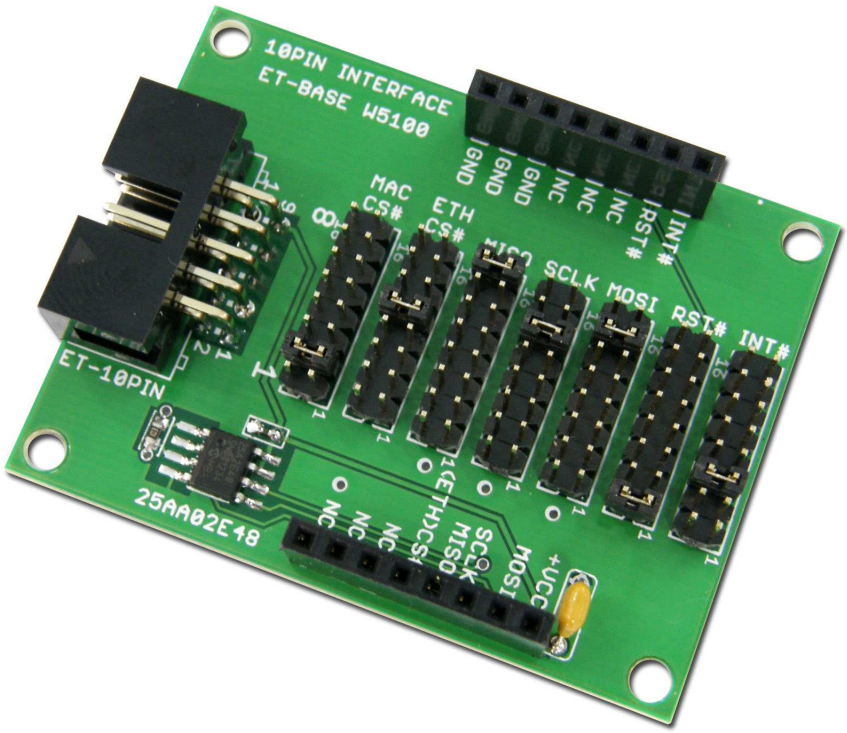
ET-BASE W5100 10PIN INTERFACE

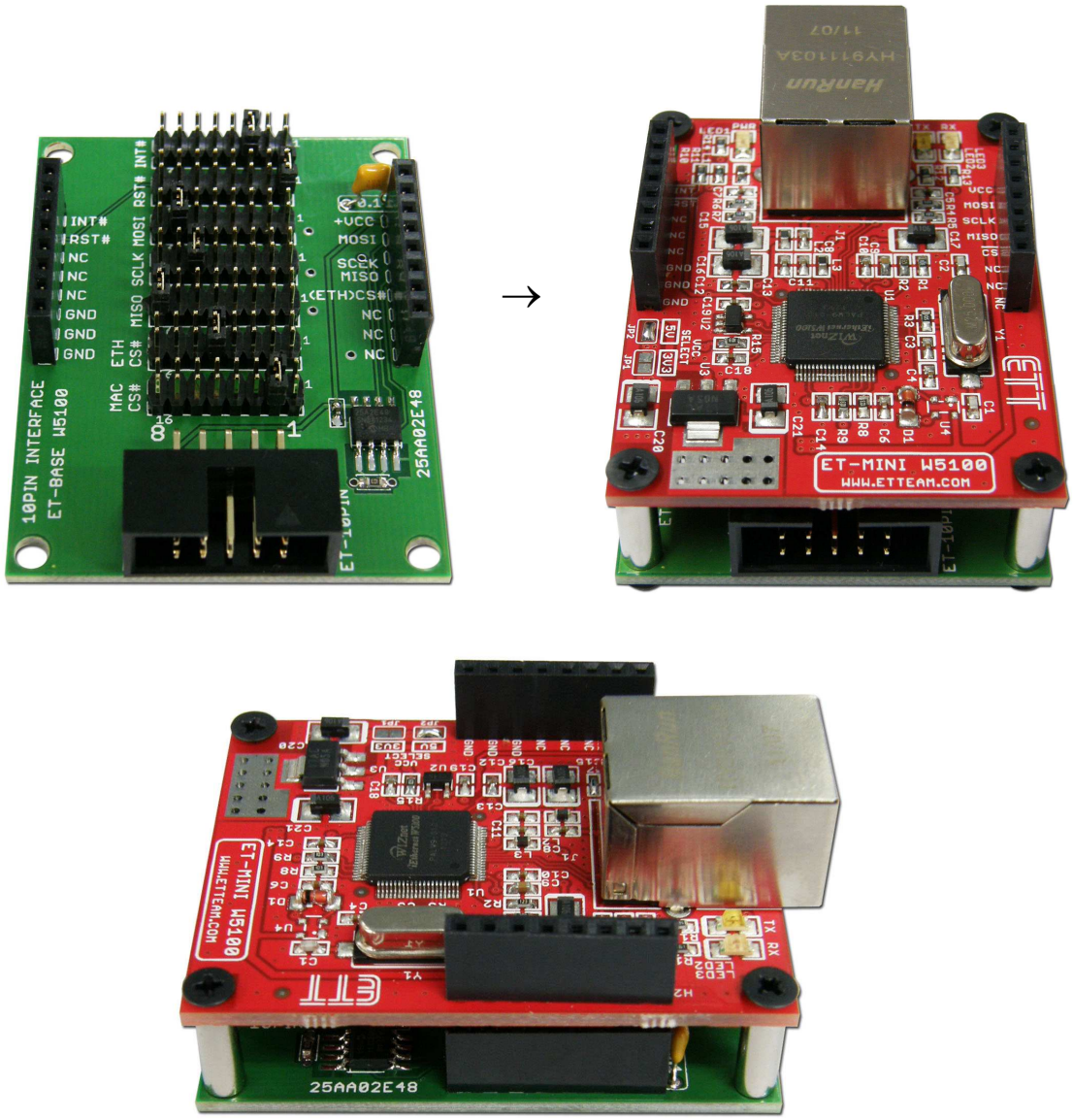


บอร์ด ET-BASE W5100 10 PIN INTERFACE เป็นชุดตัวกลางเชื่อมต่อระหว่างบอร์ด ET-MINI W5100 กับบอร์ดไมโครคอนโทรลเลอร์ของ อีทีที ที่ใช้ขั้วต่อสัญญาณแบบ IDE 10PIN โดยบนบอร์ด ET-BASE W5100 จะมีชุด Jumper สำหรับให้ผู้ใช้เลือกกำหนดสัญญาณแต่ละเส้นของ MCU ที่เชื่อมต่อผ่านมาจากขั้ว 10PIN IDE เพื่อส่งต่อไปยังบอร์ด ET-MINI W5100 ได้โดยอิสระ โดยสัญญาณแต่ละเส้นที่เชื่อมต่อไปยังชิพ W5100 ของบอร์ด ET-BASE W5100 จะมี Jumper ให้เลือกสัญญาณละ 8 ชุด เพื่อให้ผู้ใช้เลือกได้ว่าต้องการสัญญาณเส้นใดจาก 10PIN IDE เป็นสัญญาณในการใช้เชื่อมต่อสั่งงาน

