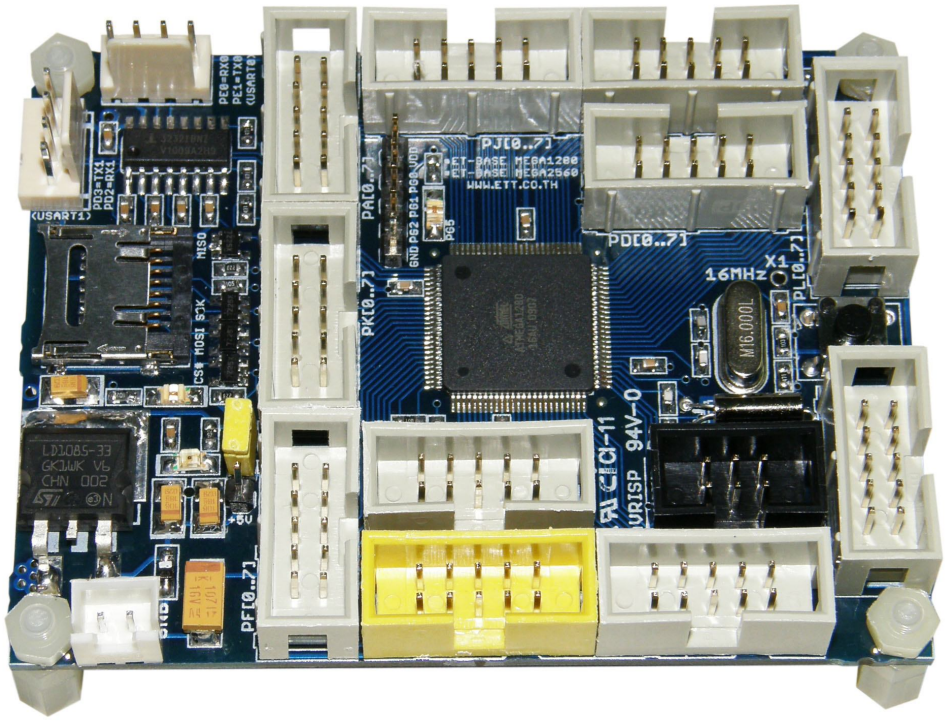
ET-BASE MEGA1280/2560



ไมโครคอนโทรลเลอร์ตระกูล AVR ของ ATMEL เป็นไมโครคอนโทรลเลอร์อีกตระกูลหนึ่งซึ่งได้รับความนิยมอย่างแพร่หลายจากผู้ใช้งานทั่วไป ซึ่งทาง ATMEL เองก็ได้มีการปรับปรุง พัฒนาขีดความสามารถของ MCU เพื่อตอบสนองความต้องการใช้งานในลักษณะต่างๆ มีการผลิตชิพ MCU ออกมาจำหน่ายเป็นจำนวนมากมายหลายเบอร์ เพื่อให้ผู้ใช้งานสามารถเลือก MCU ไปประยุกต์ใช้งานให้เหมาะสมกับงานได้ง่ายและสะดวกมากยิ่งขึ้น

ATMEGA ก็เป็น MCU ตระกูล AVR ที่มีความโดดเด่นและมีขีดความสามารถสูงในระดับต้นๆของ MCU 8 บิต ซึ่งใน ATMEGA1280/2560 เองเป็น MCU ตระกูล AVR MEGA ที่มีระบบ Peripheral I/O ต่างๆรวบรวมไว้ภายใน MCU มากมายหลากหลาย สามารถโปรแกรมโหมดการทำงานของ Peripheral I/O ใน ลักษณะต่างๆได้หลากหลาย ทำให้ง่ายและสะดวกในการนำไปดัดแปลงใช้กับงานแบบต่างๆได้โดยง่าย ซึ่งการพัฒนาโปรแกรมของบอร์ดก็สามารถเลือกใช้รูปแบบในการพัฒนาโปรแกรมด้วยโปรแกรมภาษาต่างๆที่สนับสนุนการใช้งานกับ AVR MEGA ได้ทั่วไป ตามความเหมาะสม

โดยโครงสร้างของบอร์ดได้ออกแบบให้มีความสะดวกในการพัฒนาโปรแกรม และ ประยุกต์ใช้งานได้โดยสะดวก โดยมีพอร์ตสื่อสาร RS232 และ Micro-SD Card เป็นอุปกรณ์พื้นฐานภายในบอร์ด ส่วน GPIO ต่างๆจะออกแบบเป็น IDE Connector ไว้ให้เพื่อให้เกิดความสะดวกในการเชื่อมต่อออกไปใช้งาน โดยได้เพิ่มช่องทางในการพัฒนาโปรแกรมได้ทั้งการโปรแกรมผ่าน Bootloader หรือ ISP Programmer หรือ JTAG Interface สำหรับโปรแกรมและ Debug การทำงานได้อีกด้วย

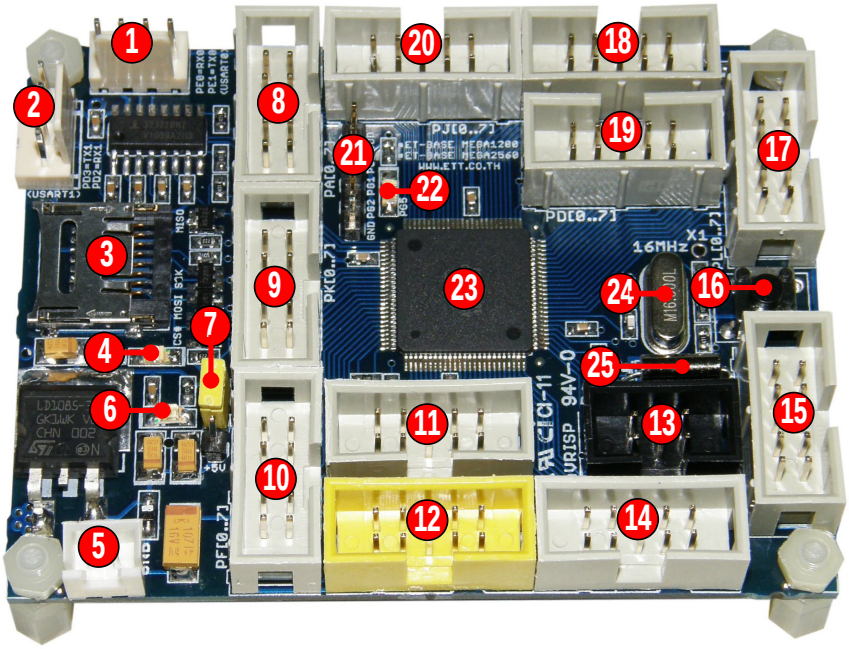
คุณสมบัติของ MCU ATMEGA1280/2560

- 128KB(ATMEGA1280,256K(ATMEGA2560) Flash / 8KB SRAM / 4KB EEPROM
- ทำงานที่แรงดัน 2.7V-5.5V (Run 16MHz ที่ 4.5V-5.5V, Run 8MHz ที่ 2.7-5.5V)
- มีวงจร Internal RC Clock 8MHz ภายใน
- มี 12 ช่อง 16Bit PWM
- มี 4 ช่อง USART
- มี 16 ช่อง 10Bit ADC(15KSPS)
- มี 86 Bit GPIO
- มี 2 ช่อง 8Bit Timer/Counter
- มี 4 ช่อง 16Bit Timer/Counter
- มี Real Time Counter
- มี 4 ช่อง 8Bit PWM
- มี 1 ช่อง I2C
- มี 1 ช่อง SPI
- มีระบบ JTAG(IEEE 1149.1 Compliant) สำหรับ Program และ Debug
- มี ISP (In System Programming) สำหรับ Program

คุณสมบัติของบอร์ด ET-BASE MEGA1280/2560

- ใช้ MCU ตระกูล MEGA AVR เบอร์ ATMEGA1280/2560 ของ ATMEL
- มีหน่วยความจำ Flash 128KB(ATMEGA1280,256K(ATMEGA2560), 8KB Boot loader, Static RAM 8KB และ EEPROM 4KB
- ใช้ Crystal 16.00 MHz
- มีวงจร Real Time Counter พร้อม XTAL ค่า 32.768KHz
- รองรับการโปรแกรมแบบ In-System Programming แบบ ISP
- มีวงจรเชื่อมต่อกับ AVR-JTAG ขนาด 10 Pin เพื่อทำการ Debug แบบ Real Time ได้
- Power Supply ใช้แรงดันไฟฟ้า +5VDC พร้อมวงจร Regulate +3V3/3A ภายในบอร์ด พร้อม Jumper เลือกระบบแหล่งจ่ายให้เป็น 3.3V หรือ 5V ได้ตามต้องการ
- มีวงจรเชื่อมต่อการ์ดหน่วยความจำแบบ SD Card(Micro SD) เชื่อมต่อแบบ SPI จำนวน 1 ช่อง
- มีวงจรสื่อสาร RS232 โดยใช้ขั้วต่อแบบ 4-PIN มาตรฐาน ETT จำนวน 2 ช่อง
- มีวงจร LED แสดงสถานะเพื่อทดลอง Output(PG5) จำนวน 1 ชุด
- มี 83 Bit GPIO อิสระ สำหรับประยุกต์ใช้งานต่างๆ เช่น A/D, I2C, SPI, USART และ Input / Output แบบต่างๆ โดยมีการจัดสรรใช้งานภายในบอร์ดไว้แล้ว จำนวน 8 เส้นสัญญาณ คือ PB[0..3] สำหรับ micro SD Card, PD[2..3] และ PE[0..1] สำหรับ RS232 แต่สัญญาณทั้ง 8 เส้น ดังกล่าวยังมีการเชื่อมต่อสัญญาณออกมาไว้ที่ขั้วต่อ 10PIN IDE ของพอร์ตด้วย โดยใช้ขั้วต่อสัญญาณแบบ 10PIN IDE จำนวน 10 ชุด มีดังนี้
  - Header 10Pin IDE (PA[0..7]) สำหรับ GPIO
  - Header 10Pin IDE (PB[0..7]) สำหรับ GPIOและ PB[0..3] ถูกใช้สำหรับ SPI ในการเชื่อมต่อกับ micro SD Card
  - Header 10Pin IDE (PC[0..7]) สำหรับ GPIO
  - Header 10Pin IDE (PD[0..7]) สำหรับ GPIO(PD[2..3] ถูกใช้สำหรับ USART0)
  - Header 10Pin IDE (PE[0..7]) สำหรับ GPIO(PE[0..1] ถูกใช้สำหรับ USART1)
  - Header 10Pin IDE (PF[0..7]) สำหรับ GPIO(PF[4..7])ถูกใช้สำหรับ JTAG)
  - Header 10Pin IDE (PH[0..7]) สำหรับ GPIO
  - Header 10Pin IDE (PJ[0..7]) สำหรับ GPIO
  - Header 10Pin IDE (PK[0..7]) สำหรับ GPIO
  - Header 10Pin IDE (PL[0..7]) สำหรับ GPIO
  - Header 1x5Pin (PG[0..2]) สำหรับ GPIO

โครงสร้างบอร์ด ET-BASE MEGA1280/2560

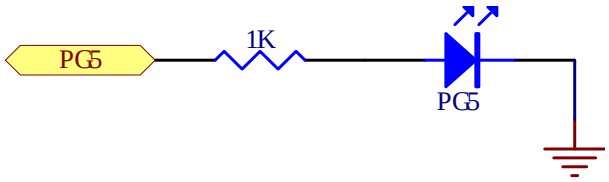


รูปแสดง โครงสร้างของบอร์ด ET-BASE MEGA1280/2560

- **หมายเลข 1,2 คือ** ขั้วต่อ USART0(RS232) และ USART1(RS232) สำหรับใช้งาน
- **หมายเลข 3,4 คือ** ช่องเสียบการ์ดหน่วยความจำ SD Card แบบ Micro-SD และ LED สถานะ
- **หมายเลข 5,6 คือ** ขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ดใช้ได้กับไฟ +5VDC และ LED Power
- **หมายเลข 7 คือ** Jumper สำหรับเลือกขนาดแรงดันทำงานของ MCU ระหว่าง 3.3V หรือ 5V
- **หมายเลข 8,9,10,11 คือ** ขั้วต่อ GPIO(PA[0..7]) ,(PK[0..7]),(PF[0..7]) และ GPIO(PE[0..7])
- **หมายเลข 12 คือ** ขั้วต่อ AVR-JTAG สำหรับ Debug แบบ Real Time
- **หมายเลข 13 คือ** ขั้วต่อ AVRISP สำหรับ Download โปรแกรม
- **หมายเลข 14,15 คือ** LED ขั้วต่อ GPIO(PH[0..7]) และ GPIO(PB[0..7])
- **หมายเลข 16 คือ** SW RESET
- **หมายเลข 17,18,19,20 คือ** ขั้วต่อ GPIO(PL[0..7]), (PC[0..7]),(PD[0..7]) และ GPIO(PJ[0..7])
- **หมายเลข 21,22 คือ** ขั้วต่อ GPIO(PG[0..2]) และ LED ใช้ทดสอบ Logic Output ของ PG5
- **หมายเลข 23 คือ** MCU เบอร์ ATMEGA1280 หรือ ATMEGA2560 (100Pin TQFP)
- **หมายเลข 24 คือ** Crystal ค่า 16.00 MHz สำหรับใช้เป็นฐานเวลาระบบให้ MCU
- **หมายเลข 25 คือ** Crystal ค่า 32.768KHz สำหรับฐานเวลาให้ RTC ภายในตัว MCU

