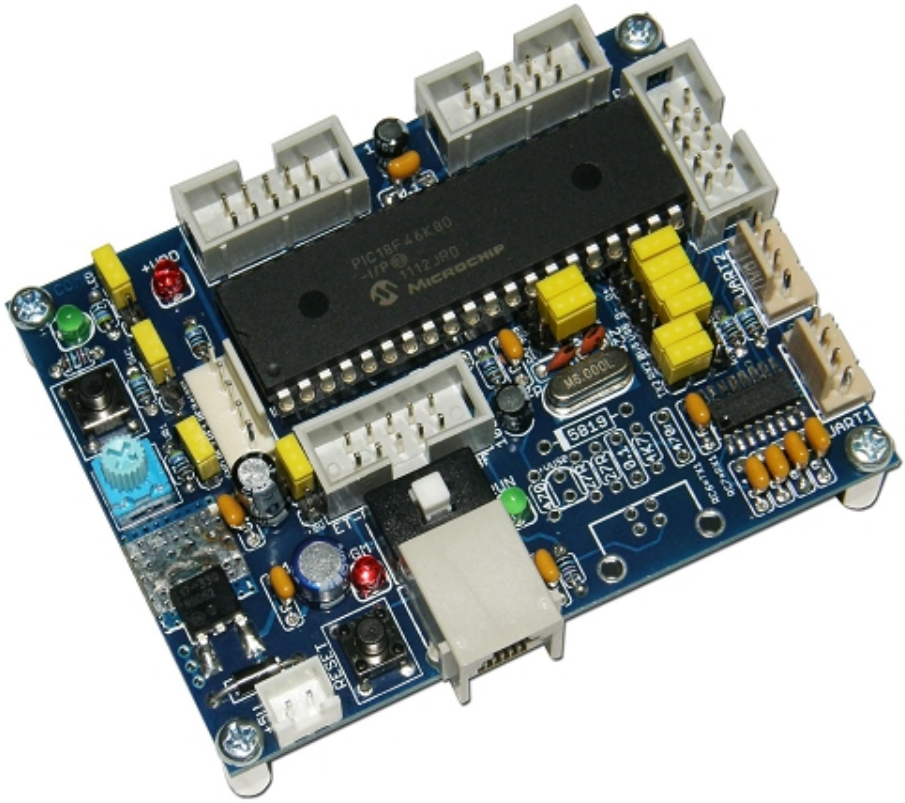
ET-BASE PIC40/46K80(ICSP)



ET-BASE PIC40/46K80(ICSP) is Board Microcontroller in the series of PIC that is designed to use and install Chip Microcontroller 40PIN(40PDIP) No.PIC18F46K80. It is compatible with Power Supply 3.3V or 5V; so, user should choose the most suitable type according to the objectives.

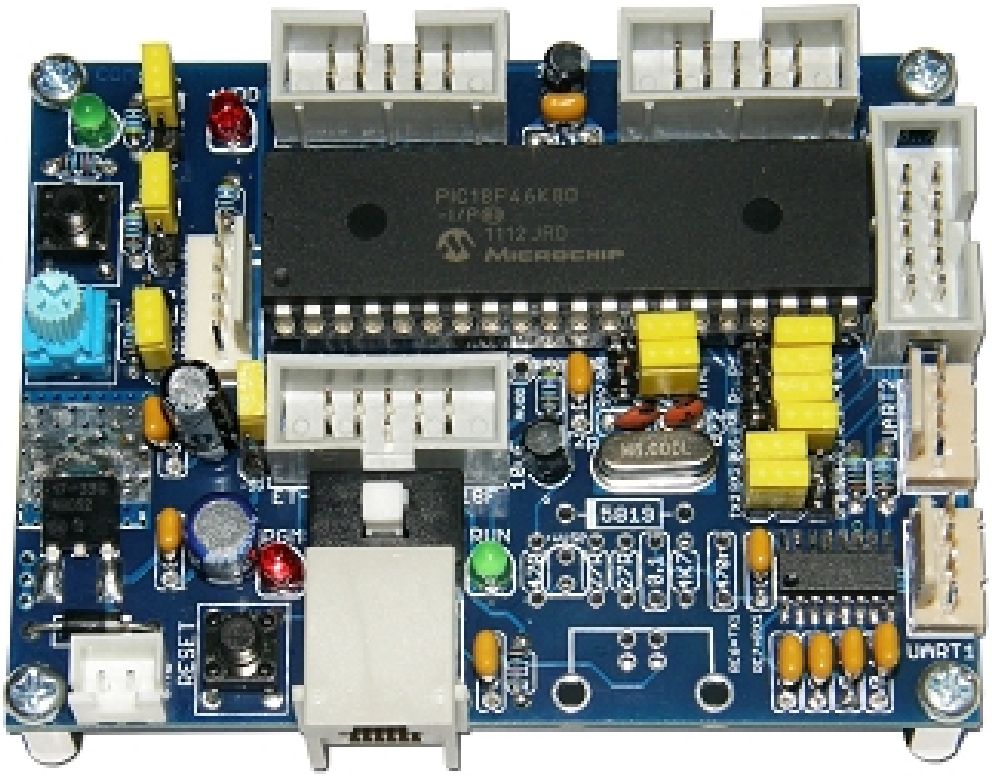
Structure of **Board ET-BASE PIC40/46K80(ICSP)** is designed to be mini board that is suitable for various applications and basic Training Kit. Internal board provides circuits that are necessary and convenient to use and develop program. It is flexible because it can adjust and modify Signal I/O for applications easily according to the preferable objectives suitably.

