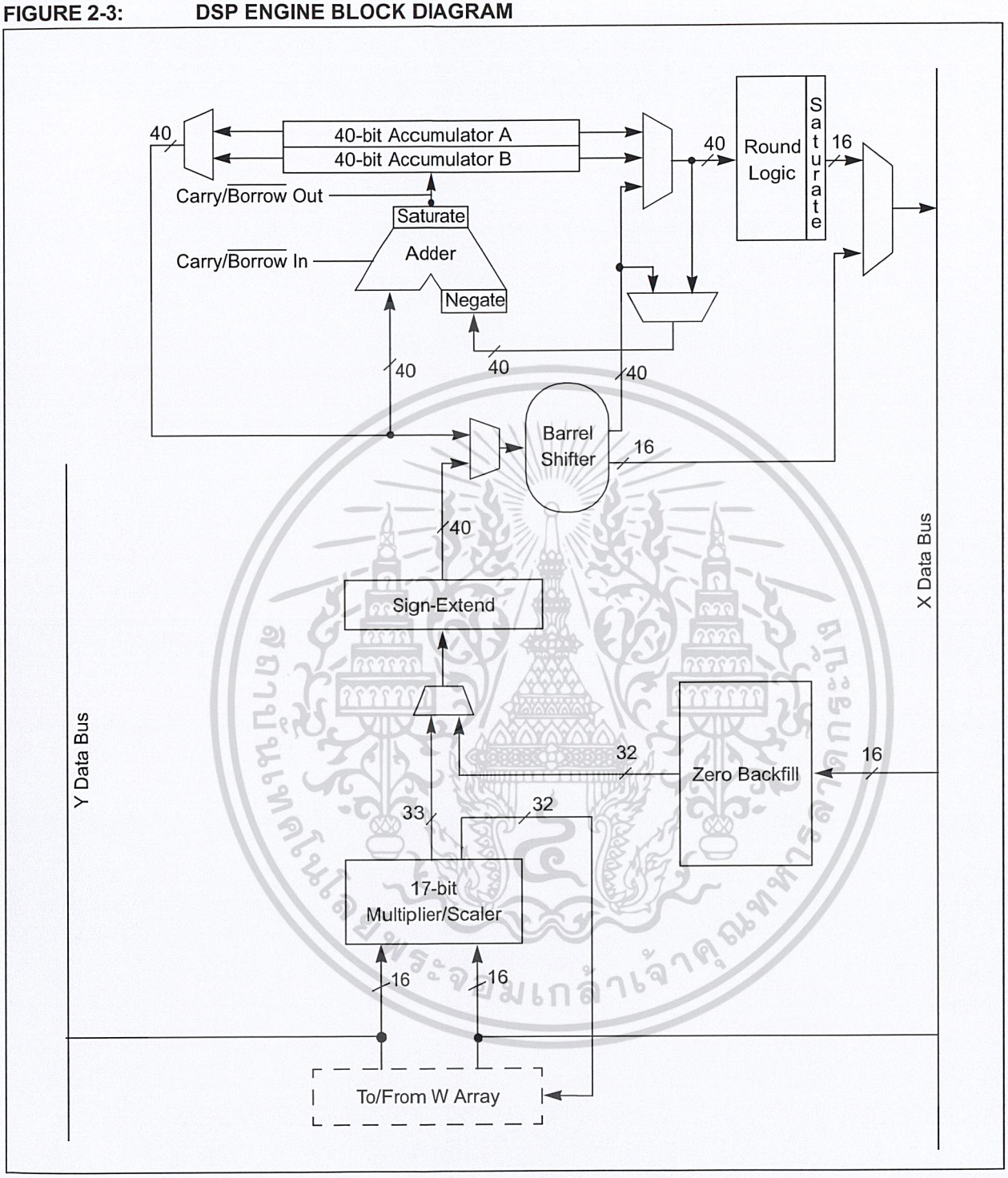
1. Fractional or integer DSP multiply (IF).
2. Signed or unsigned DSP multiply (US).
3. Conventional or convergent rounding (RND).
4. Automatic saturation on/off for AccA (SATA).
5. Automatic saturation on/off for AccB (SATB).
6. Automatic saturation on/off for writes to data memory (SATDW).
7. Accumulator Saturation mode selection (ACCSAT).

A block diagram of the DSP engine is shown in Figure 2-3.