การใช้งานวงจรขับ LED แสดงผล

LED แสดงผลของบอร์ด จะต่อวงจรแบบขับกระแส (Source Current) ทำงานด้วยโลจิก “1” และหยุดทำงานด้วยโลจิก “0” โดยควบคุมการทำงานจากขาสัญญาณ PG5 โดยวงจรในส่วนนี้จะใช้สำหรับทดสอบการทำงานของ Output จากขาสัญญาณ PG5



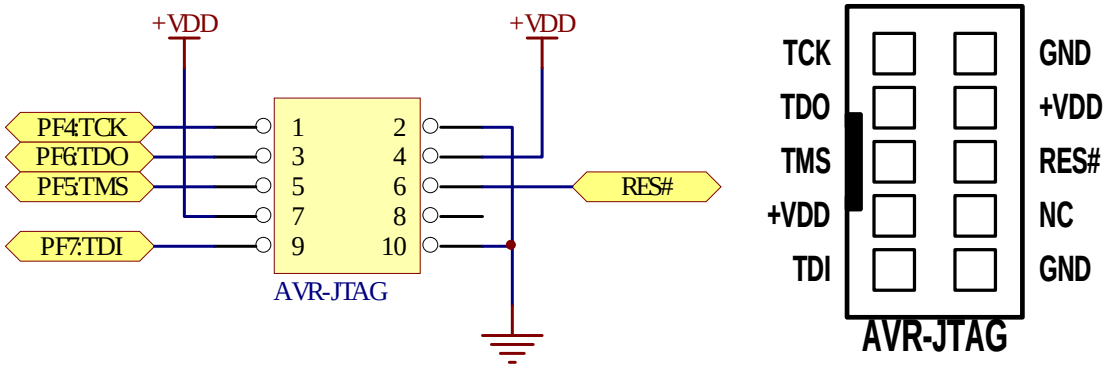
โดยเมื่อต้องการใช้งานผู้ใช้ต้องกำหนดให้PG[5] ทำหน้าที่เป็น GPIO Output Port เสียก่อนแล้วจึงควบคุม Logic ให้กับ PG[5] ตามต้องการ ดังตัวอย่าง

```
#define PORT_LED PORTG          // Port Drive LED = PG
#define DIR_LED  DDRG           // Port Direction
#define LED 5                   // Pin Drive LED = PG5
.
.
.
int main(void)
{
  DIR_LED |= (1<<LED);          // Pin Drive LED = Out
  .
  .
  .
  PORT_LED &= ~(1<<LED);        // Pin LED = 0(OFF LED)
  .
  .
  .
  PORT_LED |=  (1<<LED);        // Pin LED = 1(ON LED)
  .
  .
  .
  PORT_LED ^=  (1<<LED);        // Pin LED = Toggle
  .
  .
  .
}
```

ตัวอย่าง การกำหนดค่าการใช้งาน PG5 เป็น Output LED

การใช้งาน AVR-JTAG

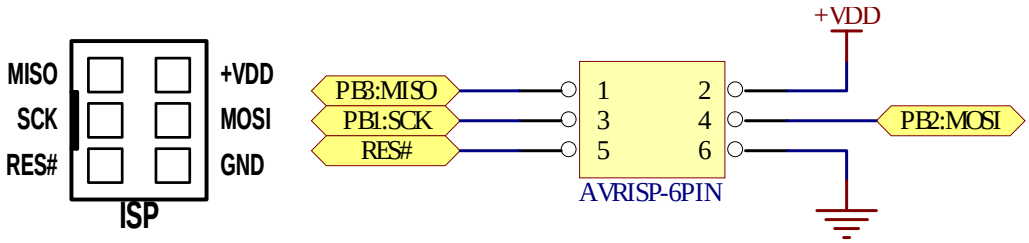
AVR-JTAG จะเป็น Connector แบบ IDE 10 Pin สำหรับ สำหรับเชื่อมต่อกับเครื่อง โปรแกรม และ Debug ภายนอกที่ใช้มาตรฐาน AVR-JTAG ซึ่งรองรับการใช้งานร่วมกับ MCU เบอร์ ATMEGA1280/2560 เช่น AVR DRAGON หรือ AVR JTAGICE mkII หรือ เทียบเท่า โดยมีการจัดวงจรและสัญญาณตามแบบ AVR-JTAG ตามมาตรฐานของ ATMEL ไว้ดังนี้

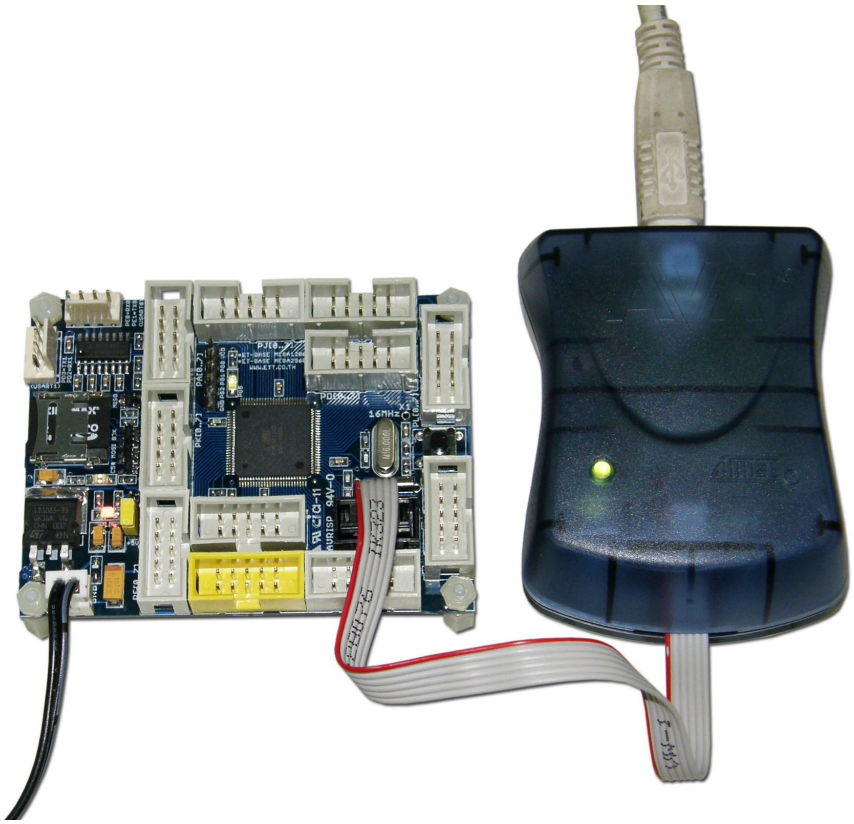


การใช้งาน ISP Program

ISP จะเป็น Connector แบบ IDE 6 Pin สำหรับ สำหรับเชื่อมต่อกับเครื่อง โปรแกรมภายนอกที่ใช้มาตรฐาน AVR ISP ซึ่งรองรับการใช้งานร่วมกับ MCU เบอร์ ATMEGA1280/2560 เช่น AVRISP mkII หรือเทียบเท่า เช่น ET-AVRISP mkII โดยมีการจัดวงจรและสัญญาณตามแบบ AVR ISP ตามมาตรฐานของ ATMEL ไว้ดังนี้

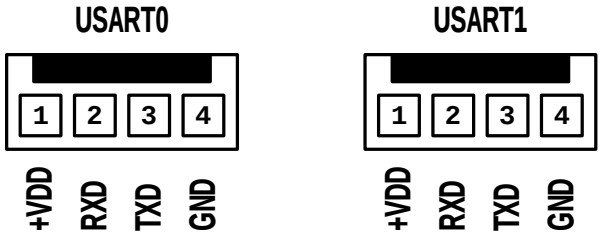
- PB1 = SCK
- PB2 = MOSI
- PB3 = MISO





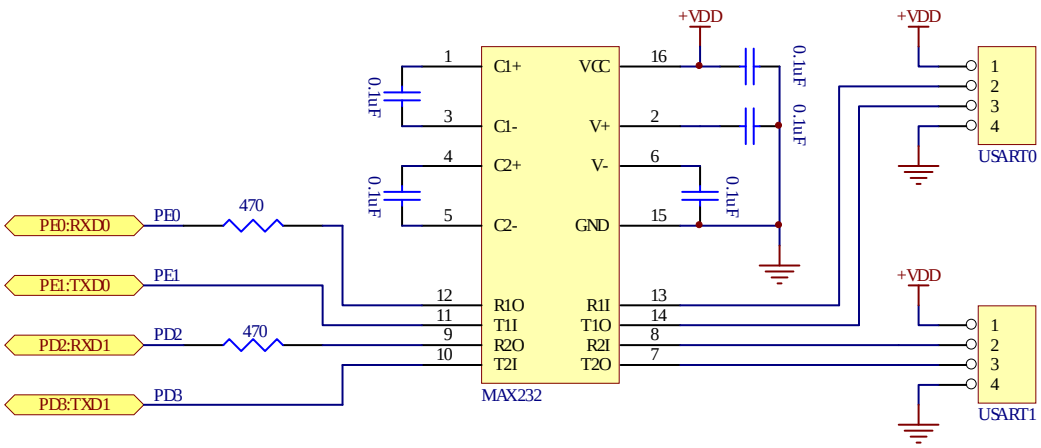
พอร์ต RS232

เป็นสัญญาณ RS232 ซึ่งผ่านวงจรแปลงระดับสัญญาณ MAX3232 เรียบร้อยแล้ว โดยมีจำนวน 2 ช่อง ด้วยกันคือ USART0 และ USART1 โดยทั้ง 2 ช่องสามารถใช้เชื่อมต่อกับสัญญาณ RS232 เพื่อรับส่งข้อมูลได้



- USART0 ใช้ขาสัญญาณจาก PE0(RXD) และ PE1(TXD)
- USART1 ใช้ขาสัญญาณจาก PD2(RXD) และ PD3(TXD)

