

## ควบคุมและแสดงผล LED บน RCM2200 ผ่าน Browser โดยใช้ AJAX

โดย : นายสมเกียรติ แข็งการ วิชา.บ.คอมพิวเตอร์ มทร. (นักศึกษาฝึกงาน)

ติดต่อ : [smokeec16@hotmail.com](mailto:smokeec16@hotmail.com)



RCM2200 เป็น Core Module Microprocessors ของตระกูล Rabbit ออกแบบมาเป็นพิเศษสำหรับการประยุกต์ใช้งานในพื้นที่ขนาดเล็ก RCM2200 ได้จัดการแก้ปัญหาการควบคุมแบบฝังตัวได้อย่างสมบูรณ์ด้วยขนาดเพียงครึ่งหนึ่งของบัตรเครดิตเท่านั้น โดยใช้ Microprocessors Rabbit2000 ที่มีประสิทธิภาพสูง RCM2200 ประกอบด้วยคุณสมบัติพิเศษสำหรับการควบคุมแบบฝังตัวเช่น หน่วยความจำโปรแกรมแบบ Flash, หน่วยความจำแบบ SRAM, พอร์ตอนุกรม, พอร์ตอินพุต / เอาต์พุต, Real-Time Clock และ พอร์ต Ethernet

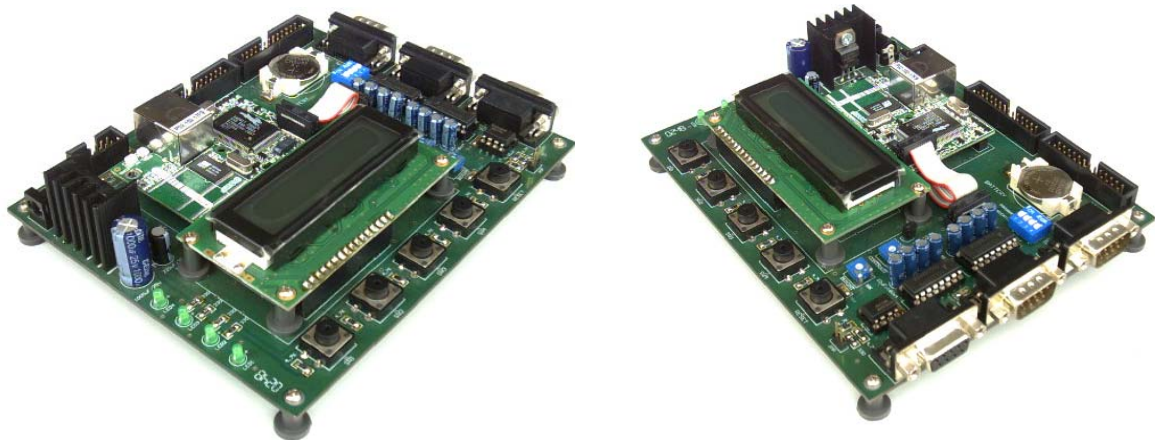
การออกแบบด้วย Rabbit Core Module หรือแบบลงบอร์ดสำเร็จรูปของตระกูล Rabbit Core ถูกออกแบบมาเพื่อให้สะดวกในการพัฒนาและการนำระบบฝังตัว (Embedded System) ไปใช้งานซึ่ง Rabbit Core ถูกขีดความสามารถโดย Microprocessors ตระกูล Rabbit ที่มีประสิทธิภาพสูงขนาด 8 Bit พร้อมด้วยคุณสมบัติเพิ่มเติมและชุดคำสั่งภาษา C ที่ถูกออกแบบมาเพื่อใช้กับระบบพัฒนา โดยที่ Rabbit Core จะติดตั้งบนแผงวงจรหลักที่ผู้ใช้ออกแบบและทำหน้าที่เป็นไมโครโปรเซสเซอร์ควบคุมระบบ แม้จะมีขนาดเล็กแต่ก็เต็มไปด้วยประสิทธิภาพ Core Module เหล่านี้บรรจุไว้ได้อย่างสมบูรณ์แบบเพื่อใช้สำหรับควบคุมและการสื่อสาร

### คุณสมบัติ

- เป็น CPU ขนาด 8 BIT ตัวถังแบบ 100PIN PQFP
- การเชื่อมต่อ Ethernet 10Base-T, RJ-45 2 LEDs
- หน่วยความจำ Flash 256K และ Static RAM 128K
- 4 CH 8 BIT TIMERS และ 2 CH 10 BIT TIMERS