นอกจากนี้แล้วบอร์ด ET-BASE W5100 ยังได้ติดตั้งชิพ 25AA02E48 ของ Microchips รวมไว้ในบอร์ดด้วย โดย 25AA02E48 เป็นชิพ SPI Serial EEPROM ขนาดความจุ 256 Byte พร้อมค่ารหัสตัวเลขแบบ Unique ซึ่งสามารถนำไปใช้อ้างอิงเป็นค่ารหัส MAC Address สำหรับใช้ในระบบ TCP/IP ได้ทั้งแบบ EUI-48(6 Byte MAC Address สำหรับระบบ มาตรฐาน IPV4) หรือ EUI-64(8 Byte MAC Address สำหรับมาตรฐาน IPV6) โดยชิพแต่ละตัวจะมี ค่ารหัส MAC Address ที่ไม่ซ้ำกัน และมีความถูกต้องเป็นมาตรฐาน สามารถใช้อ้างอิงเป็นรหัส MAC Address เพื่อใช้เป็นรหัสอ้างอิงตัวตนของอุปกรณ์ในการติดต่อสื่อสารกับระบบ TCP/IP ได้โดยไม่เกิดปัญหา แต่สำหรับ W5100 จะเป็น IPV4 จะใช้รหัสแบบ EUI-48 เป็นรหัส MAC Address

