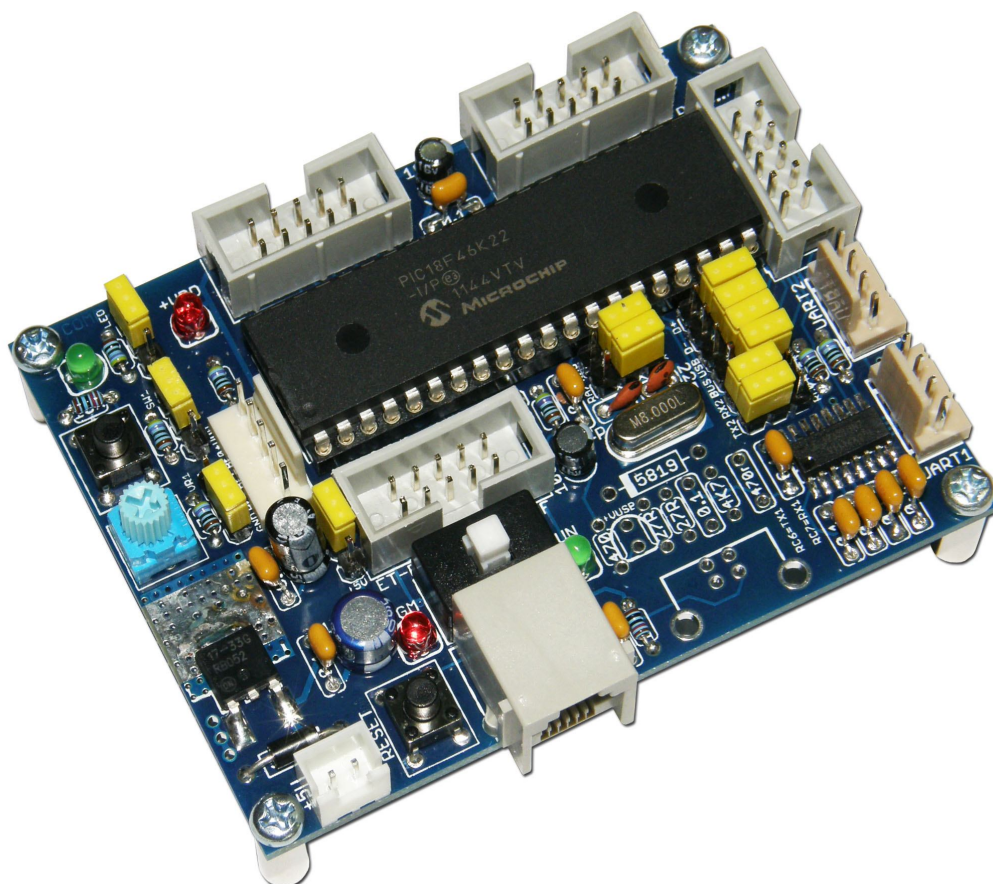


ET-BASE PIC40/46K22(ICSP)

ET-BASE PIC40/46K22(ICSP) เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล PIC ซึ่งออกแบบให้รองรับการติดตั้งใช้งานกับชิพไมโครคอนโทรลเลอร์รุ่น 40 Pin(40PDIP) เบอร์ PIC18F46K22 สามารถเลือกใช้กับระบบแหล่งจ่ายไฟเลี้ยงที่เป็น 3.3V หรือ 5V ให้เหมาะสมกับจุดประสงค์ในการใช้งานได้อีกด้วย

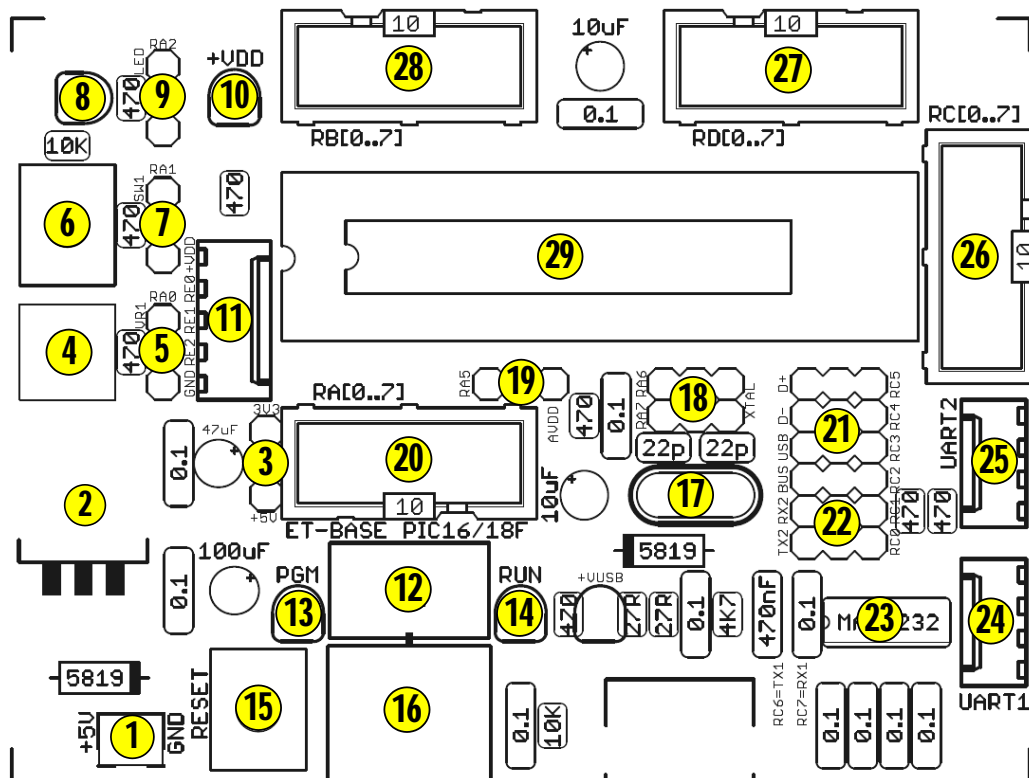
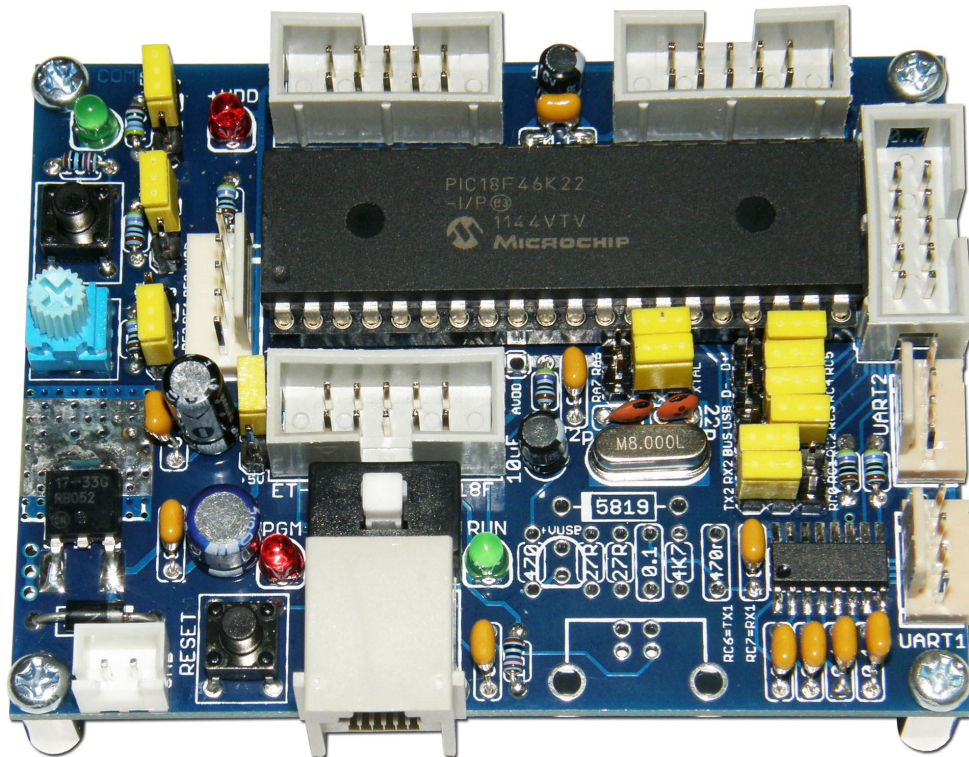
โดยโครงสร้างของบอร์ด ET-BASE PIC40/46K22(ICSP) ได้รับการออกแบบให้บอร์ดมีขนาดเล็กเหมาะต่อการนำไปประยุกต์ใช้งานและยังสามารถใช้เป็นชุดทดลองเรียนรู้เบื้องต้นได้อีกด้วย โดยภายในบอร์ดได้บรรจุเอาวางจอร์ที่จำเป็นต่อการใช้งาน และสะดวกต่อการพัฒนาโปรแกรม มีความยืดหยุ่น สามารถปรับเปลี่ยนสัญญาณ I/O เพื่อนำไปประยุกต์ใช้งานในลักษณะต่างๆ ให้สอดคล้องและเหมาะสมกับความต้องการใช้งานได้ในหลายๆลักษณะตามต้องการ