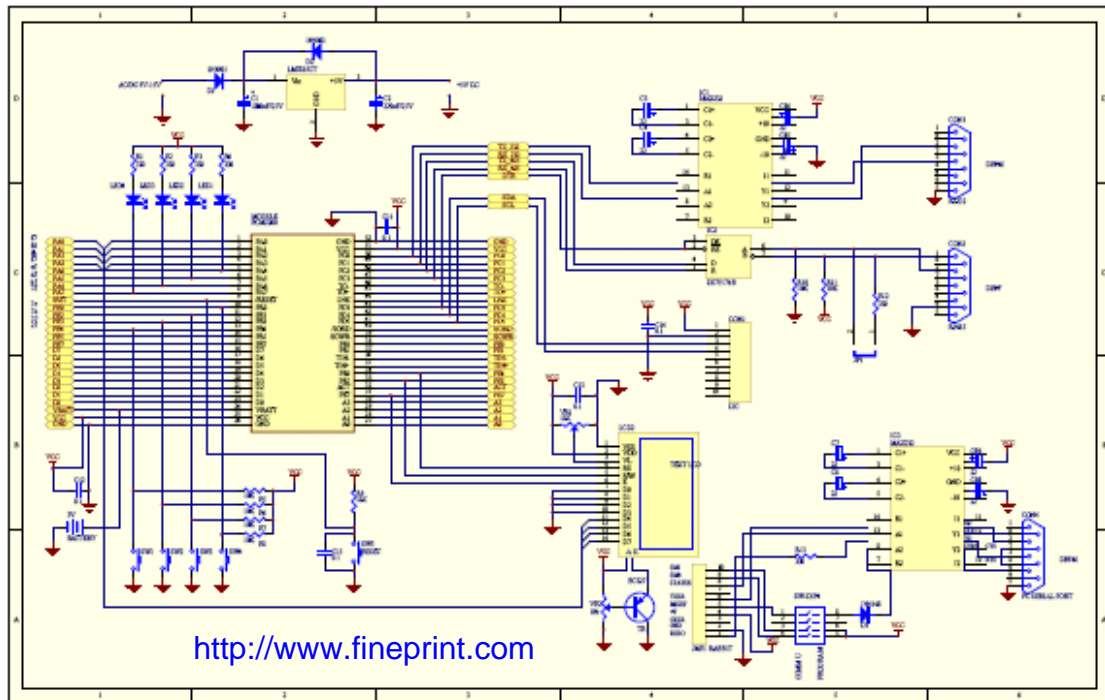
- ใช้งานได้ที่แรงดัน 4.75-5.25 V DC, 134 mA
- อุณหภูมิ -40°C to + 70°C
- ขนาดของตัวบอร์ด 2.3" x 1.6" x 0.86" (59 x 41 x 22 mm)
- 5 CH Parallel Port, 4 CH Serial Port, สามารถต่อหน่วยความจำต่างๆ ได้โดยตรงจาก CPU
- ความถี่ที่ใช้งาน 22.1 MHz

### ชุดพัฒนา RCM2200

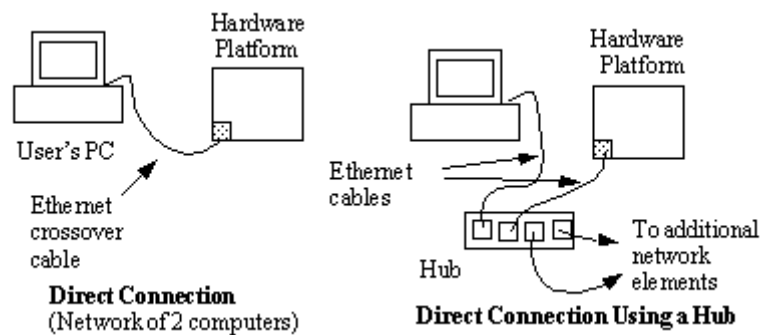


โดยภายในชุดพัฒนานี้จะประกอบไปด้วยอุปกรณ์ต่างๆมากมาย อย่างเช่น LED, Switch, LCD, Serial Port เป็นต้น สะดวกต่อการพัฒนาอย่างมาก

### Schematic



### การเชื่อมต่อ



การเชื่อมต่อใช้มาตรฐาน Ethernet 10Base-T, RJ-45 สามารถที่จะเชื่อมต่อผ่าน Hub/Switch หรือ เชื่อมต่อเข้ากับ PC โดยตรงผ่านสาย Cross ก็ได้