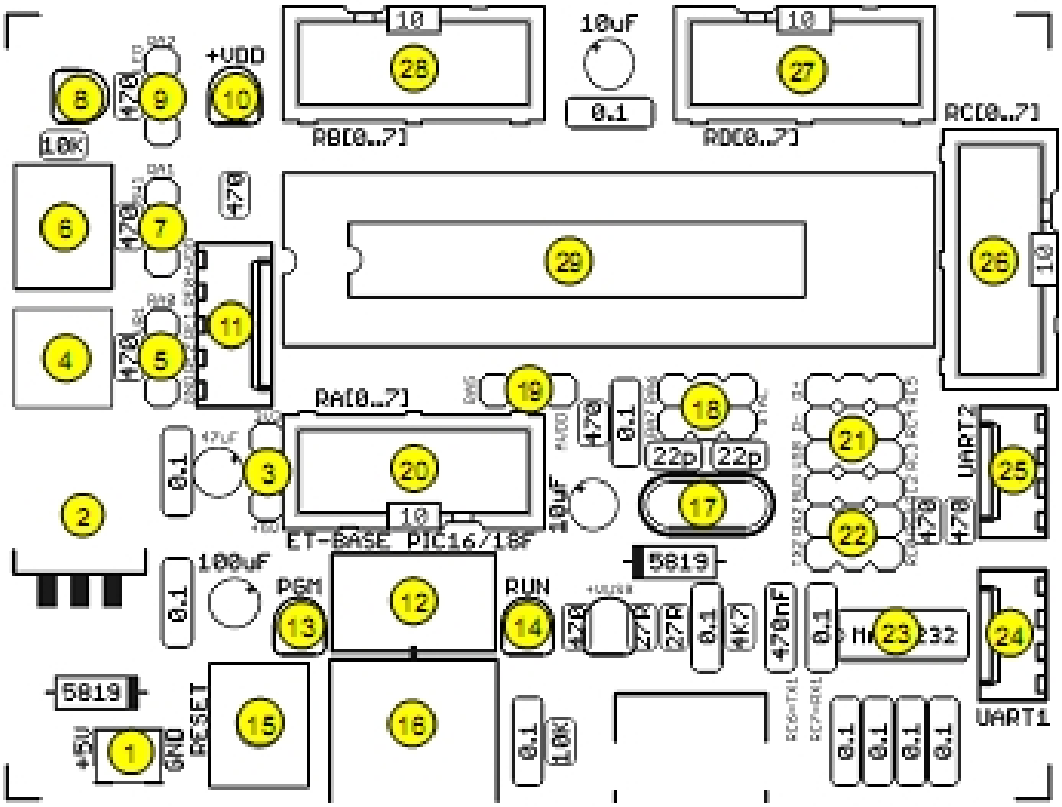
Moreover, it adds more devices for initially testing Input/Output internal board; so, user can use these devices to be tool while testing and developing the program. For example, it uses Adjustable VR to test the operation of ADC; or, it uses Push Button Switch to test Input Logic; or, it uses LED to test the operation of Output logic.