นอกจากนี้แล้วยังได้เพิ่มเติมอุปกรณ์ทดสอบ Input/Output แบบต่างๆในเบื้องต้นไว้ภายในบอร์ด เพื่อให้ผู้ใช้ได้ใช้เป็นเครื่องมือในการทดลองระหว่างการพัฒนาโปรแกรม เช่น VR ปรับค่าได้สำหรับทดสอบการทำงานของ ADC หรือ Push Button Switch สำหรับทดสอบ Input Logic หรือ LED สำหรับใช้ทดสอบการทำงานของ Output Logic เป็นต้น

คุณสมบัติของบอร์ด

- ใช้ MCU เบอร์ PIC18F46K22(40PDIP) เป็น MCU ประจำบอร์ด Run ความถี่สูงสุด 64MHz
 - 64KByte(32KWord) Flash / 1024 Byte EEPROM/ 3896 Byte SRAM
 - 36 GPIO
 - 30 Channel 10Bit ADC
 - 2 Channel Comparator
 - 3 Channel 8 Bit Timer / 4 Channel 16 Bit Timer
 - 2 Channel EUART
 - 2 Channel I2C / 2 Channel SPI
 - 2 Channel ECCP Full Bridge / 1 Channel ECCP Half Bridge
 - 2 Channel CCP
- มี Crystal ความถี่ 8.00MHz พร้อม Jumper ตัดต่อเมื่อไม่ต้องการใช้งาน
- มีวงจร Line Driver สำหรับพอร์ตสื่อสารอนุกรม UART แบบ RS232 จำนวน 2 ช่อง โดยใช้ขั้วต่อ UART แบบ CPA-4 Pin มาตรฐาน อีทีที
 - 1 ช่อง สำหรับ Hardware UART1 โดยใช้ Pin RC6(TX1) และ RC7(RX1) มาตรฐาน PIC
 - 1 ช่อง สำหรับ Software UART โดยใช้ Pin RC0(TX2) และ RC1(RX2) พร้อม Jumper สำหรับเลือกใช้งาน UART2(Software UART) หรือ GPIO ได้ตามต้องการ
- มีขั้ว ICSP มาตรฐาน ICD2 แบบ RJ11 สำหรับใช้ร่วมกับชุดพัฒนาโปรแกรมและ Debugger ที่รองรับการทำงานตามมาตรฐาน ICSP ของ Microchips เช่น ICD2/ICD3 หรือ Pickit2/Pickit3 ได้
- มี Switch สำหรับสลับสัญญาณระหว่าง Program/Debug(PGM) และ ใช้งานปกติ(RUN) พร้อม LED แสดงโหมดการทำงานของบอร์ด
- มีขั้วต่อสัญญาณ I/O แบบ Header ขนาด 2x5 จำนวน 4 ชุด และ Header CPA-5 Pin อีก 1 ชุด
- มี Switch Reset สำหรับสั่ง Reset การทำงานของ MCU ภายในบอร์ด
- มี VR ปรับค่าสำหรับทดสอบการทำงาน ADC Input โดยใช้ RA0 พร้อม Jumper ตัดต่อสัญญาณ
- มี Switch สำหรับทดสอบการทำงาน Digital Input โดยใช้ RA1 พร้อม Jumper ตัดต่อสัญญาณ
- มี LED สำหรับทดสอบการทำงาน Digital Output โดยใช้ RA2 พร้อม Jumper ตัดต่อสัญญาณ
- Power +5VDC Input พร้อม Regulate แบบ 3.3V/1A และ LED แสดงสถานะแหล่งจ่าย Power Jumper สำหรับ เลือกแหล่งจ่ายไฟเลี้ยงให้ MCU ว่าจะใช้เป็น +5VDC หรือ 3.3VDC
- ขนาด PCB Size เล็กเพียง 8 x 6 cm.

โครงสร้างบอร์ด ET-BASE PIC40/46K22(ICSP)



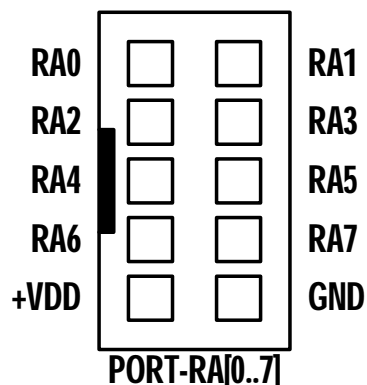
รูปแสดง โครงสร้างของบอร์ด ET-BASE PIC40/46K22(ICSP)

- หมายเลข 1 คือ ขั้วต่อแหล่งจ่ายไฟเลี้ยงวงจรของบอร์ด ใช้กับแหล่งจ่ายไฟ +5VDC
- หมายเลข 2 คือ IC Regulate ขนาด 3.3VDC/1A
- หมายเลข 3 คือ Jumper สำหรับเลือกขนาดแรงดันไฟเลี้ยง MCU(+VDD) ระหว่าง 3.3V หรือ 5V
- หมายเลข 4 คือ VR ปรับค่าแรงดัน สำหรับใช้ทดสอบการทำงานของ Input Analog(ADC)
- หมายเลข 5 คือ Jumper สำหรับ ตัดต่อ สัญญาณ RA0 กับแรงดันปรับค่าจาก VR1
- หมายเลข 6 คือ Switch กดติดปล่อยดับ สำหรับใช้ทดสอบการทำงานของ Digital Input
- หมายเลข 7 คือ Jumper สำหรับตัดต่อสัญญาณ RA1 กับ Digital Input จาก SW1
- หมายเลข 8 คือ LED สำหรับใช้ทดสอบการทำงานของ Digital Output
- หมายเลข 9 คือ Jumper สำหรับตัดต่อสัญญาณ RA2 กับ Digital Output ให้กับ LED
- หมายเลข 10 คือ LED แสดงสถานะของแหล่งจ่ายไฟ +VDD
- หมายเลข 11 คือ ขั้วต่อสัญญาณ RE[0..2] ซึ่งจะถูกใช้งานในกรณีติดตั้ง MCU รุ่น 40Pin เท่านั้น
- หมายเลข 12 คือ สวิตช์ สำหรับเลือกโหมดการทำงานระหว่าง Run(RUN) และ Program(PGM)
- หมายเลข 13 คือ LED สีแดง แสดงสถานะ PGM เมื่อบอร์ดทำงานใน Program Mode
- หมายเลข 14 คือ LED สีเขียว แสดงสถานะ RUN เมื่อบอร์ดทำงานใน Run Mode
- หมายเลข 15 คือ สวิตช์ Reset สำหรับ Reset การทำงานของ MCU เมื่ออยู่ในโหมด Run
- หมายเลข 16 คือ ขั้วต่อ ICSP สำหรับใช้เชื่อมต่อกับเครื่องโปรแกรมและดีบั๊กตามมาตรฐาน ICD2
- หมายเลข 17 คือ Crystal ค่าความถี่ 8.00MHz
- หมายเลข 18 คือ Jumper สำหรับเลือกขาสัญญาณ RA6,RA7 ของ MCU ว่าจะให้เชื่อมต่อไปเป็น GPIO ที่ขั้ว RA[0..7] หรือวงจร Crystal
- หมายเลข 19 คือ Jumper สำหรับเลือกหน้าที่ของ Pin7 ของ MCU รุ่น 28Pin ว่าเป็นสัญญาณ RA5 ของ MCU โดยให้เชื่อมต่อไปเป็น GPIO ที่ขั้ว RA[0..7] หรือเป็นขา +AVDD โดยในกรณีของ PIC18F46K22 นั้น Jumper นี้จะกำหนดตายตัวไว้เป็น RA5 เสมอ
- หมายเลข 20 คือ ขั้วต่อ IDE10Pin ของ RA[0..7]
- หมายเลข 21 คือ Jumper สำหรับเลือกขาสัญญาณ RC2,RC3,RC4,RC5 ของ MCU ว่าจะให้เชื่อมต่อไปเป็น GPIO ที่ขั้ว RC[0..7] หรือใช้เชื่อมต่อกับ USB Bus(VUSB,VBUS,D-,D+)
- หมายเลข 22 คือ Jumper สำหรับเลือกขาสัญญาณ RC0,RC1 ของ MCU ว่าจะให้เชื่อมต่อไปเป็น GPIO ที่ขั้ว RC[0..7] หรือ TX2,RX2 ของ UART2
- หมายเลข 23 คือ ไอซี Line Driver ของ RS232(MAX3232/ICL3232) ใช้สำหรับแปลง Level ระหว่างสัญญาณของ UART Logic กับ สัญญาณมาตรฐาน RS232 Level ของ UART1,UART2