TABLE 2-1: DSP INSTRUCTIONS SUMMARY

Instruction	Algebraic Operation	ACC Write Back
CLR	A = 0	Yes
ED	A = $(x - y)^2$	No
EDAC	A = A + $(x - y)^2$	No
MAC	A = A + (x * y)	Yes
MAC	A = A + x^2	No
MOVSAC	No change in A	Yes
MPY	A = x * y	No
MPY	A = x^2	No
MPY.N	A = – x * y	No
MSC	A = A – x * y	Yes

FIGURE 2-3: DSP ENGINE BLOCK DIAGRAM



2.6.1 MULTIPLIER

The 17-bit x 17-bit multiplier is capable of signed or unsigned operation and can multiplex its output using a scaler to support either 1.31 fractional (Q31) or 32-bit integer results. Unsigned operands are zero-extended into the 17th bit of the multiplier input value. Signed operands are sign-extended into the 17th bit of the multiplier input value. The output of the 17-bit x 17-bit multiplier/scaler is a 33-bit value which is sign-extended to 40 bits. Integer data is inherently represented as a signed two's complement value, where the MSb is defined as a sign bit. Generally speaking, the range of an N-bit two's complement integer is -2^{N-1} to $2^{N-1} - 1$. For a 16-bit integer, the data range is -32768 (0x8000) to 32767 (0x7FFF) including '0'. For a 32-bit integer, the data range is -2,147,483,648 (0x8000 0000) to 2,147,483,647 (0x7FFF FFFF).

When the multiplier is configured for fractional multiplication, the data is represented as a two's complement fraction, where the MSb is defined as a sign bit and the radix point is implied to lie just after the sign bit (QX format). The range of an N-bit two's complement fraction with this implied radix point is -1.0 to $(1 - 2^{1-N})$. For a 16-bit fraction, the Q15 data range is -1.0 (0x8000) to 0.999969482 (0x7FFF) including '0' and has a precision of 3.01518×10^{-5}. In Fractional mode, the 16 x 16 multiply operation generates a 1.31 product which has a precision of 4.65661×10^{-10}.

The same multiplier is used to support the MCU multiply instructions which include integer 16-bit signed, unsigned and mixed sign multiplies.

The MUL instruction may be directed to use byte or word sized operands. Byte operands will direct a 16-bit result, and word operands will direct a 32-bit result to the specified register(s) in the W array.

2.6.2 DATA ACCUMULATORS AND ADDER/SUBTRACTER

The data accumulator consists of a 40-bit adder/subtracter with automatic sign extension logic. It can select one of two accumulators (A or B) as its pre-accumulation source and post-accumulation destination. For the ADD and LAC instructions, the data to be accumulated or loaded can be optionally scaled via the barrel shifter prior to accumulation.

2.6.2.1 Adder/Subtracter, Overflow and Saturation

The adder/subtracter is a 40-bit adder with an optional zero input into one side, and either true, or complement data into the other input. In the case of addition, the carry/borrow input is active-high and the other input is true data (not complemented), whereas in the case of subtraction, the carry/borrow input is active-low and the other input is complemented. The adder/subtracter generates Overflow Status bits, SA/SB and OA/OB, which are latched and reflected in the STATUS register:

- Overflow from bit 39: this is a catastrophic overflow in which the sign of the accumulator is destroyed.
- Overflow into guard bits 32 through 39: this is a recoverable overflow. This bit is set whenever all the guard bits are not identical to each other.

The adder has an additional saturation block which controls accumulator data saturation, if selected. It uses the result of the adder, the Overflow Status bits described above and the SAT<A:B> (CORCON<7:6>) and ACCSAT (CORCON<4>) mode control bits to determine when and to what value to saturate.

Six STATUS register bits have been provided to support saturation and overflow; they are:

1. OA:
   AccA overflowed into guard bits
2. OB:
   AccB overflowed into guard bits
3. SA:
   AccA saturated (bit 31 overflow and saturation)
   or
   AccA overflowed into guard bits and saturated (bit 39 overflow and saturation)
4. SB:
   AccB saturated (bit 31 overflow and saturation)
   or
   AccB overflowed into guard bits and saturated (bit 39 overflow and saturation)
5. OAB:
   Logical OR of OA and OB
6. SAB:
   Logical OR of SA and SB

The OA and OB bits are modified each time data passes through the adder/subtracter. When set, they indicate that the most recent operation has overflowed into the accumulator guard bits (bits 32 through 39). The OA and OB bits can also optionally generate an arithmetic warning trap when set and the corresponding Overflow Trap Flag Enable bits (OVATE, OVBTE) in the INTCON1 register (refer to **Section 6.0 "Interrupt Controller"**) are set. This allows the user to take immediate action, for example, to correct system gain.