### การโปรแกรม RCM2200

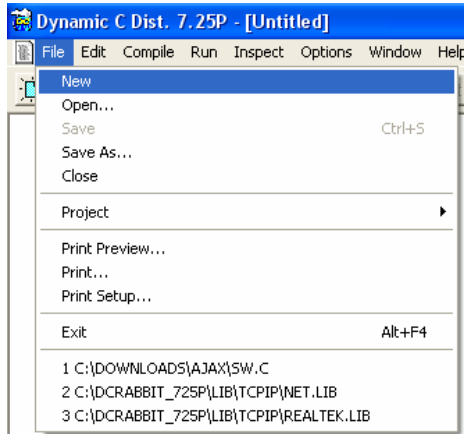
โปรแกรมที่ใช้พัฒนาคือโปรแกรม Dynamic C ของ Z-WORLD และยังมี Library และ โปรแกรมตัวอย่างต่างๆ ให้ผู้ใช้ด้วย

### เริ่มต้นเขียนโปรแกรม

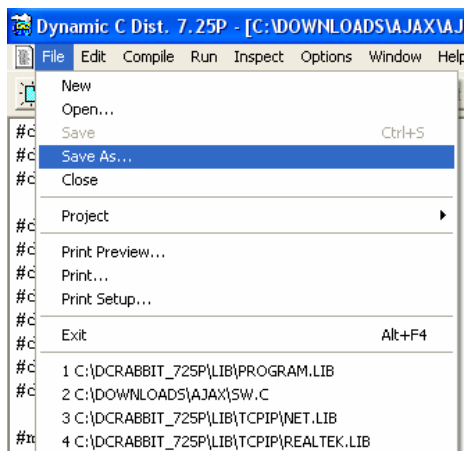
1. เลือกโปรแกรมโดยไปที่ Start > All Programs > Dynamic C Premier 7.25P > Dynamic C 7.25P ดังรูป



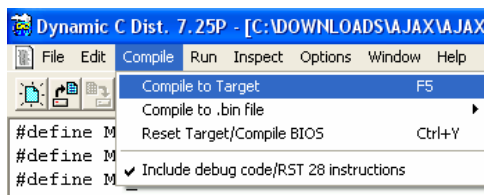
2. เลือกที่ File > New เพื่อสร้างงานชิ้นใหม่ จากนั้นพิมพ์ Code โปรแกรมตามตัวอย่างข้างล่างนี้



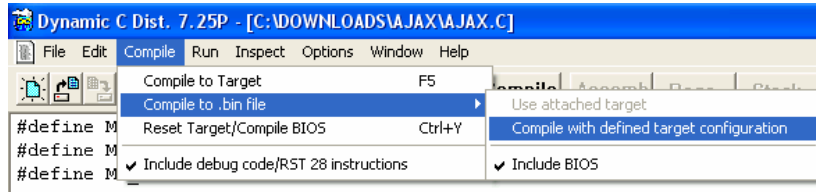
3. จากนั้นทำการ Save โปรแกรมให้เป็น File.c โดยเลือกที่ File > Save As...



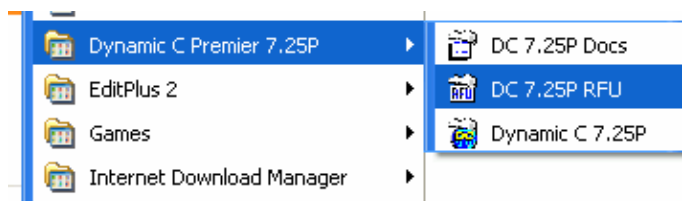
4. หลังจากนั้นก็ทำการ Compile โปรแกรมโดยเลือกไปที่ Compile > Compile to Target หรือ กดปุ่ม F5 ก็ได้



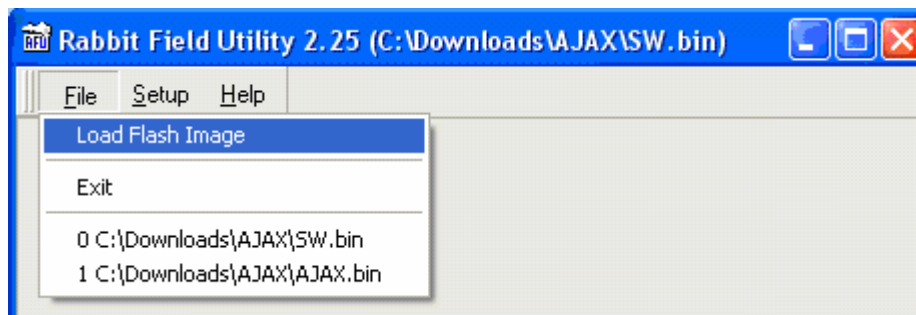
5. หากทำการเขียนโปรแกรมถูกต้อง จะสามารถ Compile ได้จนเสร็จสมบูรณ์ ขั้นตอนต่อไปคือการทำให้เป็น File.bin เพื่อที่จะ Load ไปยัง Rabbit Board ต่อไป โดยทำการเลือกที่ Compile > Compile to .bin file > Compile with defined target configuration โดย File.bin จะอยู่ใน Directory เดียวกันกับ File.c ที่ทำการ Save ไปก่อนหน้านี้