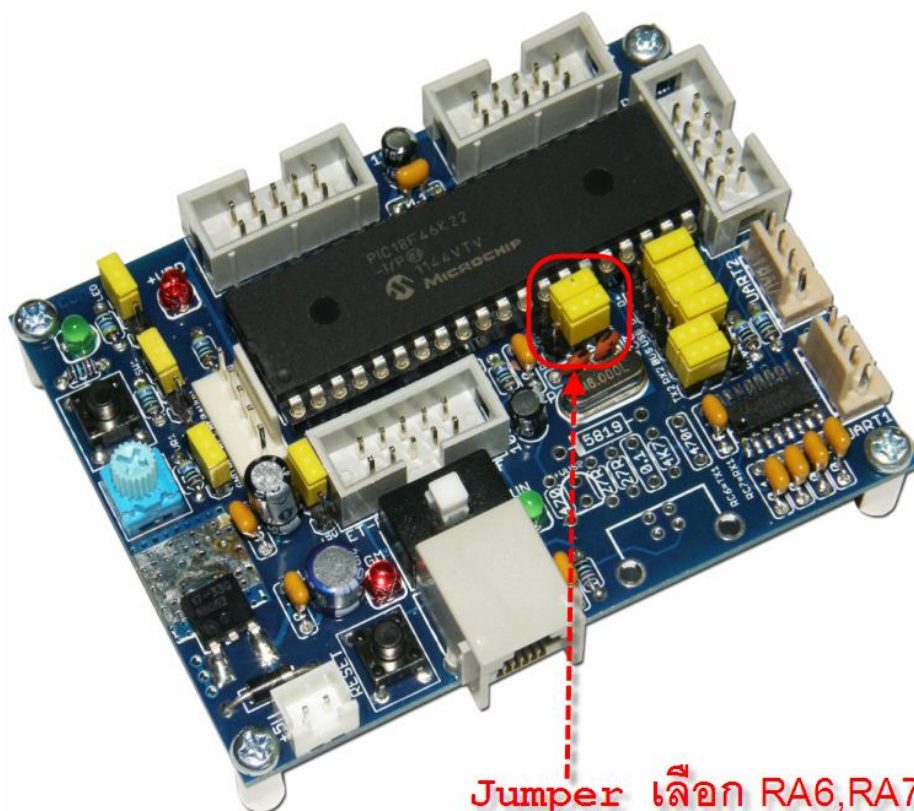
- หมายเลข **24** คือ ขั้วต่อ UART1 โดยเป็นสัญญาณแบบ RS232 รองรับ Hardware UART ซึ่งใช้ Pin ของ RC6(TX1) และ RC7(RX1) เป็นสัญญาณเชื่อมต่อ
- หมายเลข **25** คือ ขั้วต่อ UART2 โดยเป็นสัญญาณแบบ RS232 รองรับ Software UART ซึ่งใช้ Pin ของ RC0(TX2),RC1(RX2) เป็นสัญญาณเชื่อมต่อ
- หมายเลข **26** คือ ขั้วต่อ IDE10Pin ของ RC[0..7]
- หมายเลข **27** คือ ขั้วต่อ IDE10Pin ของ RD[0..7]
- หมายเลข **28** คือ ขั้วต่อ IDE10Pin ของ RB[0..7]
- หมายเลข **29** คือ MCU ประจำบอร์ด เบอร์ PIC18F46K22

หัวข้อสัญญาณต่างๆ

PORT RA[0..7]

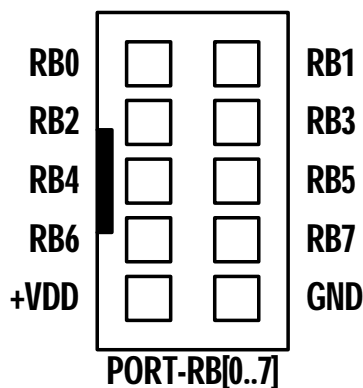


PORT-RA[0..7] เป็นสัญญาณจาก RA0...RA7 ของ MCU โดย RA0...RA5 จะเป็นสัญญาณที่ต่อตรงมาจาก Pin ของ MCU โดยตรง ส่วนสัญญาณ RA6 และ RA7 จะมี Jumper สำหรับเลือกตัดต่อสัญญาณร่วมอยู่ด้วยว่าต้องการใช้ RA6,RA7 เป็น GPIO หรือ ใช้เชื่อมต่อกับวงจรกำเนิดความถี่ Crystal ซึ่งในกรณีของ PIC18F46K22 จะสามารถเลือกใช้สัญญาณนาฬิกาจากภายใน Internal RC แล้วนำสัญญาณ RA6,RA7 มาใช้งานเป็น GPIO ตามปกติได้

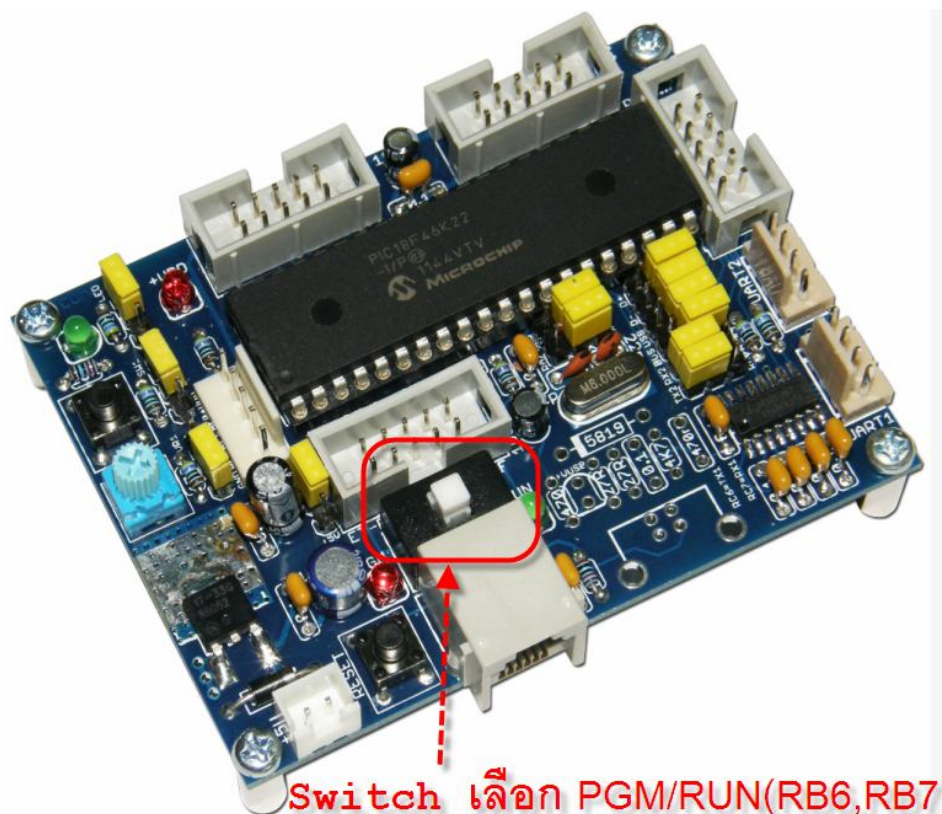


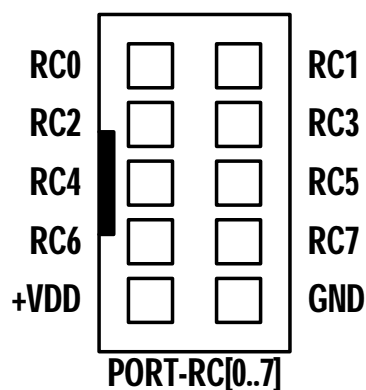
Jumper เลือก RA6,RA7

PORT RB[0...7]



PORT-RB[0..7] เป็นสัญญาณจาก RB0...RB7 ของ MCU โดย RB0...RB5 จะเป็นสัญญาณที่ต่อตรงมาจาก Pin ของ MCU โดยตรง ส่วนสัญญาณ RB6 และ RB7 จะมี Switch สำหรับเลือกตัดต่อสัญญาณร่วมอยู่ด้วยว่าต้องการใช้ RB6,RB7 เป็น GPIO หรือ ใช้เชื่อมต่อเครื่องโปรแกรมแบบ ICSP ทางหัว ICSP(RJ11) โดยถ้าเลือกสวิตช์ไว้ในตำแหน่ง RUN จะเป็นการเชื่อมต่อขา RB6,RB7 มายังหัวต่อนี้ แต่ถ้าเลือกสวิตช์ไว้ในตำแหน่ง PGM จะเป็นการเลือกเชื่อมต่อสัญญาณ RB6,RB7 ของ MCU เข้ากับเครื่องโปรแกรม ผ่านทางหัว ICSP(RJ11) แทน