ตัวอย่างการติดตั้งใช้งาน

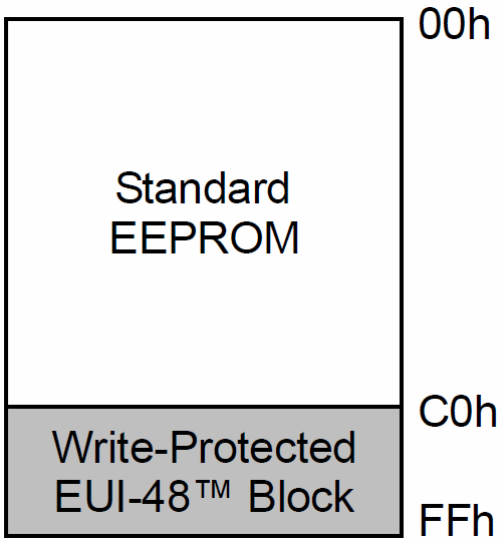
ในการติดตั้งใช้งาน ก่อนอื่นต้องตรวจสอบและเลือกกำหนด Jumper ของสัญญาณทั้ง 7 ชุด ว่าจะให้สัญญาณเส้นใดของ ET-MINI W5100 เชื่อมต่อกับสัญญาณเส้นใดจาก 10PIN IDE ของบอร์ด ET-BASE W5100 โดย Jumper ทั้ง 7 ชุด จะสามารถเลือกกำหนดสัญญาณได้อิสระจาก 1-8 ไม่ซ้ำกันตามความต้องการ เมื่อเลือกกำหนด Jumper เสร็จเรียบร้อยแล้ว ให้ทำการเสียบบอร์ด ET-MINI W5100 ซ้อนทับไปบนบอร์ด ET-BASE W5100 ตามทิศทางและตำแหน่งที่กำหนดไว้แล้วทำการยึดน๊อตให้เรียบร้อยดังตัวอย่าง



รูปแสดงตัวอย่างการติดตั้งใช้งานบอร์ด ET-BASE W5100 กับ ET-MINI W5100

การอ่านค่ารหัส MAC Address ของ 25AA02E48

25AA02E48 จะมีพื้นที่ของหน่วยความจำ ขนาด 256Byte โดยแบ่งเป็นพื้นที่ของหน่วยความจำ EEPROM ขนาด 8บิต สำหรับให้ใช้งานในการอ่านเขียน และเก็บข้อมูลต่างๆได้โดยอิสระจำนวน 192 Byte และจะสงวนพื้นที่ไว้ 64Byte สำหรับเก็บค่ารหัส Unique ซึ่งสามารถใช้แบบ EUI-48 หรือ EUI-64 ก็ได้ โดยรหัส Unique จะเป็นรหัสตัวเลขที่มีค่าไม่ซ้ำกัน ซึ่งในระบบการสื่อสาร TCP/IP จะใช้รหัส Unique เป็นค่ารหัส MAC Address สำหรับระบุตัวตนของอุปกรณ์ในเครือข่ายการสื่อสารแบบ TCP/IP ซึ่งอุปกรณ์ทุกๆตัวจะต้องมี ค่าตัวเลขรหัสประจำตัวที่ไม่ซ้ำกัน โดยในปัจจุบันการสื่อสาร TCP/IP จะมีการใช้รหัส MAC Address อยู่ 2 แบบ คือ EUI-48 ซึ่งมีค่ารหัสขนาด 6ไบท์ ใช้ใน IPV4 และรหัส EUI-64 ซึ่งเป็นรหัสตัวเลขขนาด 8ไบท์ ใช้ใน IPV6 โดยตำแหน่งหน่วยความจำที่ใช้อ่านผลลัพธ์ของค่ารหัส Unique จะอยู่ที่ตำแหน่งแอดเดรส 0xFA ถึง 0xFF ตามลำดับดังรูป