6. ต่อไปจะเป็นการ Load โปรแกรมไปที่ตัว Rabbit Board โดยเลือกที่ Start > All Programs > Dynamic C Premier 7.25P > Dynamic C 7.25P RFU ดังรูป



7. เลือก File.bin ที่จะทำการ Load โดยเลือก File > Load Flash Image จากนั้นก็เลือก File ที่ต้องการแล้วกด OK โปรแกรมจะถูก Load ไปยัง Rabbit Board



8. หลังจาก Load โปรแกรมเสร็จแล้ว ขั้นตอนสุดท้ายก่อนการใช้งานจะต้องทำการเลื่อน Dip Switch ที่ชุดพัฒนาไปที่ตำแหน่ง Communication จากตำแหน่งเดิมคือ Program เท่านั้นเป็นอันเสร็จ

### การเขียนโปรแกรม

โดยตัวโปรแกรมจะแบ่งออกเป็นสองส่วนใหญ่ๆคือส่วนที่เป็นภาษา c ที่คอยให้บริการ HTTP Server จะรองรับคำสั่งจาก Browser และจาก Switch สั่งให้ LED ติด/ดับ และอีกส่วนคือส่วนที่เป็น Browser เขียนด้วยภาษา html ทำหน้าที่สั่งงานให้ LED ติด/ดับ และเป็น LED Monitor โดยใช้ AJAX ซึ่ง Web Page จะไม่ Refresh หรือไม่กระพริบขณะทำงานให้เห็น

1. ส่วนที่เป็น HTTP Server โดยเขียนด้วยภาษา C (File.c)

```

#define MY_IP_ADDRESS "192.168.1.100" //กำหนดค่า IP Address,
#define MY_GATEWAY "192.168.1.1" //Gateway และ
#define MY_NETMASK "255.255.255.0" // Subnet Mask

#mmap xmem
#use "dcrtcp.lib" //Library ที่จำเป็นต้องใช้ทางด้าน Network
#use "http.lib"

#ximport "C:\Downloads\AJAX\ajax.shtml" index_html //นำเข้า File จากภายนอก
#ximport "C:\Downloads\AJAX\on.gif" ledon.gif //แล้วทำการตั้งชื่อใหม่
#ximport "C:\Downloads\AJAX\off.gif" ledoff.gif //นั่นก็คือที่เป็น Browser

const HttpType http_types[] ={
    { ".shtml", "text/html", shtml_handler}, // ssi
    { ".html", "text/html", NULL}, // html
    { ".cgi", "", NULL}, // cgi
    { ".gif", "image/gif", NULL}
};

char buffer[256]; //ประกาศตัวแปรเพื่อกำหนดขนาดของข้อความที่จะส่งไปยัง Browser
char buffer2[256];
const char teststron[] = "%d"; //ประกาศตัวแปรแบบ Static Char
const char teststroff[] = "%d"; //เพื่อเก็บข้อความที่จะส่งไปยัง Browser
const char ledon[] = "%d on";
const char ledoff[] = "%d off";

int i_led;
int status(HttpState* state){ //ประกาศ Function ตรวจสอบสถานะของ LEDs
    buffer2[0]=0; //กำหนดค่าเริ่มต้นที่ตำแหน่งแรก
    for(i_led=4;i_led<=7;i_led++){
        if(BitRdPortI(PADR,4) == 1){ //ถ้า LED ดับ
            sprintf(buffer,teststroff,0); //ให้นำค่าใน teststroff ไปเก็บในตัวแปร
            //buffer
        }
        else{ //ถ้า LED ติด
            sprintf(buffer,teststron,1); //ให้นำค่าใน teststron ไปเก็บในตัวแปร
            //buffer
        }
        strcat(buffer2, buffer); //นำข้อความ buffer ไปต่อท้าย buffer2
    }
    cgi_sendstring(state,buffer2); //แล้วส่งค่าใน buffer ไปยัง Browser
    return 0;
}