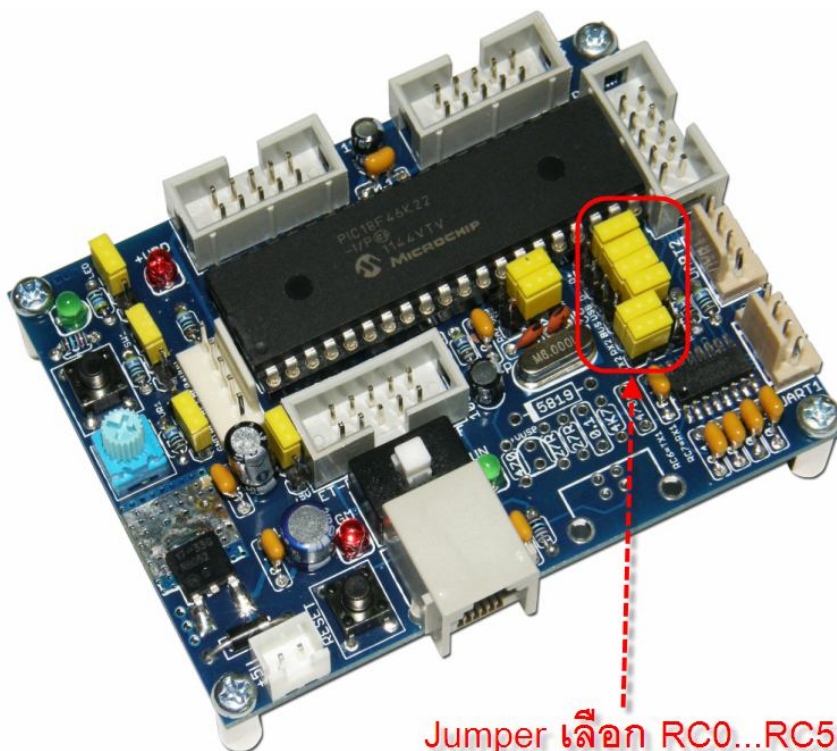


PORT-RC[0...7]

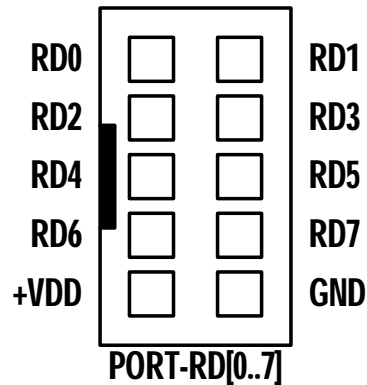
PORT-RC[0..7] เป็นสัญญาณจาก RC0...RC7 ของ MCU โดย RC0...RC5 จะเป็นสัญญาณที่ผ่านการเลือกจาก Jumper มาก่อน ส่วน RC6 และ RC7 จะเป็นสัญญาณที่ต่อตรงจาก Pin ของ MCU

RC0,RC1 จะมี Jumper สำหรับเลือกว่าจะให้ขา RC0 และ RC1 เชื่อมต่อมายัง Connector นี้ หรือจะเชื่อมต่อ RC0,RC1 ไปยังวงจร Line Driver ของ UART2 (Software UART)

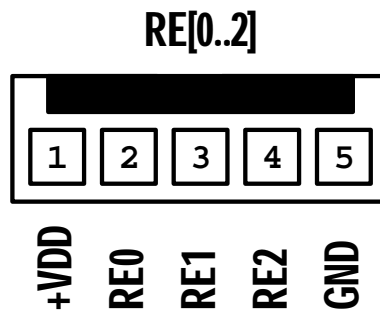
RC2...RC5 จะมี Jumper สำหรับเลือกว่าจะให้ขา RC2...RC5 เชื่อมต่อมายัง Connector นี้ หรือจะเชื่อมต่อไปยังวงจรของการเชื่อมต่อกับ USB ซึ่งในกรณีของ PIC18F46K22 ซึ่งโครงสร้างภายใน MCU ไม่มีวงจรการเชื่อมต่อกับ USB อยู่ด้วย ต้องเลือกกำหนด Jumper ของ RC2,RC3,RC4,RC5 ไว้ทางด้าน GPIO เพื่อเชื่อมต่อขา RC2,RC3,RC4,RC5 มายัง Connector นี้เสมอ ดังรูป



Jumper เลือก RC0...RC5

PORT RD[0...7]

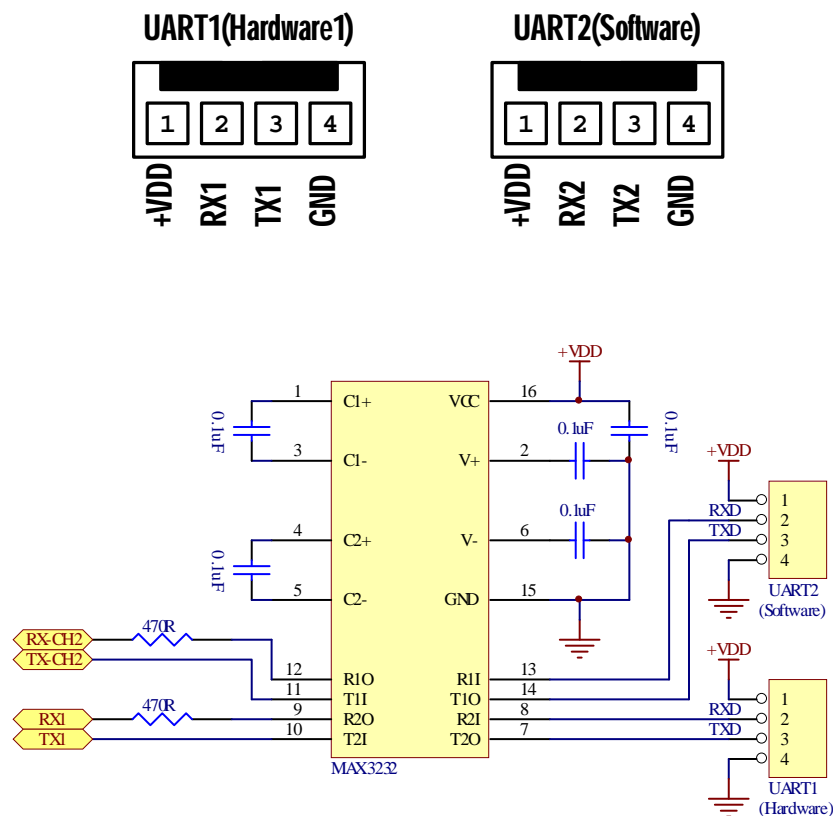
PORT-RD[0..7] เป็นสัญญาณจาก RD0...RD7 ของ MCU โดยเป็นสัญญาณที่ต่อตรงมาจาก Pin ของ MCU โดยตรงทั้ง 8 เส้น

PORT-RE[0..2]

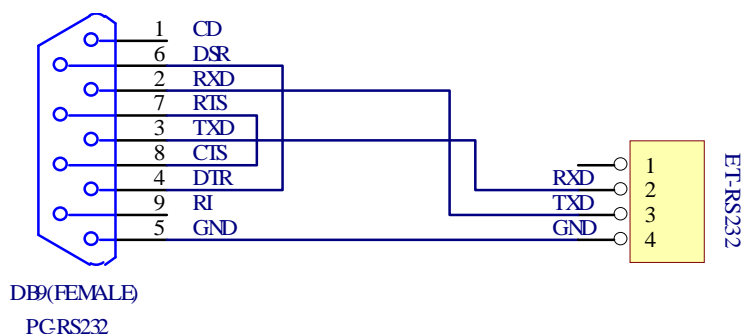
PORT-RE[0...2] เป็นสัญญาณจาก RE0...RE2 ของ MCU โดยเป็นสัญญาณที่ต่อตรงมาจาก Pin ของ MCU โดยตรงทั้ง 3 เส้น