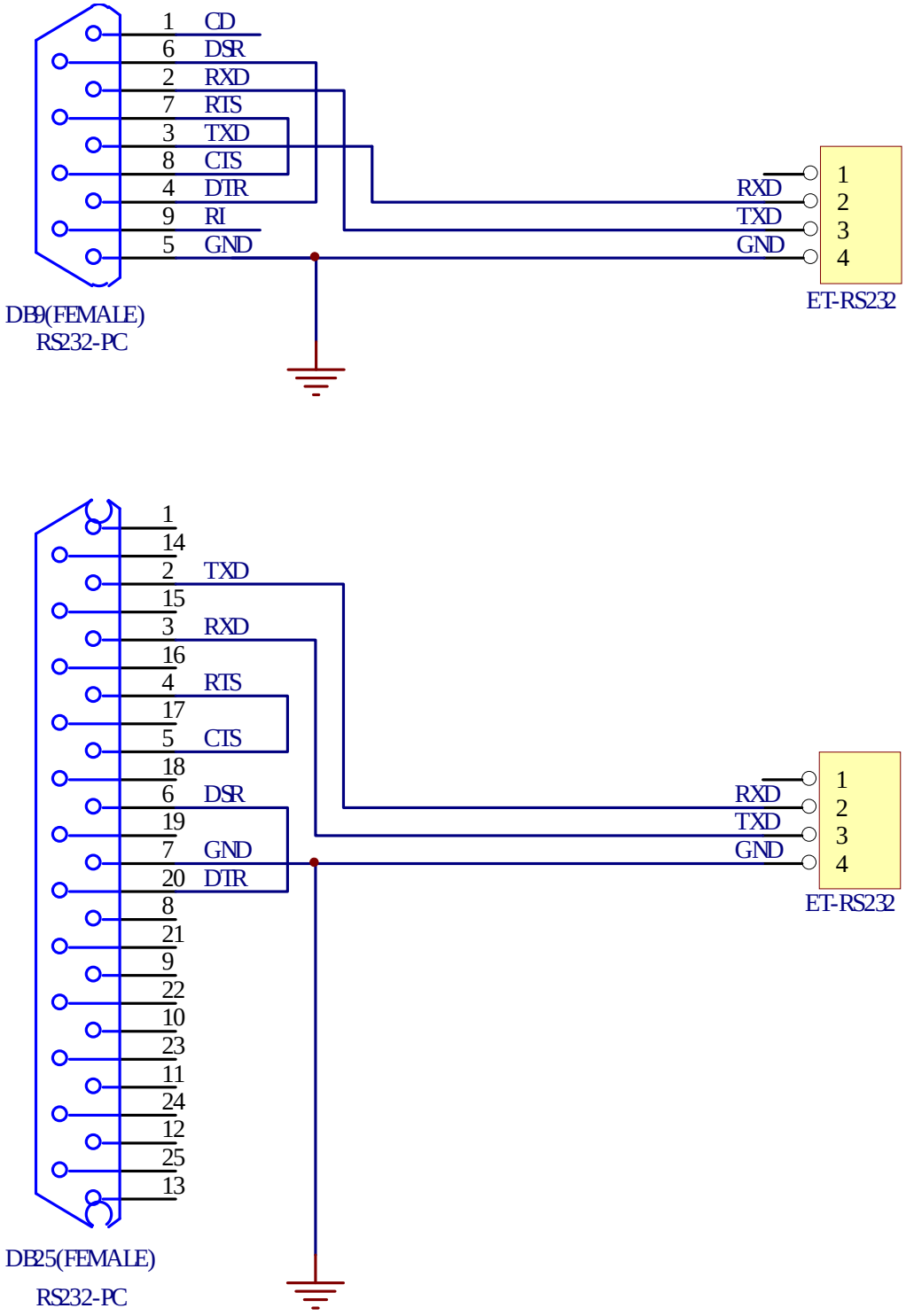
เนื่องจากระบบ Hardware USART ของ ATMEGA1280/2560 นั้นจะมี USART ไว้ใช้งานมากถึง 4 ชุด คือ PE0,PE1 และ PD2,PD3 และ PH0,PH1 และ PJ0,PJ1 โดยในกรณีของบอร์ด ET-BASE MEGA1280/2560 จะจัดวงจร USART ร่วมกับวงจร Line Driver ของ RS232 ไว้ให้จำนวน 2 ช่อง คือ PE0,PE1 และ PD2,PD3 ดังวงจร



```
#include <avr/io.h>
#include <stdio.h>
#define F_CPU 16000000UL                    //16.0 MHz
#define BAUD 9600                           //9600 BPS
#define MYUBRR F_CPU/16/BAUD-1

void init_serial(unsigned int ubrr);       //Initil UART
int   my_putchar(char c, FILE *stream);
int   my_getchar(FILE *stream);

/* Retarget STDIO(putchar,getchar of printf) to My Function */
FILE uart_str = FDEV_SETUP_STREAM(my_putchar, my_getchar, _FDEV_SETUP_RW);

int main(void)
{
  init_serial(MYUBRR);                      //Initilial UART0 = 9600,N,8,1
  stdout = stdin = &uart_str;

  printf("Hello World\n\r");
  while(1)
  {
    putchar(getchar());                     //Echo Receive Char
  }
}

/****************/
/* Initial UART */
/****************/
void init_serial(unsigned int ubrr)
{
  UBRR0H = (unsigned char)(ubrr>>8);       //Set baud rate
  UBRR0L = (unsigned char)ubrr;
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);          //Enable RX and TX
  UCSR0C = (1<<USBS0)|(3<<UCSZ00);         //8data, 2stop bit
}

/*****************************/
/* Write 1 Character To UART */
/*****************************/
int my_putchar(char c, FILE *stream)
{
  while (!(UCSR0A & (1<<UDRE0)));          // Wait TXD Buffer Empty
  UDR0 = c;
  return 0;
}

/*****************************/
/* Get 1 Character From UART */
/*****************************/
int my_getchar(FILE *stream)
{
  while(!(UCSR0A & (1<<RXC0)));            // Wait RXD Receive Data Ready
  return (UDR0);                           // Get Receice Data & Return
}
```

ตัวอย่างการใช้งาน USART0 รับส่งข้อมูล

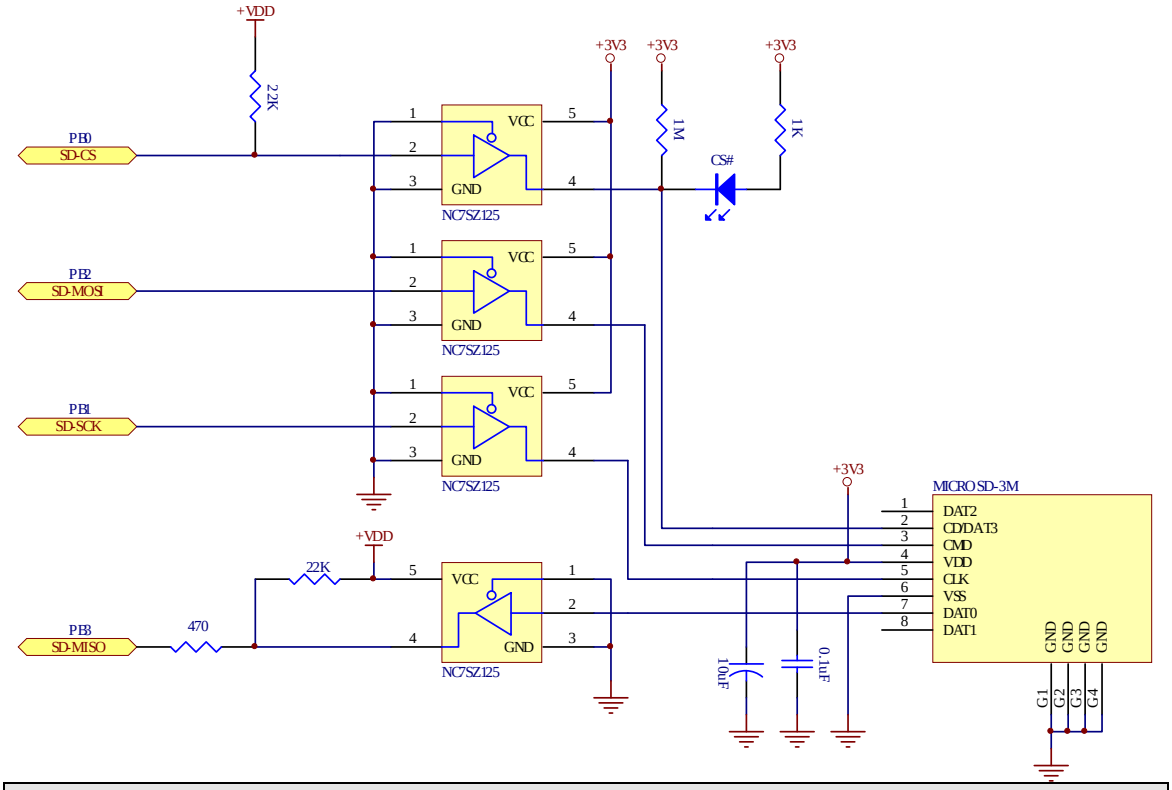
สำหรับ Cable ที่จะใช้ในการเชื่อมต่อ RS232 ระหว่าง Comport ของเครื่องคอมพิวเตอร์ PC เข้ากับขั้วต่อ USART0 และ USART1 ของบอร์ด ET-BASE MEGA1280/2560 นั้น เป็นดังนี้



รูป แสดงวงจรสาย Cable สำหรับ RS232

การ์ดหน่วยความจำ SD Card แบบ Micro-SD

บอร์ด ET-BASE MEGA1280/2560 รองรับการเชื่อมต่อกับการ์ดหน่วยความจำ SD Card แบบ Micro-SD โดยใช้การเชื่อมต่อแบบ SPI โดยใช้ขาสัญญาณ PB[0..3] ในการเชื่อมต่อกับการ์ด ซึ่งในการติดต่อสั่งงาน การ์ดนั้น สามารถโปรแกรม Pin I/O ของ PB[0..3] ให้ทำงานในโหมด SPI โดยต้องกำหนดหน้าที่ของขาสัญญาณ PB[0..3] ของ MCU เป็นดังนี้

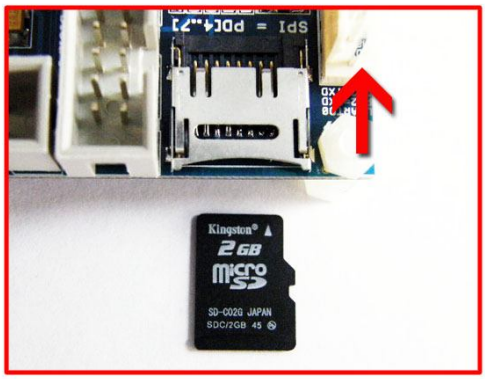



```
#define DDR_SPI DDRB
#define SD_CS   DDB0
#define SD_SCK  DDB1
#define SD_MOSI DDB2
.
.
// Set as master, clock and chip select output
DDR_SPI |= (1<<SD_MOSI)|(1<<SD_SCK)|(1<<SD_CS);  //MOSI,SCK,CS = Out
SPCR     = (1<<SPE)|(1<<MSTR);                   //SPI, Master

// SPI = Low Speed(Max. 400 kBit used in Card Initialization)
// SPI2X:SPR1:SPR0 = 0:1:1 (SCK = Fosc/128)
// SCK = Fosc  / 128
//     = 16MHz / 128
//     = 126KHz
SPSR &= ~(1 << SPI2X);                           // SPI2X = 0
SPCR |=  ((1 << SPR1) | (1 << SPR0));            // SPR1:SPR0 = 1:1
```

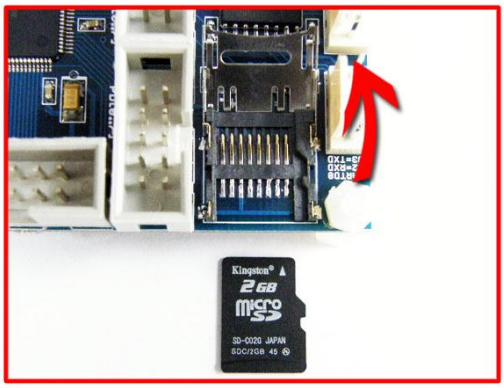
ตัวอย่าง การกำหนดค่า Pin สำหรับใช้งาน SD Card

การใส่การ์ดหน่วยความจำใน Socket

สำหรับ Socket สำหรับติดตั้งการ์ดหน่วยความจำของบอร์ด ET-BASE MEGA1280/2560 นั้น จะใช้ Socket หน่วยความจำแบบใส่การ์ดจากด้านบน แบบเดียวกันกับซิมการ์ดของโทรศัพท์มือถือ โดยเมื่อต้องการจะใส่หรือถอดการ์ดหน่วยความจำจะต้องทำการเปิดฝาครอบ Socket ออกเสียก่อน จากนั้นจึงจะสามารถใส่หรือถอดการ์ดหน่วยความจำได้ โดยการเปิด หรือ ปิด ฝาครอบ Socket จะใช้การกดเลื่อนฝาครอบเข้าหรือออก ซึ่งถ้าเลื่อนฝาครอบเข้าด้านในจะเป็นการเลื่อนเพื่อเปิดฝา แต่ถ้าเลื่อนออกด้านนอกจะเป็นการเลื่อนเพื่อล๊อคฝาครอบดังรูป