โดยในการอ่านค่ารหัส Unique จาก 25AA02E48 นั้นจะเริ่มอ่านจากตำแหน่งแอดเดรส 0xFA เป็นตำแหน่งแรก และจะอ่านต่อเนื่องไปจนถึงตำแหน่งแอดเดรส 0xFF โดยค่าของรหัสใน 3 ตำแหน่งแรกจะมีค่าคงที่ คือ 0x00, 0x04 และ 0xA3 ตามลำดับ ส่วนอีก 3ไบท์ถัดไปจะมีค่ารหัสที่ไม่ซ้ำกันในชิพแต่ละตัว โดยถ้าเราต้องการนำรหัส Unique นี้มาใช้แบบ EUI-48 ก็จะใช้รหัสที่อ่านได้ทั้ง 6 Byte เป็นค่ารหัส MAC Address ใช้งานได้เลย แต่ถ้าต้องการใช้รหัส Unique ให้เป็นแบบ EUI-64 จะต้องทำการแทรกค่า 0xFF และ 0xFE คั่นกลางระหว่างไบท์ที่ 3 และ ไบท์ที่4 ตัวอย่างเช่น ถ้าชิพมีรหัสประจำตัว 0x123456 ค่าที่อ่านได้จะเป็นดังตัวอย่าง

Description	24-bit Organizationally Unique Identifier			24-bit Extension Identifier		
Data	00h	04h	A3h	12h	34h	56h
Array Address	FAh					FFh

รูปแสดง โครงสร้างการจัดเก็บข้อมูลของ 25AA02E48

จากตัวอย่างข้างต้นจะแสดงให้เห็นถึงรูปแบบการจัดเก็บข้อมูลที่เป็นค่ารหัส Unique ของชิพหน่วยความจำเบอร์ 25AA02E48 โดยในตัวอย่างที่แสดงเป็นการสมมุตค่าของชิพที่บรรจุรหัสประจำตัวเป็น 12-34-56 ไว้ โดยชิพแต่ละตัวที่ผลิตขึ้นมาจำหน่ายจะมีรหัสส่วนนี้ไม่ซ้ำกัน

- ถ้าใช้รหัส Unique แบบ EUI-48 จะได้รหัส เป็น 00-04-A3-12-34-56
- ถ้าใช้รหัส Unique แบบ EUI-64 จะได้รหัส เป็น 00-04-A4-FF-FE-12-34-56

โดยในการอ่านค่าจาก 25AA02E48 นั้นจะใช้การสื่อสารแบบ SPI ซึ่งรูปแบบการสื่อสารนั้นจะเริ่มต้นนับจากจุดที่สัญญาณ Chips Select (CS) ของชิพ เปลี่ยนเป็นโลจิก “0” โดยข้อมูลไบท์แรกที่จะเขียนไปยัง 25AA02E48 จะต้องเป็น OP Code ของรหัสคำสั่งอ่าน ซึ่งมีค่าคงที่เป็น 0x03 ส่วนข้อมูลไบท์ที่2 จะเป็นตำแหน่งแอดเดรสเริ่มต้นที่ต้องการอ่าน ซึ่งในกรณีของรหัส Unique จะเริ่มต้นที่ตำแหน่งแอดเดรส 0xFA ส่วนไบท์ที่3 จะเป็นค่า Dummy สำหรับอ่านค่ารหัส Unique กลับออกมาจากชิพ ตามปรกติใช้ 0xFF โดยจะอ่านค่าต่อเนื่องกันไปจนครบ 6 ไบท์แล้วจึงทำการเปลี่ยนโลจิกของสัญญาณ CS กลับเป็น “1” เพื่อสิ้นสุดการสื่อสารกับชิพ ดังตัวอย่าง

```
/*
 *@brief : Get_MAC_Address()
 *       : Read MAC Address From 25AA02E48 to MAC Buffer
 */
void Get_MAC_Address(unsigned char *buf)  // Pointer Buffer
{
  unsigned char idx = 0;
  unsigned char dummy_data;

  MAC_CS_LOW();                           // CS = 0(SPI Start)
  dummy_data = SPI_SendByte(0x03);        // 1 = Opcode MAC Read
  dummy_data = SPI_Sendbyte(0xFA);        // 2 = MAC Address Read
  for (idx = 0; idx < 6; idx++)           // 6 Byte MAC Address Read
  {
    dummy_data = SPI_SendByte(0xFF);      // 3...8(Echo = MAC Data)
    buf[idx] = dummy_data;
  }
  MAC_CS_HIGH();                          // CS = 1(SPI Stop)
}
```