การใช้งาน RS232

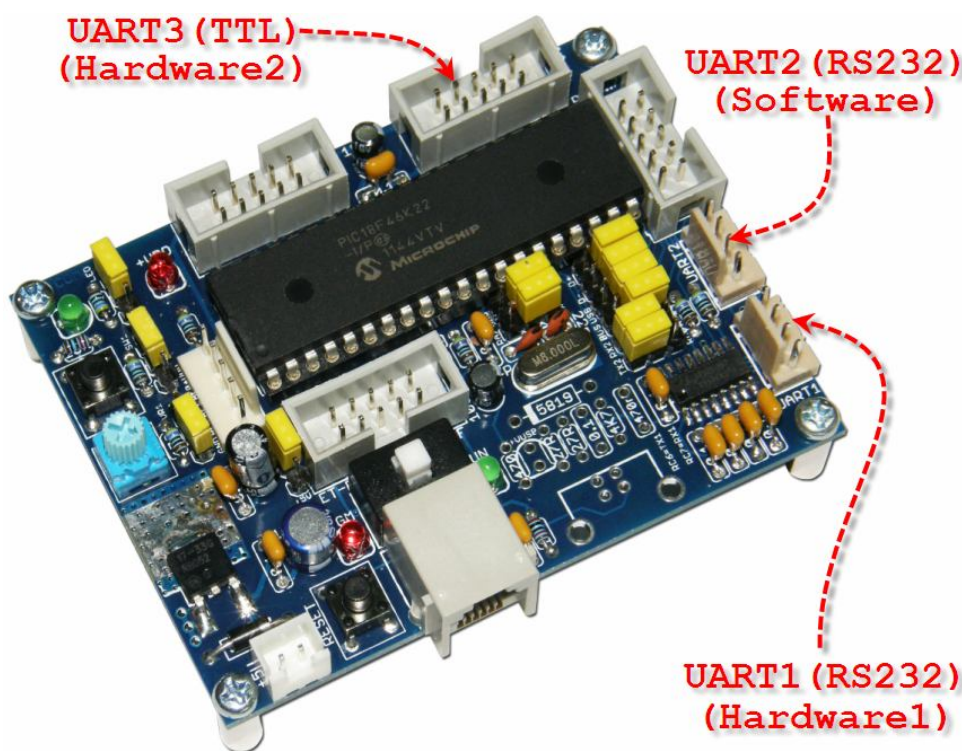
พอร์ต **RS232** เป็นสัญญาณ RS232 ซึ่งผ่านวงจรแปลงระดับสัญญาณจาก MAX3232 เรียบร้อยแล้ว โดยในกรณีของ PIC18F46K22 นั้นตามปกติแล้วจะมี Hardware UART จำนวน 2 ช่อง โดยใช้ขาสัญญาณ RC6(TX1) และ RC7(RX1) และ RD6(TX2), RD7(RX2) ในการเชื่อมต่อ แต่ในส่วนวงจรของบอร์ด จะออกแบบให้ใช้ Hardware UART เพียง 1 ช่องสำหรับเชื่อมต่อกับวงจร Line Driver แบบ RS232 คือ UART1(Hardware UART1) ส่วน Hardware UART อีก 1 ช่อง (Hardware UART2:RD6/RD7) จะปล่อยว่างเป็นอิสระไว้สำหรับให้ผู้ใช้นำสัญญาณดังกล่าวไปออกแบบใช้งานต่างๆ ได้โดยอิสระตามความต้องการ เช่น ใช้เป็น RS422/RS485 หรือ ใช้งานสื่อสารแบบ UART กับอุปกรณ์อื่นๆ โดยตรงแบบ TTL Level และวงจรของบอร์ดได้เพิ่มเติมวงจร Line Driver สำหรับ UART2(Software UART) จัดเตรียมเพิ่มเติมไว้ให้ด้วย โดยในช่องที่2 จะใช้ RC0(TX2) และ RC1(RX2) พร้อม Jumper เลือกตัดต่อสัญญาณของ UART2 ว่าจะใช้หรือไม่งานได้ตามต้องการ โดยสัญญาณของ RS232 แต่ละช่อง จะจัดหัวเป็นแบบ CPA-4PIN (RS232) ดังรูป



สำหรับ Cable ที่จะใช้ในการเชื่อมต่อ RS232 ระหว่าง Comport ของเครื่องคอมพิวเตอร์ PC เข้ากับหัวต่อ RS232 ของบอร์ด ET-BASE PIC40/46K22(ICSP) นั้น เป็นดังนี้



รูป แสดงวงจรสาย Cable สำหรับ RS232



หมายเหตุ สำหรับ UART2 (Software UART) นั้น ในการเขียนใช้งานต้องขึ้นอยู่กับความสามารถของ Compiler ด้วยว่าสนับสนุน Library แบบ Software UART ด้วยหรือไม่ ซึ่งกรณีใช้ตัวแปลภาษาของ PIC CCS Compiler เป็นตัวแปลภาษา จะสามารถใช้ความสามารถของ Software UART ได้โดยง่าย ในกรณีไม่ต้องการใช้งานขา RC0/RC1 เป็น UART2 ก็สามารถเลือก Jumper เพื่อกำหนดหน้าที่ของขาสัญญาณทั้ง 2 ไปเป็น GPIO ได้ตามต้องการ เนื่องจากวงจร Driver ในส่วนของ UART2 ถูกออกแบบให้มีความยืดหยุ่นต่อการใช้งาน โดยมี Jumper RC0/TX2 และ RC1/RX2 ในการติดต่อสัญญาณเพื่อให้ผู้ใช้สามารถเลือกใช้หรือไม่ใช้ได้ตามต้องการ ดังตัวอย่าง

```

/*****
/* Demo Program For ET-BASE PIC40/46K22 UART Demo */
/* MCU Control : PIC18F46K22 */
/*          : Run 64MHz (Internal 16 MHz + PLL) */
/* +VDD Power : +5V/+3V3 Operate Voltage */
/* Function   : Demo UART1, UART2 Echo Test */
*****/
#include <18F46K22.h>
#include <stdlib.h>

// Fuses: PIC18F46K22 (CCS Compiler)
// Fuses: LP, XT, HSH, HSM, ECH, ECH_IO, RC, RC_IO, INTRC_IO, INTRC, ECM, ECM_IO
// Fuses: ECL, ECL_IO, NOPLLEN, PLEN, PRIMARY_SW, PRIMARY_ON, NOFCMEN, FCMEN
// Fuses: NOIESO, IESO, PUT, NOPUT, NOBROWNOUT, BROWNOUT_SW, BROWNOUT_NOSL
// Fuses: BROWNOUT, BORV45, BORV42, BORV27, BORV20, NOWDT, WDT_NOSLEEP
// Fuses: WDT_SW, WDT, WDT1, WDT2, WDT4, WDT8, WDT16, WDT32, WDT64, WDT128
// Fuses: WDT256, WDT512, WDT1024, WDT2048, WDT4096, WDT8192, WDT16384
// Fuses: WDT32768, CCP2B3, CCP2C1, NOPBADEN, PBADEN, CCP3E0, CCP3B5
// Fuses: NOHFOFST, HFOFST, TIMER3B5, TIMER3C0, CCP2C0, CCP2D2, NOMCLR, MCLR
// Fuses: NOSTVREN, STVREN, NOLVP, LVP, NOXINST, XINST, DEBUG, NODEBUG
// Fuses: PROTECT, NOPROTECT, CPB, NOCPB, CPD, NOCPD, WRT, NOWRT, WRTC, NOWRTC
// Fuses: WRTB, NOWRTB, WRTD, NOWRTD, EBTR, NOEBTR, EBTRB, NOEBTRB
#fuses INTRC, NOWDT, NOPROTECT, NODEBUG, PLEN //Internal 16MHz + PLL(4)
#use delay(clock=64000000)

/* Config and Enable Hardware UART1(RC6=TX1, RC7=RX1) */
#use rs232(uart1, baud=9600, stream=CH1)

/* Config and Enable Software UART2(RC0=TX2, RC1=RX2) */
#define TX2 PIN_C0
#define RX2 PIN_C1
#use rs232(baud=9600, xmit=TX2, rcv=RX2, stream=CH2)

/* Config and Enable Hardware UART2(RD6=TX3, RD7=RX3) */
#use rs232(uart2, baud=9600, stream=CH3)

/*****
/* Main Program */
*****/
void main()
{
    char rx_buff;

    //Start-Up UART1
    fprintf(CH1, "\n\n\n\r");
    fprintf(CH1, "Demo UART1 ET-BASE PIC40/46K22(ICSP)\n\r");
    fprintf(CH1, "Run 64.00 MHz (Internal RC 16 MHz + PLL)\n\r");
    fprintf(CH1, "UART1>");

    //Start-Up UART2
    fprintf(CH2, "\n\n\n\r");
    fprintf(CH2, "Demo UART2 ET-BASE PIC40/46K22(ICSP)\n\r");
    fprintf(CH2, "Run 64.00 MHz (Internal RC 16 MHz + PLL)\n\r");
    fprintf(CH2, "UART2>");

    //Start-Up UART3
    fprintf(CH3, "\n\n\n\r");
    fprintf(CH3, "Demo UART3 ET-BASE PIC40/46K22(ICSP)\n\r");
    fprintf(CH3, "Run 64.00 MHz (Internal RC 16 MHz + PLL)\n\r");
    fprintf(CH3, "UART3>");
}