```

```

int toggle1(HttpState* state){           //ประกาศ Function สั่งให้ LED1 ติด/ดับ
    if(BitRdPortI(PADR,4) == 1){        //ถ้า LED1 ดับ
        BitWrPortI(PADR,&PADRShadow,0,4); //สั่งให้ LED1 ติด
        sprintf(buffer, ledon,1);        //ให้นำค่าใน teststron ไปเก็บในตัวแปร buffer
        cgi_sendstring(state,buffer);    //แล้วส่งค่าใน buffer ไปยัง Browser
    }
    else{                                //ถ้า LED1 ติด
        BitWrPortI(PADR,&PADRShadow,1,4); //สั่งให้ LED1 ดับ
        sprintf(buffer, ledoff,1);       //ให้นำค่าใน teststroff ไปเก็บในตัวแปร buffer
        cgi_sendstring(state,buffer);    //แล้วส่งค่าใน buffer ไปยัง Browser
    }
    return 0;
}
int toggle2(HttpState* state){
    if(BitRdPortI(PADR,5) == 1){
        BitWrPortI(PADR,&PADRShadow,0,5);
        sprintf(buffer, ledon,2);
        cgi_sendstring(state,buffer);
    }
    else{
        BitWrPortI(PADR,&PADRShadow,1,5);
        sprintf(buffer, ledoff,2);
        cgi_sendstring(state,buffer);
    }
    return 0;
}
int toggle3(HttpState* state){
    if(BitRdPortI(PADR,6) == 1){
        BitWrPortI(PADR,&PADRShadow,0,6);
        sprintf(buffer, ledon,3);
        cgi_sendstring(state,buffer);
    }
    else{
        BitWrPortI(PADR,&PADRShadow,1,6);
        sprintf(buffer, ledoff,3);
        cgi_sendstring(state,buffer);
    }
    return 0;
}
int toggle4(HttpState* state){
    if(BitRdPortI(PADR,7) == 1){
        BitWrPortI(PADR,&PADRShadow,0,7);
        sprintf(buffer, ledon,4);
        cgi_sendstring(state,buffer);
    }
    else{
        BitWrPortI(PADR,&PADRShadow,1,7);

```

```

        sprintf(buffer, ledoff,4);
        cgi_sendstring(state,buffer);
    }
    return 0;
}

void LightLED(int ledport){ //ประกาศ Function สั่งให้ LED ติด/ดับ จากการกด Switch
    if(ledport == 4) ledport = 4; //กำหนด Port ให้กับ LED
    if(ledport == 3) ledport = 5; //4 = LED1, 5 = LED2
    if(ledport == 2) ledport = 6; //6 = LED3, 7 = LED4
    if(ledport == 0) ledport = 7;

    if(BitRdPortI(PADR,ledport) == 1) //ถ้า LED ดับ
        BitWrPortI(PADR,&PADRShadow,0 , ledport); //สั่งให้ LED ติด
    else
        BitWrPortI(PADR,&PADRShadow,1 , ledport); //สั่งให้ LED ดับ
}

main(){

    int vs1,vs2,vs3,vs4;
    int sport,vswitch;
    int i;
    i=4;
    vs1=vs2=vs3=vs4=0;

    WrPortI(SPCR, & SPCRShadow, 0x84); //Reset Port ทั้งหมด
    WrPortI(PADR, & PADRShadow, 0xff); //โดยให้ LEDs ดับทั้งหมด

    sock_init();
    http_init();
    tcp_reserveport(80); //กำหนด Port สื่อสาร

    while(1){ //แบ่งการทำงานออกเป็น 2 Task หลักๆ
        costate{ //Task1 ทำงานด้าน Web Server
            http_handler();
        }
        costate{ //Task2 ทำงานด้านการตรวจสอบ Switch
            while(i>=1){
                if(i == 1) i=0;
                waitFor(DelayMs(50)); //หน่วงเวลารอการกด Switch
                if(!BitRdPortI(PBDR,i)){ //ตรวจสอบการกด Switch
                    LightLED(i); //ถ้ามีการกด Switch
                    i--; //ให้แสดงผล LED ติด/ดับ
                    if(i <= 0) i=4; //และลดค่า i เพื่อตรวจสอบSwitchต่อไป
                }
            }
        }
    }
}