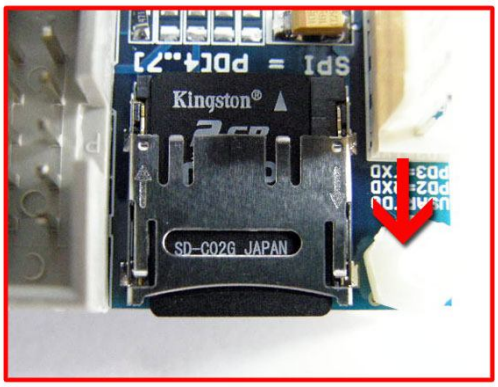


1. กดฝาครอบและเลื่อนเข้าเพื่อเปิดล๊อค



2. เปิดฝาครอบ Socket ออก

3. ใสการ์ดหน่วยความจำใน Socket

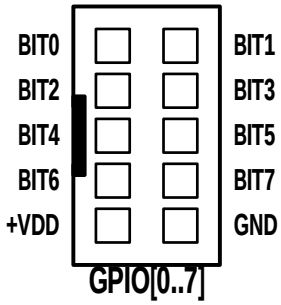


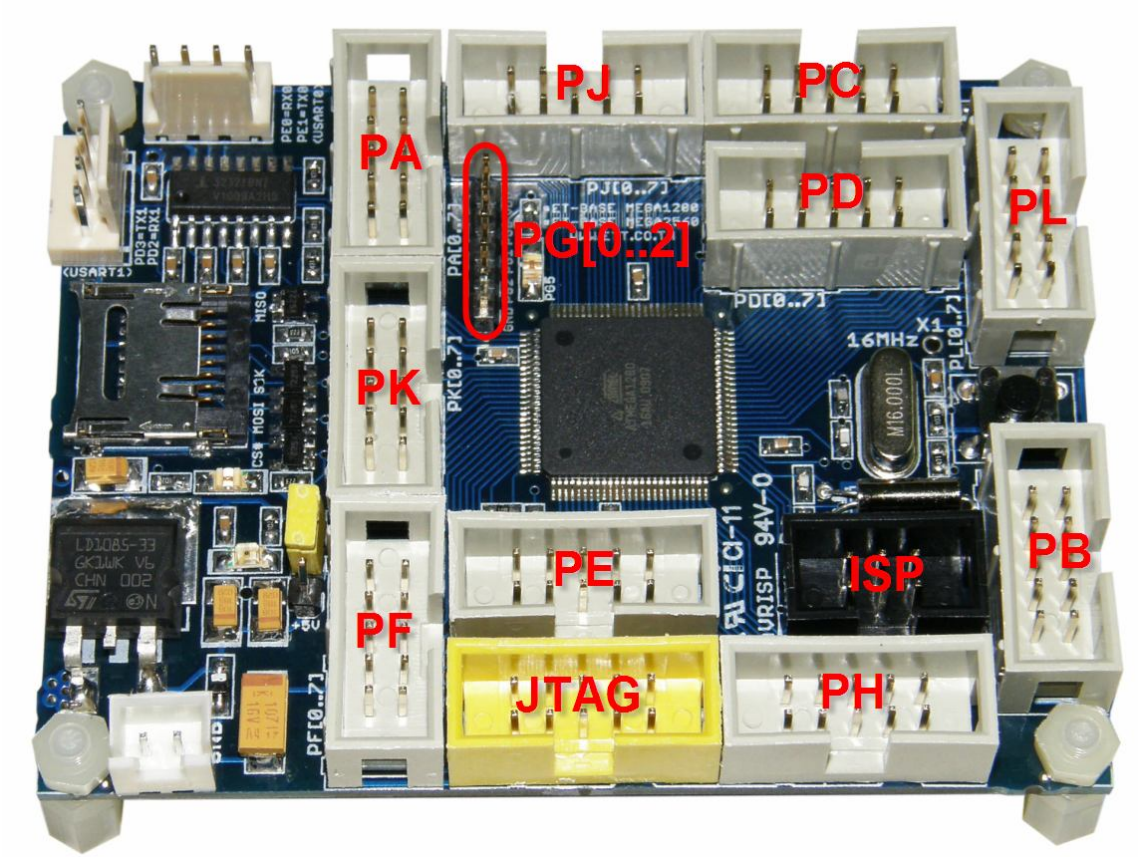
4. ปิดฝาครอบและกดเลื่อนออกเพื่อล๊อค

แสดงลำดับขั้นตอนการใส่การ์ดหน่วยความจำ

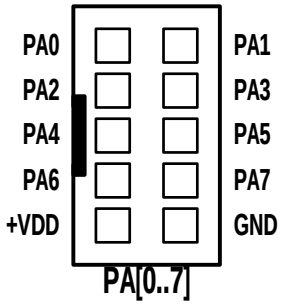
ขั้วต่อ Port I/O ต่าง ๆของบอร์ด

สำหรับขั้วต่อ Port I/O ของ MCU ของบอร์ด ET-BASE MEGA1280/2560 นั้น จะจัดเรียงออกมารอไว้ยังขั้วต่อแบบ IDE 10 Pin จำนวน 10 ชุดๆละ 8บิต สำหรับให้ผู้ใช้เลือกต่อออกไปใช้งานตามต้องการ โดยรูปแบบการจัดเรียงสัญญาณของแต่ละพอร์ต จะเรียงลำดับตำแหน่งบิต I/O เหมือนกัน โดยมีรายละเอียดของสัญญาณแต่ละชุดดังนี้

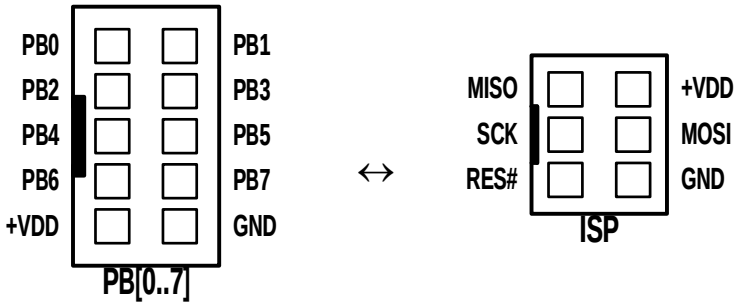




- ขั้วต่อ IDE 10 Pin ของ PA[0..7] โดย Port PA ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 เองนั้นสัญญาณทั้ง 8 เส้น จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้

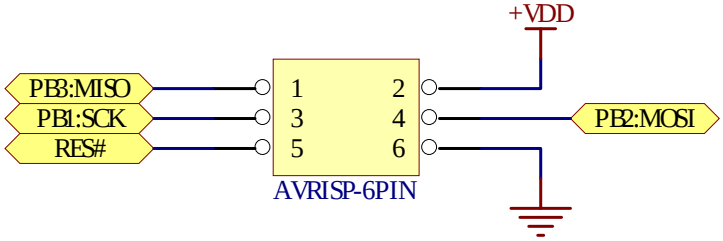


- ขั้วต่อ IDE 10 Pin ของ PB[0..7] โดย Port PB ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 นั้น จะมีการเชื่อมต่อ PB[1..3] ไปยังขั้วต่อ AVRISP และ PB[0..3] ก็จะถูกเชื่อมต่อเข้ากับวงจรของ Socket Micro SD-Card ด้วย ส่วน PB[4...7] จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ ซึ่งถ้ามีการต่อ PB[1..3] กับอุปกรณ์ภายนอกไว้ เมื่อต้องการโปรแกรมผ่าน AVRISP จะต้องปลดขาสัญญาณชุดนี้ให้เป็นอิสระจากอุปกรณ์ภายนอกด้วย ไม่เช่นนั้นอาจไม่สามารถโปรแกรมผ่าน AVRISP ได้ โดย PB[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

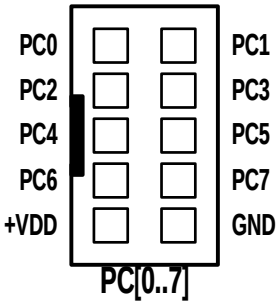


*****หมายเหตุ***** PB1...PB3 จะซ้อนทับกับ ISP Function ด้วย ซึ่งขาสัญญาณ PB1...PB3 ของบอร์ดจะถูกเชื่อมต่อไปยังขั้วต่อ AVRISP และ Micro SD-Card ด้วย โดยขาสัญญาณ PB1...PB3 เมื่อทำหน้าที่เป็น AVRISP จะมีหน้าที่ดังนี้

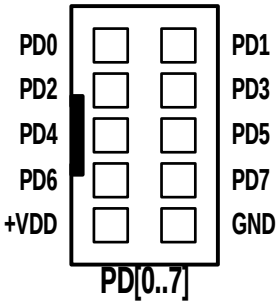
- PB1 = SCK
- PB2 = MOSI
- PB3 = MISO



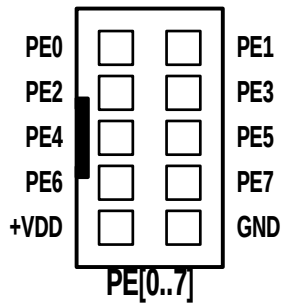
- ขั้วต่อ IDE 10 Pin ของ PC[0..7] โดย Port PC ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 เองนั้นสัญญาณทั้ง 8 เส้น จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้



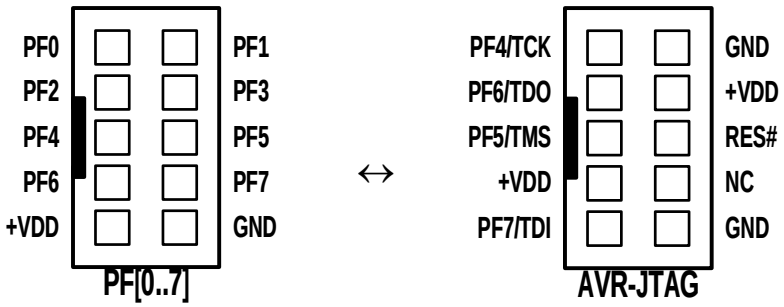
- ขั้วต่อ IDE 10 Pin ของ PD[0..7] โดย Port PD ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 นั้น จะมีการเชื่อมต่อ PD2...PD3 ไปยังวงจร USART1 ด้วย ส่วนสัญญาณ PD0...PD1 และ PD[4..7] จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้



- ขั้วต่อ IDE 10 Pin ของ PE[0..7] โดย Port PE ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 นั้น จะมีการเชื่อมต่อ PE0...PE1 ไปยังวงจร USART0ด้วย ส่วน PE[2...7] จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้

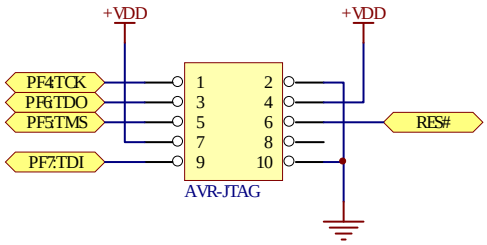


- ขั้วต่อ IDE 10 Pin ของ PF[0..7] โดย Port PF ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 นั้น จะมีการเชื่อมต่อ PF[4..7] ไปยังขั้วต่อสำหรับ AVR-JTAG ด้วย ส่วน PF[0...3] จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ ซึ่งถ้าต้องการใช้งาน PF[4..7] ต้องไปสั่ง Disable Fuse Bit ของ JTAGEN ออกเสียก่อนด้วยเครื่องโปรแกรมทาง ISP Port โดย PF[0..7] มีการจัดเรียงสัญญาณไว้ดังนี้

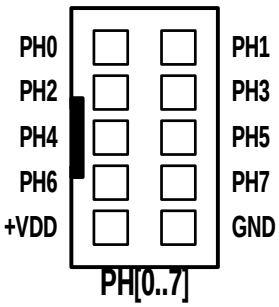


*****หมายเหตุ***** PF4...PF7 จะซ้อนทับกับ JTAG Function ด้วย ซึ่งขาสัญญาณ PF4...PF7 ของบอร์ดจะถูกเชื่อมต่อไปยังขั้วต่อ AVR-JTAG ด้วย ซึ่งถ้าผู้ใช้ได้เชื่อมต่อ AVR-JTAG ไว้ด้วย จะไม่สามารถใช้งานขาสัญญาณ PF4...PF7 ได้ โดยขาสัญญาณ PF4...PF7 เมื่อทำหน้าที่เป็น JTAG จะมีหน้าที่ดังนี้

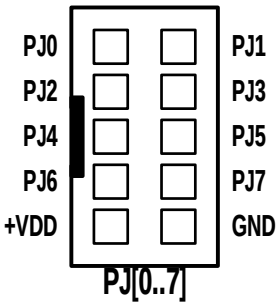
- PF4 = TCK JTAG
- PF5 = TMS JTAG
- PF6 = TDO JTAG
- PF7 = TDI JTAG



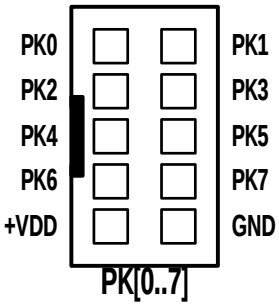
- ขั้วต่อ IDE 10 Pin ของ PH[0..7] โดย Port PH ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 เองนั้นสัญญาณทั้ง 8 เส้น จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้



- ขั้วต่อ IDE 10 Pin ของ PJ[0..7] โดย Port PJ ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 เองนั้นสัญญาณทั้ง 8 เส้น จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้