```



```

while(true)
{
    //Verify & Echo UART1
    if(kbhit(CH1))
    {
        rx_buff = fgetc(CH1);
        if (rx_buff == 0x0D)
        {
            fprintf(CH1, "\n\r");
            fprintf(CH1, "ET-BASE PIC40/46K22(ICSP)\n\r");
            fprintf(CH1, "UART1>");
        }
        else
        {
            fputc(rx_buff, CH1);           // Echo Received Characters
        }
    }

    //Verify & Echo UART2
    if(kbhit(CH2))
    {
        rx_buff = fgetc(CH2);
        if (rx_buff == 0x0D)
        {
            fprintf(CH2, "\n\r");
            fprintf(CH2, "ET-BASE PIC40/46K22(ICSP)\n\r");
            fprintf(CH2, "UART2>");
        }
        else
        {
            fputc(rx_buff, CH2);           // Echo Received Characters
        }
    }

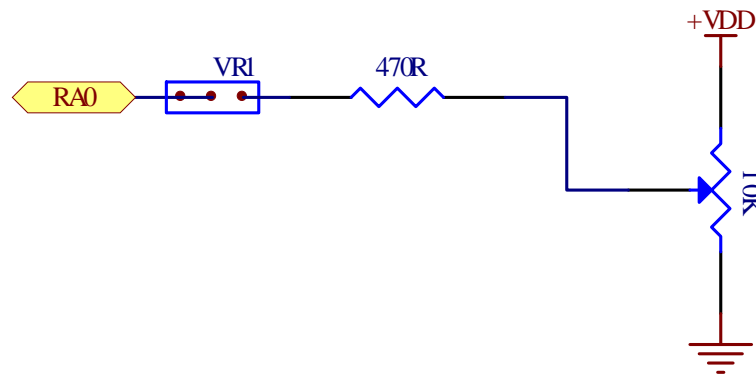
    //Verify & Echo UART3
    if(kbhit(CH3))
    {
        rx_buff = fgetc(CH3);
        if (rx_buff == 0x0D)
        {
            fprintf(CH3, "\n\r");
            fprintf(CH3, "ET-BASE PIC40/46K22(ICSP)\n\r");
            fprintf(CH3, "UART3>");
        }
        else
        {
            fputc(rx_buff, CH3);           // Echo Received Characters
        }
    }
}
}

```

แสดงตัวอย่าง Code ติดต่อ UART ภาษาซี (PICC CCS Compiler)

การใช้งาน VR ปรับแรงดัน

VR1 เป็นตัวต้านทานปรับค่าได้ใช้สำหรับปรับค่าแรงดันระหว่าง +VDD และ GND เพื่อใช้สร้างระดับแรงดัน สำหรับใช้ทดสอบการทำงานของ Input แบบ Analog (ADC) โดยแรงดันที่ได้จากการปรับ VR1 จะถูกเชื่อมต่อไปยัง ขาสัญญาณ RA0 ของ MCU โดยมี Jumper ตัดต่อสัญญาณเพื่อให้ผู้ใช้สามารถเลือกใช้หรือไม่ใช้แรงดันทดสอบที่ได้จากการปรับ VR1 นี้ ได้โดยอิสระ



เนื่องจาก Pin RA0 ของ PIC18F46K22 สามารถโปรแกรมหน้าที่การทำงานได้หลายหน้าที่ ในกรณีที่ต้องการใช้เป็น ADC(AN0) ต้องทำการกำหนดค่าบิต ANSA0 ของรีจิสเตอร์ ANSELA ให้มีค่าเป็น "1" ก่อนเสมอ แล้วจึงสั่ง Initial การทำงานของ ADC ในส่วนอื่นๆตามต้องการ ไมเช่นนั้นอาจจะไม่สามารถใช้งานขา RA0 เป็น ADC ได้ ดังตัวอย่าง

```
...
void main(void)
{
    unsigned char Result;

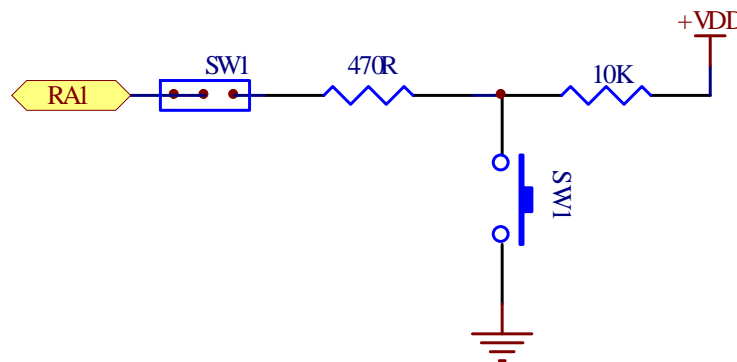
    //Initial RA0 = ADC(AN0)
    ANSELAbits.ANSA0 = 1; // RA0 = Analog Function
    TRISAbits.TRISA0 = 1; // RA0 = Analog Input
    ADCON2bits.ADFM = 0; // ADC Result = Left justified
    ADCON2bits.ACQT = 7; // ADC Acquisition Time = 20TAD
    ADCON2bits.ADCS = 6; // ADC Conversion Clock=Fosc/64
    ADCON1bits.PVCFG = 0; // ADC VREF+ connected to internal signal, AVDD
    ADCON1bits.NVCFG = 0; // ADC VREF- connected to internal signal, AVSS
    ADCON0bits.CHS = 0; // Select ADC Channel = AN0
    ADCON0bits.ADON = 1; // Start ADC

    ...
    ADCON0bits.GO = 1; // Start Conversion
    while(ADCON0bits.NOT_DONE == 1); // wait for it to complete
    Result = ADRESH; // return high byte of result
}
```

แสดง ตัวอย่าง Code (PIC-C18 Compiler) สำหรับ Initial ADC

การใช้วงจร SW1

SW1 เป็นวงจรสวิตช์ กดติด-ปล่อยดับ ใช้สำหรับสร้างสัญญาณลอจิก "0" และ "1" เพื่อทดสอบการทำงานของ Input แบบ Logic เช่น ทดลองการตรวจจับค่าการกดสวิตช์ โดยเมื่อสวิตช์ไม่ถูกกดจะได้ค่าสถานะทางลอจิกเป็น "1" แต่ถ้าสวิตช์ถูกกดจะได้ค่าสถานะทางลอจิกเป็น "0" โดยสัญญาณลอจิกที่ได้จากวงจรมี จะถูกเชื่อมต่อไปยัง ขาสัญญาณ RA1 ของ MCU โดยมี Jumper ตัดต่อสัญญาณเพื่อให้ผู้ใช้สามารถเลือกใช้หรือไม่ใช้ สัญญาณลอจิกที่ได้จากการกดสวิตช์ SW1 นี้ได้โดยอิสระ



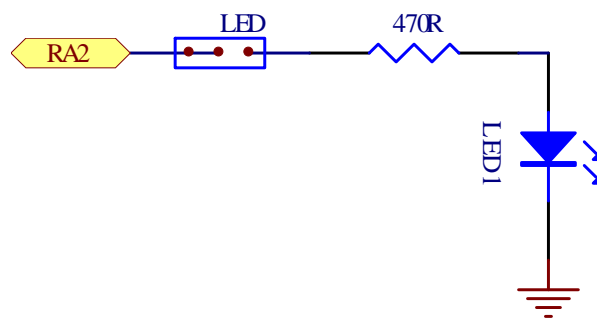
เนื่องจาก Pin RA1 ของ PIC18F46K22 สามารถโปรแกรมหน้าที่การทำงานได้หลายหน้าที่ ในกรณีที่ต้องการใช้เป็น SW ต้องทำการกำหนดค่าบิต ANSA1 ของรีจิสเตอร์ ANSELA ให้มีค่าเป็น "0" ก่อนเสมอ แล้วจึงสั่ง Initial การทำงานของ RA1 ในส่วนอื่น ๆ ตามต้องการ ไม่เช่นนั้นอาจจะไม่สามารถใช้งานขา RA1 เป็น Digital Input ได้ ดังตัวอย่าง

```
#define sw1          PORTAbits.RA1
...
void main(void)
{
    ...
    //Initial RA1 = Digital Input(SW1)
    ANSELAbits.ANSA1 = 0; // RA1 = Digital Function
    TRISAbits.TRISA1 = 1; // RA1 = Digital Input
    ...
    if(sw1 != 1)
    {
        ...
        // SW1 Press(0)
    }
    else
    {
        ...
        // SW1 Release(1)
    }
    ...
}
```

แสดง ตัวอย่าง Code (PIC-C18 Compiler) สำหรับ Initial SW1

การใช้งานวงจร LED

LED เป็นวงจรแสดงผลทางโลจิก โดยใช้สำหรับแสดงผลทางโลจิกเพื่อให้ผู้ใช้ได้รับรู้ โดยใช้กับสัญญาณ Output แบบ Logic โดยถ้าได้รับโลจิก "1" จะทำให้ LED ติดสว่าง และ ถ้าได้รับโลจิก "0" จะทำให้ LED ดับ โดยสัญญาณโลจิกที่จะใช้ขับเคลื่อนการแสดงผลของ LED ในวงจรนี้ จะถูกเชื่อมต่อมาจากขาสัญญาณ RA2 ของ MCU โดยมี Jumper ตัดต่อสัญญาณเพื่อให้ผู้ใช้สามารถเลือกใช้หรือไม่ใช้สัญญาณโลจิกที่ได้จากขาสัญญาณ RA2 มาขับ LED นี้ได้โดยอิสระ