Specifications of Board ET-ET-BASE PIC40/46K80(ICSP)

- Use MCU No.PIC18F46K80(40PDIP) on board; Run by Frequency 64MHz at the maximum
  - 64KByte(32KWord) Flash/ 1024 Byte EEPROM/ 3896 Byte SRAM
  - 35 GPIO
  - 11-Channel 12Bit ADC
  - 2 Comparator
  - 2-Channel 8 Bit Timer/ 3-Channel 16 Bit Timer
  - 2-Channel EUART
  - 2-Channel I2C/ 2-Channel SPI
  - 1-Channel ECCP/ 4-Channel CCP
- Has Crystal 8.00MHz and Jumper to connect/disconnect signal
- Has 2-Channel Circuit Line Driver for UART RS232 Serial Port Communication; it uses Connector UART CPA-4Pin according to the ETT Standard
  - 1-Channel for Hardware UART1; it uses Pin RC6(TX1) and Pin RC7(RX1) according to the PIC Standard
  - 1-Channel for Software UART; it uses Pin RC0(TX2) and Pin RC1(RX2) with Jumper to choose the operation modes either to be UART or GPIO as required.
  - 1-Channel for Hardware UART2 in the format of TTL Level; it is used for general applications independently. For example, it interfaces with UART in the format of TTL Level or it interfaces with Line Driver to be RS422/485.
- Has Connector ICSP RJ11 according to the ICD2 standard; it is used with Programmer and Debugger that supports the operation according to the ICSP standard of Microchips such as ICD2/ICD3 or Pickit2/Pickit3.
- Has Switch to alternate signals between Program/Debug(PGM) and Normal Run(RUN); moreover, there is LED to display operation modes of board.
- Has 4 of Header I/O 2x5 and 1 of Header CPA-5Pin
- Has Switch RESET to reset the operation of MCU internal board
- Has Adjustable VR to test the operation of ADC Input; it uses RA0 with Jumper to connect/disconnect signal.
- Has Switch to test the operation of Digital Input; it uses RA1 with Jumper to connect/disconnect signal.
- Has LED to test the operation of Digital Output; it uses RA2 with Jumper to connect/disconnect signal.
- Has Power +5VDC Input with 3.3V/1V Regulate and LED to display the operating status of Power Supply; moreover, there is Jumper to choose Power Supply for MCU either to be +5VDC or 3.3VDC
- Mini PCB Size: 8 x 6 cm.

Structure of Board ET-BASE PIC40/46K80(ICSP)





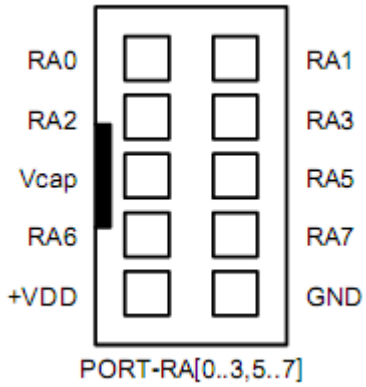
Picture shows structure of Board ET-BASE PIC40/46K80(ICSP)