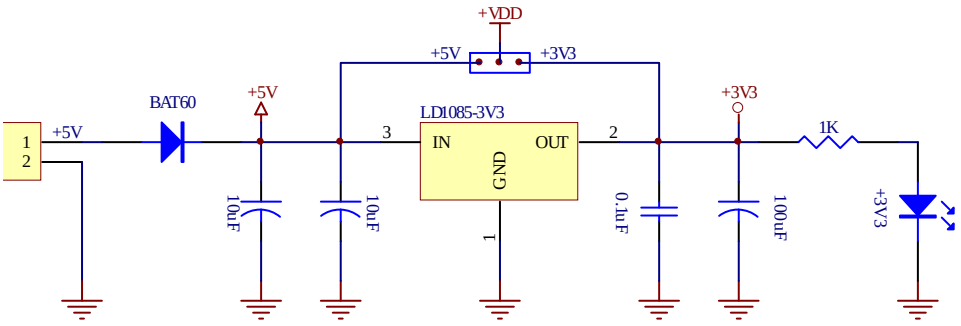
- ขั้วต่อ IDE 10 Pin ของ PK[0..7] โดย Port PK ของ ATMEGA1280/2560 จะมีทั้งหมดจำนวน 8 บิต ซึ่งในบอร์ด ET-BASE MEGA1280/2560 เองนั้นสัญญาณทั้ง 8 เส้น จะยังคงอิสระปล่อยว่างไว้ให้ผู้ใช้นำไปต่อประยุกต์ใช้งานได้เองตามความต้องการ โดยมีการจัดเรียงสัญญาณไว้ดังนี้

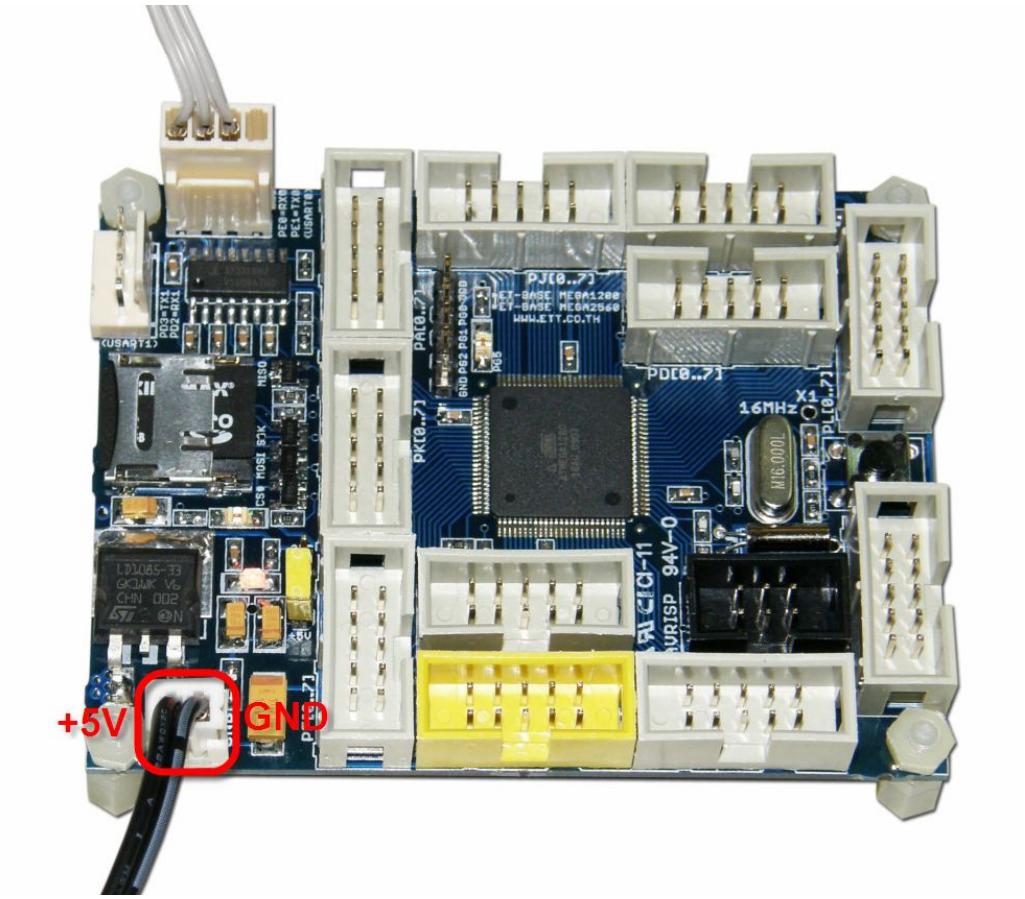


วงจรแหล่งจ่ายไฟ

วงจรแหล่งจ่ายไฟของบอร์ดใช้งานได้กับไฟ DC ขนาด +5V โดยใช้ขั้วต่อแบบ 2 Pin Block ป้องกันการเสียบสายกลับขั้ว พร้อมวงจร Regulate ขนาด +3V3/3A

โดยวงจรภาคแหล่งจ่ายไฟในส่วนที่เป็นวงจร Regulate ขนาด 3.3V นั้นจะจ่ายตรงไปให้กับ วงจร SD-Card ส่วนวงจรของ MPU และวงจร I/O ของบอร์ดนั้น จะสามารถเลือกขนาดแรงดันใช้งานได้จาก Jumper ว่าต้องการใช้งานเป็น 3V3 หรือ 5V ดังรูป

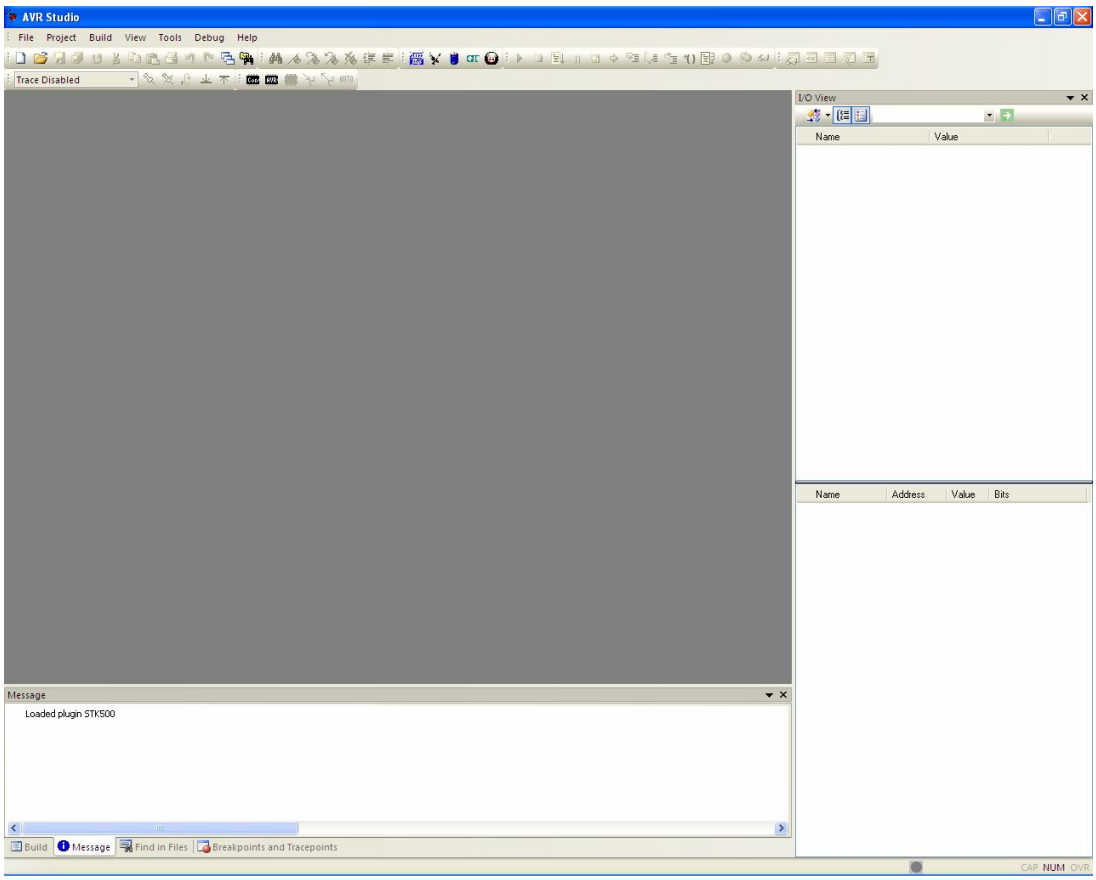




ตัวอย่างการพัฒนาโปรแกรมด้วย WinAVR ร่วมกับ AVR Studio4

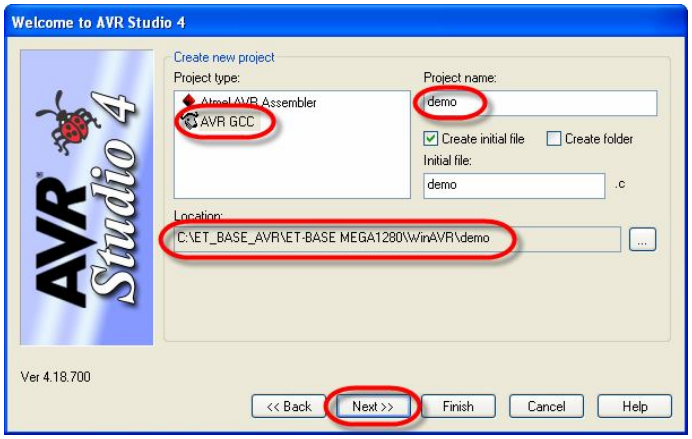
ในการพัฒนาโปรแกรมของ AVR เบอร์ ATMEGA1280/2560 นั้น ตามปรกติแล้วจะสามารถเลือกใช้ Compiler ต่างๆที่สนับสนุน MCU เบอร์นี้อยู่ได้ทั้งหมด แต่ในที่นี้จะขอแนะนำให้ใช้โปรแกรม AVR Studio4 ร่วมกับ WinAVR ซึ่งเป็นชุดโปรแกรมที่แจกจ่ายให้ใช้ได้ฟรีๆไม่เสียค่าใช้จ่าย มีการพัฒนาปรับปรุงความสามารถของโปรแกรมกันอย่างต่อเนื่องและมีผู้ใช้งานกันอย่างแพร่หลายทั่วโลก สามารถค้นหาตัวอย่างโปรแกรม และ Library ต่างๆที่ผู้ใช้ต่างๆจำนวนมากได้สร้างและเผยแพร่ไว้มาเป็นแนวทางในการศึกษาได้มากมายซึ่งปัจจุบัน (สิงหาคม 2553) ทาง ATMEL ได้ทำการปรับปรุงโปรแกรม AVR Studion4 เป็นรุ่น V4.18 แล้ว ส่วนโปรแกรม WinAVR สามารถ Download ได้จาก http://winavr.sourceforge.net/ จะปรับปรุงเป็น WINAVR-20100110 แล้ว ซึ่งผู้ใช้สามารถไป Download มาติดตั้งใช้งานได้ฟรี โดยในที่นี้จะขอแนะนำแนวทางการพัฒนาโปรแกรมแบบพอสังเขป เพื่อเป็นแนวทางสำหรับผู้เริ่มต้น ซึ่งรายละเอียดต่างๆสามารถศึกษาได้จากคู่มือของโปรแกรมได้ โดยแนวทางการพัฒนาโปรแกรมของ ATMEGA1280 และ ATMEGA2560 โดยใช้โปรแกรม AVR Studio4 ร่วมกับ WinAVR มีลำดับขั้นตอนดังนี้

1. สั่ง Run Program AVR Studio4 ซึ่งจะได้ผลดังรูป

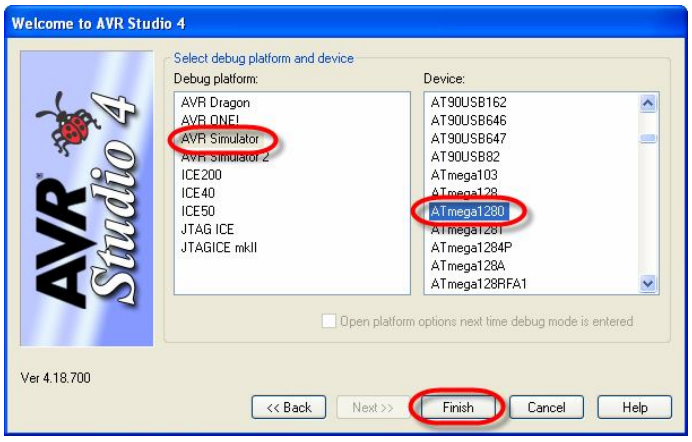


2. สั่งสร้าง project ใหม่ โดยเลือกที่ project → New project จากนั้นเลือกกำหนดตัวเลือกต่างๆ ให้กับโปรแกรมดังนี้

   - Project type เลือกกำหนดเป็น AVR GCC
   - Location สำหรับบันทึก project ให้ระบุตำแหน่ง Folder ที่ต้องการใช้บันทึกไฟล์ และ Codeต่างของ project
   - Project name ให้กำหนดชื่อ project ตามต้องการในตัวอย่างกำหนดเป็น “demo” และให้เลือก Create initial file ไว้ด้วย ซึ่งเมื่อเราทำการกำหนดชื่อ project name เสร็จแล้ว โปรแกรมจะสร้างไฟล์ ที่มีชื่อเดียวกันกับ project name ให้เองโดยอัตโนมัติ ซึ่งถ้าต้องการกำหนดชื่อไฟล์เป็นชื่ออื่น ก็ไม่ต้องเลือก Create initial file