The SA and SB bits are modified each time data passes through the adder/subtracter, but can only be cleared by the user. When set, they indicate that the accumulator has overflowed its maximum range (bit 31 for 32-bit saturation or bit 39 for 40-bit saturation) and will be saturated (if saturation is enabled). When saturation is not enabled, SA and SB default to bit 39 overflow and, thus, indicate that a catastrophic overflow has occurred. If the COVTE bit in the INTCON1 register is set, SA and SB bits will generate an arithmetic warning trap when saturation is disabled.

The Overflow and Saturation Status bits can optionally be viewed in the STATUS Register (SR) as the logical OR of OA and OB (in bit OAB) and the logical OR of SA and SB (in bit SAB). This allows programmers to check one bit in the STATUS register to determine if either accumulator has overflowed, or one bit to determine if either accumulator has saturated. This would be useful for complex number arithmetic which typically uses both the accumulators.

The device supports three Saturation and Overflow modes:

1. Bit 39 Overflow and Saturation:
   When bit 39 overflow and saturation occurs, the saturation logic loads the maximally positive 9.31 (0x7FFFFFFFFF), or maximally negative 9.31 value (0x8000000000), into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. This is referred to as 'super saturation' and provides protection against erroneous data or unexpected algorithm problems (e.g., gain calculations).
2. Bit 31 Overflow and Saturation:
   When bit 31 overflow and saturation occurs, the saturation logic then loads the maximally positive 1.31 value (0x007FFFFFFF), or maximally negative 1.31 value (0x0080000000), into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. When this Saturation mode is in effect, the guard bits are not used (so the OA, OB or OAB bits are never set).
3. Bit 39 Catastrophic Overflow:
   The bit 39 Overflow Status bit from the adder is used to set the SA or SB bit, which remains set until cleared by the user. No saturation operation is performed and the accumulator is allowed to overflow (destroying its sign). If the COVTE bit in the INTCON1 register is set, a catastrophic overflow can initiate a trap exception.

2.6.2.2 Accumulator 'Write Back'

The MAC class of instructions (with the exception of MPY, MPY.N, ED and EDAC) can optionally write a rounded version of the high word (bits 31 through 16) of the accumulator that is not targeted by the instruction into data space memory. The write is performed across the X bus into combined X and Y address space. The following addressing modes are supported:

1. W13, Register Direct:
   The rounded contents of the non-target accumulator are written into W13 as a 1.15 fraction.
2. [W13]+ = 2, Register Indirect with Post-Increment:
   The rounded contents of the non-target accumulator are written into the address pointed to by W13 as a 1.15 fraction. W13 is then incremented by 2 (for a word write).

2.6.2.3 Round Logic

The round logic is a combinational block which performs a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the RND bit in the CORCON register. It generates a 16-bit, 1.15 data value which is passed to the data space write saturation logic. If rounding is not indicated by the instruction, a truncated 1.15 data value is stored and the least significant word is simply discarded.

Conventional rounding zero-extends bit 15 of the accumulator and adds it to the ACCxH word (bits 16 through 31 of the accumulator). If the ACCxL word (bits 0 through 15 of the accumulator) is between 0x8000 and 0xFFFF (0x8000 included), ACCxH is incremented. If ACCxL is between 0x0000 and 0x7FFF, ACCxH is left unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value tends to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding, except when ACCxL equals 0x8000. In this case, the Least Significant bit (bit 16 of the accumulator) of ACCxH is examined. If it is '1', ACCxH is incremented. If it is '0', ACCxH is not modified. Assuming that bit 16 is effectively random in nature, this scheme removes any rounding bias that may accumulate.

The SAC and SAC.R instructions store either a truncated (SAC), or rounded (SAC.R) version of the contents of the target accumulator to data memory via the X bus, subject to data saturation (see **Section 2.6.2.4 "Data Space Write Saturation"**). For the MAC class of instructions, the accumulator write-back operation will function in the same manner, addressing combined MCU (X and Y) data space though the X bus. For this class of instructions, the data is always subject to rounding.

2.6.2.4 Data Space Write Saturation

In addition to adder/subtracter saturation, writes to data space can also be saturated but without affecting the contents of the source accumulator. The data space write saturation logic block accepts a 16-bit, 1.15 fractional value from the round logic block as its input, together with overflow status from the original source (accumulator) and the 16-bit round adder. These inputs are combined and used to select the appropriate 1.15 fractional value as output to write to data space memory.