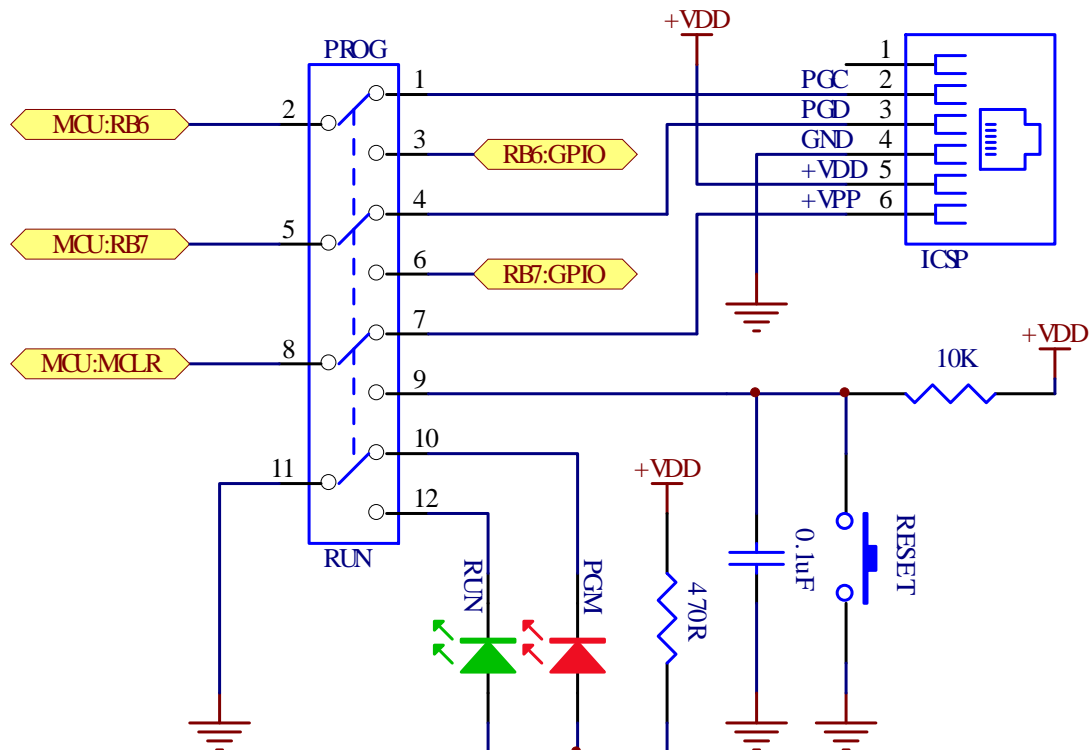
เนื่องจาก Pin RA2 ของ PIC18F46K22 สามารถโปรแกรมหน้าที่การทำงานได้หลายหน้าที่ ในกรณีที่ต้องการใช้เป็น LED ต้องทำการกำหนดค่าบิต ANSA2 ของรีจิสเตอร์ ANSELA ให้มีค่าเป็น "0" ก่อนเสมอ แล้วจึงสั่ง Initial การทำงานของ RA2 ในส่วนอื่นๆตามต้องการ ไม่เช่นนั้นอาจจะไม่สามารถใช้งานขา RA2 เป็น Digital Output ได้ ดังตัวอย่าง

```
...
#define LED1          LATAbits.LATA2
#define LED1_On()     LED1 = 1;
#define LED1_Off()    LED1 = 0;
#define LED1_Toggle() LED1 = !LED1;
...
void main(void)
{
    ...
    //Initial RA2 = Output(LED)
    ANSELAbits.ANSA2 = 0; // RA2 = Digital Function
    TRISAbits.TRISA2 = 0; // RA2 = Digital Output
    ...
    LED1_On();           // ON LED
    LED1_Off();          // OFF LED
    LED1_Toggle();       // Toggle LED
    ...
}
```

แสดง ตัวอย่าง Code (PIC-C18 Compiler) สำหรับ Initial LED

การใช้งาน ICSP

ICSP จะเป็น Connector แบบ RJ11 สำหรับ Interface กับเครื่องมือพัฒนาโปรแกรมตระกูล PIC ที่มีการจัดขั้วตามมาตรฐาน ICSP ของ Microchips เช่น ICD2, ICD3, Pickit2 หรือ Pickit3 ซึ่งสามารถใช้งานร่วมกับเครื่องมือพัฒนาของ Microchips หรือ เทียบเท่า เช่น ET-PGMPIC USB(เทียบเท่า Pickit2) หรือ ET-PGMPIC-PK3(เทียบเท่า Pickit3) หรือ ET-ICDX(เทียบเท่า ICD2) ส่วนการจะเลือกใช้เครื่องมือรุ่นใดนั้นให้พิจารณาจากเบอร์ MCU ที่จะใช้งานเป็นหลักว่า เบอร์ MCU ที่จะใช้งานนั้น มีเครื่องโปรแกรมรุ่นใดรองรับบ้าง เพราะความสามารถของเครื่องโปรแกรมที่ยกตัวอย่างให้ทราบข้างต้นนั้น รองรับการใช้งานกับ MCU ได้จำนวนเบอร์มากน้อยไม่เท่ากัน เพียงแต่มีการจัดเรียงขาสัญญาณในการโปรแกรม MCU เป็นมาตรฐานแบบเดียวกันเท่านั้น โดยวงจรของบอร์ดในส่วนนี้จะมีสวิตช์สำหรับเลือกติดต่อสัญญาณของ RB6, RB7 และ MCLR สำหรับใช้ทำหน้าที่เชื่อมต่อกับ Programmer/Debugger หรือ ใช้งานตามปกติได้พร้อม LED แสดงสถานะ ว่าการทำงานของสวิตช์อยู่ในตำแหน่งใด โดยถ้าเลือกสวิตช์ไว้ทางด้าน Programmer/Debugger จะเห็น LED สีแดงของ PGM ติดสว่าง แต่ถ้าตำแหน่งของสวิตช์อยู่ด้าน Run จะเห็น LED สีเขียว(RUN) ติดสว่าง โดยมีการจัดเรียงขาสัญญาณตามมาตรฐานของ ICSP ดังนี้



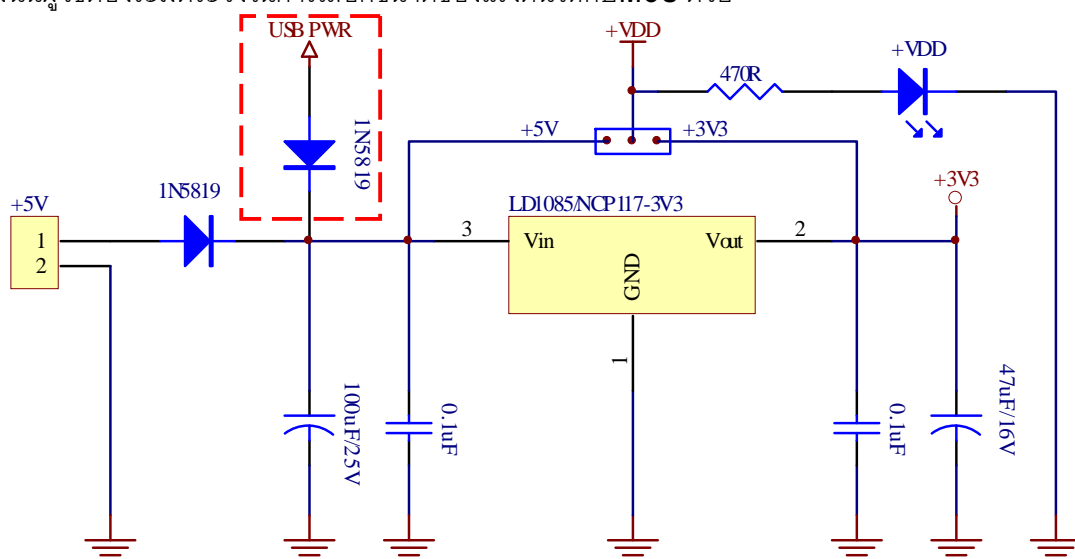
รูปแสดง โครงสร้างวงจรส่วนการเชื่อมต่อ ICSP ของบอร์ด ET-BASE PIC40/46K22(ICSP)

วงจรภาคจ่ายไฟ

สำหรับวงจรภาคจ่ายไฟของบอร์ด จะใช้ได้กับแรงดันภายนอกขนาด +5VDC โดยภายในบอร์ดจะมี วงจร Regulate ขนาด 3.3V/1A พร้อม Jumper สำหรับเลือกใช้แรงดัน +5VDC หรือ +3V3 ให้เป็น แรงดันไฟเลี้ยงของ MCU (+VDD) และวงจร I/O ภายในบอร์ด



สำหรับการเลือกระดับแรงดัน +VDD ให้กับ MCU ภายในบอร์ดนั้น สามารถทำได้โดยการเลือก กำหนดจาก Jumper 3V3/+5V ซึ่งการจะเลือกใช้ +VDD เป็นเท่าใดนั้น ให้พิจารณาจากจุดประสงค์การใช้งานและอุปกรณ์การเชื่อมต่อที่ต้องการ ซึ่ง MCU เบอร์ PIC18F46K22 สามารถทำงานได้ดีในย่านแรงดัน 1.8V ถึง 5.5V ซึ่งบนบอร์ดจะสามารถเลือกระบบแรงดัน ได้ 2 ย่าน คือ +5V และ +3.3V ซึ่งผู้ใช้ต้องระลึกไว้เสมอว่า ถ้าเลือกใช้แรงดันของแหล่งจ่ายให้บอร์ดในย่านใด ระดับสัญญาณลอจิกในการเชื่อมต่อกับอุปกรณ์ภายนอกต้องอยู่ในระดับที่มีค่าไม่เกินแหล่งจ่ายด้วย เช่น ถ้าเลือก แต่ที่จะใช้แหล่งจ่ายเป็น 3.3V สัญญาณที่จะเชื่อมต่อก็คือต้องเป็น 3.3V ด้วย ถ้านำสัญญาณ 5V มาเชื่อมต่ออาจทำให้ MCU เกิดความเสียหายได้ ดังนั้นผู้ใช้ต้องระมัดระวังในการเลือกขนาดของแรงดันให้กับ MCU ด้วย