- **No.1:** It is Connector +5VDC Power Supply of Board.
- **No.2:** It is IC Regulate 3.3VDC/1A.
- **No.3:** It is Jumper to choose either to be 3.3V or 5V that is voltage level of Power Supply for MCU(+VDD).
- **No.4:** It is Adjustable VR(VR1) to test the operation of Input Analog (ADC).
- **No.5:** It is Jumper to connect/disconnect Signal RA0 and adjusted voltage from VR1.
- **No.6:** It is Push Button Switch (SW1) to test the operation of Digital Input.
- **No.7:** It is Jumper to connect/disconnect Signal RA1 and Digital Input from SW1.
- **No.8:** It is LED to test the operation of Digital Output.
- **No.9:** It is Jumper to connect/disconnect Signal RA2 and Digital Output for LED.
- **No.10:** It is LED to display the status of Power Supply +VDD.
- **No.11:** It is Connector RE[0..2] that is used in case of installing MCU 40Pin only.
- **No.12:** It is Switch to choose the operation modes either to be Run Mode(RUN) and Program Mode(PGM).
- **No.13:** It is red LED to display status of PGM when the board is running in Program Mode.
- **No.14:** It is green LED to display status of RUN when the board is running in Run Mode.
- **No.15:** It is Switch RESET to reset the operation of MCU when it is running in Run Mode.
- **No.16:** It is Connector ICSP to interface with Programmer and Debugger, according to the ICD2 standard.
- **No.17:** It is Crystal 8.00MHz.
- **No.18:** It is Jumper to choose Pin RA6,RA7 of MCU either to interface to be GPIO at Connector RA[0…3,5…7] or Circuit Crystal.
- **No.19:** It is Jumper to choose functions of Pin7 of MCU 28Pin either to be Signal RA5 of MCU by interfacing to be GPIO at Connector RA[0..7] or to be Pin +AVDD. In case of PIC18F46K280, it always sets Jumper at the position of RA5.
- **No.20:** It is Connector IDE10Pin of RA[0…3,5…7].
- **No.21:** It is Jumper to choose Pin RC2,RC3,RC4,RC5 of MCU either to interface to be GPIO at Connector RC[0..7] or to interface with USB Bus(VUSB,VBUS,D-,D+). In case of PIC18F46K80, all 4 Jumpers always are set to be GPIO.
- **No.22:** It is Jumper to choose Pin RC0,RC1 of MCU either to interface to be GPIO at Connector RC[0..7] or TX2,RX2 of UART2 (Software UART).

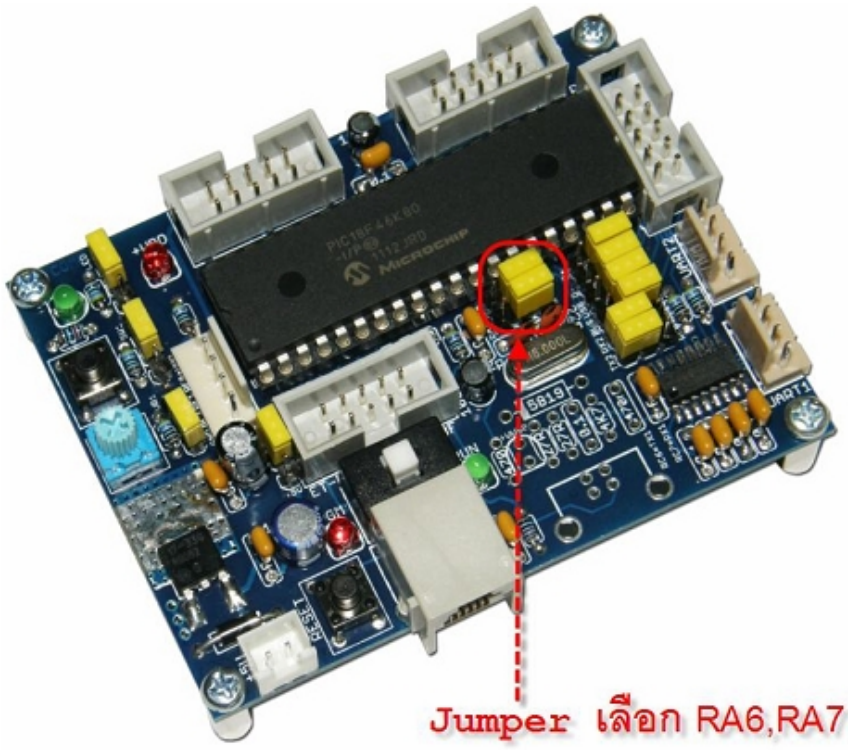
- **No.23:** It is IC Line Driver of RS232(MAX3232/ICL3232) to convert Signal Level between signal of UART Logic and standard Signal RS232 Level of UART1,UART2.
- **No.24:** It is Connector UART1 that is Signal RS232. It supports Hardware UART1 that uses Pin RC6(TX1)and RC7(RX1) for connection.
- **No.25:** It is Connector UART2 that is Signal RS232. It supports Software UART that uses Pin RC0(TX2) and RC1(RX2) for connection.
- **No.26:** It is Connector IDE10Pin of RC[0..7].
- **No.27:** It is Connector IDE10Pin of RD[0..7].
- **No.28:** It is Connector IDE10Pin of RB[0..7].
- **No.29:** It is MCU No.PIC18F46K80 on board.