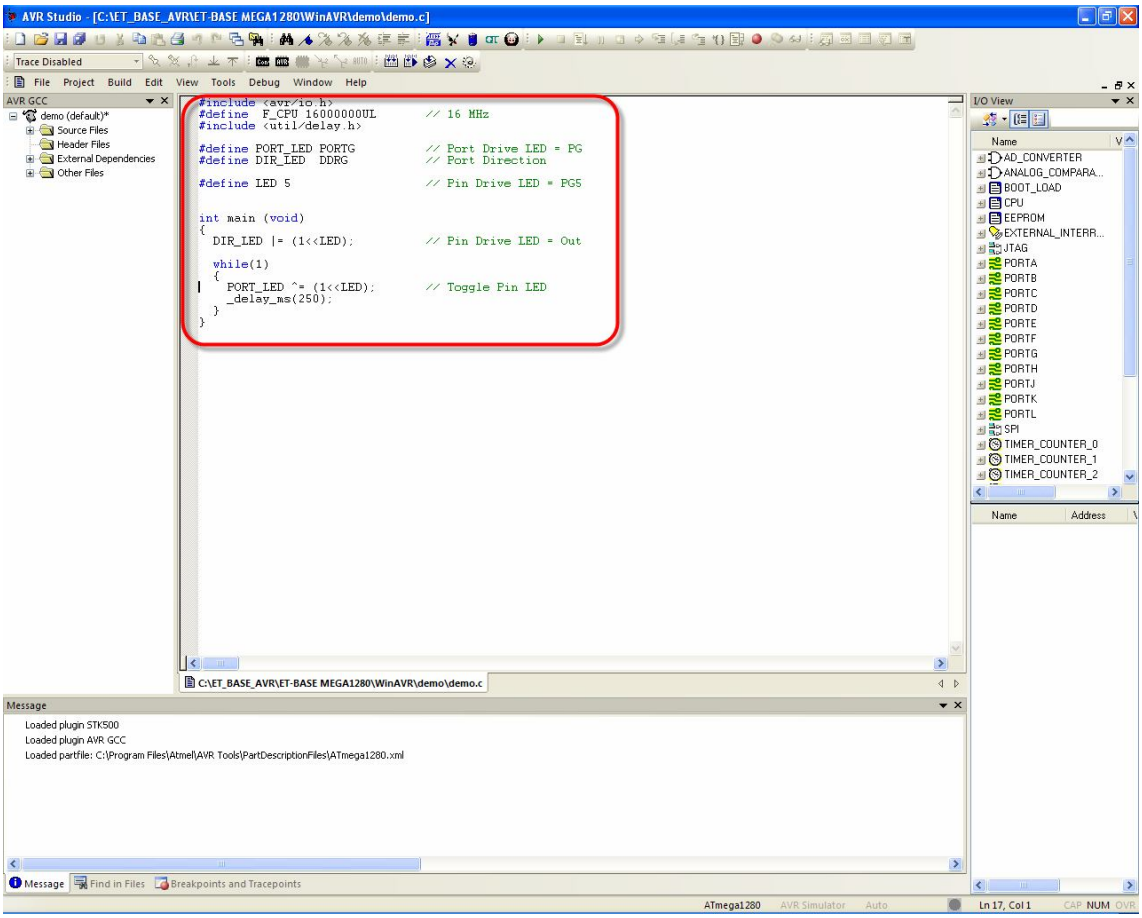
3. เมื่อกำหนดค่าตัวเลือกต่างๆ ให้กับโปรแกรมเรียบร้อยแล้ว ให้เลือกที่ Next แล้วกำหนดค่าใน Debug platform เป็น AVR Simulator และเลือก Device เป็น ATmega1280 ซึ่งเมื่อสร้าง project เสร็จโปรแกรมจะสร้างไฟล์ภาษาซีให้ โดยมีชื่อเดียวกับ project ไฟล์ ซึ่งในที่นี้จะเป็นไฟล์ชื่อ demo.c ให้เองโดยอัตโนมัติ เพียงแต่ไฟล์ดังกล่าวจะยังไม่มี code ใดๆบรรจุไว้ให้ เป็นเพียงหน้ากระดาษเปล่าๆ ซึ่งต้องรอให้เราเขียน code เพิ่มเข้าไปเอง ดังรูป



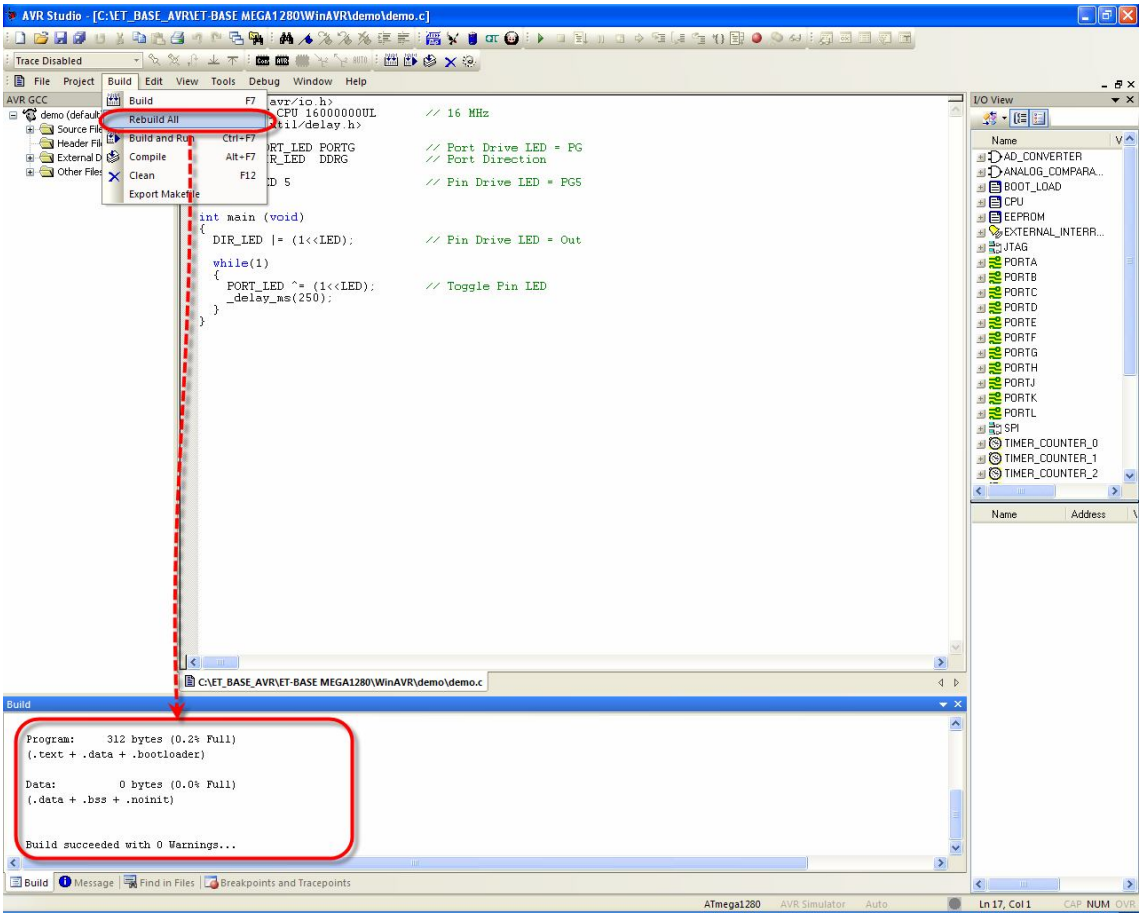
4. ให้พิมพ์คำสั่งของโปรแกรมสำหรับทดสอบการทำงาน ในหน้าต่าง Text Editor ของโปรแกรม โดยในที่นี้จะทดสอบด้วย Codeโปรแกรม สำหรับทำหน้าที่ทดสอบการทำงานของบอร์ดในเบื้องต้น โดยทำหน้าที่ ON/OFF หลอดแสดงผล LED ซึ่งต่อควบคุมจากขา PG5 ดังตัวอย่าง

```
#include <avr/io.h>
#define  F_CPU 16000000UL              // 16 MHz
#include <util/delay.h>

#define PORT_LED PORTG                 // Port Drive LED = PG
#define DIR_LED  DDRG                  // Port Direction
#define LED 5                          // Pin Drive LED = PG5

int main (void)
{
  DIR_LED |= (1<<LED);                 // Pin Drive LED = Out

  while(1)
  {
    PORT_LED ^= (1<<LED);              // Toggle Pin LED
    _delay_ms(250);
  }
}
```

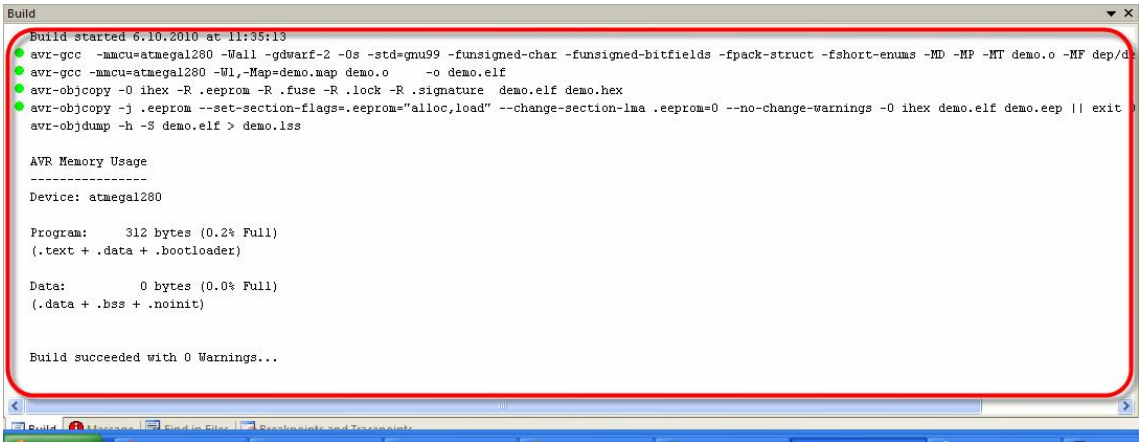
↓



5. หลังจากพิมพ์ Code โปรแกรมเสร็จแล้วให้สั่งแปลโปรแกรม โดยเลือกที่ build →rebuild all ซึ่งถ้าทุกอย่างถูกต้อง ผลการแปลคำสั่งจะได้ผลลัพธ์เป็น “Build succeeded with 0 Warnings... “ และจะรายงานผลการแปลพร้อมขนาดหน่วยความจำที่ใช้ไป และจะได้ Output เป็น HEX File ที่มีชื่อเดียวกันกับ project ที่สร้างไว้ โดยจะบรรจุอยู่ในDirectory ย่อยชื่อ default ดังรูป

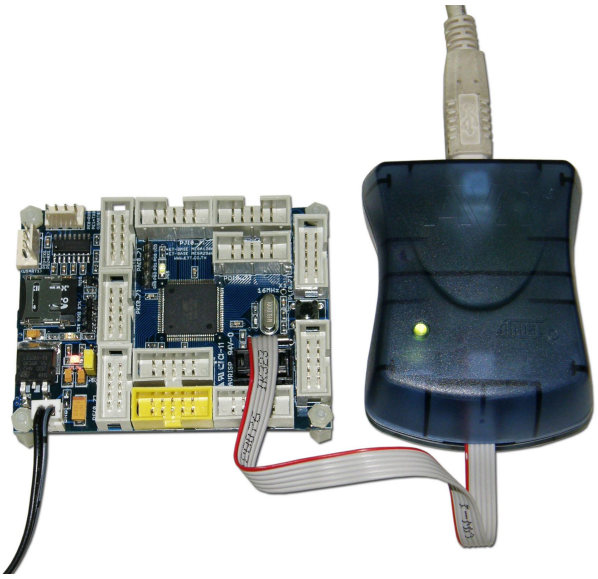


↓

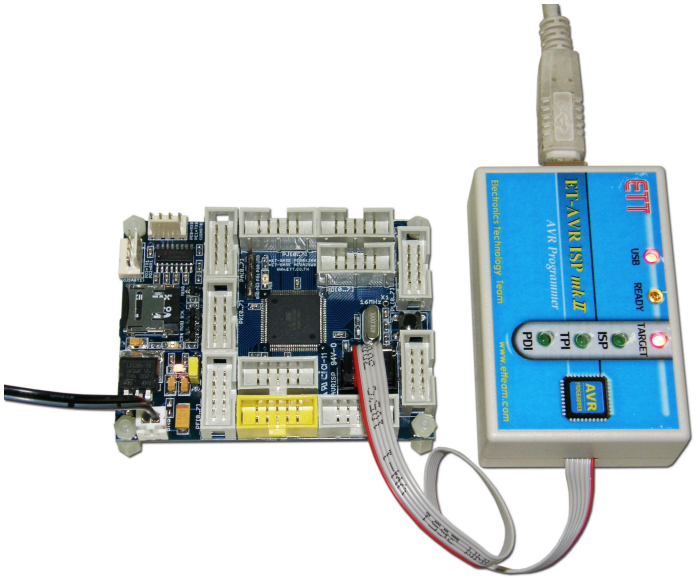


การโปรแกรม Hex File ให้บอร์ดโดยใช้ AVRISP mkII

ซึ่งเมื่อผ่านกระบวนการ Compile Code จนได้ Hex File มาแล้ว ขั้นตอนต่อไป จะเป็นขั้นตอนของการนำข้อมูล Code ใน Hex File เขียนเข้าไปในหน่วยความจำของ MCU ซึ่งขั้นตอนนี้สามารถทำได้หลายแนวทาง แต่ในที่นี้จะแนะนำให้ใช้เครื่องโปรแกรม AVRISP mkII หรือ อุปกรณ์อื่นที่มีความสามารถเทียบเท่ากัน เช่นเครื่องโปรแกรมรุ่น ET-AVRISP mkII ของ อีทีที โดยสามารถสั่งงานผ่านโปรแกรม AVR Studio4 ได้เลย ซึ่งในการเชื่อมต่อกับเครื่องโปรแกรมจะใช้ขั้วต่อISP ดังตัวอย่าง