ตัวอย่างการใช้งานกับ **Arduino**

```
// ET-BASE AVR EASY(328) <--> ET-BASE W5100 10PIN Interface
// Digital-8(PB0)  --> MAC_CS#(25AA02E48)      -> Jumper ON PIN1
// Digital-9(PB1)  --> ETH_RES#(W5100)         -> Jumper ON PIN2
// Digital-10(PB2) --> ETH_CS#(W5100)          -> Jumper ON PIN3
// Digital-11(PB3) --> MOSI(W5100,25AA02E48)   -> Jumper ON PIN4
// Digital-12(PB4) --> MISO(W5100,25AA02E48)   -> Jumper ON PIN5
// Digital-13(PB5) --> SCLK(W5100,25AA02E48)   -> Jumper ON PIN6
// NC              <-- INT#                    -> Jumper ON PIN8(NC)

#include <SPI.h>
#include <Ethernet.h>

byte mac[6] = {0xDE,0xAD,0xBE,0xEF,0xFE,0xED};
IPAddress ip(192,168,1,234);          // IP address
#define CS_MAC    8                   // CS#  -> 25AA02E48
#define RES_ETH   9                   // RES# -> W5100
#define CS_ETH    10                  // CS#  -> W5100
#define MOSI_SPI  11                  // MOSI -> W5100,25AA02E48
#define MISO_SPI  12                  // MISO <- W5100,25AA02E48
#define SCLK_SPI  13                  // SCLK -> W5100,25AA02E48

void setup()
{
  pinMode(CS_MAC, OUTPUT);            // CS_MAC#
  pinMode(RES_ETH, OUTPUT);           // RES_ETH#
  pinMode(CS_ETH, OUTPUT);            // CS_ETH#
  pinMode(MOSI_SPI, OUTPUT);          // MOSI
  pinMode(MISO_SPI, INPUT);           // MISO
  pinMode(SCLK_SPI, OUTPUT);          // SCLK
  digitalWrite(CS_MAC,HIGH);          // Default : CS_MAC#
  digitalWrite(CS_ETH,HIGH);          // Default : CS_ETH#
  digitalWrite(RES_ETH,HIGH);         // Default : RES_ETH#

  digitalWrite(RES_ETH,LOW);         // Active Reset W5100
  delay(10);
  digitalWrite(RES_ETH,HIGH);        // Release Reset W5100

  //Start of Read MAC Address From 25AA02A48
  SPI.begin();                        // Initial SPI Read MAC Address
  digitalWrite(CS_MAC,LOW);           // Start CS_MAC#
  SPI.transfer(0x03);                 // 25AA02E48 Read OP-Code
  SPI.transfer(0xFA);                 // Address Start Read MAC Address
  mac[0]=SPI.transfer(0xFF);          // 1st Byte MAC Address(0x00)
  mac[1]=SPI.transfer(0xFF);          // 2nd Byte MAC Address(0x04)
  mac[2]=SPI.transfer(0xFF);          // 3rd Byte MAC Address(0xA3)
  mac[3]=SPI.transfer(0xFF);          // 4th Byte MAC Address(??)
  mac[4]=SPI.transfer(0xFF);          // 5th Byte MAC Address(??)
  mac[5]=SPI.transfer(0xFF);          // 6th Byte MAC Address(??)
  digitalWrite(CS_MAC,HIGH);          // Stop CS_MAC#
  //End of Read MAC Address From 25AA02E48

  Ethernet.begin(mac, ip);            // Iinitialize W5100
  .
  .
}
```