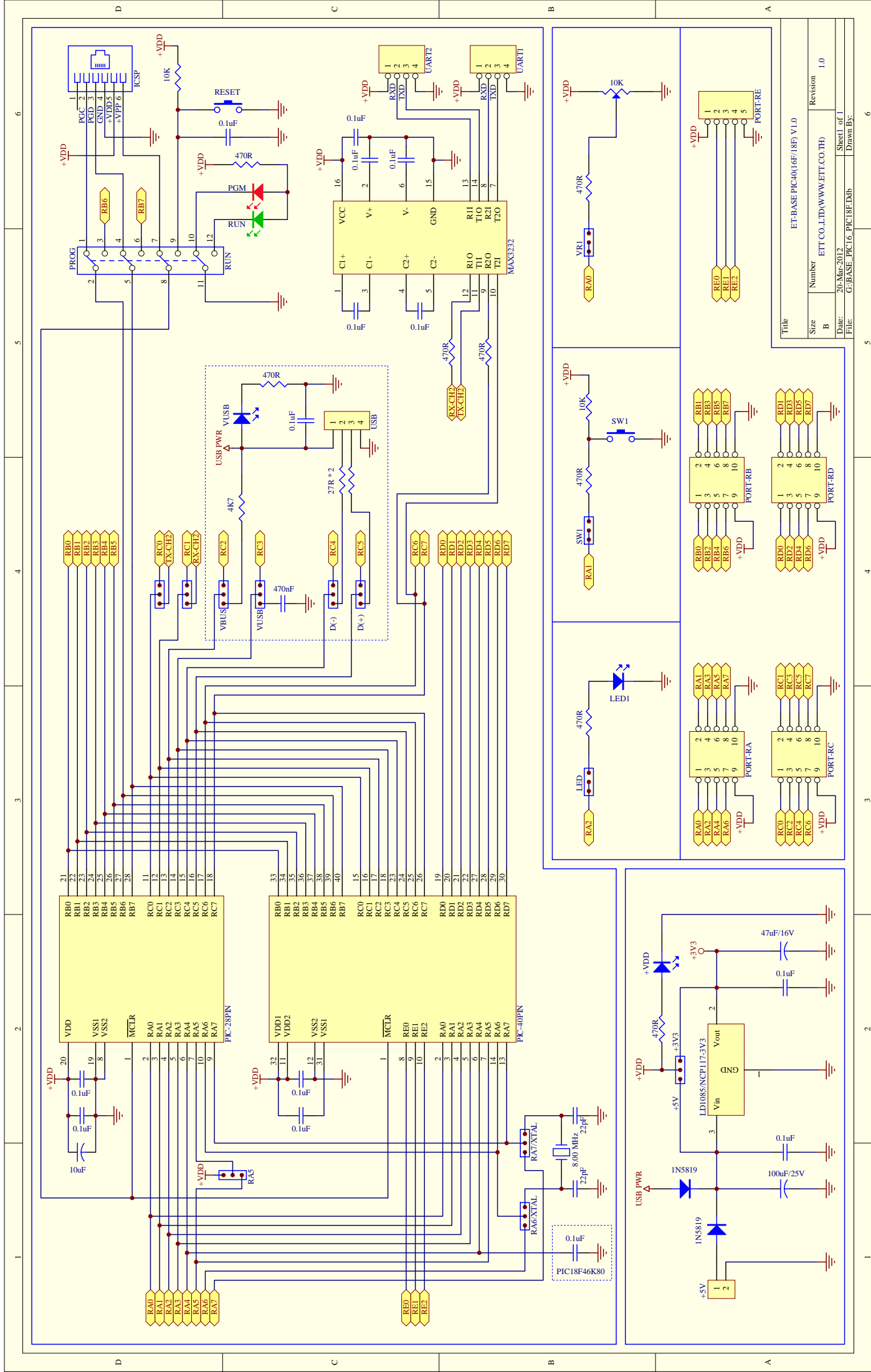


การกำหนดค่า Configuration สำหรับบอร์ด ET-BASE PIC40/46K22(ICSP)

```
//ETT:ET-BASE PIC40/46K22(PIC18F46K22)
#pragma config FOSC      = INTIO7       // Internal OSC ,CLKOUT on OSC2
#pragma config PLLCFG    = ON           // Oscillator multiplied by 4
#pragma config PRICLK    = ON           // Primary clock is always enabled
#pragma config FCEN      = ON           // Fail-Safe Clock Monitor enabled
#pragma config IESO      = ON           // Oscillator Switchover mode enabled
#pragma config PWRTEN    = ON           // Power up timer enabled
#pragma config BOREN     = SBORDIS      // Brown-out Reset enabled
#pragma config BORV      = 285          // VBOR set to 2.85 V nominal
#pragma config WDTE      = OFF          // Watch dog timer is always disabled.
#pragma config WDTPS     = 32768        // 1:32768
#pragma config CCP2MX    = PORTB3       // CCP2 I/O is multiplexed with RB3
#pragma config PBAEN     = OFF          // PORTB<5:0> = digital I/O on Reset
#pragma config CCP3MX    = PORTB5       // P3A/CCP3 I/O is multiplexed with RB5
#pragma config HFOFST    = OFF          // HFINTOSC output and ready status
#pragma config T3CMX     = PORTB5       // T3CKI is on RB5
#pragma config P2BMX     = PORTD2       // P2B is on RD2
#pragma config MCLRE     = EXTMCLR      // MCLR pin enabled
#pragma config LVP       = ON           // Single-Supply ICSP enabled
#pragma config XINST     = OFF          // Instruction set extension
#pragma config DEBUG     = ON           // ICSP Debug Enabled

//Disable All Protect
#pragma config CP0       = OFF          // Block0(000800-003FFFh) not protected
#pragma config CP1       = OFF          // Block1(004000-007FFFh) not protected
#pragma config CP2       = OFF          // Block2(008000-00BFFFh) not protected
#pragma config CP3       = OFF          // Block3(00C000-00FFFFh) not protected
#pragma config CPB       = OFF          // Boot block(00-0007FFh) not protected
#pragma config CPD       = OFF          // Data EEPROM not protected
#pragma config WRT0      = OFF          // Block0(000800-003FFFh) not protected
#pragma config WRT1      = OFF          // Block1(004000-007FFFh) not protected
#pragma config WRT2      = OFF          // Block2(008000-00BFFFh) not protected
#pragma config WRT3      = OFF          // Block3(00C000-00FFFFh) not protected
#pragma config WRTB      = OFF          // Boot Block(00-0007FFh) not protected
#pragma config WRTC      = OFF          // reg.(300000-3000FFh) not protected
#pragma config WRTD      = OFF          // Data EEPROM not write-protected
#pragma config EBTR0     = OFF          // Block0(000800-003FFFh) not protected
#pragma config EBTR1     = OFF          // Block1(004000-007FFFh) not protected
#pragma config EBTR2     = OFF          // Block2(008000-00BFFFh) not protected
#pragma config EBTR3     = OFF          // Block3(00C000-00FFFFh) not protected
#pragma config EBTRB     = OFF          // Boot Block(00-0007FFh) not protected
...
...
void main (void)
{
    //Config PIC18F46K22 Oscillator : Run 64MHz from Internal 16MHz + PLL(4)
    OSCCONbits.IRCF = 7;           // Internal RC = HFINTOSC - (16 MHz)
    OSCCONbits.OSTS = 0;           // Oscillator Start-up Time-out
                                   // Device is running from the internal
                                   // oscillator (HFINTOSC, MFINTOSC or
                                   // LFINTOSC)
    OSCCONbits.HFIOFS = 1;         // HFINTOSC frequency is stable
    OSCCONbits.SCS = 0;            // Primary clock(FOSC<3:0> in CONFIG1H)
    OSTUNEbits.PLEN = 1;          // x4 PLL enabled = 64MHz
    ...
}
```

ตัวอย่างการกำหนดค่า Configuration ให้กับ PIC18F46K22 ด้วย PIC-C18



Title				ET-BASE PIC400(6F/18F) V1.0			
Size				Number			
B				ETT CO., LTD./WWW.ETT.CO.TH			
Date:				20-Mar-2012			
File:				G:\BASE-PIC16-PIC18F.Ddb			
Revision				1.0			