If the SATDW bit in the CORCON register is set, data (after rounding or truncation) is tested for overflow and adjusted accordingly, For input data greater than 0x007FFF, data written to memory is forced to the maximum positive 1.15 value, 0x7FFF. For input data less than 0xFF8000, data written to memory is forced to the maximum negative 1.15 value, 0x8000. The Most Significant bit of the source (bit 39) is used to determine the sign of the operand being tested.

If the SATDW bit in the CORCON register is not set, the input data is always passed through unmodified under all conditions.

2.6.3 BARREL SHIFTER

The barrel shifter is capable of performing up to 16-bit arithmetic or logic right shifts, or up to 16-bit left shifts in a single cycle. The source can be either of the two DSP accumulators or the X bus (to support multi-bit shifts of register or memory data).

The shifter requires a signed binary value to determine both the magnitude (number of bits) and direction of the shift operation. A positive value shifts the operand right. A negative value shifts the operand left. A value of '0' does not modify the operand.

The barrel shifter is 40 bits wide, thereby obtaining a 40-bit result for DSP shift operations and a 16-bit result for MCU shift operations. Data from the X bus is presented to the barrel shifter between bit positions 16 to 31 for right shifts, and between bit positions 0 to 16 for left shifts.

10.0 I/O PORTS

Note:	This data sheet summarizes the features of this group of dsPIC33F devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to the *"dsPIC30F Family Reference Manual"* (DS70046).

All of the device pins (except VDD, VSS, MCLR and OSC1/CLKIN) are shared between the peripherals and the parallel I/O ports. All I/O input ports feature Schmitt Trigger inputs for improved noise immunity.

10.1 Parallel I/O (PIO) Ports

A parallel I/O port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pin. The logic also prevents "loop through", in which a port's digital output can drive the input of a peripheral that shares the same pin. Figure 10-1 shows how ports are shared with other peripherals and the associated I/O pin to which they are connected.

When a peripheral is enabled and actively driving an associated pin, the use of the pin as a general purpose output pin is disabled. The I/O pin may be read, but the output driver for the parallel port bit will be disabled. If a peripheral is enabled, but the peripheral is not actively driving a pin, that pin may be driven by a port.

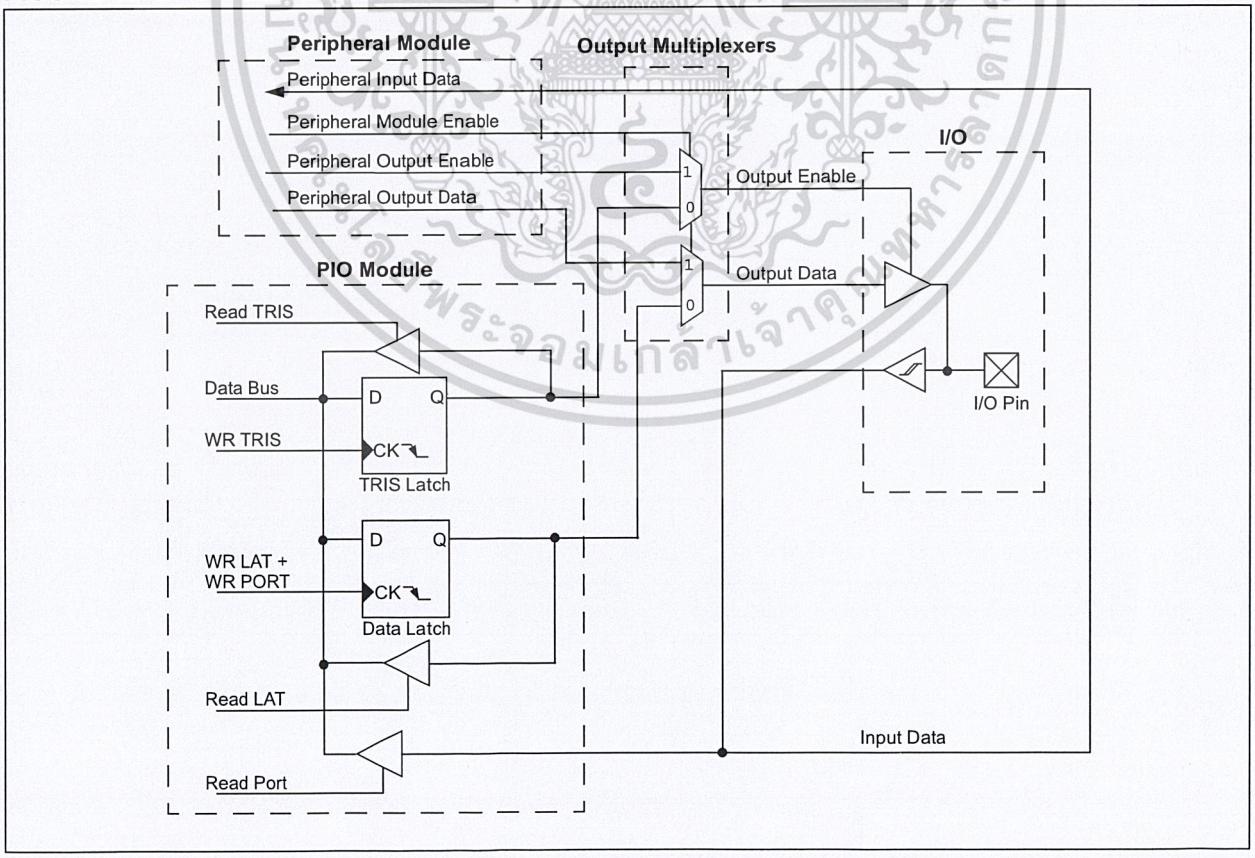
All port pins have three registers directly associated with their operation as digital I/O. The data direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx), read the latch. Writes to the latch, write the latch. Reads from the port (PORTx), read the port pins, while writes to the port pins, write the latch.