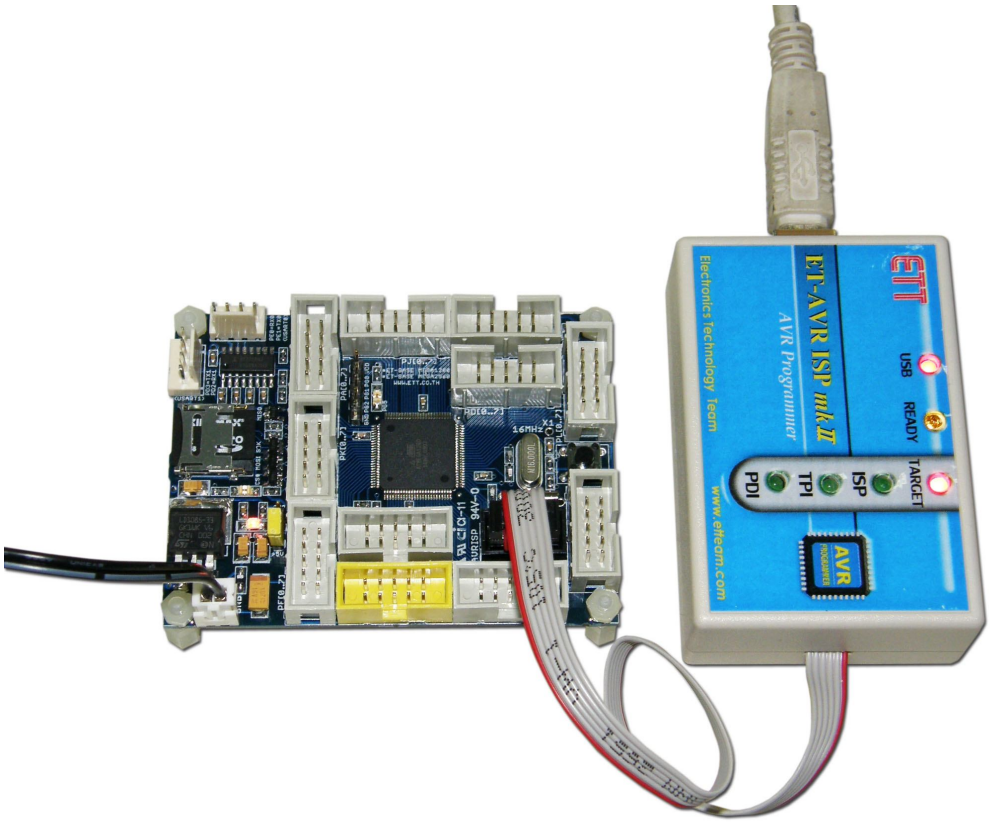
รูปแสดงตัวอย่างการต่อ AVRISP mkII ของ ATMEL



รูปแสดงตัวอย่างการต่อ ET-AVRISP mkII ของ อีทีที

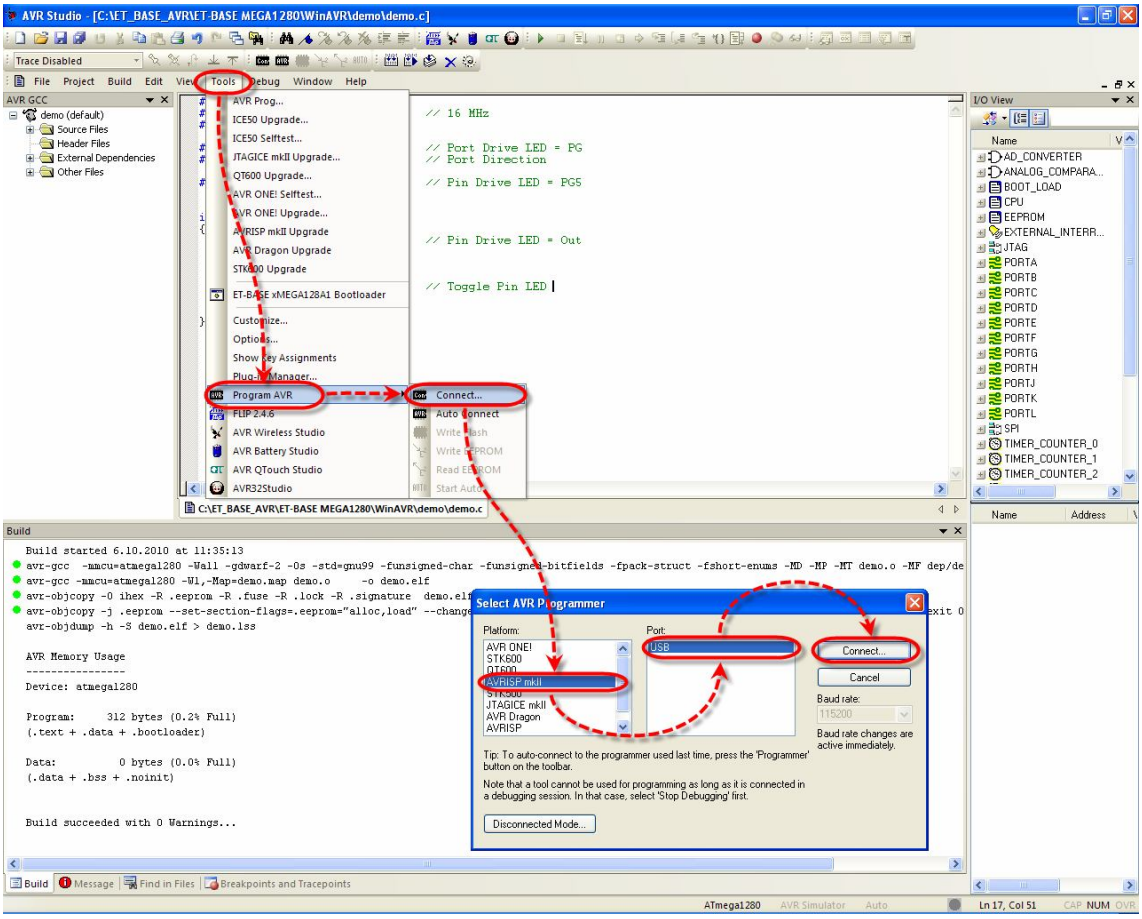
สำหรับลำดับขั้นตอนการโปรแกรม Hex File ด้วย AVRISP mkII มีดังนี้

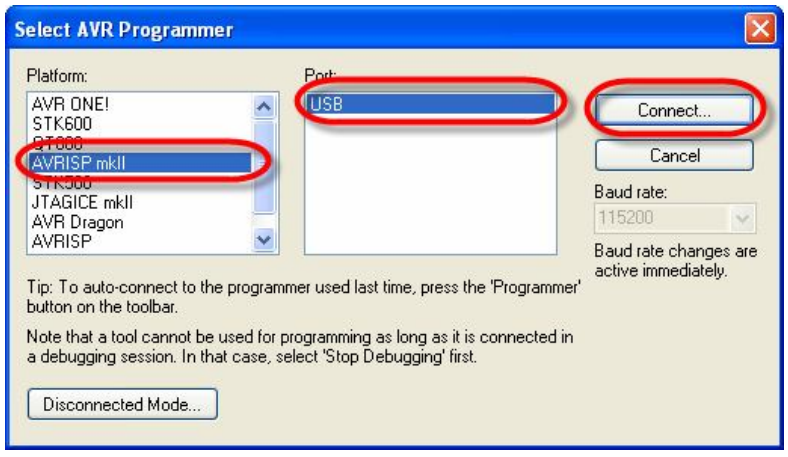
1. จ่ายไฟให้บอร์ดโดยใช้ Adapter จ่ายไฟขนาด 5VDC โดยให้ระมัดระวัง และ ตรวจสอบขั้วของแหล่งจ่ายไฟให้ถูกต้องด้วย ซึ่งถ้าถูกต้องควรจะเห็น LED Power ติดสว่างให้เห็น
2. ต่อสายสัญญาณโปรแกรมของ ISP ระหว่างเครื่องโปรแกรม AVRISP mkII เข้ากับขั้วต่อ ISP ของบอร์ด ET-BASE MEGA1280/2560 โดยให้ตรวจตำแหน่งของขาสัญญาณให้ดี ระวังอย่าเสียบสายกลับด้าน ซึ่งถ้าเป็นเครื่องโปรแกรมและบอร์ดของอีทีที จะเลือกใช้ Connector IDE แบบ 6Pin ชนิดที่ป้องกันการเสียบสายกลับด้านเพื่อป้องกันไว้อยู่แล้ว ถ้าพบผิดความผิดปรกติเช่น LED Power ดับขณะเสียบสายให้รีบถอดสายออกและตรวจสอบสาเหตุความผิดพลาดทันที



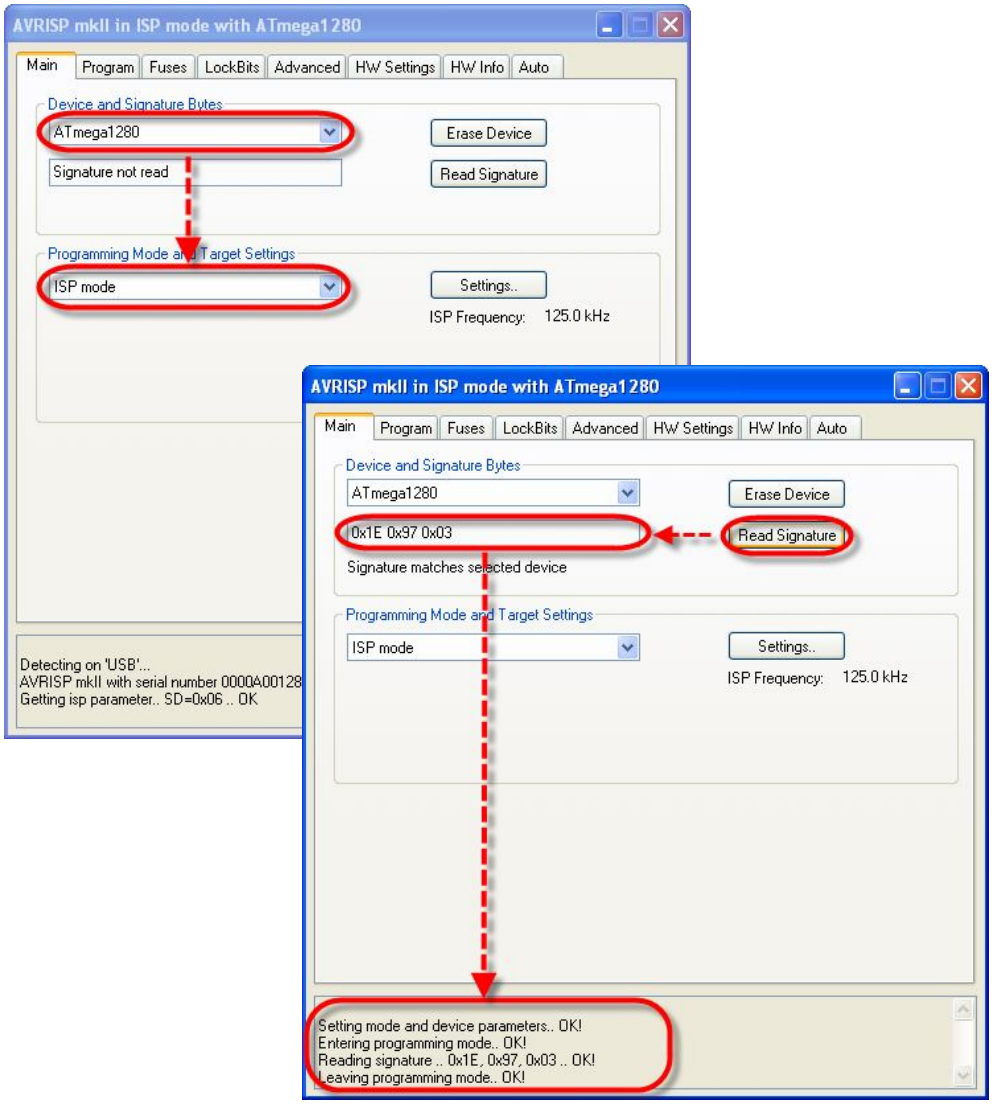
3. เสียบสาย USB ของเครื่อง AVRISP mkII เข้ากับเครื่องคอมพิวเตอร์ PC ซึ่งถ้าเป็นการใช้งานครั้งแรก Windows จะแจ้งว่าพบ new hardware และถามหาการติดตั้ง Driver ให้สั่งติดตั้ง Driver ให้เรียบร้อย โดยใน AVR Studio จะมีไฟล์ Driver ของ AVRISP mkII เตรียมไว้ให้ด้วยแล้ว โดยจะอยู่ใน C:\Program Files\Atmel\AVR Tools\usb ให้ทำการติดตั้ง Driver ให้เรียบร้อย (รายละเอียดศึกษาได้จากคู่มือการใช้งานของเครื่องโปรแกรม ET-AVRISP mkII)