```



```

if(!xmlhttp && document.createElement){
    xmlhttp = new XMLHttpRequest();
}
return xmlhttp;
}
//จบการประกาศ XmlHttp Function รูปแบบของการใช้งาน AJAX
function toggle(){
    //ประกาศ Function เพื่อสั่งให้ LEDs ติด/ดับ
    stopInterval();
    var url = "http://" + window.location.hostname + "/toggle" +
this.form.data.value + ".cgi";
    xmlhttp = uzXmlHttp(); //เรียกใช้ AJAX
    xmlhttp.open("POST", url, true); //ส่งข้อมูลแบบ POST ไปยัง url ที่กำหนด
    xmlhttp.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded"); //กำหนดส่วนหัวของ Browser
    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4) { //ตรวจสอบการเชื่อมต่อ
            if (xmlhttp.status == 200) { //รวมไปถึงสถานะการส่งข้อมูล
                saveResult(xmlhttp.responseText); //ถ้าถูกต้อง Browser
            } //จะได้รับ Text
            else { //กลับมา
                saveResult(xmlhttp.responseText); //และนำแสดงผลต่อไป
            }
        }
    };
    xmlhttp.send(null); //คำสั่งส่งข้อมูลไปยัง url ที่กำหนด
    startInterval();
}
function status(){
    //ประกาศ Function เพื่อตรวจสอบสถานะของ LEDs
    var url = "http://" + window.location.hostname + "/status.cgi";
    xmlhttp = uzXmlHttp();
    xmlhttp.open("POST", url, true);
    xmlhttp.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4) {
            if (xmlhttp.status == 200) {
                saveStatus (xmlhttp.responseText);
            }
            else {
                saveStatus (xmlhttp.responseText);
            }
        }
    };
    xmlhttp.send(null);
}
function saveStatus(statusText){
    //ประกาศ Function เพื่อแสดงผล LEDs บน Browser
    for(var i_led=0;i_led<=3;i_led++){
        var Textstatus = statusText.substring(i_led,i_led+1);
    }
}

```

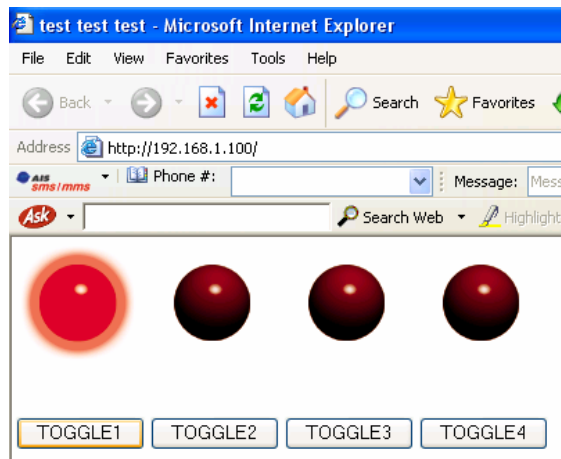


```
</table>  
</form>  
</body>  
</html>
```

<!-- ติด/ดับ -->

#### การทดสอบโปรแกรม

1. เปิดโปรแกรม Web Browser ขึ้นมา อาจจะใช้ Microsoft Internet Explorer ก็ได้ แล้วพิมพ์หมายเลข IP Address ลงไปในช่อง Address ในที่นี้ใช้ IP Address : 192.168.1.100 จะพบกับ Web Page ที่เราทำการ Import เข้าไปเป็นหน้าแรก โดย Web Server จะทำการ Set LED ให้ดับทั้งหมดก่อน เมื่อกดที่ปุ่ม TOGGLE1 ที่ Web Browser LED1 จะติดทั้งที่ Rabbit Board และ Web Browser และเมื่อกดที่ปุ่ม TOGGLE1 อีกครั้ง LED1 ก็จะดับทั้งที่ Rabbit Board และ Web Browser เช่นกัน โดย Web Page จะไม่ Refresh หรือกระพริบให้เห็นเป็นที่รำคาญตาด้วยคุณสมบัติของ AJAX



#### เอกสารอ้างอิง

- JAN AXELSON, Embedded ETHERNET AND INTERNET COMPLETE, Lakeview Research LLC, 2003
- ฉลองชัย จงประเสริฐพร, วรวิภา ทำพระนา, CGI WEB Programming การพัฒนาโปรแกรมใช้งานบนเครือข่ายอินเทอร์เน็ต, บริษัท จีรวัฒน์ เอ็กเพรส จำกัด,
- <http://www.rabbitsemiconductor.com>
- <http://www.embeddedj.co.th>