Any bit and its associated data and control registers that are not valid for a particular device will be disabled. That means the corresponding LATx and TRISx registers and the port pins will read as zeros.

When a pin is shared with another peripheral or function that is defined as an input only, it is nevertheless regarded as a dedicated port because there is no other competing source of outputs. An example is the INT4 pin.

Note:	The voltage on a digital input pin can be between -0.3V to 5.6V.

FIGURE 10-1: BLOCK DIAGRAM OF A TYPICAL SHARED PORT STRUCTURE



10.2 Open-Drain Configuration

In addition to the PORT, LAT and TRIS registers for data control, each port pin can also be individually configured for either digital or open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each port. Setting any of the bits configures the corresponding pin to act as an open-drain output.

The open-drain feature allows the generation of outputs higher than VDD (e.g., 5V) on any desired digital only pins by using external pull-up resistors. (The open-drain I/O feature is not supported on pins which have analog functionality multiplexed on the pin.) The maximum open-drain voltage allowed is the same as the maximum VIH specification. The open-drain output feature is supported for both port pin and peripheral configurations.

10.3 Configuring Analog Port Pins

The use of the ADxPCFGH, ADxPCFGL and TRIS registers control the operation of the ADC port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) is converted.

Clearing any bit in the ADxPCFGH or ADxPCFGL register configures the corresponding bit to be an analog pin. This is also the Reset state of any I/O pin that has an analog (ANx) function associated with it.

Note:	In devices with two ADC modules, if the corresponding PCFG bit in either AD1PCFGH(L) and AD2PCFGH(L) is cleared, the pin is configured as an analog input.

When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level).

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins) can cause the input buffer to consume current that exceeds the device specifications.

Note:	The voltage on an analog input pin can be between -0.3V to (VDD + 0.3 V).

10.4 I/O Port Write/Read Timing

One instruction cycle is required between a port direction change or port write operation and a read operation of the same port. Typically, this instruction would be a NOP.

10.5 Input Change Notification

The input change notification function of the I/O ports allows the dsPIC33F devices to generate interrupt requests to the processor in response to a change-of-state on selected input pins. This feature is capable of detecting input change-of-states even in Sleep mode, when the clocks are disabled. Depending on the device pin count, there are up to 24 external signals (CN0 through CN23) that can be selected (enabled) for generating an interrupt request on a change-of-state.

There are four control registers associated with the CN module. The CNEN1 and CNEN2 registers contain the CN interrupt enable (CNxIE) control bits for each of the CN input pins. Setting any of these bits enables a CN interrupt for the corresponding pins.

Each CN pin also has a weak pull-up connected to it. The pull-ups act as a current source that is connected to the pin and eliminate the need for external resistors when push button or keypad devices are connected. The pull-ups are enabled separately using the CNPU1 and CNPU2 registers, which contain the weak pull-up enable (CNxPUE) bits for each of the CN pins. Setting any of the control bits enables the weak pull-ups for the corresponding pins.

Note:	Pull-ups on change notification pins should always be disabled whenever the port pin is configured as a digital output.

EXAMPLE 10-1: PORT WRITE/READ EXAMPLE

```
MOV    0xFF00, W0          ; Configure PORTB<15:8> as inputs
MOV    W0, TRISBB          ; and PORTB<7:0> as outputs
NOP                        ; Delay 1 cycle
btss   PORTB, #13          ; Next Instruction
```