4. เลือกคลิกเมาส์ที่ Tools → Program AVR → Connect.. → AVRISP mkII จากนั้นก็ให้เลือก port เป็น USB พร้อมกับเลือก Connect ดังรูป



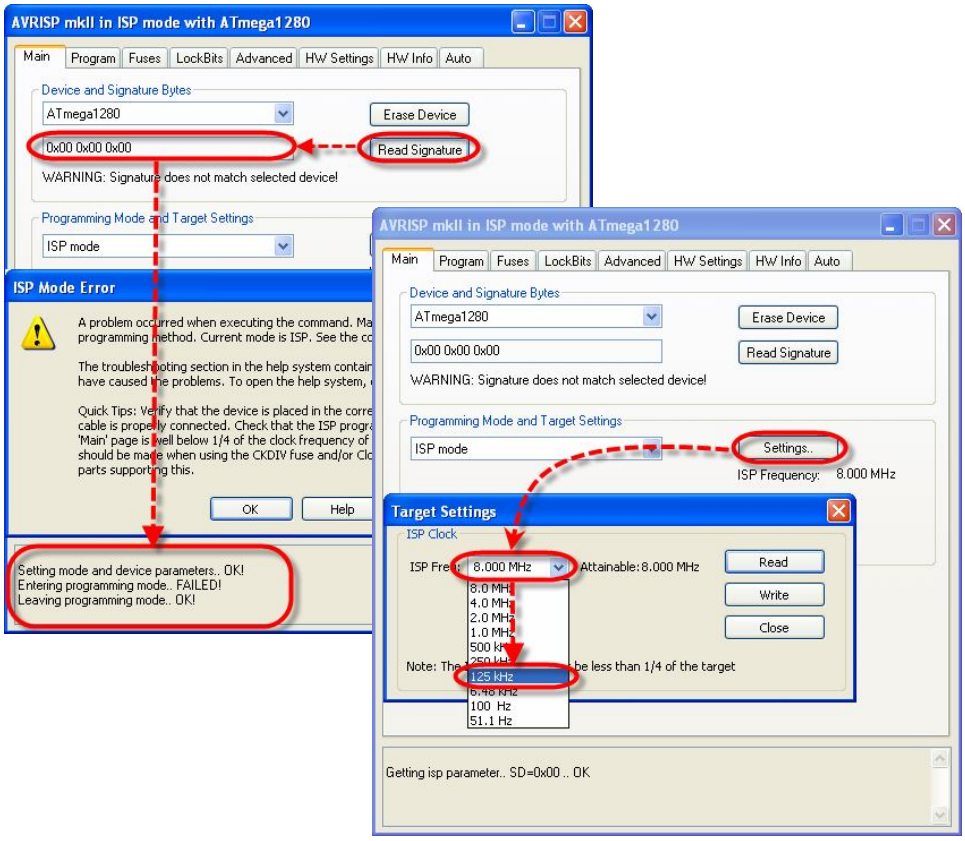


5. ถ้าทุกอย่างถูกต้องโปรแกรมจะเข้าสู่หน้าต่างโปรแกรมของ AVRISP mkII ให้ลองทดสอบการเชื่อมต่อระหว่าง ATMEGA1280/2560 กับ AVRISP mkII ดู ว่าสามารถสื่อสารกันได้เรียบร้อยหรือยัง โดยให้เลือกที่ tab ของ Main แล้วเลือกกำหนดเบอร์ MCU เป็น ATmega1280 หรือ ATmega2560 ให้ตรงกับเบอร์ที่ต่อไว้จริง พร้อมกับเลือกการเชื่อมต่อเป็น ISP mode แล้วลองเลือก Read Signature ดู ซึ่งถ้าทุกอย่างถูกต้องโปรแกรมควรต้องอ่านค่า Signature ของ ATMEGA1280 หรือ ATMEGA2560 ได้อย่างถูกต้อง ดังรูป



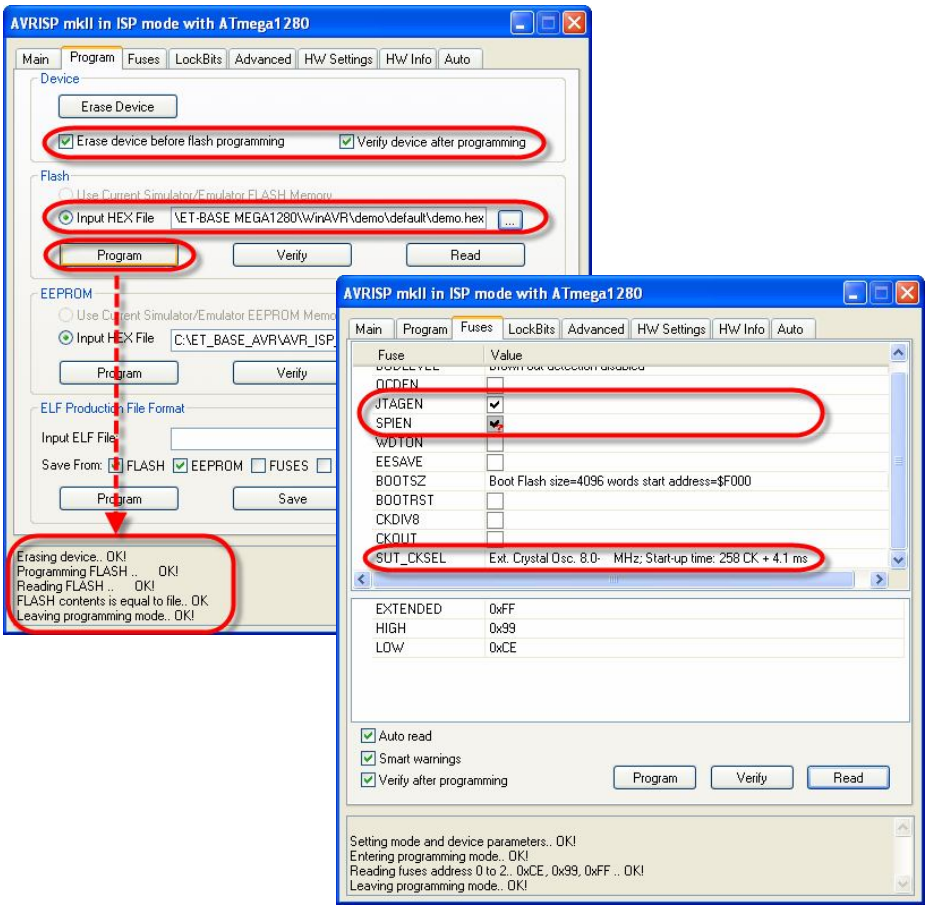
ในกรณีที่เกิดความผิดพลาดขึ้น ให้ลองตรวจสอบหาสาเหตุของความผิดพลาดและแก้ไขให้ถูกต้อง โดยมีแนวทางดังนี้

- การเชื่อมต่อสายระหว่างบอร์ด ET-BASE MEGA1280/2560 กับเครื่องโปรแกรม ถูกต้องหรือยัง บอร์ดอยู่ในสภาวะพร้อมที่จะทำงานหรือยัง
- ขาสัญญาณที่ทำหน้าที่เป็น ISP(PB1,PB2,PB3) มีการนำไปต่อใช้งานอย่างอื่นในขณะทำการโปรแกรมอยู่หรือไม่ ถ้ามีให้ปลดการเชื่อมต่อเหล่านั้นออกให้หมดเพื่อให้ขาสัญญาณดังกล่าวเป็นอิสระเสียก่อน
- กำหนดค่าความถี่ของ ISP Frequency ของเครื่องโปรแกรมไว้ สัมพันธ์สอดคล้องกันกับค่าความถี่ของ MCU ที่ทำงานใน Target Board หรือไม่ ซึ่งค่าความถี่ที่ถูกต้องของ ISP Frequency ต้องไม่เกิน 1/4 ของค่าความถี่ที่ MCU ทำงานอยู่ในขณะนั้น เช่น ถ้า MCU ทำงานที่ความถี่ 1MHz ค่าความถี่ของ ISP Frequency ต้องไม่เกิน 250KHz ตามปรกติถ้าเป็น MCU ตัวใหม่ๆจากโรงงานผลิต ค่าความถี่ของ MCU จากโรงงานจะถูกกำหนดให้ Run จากความถี่ 1MHz (Internal RC 8MHz / 8) ซึ่งถ้าไม่แน่ใจอาจทดลองปรับค่าความถี่ ISP Frequency ให้มีค่าต่ำๆดูก่อน เมื่อสามารถติดต่อสื่อสารกับ MCU ได้เรียบร้อยแล้วจึงค่อยเข้าไปตรวจสอบค่า Fuse Bit ของ MCU ที่เกี่ยวกับระบบสัญญาณนาฬิกา Clock ต่างๆ ในภายหลัง ดังรูป



6. เมื่อทุกอย่างถูกต้องให้เลือกไปที่ tab ของ Program พร้อมทั้งเลือก ตัวเลือกต่างๆดังนี้

- Device ให้เลือก Erase device before flash programming และ Verify device after programming
- Flash ให้เลือก Input HEX File ที่ต้องการจะโปรแกรมให้กับ MCU บนบอร์ด
- Fuses และ Lock Bits สามารถเลือกกำหนด และสั่งโปรแกรมค่าได้ตามต้องการ ซึ่งก่อนจะสั่ง Program ค่าของ Fuse Bit ผู้ใช้ควรต้องศึกษารายละเอียดในการกำหนดค่าให้เข้าใจ ซึ่งจะต้องสัมพันธ์สอดคล้องกับความต้องการของระบบ Hardware ที่ใช้อยู่ด้วย ถ้ายังไม่แน่ใจในรายละเอียดไม่ควรไปสั่งโปรแกรมค่า ของ Fuse Bit เหล่านี้ เพราะถ้ามีการโปรแกรมค่าของ Fuse Bit ผิดไปอาจส่งผลให้ MCU ทำงานผิดพลาดไปจากความต้องการ ซึ่งในเบื้องต้น ค่า Fuse Bit และ Lock Bit แนะนำให้ข้ามไปก่อนยังไม่ต้องสั่ง โปรแกรมค่าทั้ง สองนี้ ให้จัดการเฉพาะส่วนของการโปรแกรม Flash ด้วย Hex ดังรูป



ซึ่งหลังจากสั่งโปรแกรมเสร็จ MCU จะเริ่มต้นทำงานตามคำสั่งใน Hex ที่ได้สั่งโปรแกรมไปแล้วทันที ถ้าใช้เครื่องโปรแกรม ET-AVRISP mkII แต่ถ้าเป็นเครื่องโปรแกรมรุ่นอื่นอาจต้องทำการกดสวิตช์รีเซ็ตที่บอร์ดก่อน 1 ครั้ง เพื่อสั่งให้ MCU เริ่มต้นทำงานหลังจากโปรแกรมเสร็จ