CONNECTORS

PORT RA[0..7]

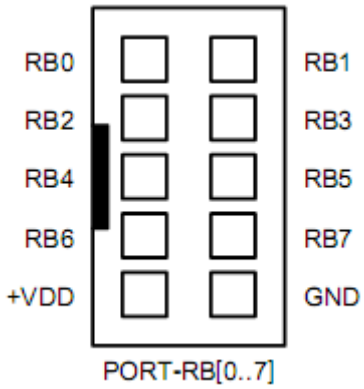


PORT-RA[0…3, 5…7]: It is Signal from RA0…RA3, RA5…RA7 of MCU. Signal RA0…RA5 is directly interfaced from Pin of MCU but Signal RA6 and RA7 has Jumper to connect/disconnect signal; so, user can choose either to interface Signal RA6,RA7 to be GPIO or to interface with Circuit Crystal Generator. In case of PIC18F46K80, user can use Signal Clock from internal RC and then user can use Signal RA6,RA7 to be GPIO as normal.

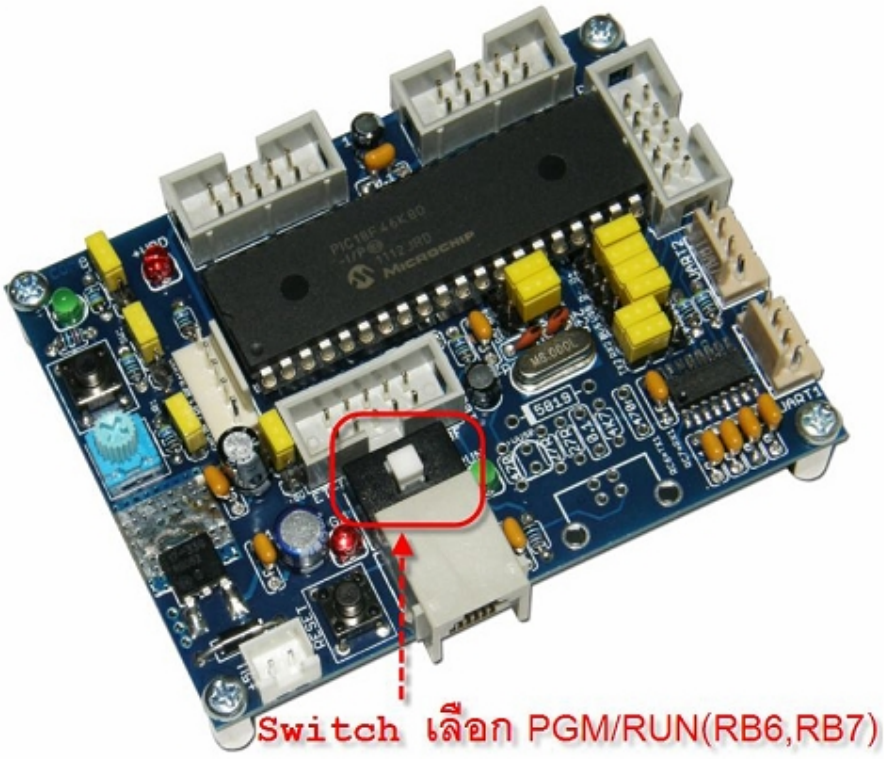


*****NOTE***** PIC18F46K80 has not Pin RA4; so, this pin position becomes Pin V_{DDCORE}/V_{CAP} instead.

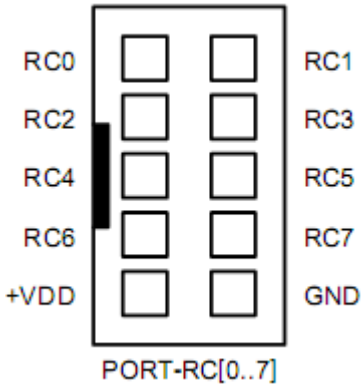
PORT RB[0…7]



PORT-RB[0…7]: It is Signal from RB0…RB7 of MCU. Signal RB0…RB5 is directly interfaced from Pin of MCU. Signal RB6 and RB7 has Switch to connect/disconnect signal; in this case, user can choose either to use Signal RB6,RB7 to be GPIO or to connect with ICSP Programmer through Connector ICSP(RJ11). If it shifts the Switch to position of RUN, it interfaces Pin B6,RB7 to this Connector; on the other hand, if it shifts the Switch to position of PGM, it interfaces Signal RB6,RB7 of MCU with Programmer through Connector ICSP(RJ11) instead.



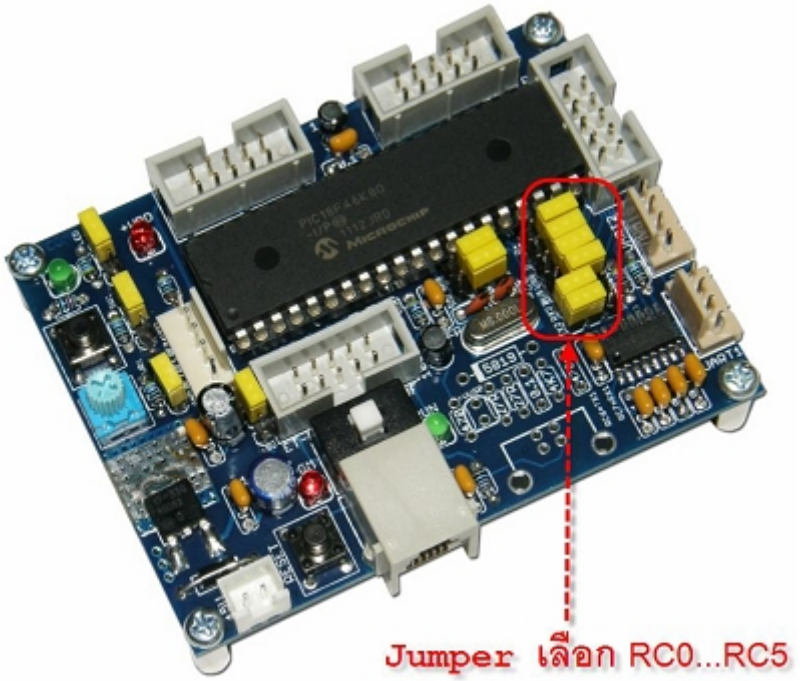
PORT-RC[0...7]



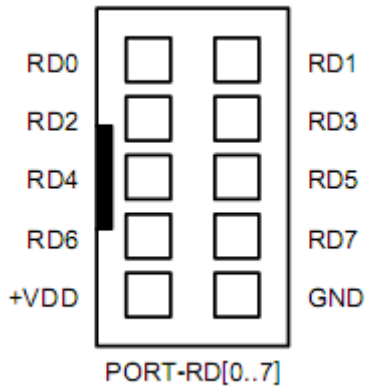
PORT-RC[0…7]: It is Signal from RC0…RC7 of MCU. Signal RC0…RC3 has been set by Jumper first while Signal RC6 and RC7 are directly interfaced from Pin of MCU.

RC0,RC1: It has Jumper to choose Pin RC0 and RC1 either to interface RC0,RC1 to this Connector or to interface RC0,RC1 to Circuit Line Driver of UART2(Software UART).

RC2…RC5: There is Jumper to choose Pin RC2…RC5 either to interface to this Connector or to interface to Circuit USB. In case of PIC18F46K80, there is no circuit internal MCU for connecting with USB; so, user has to set Jumper of RC2,RC3, RC4,RC5 to the side of GPIO to always interface Pin RC2,RC3, RC4,RC5 to this Connector as shown in picture below;

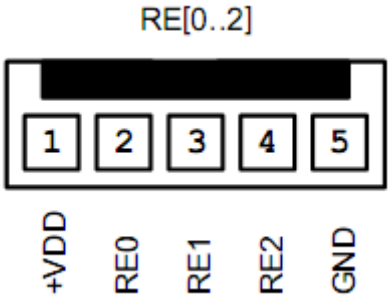


PORT RD[0…7]



PORT RD[0..7]: It is Signal from RD0…RD7 of MCU. All 8 Signals are directly interfaced from Pin of MCU.

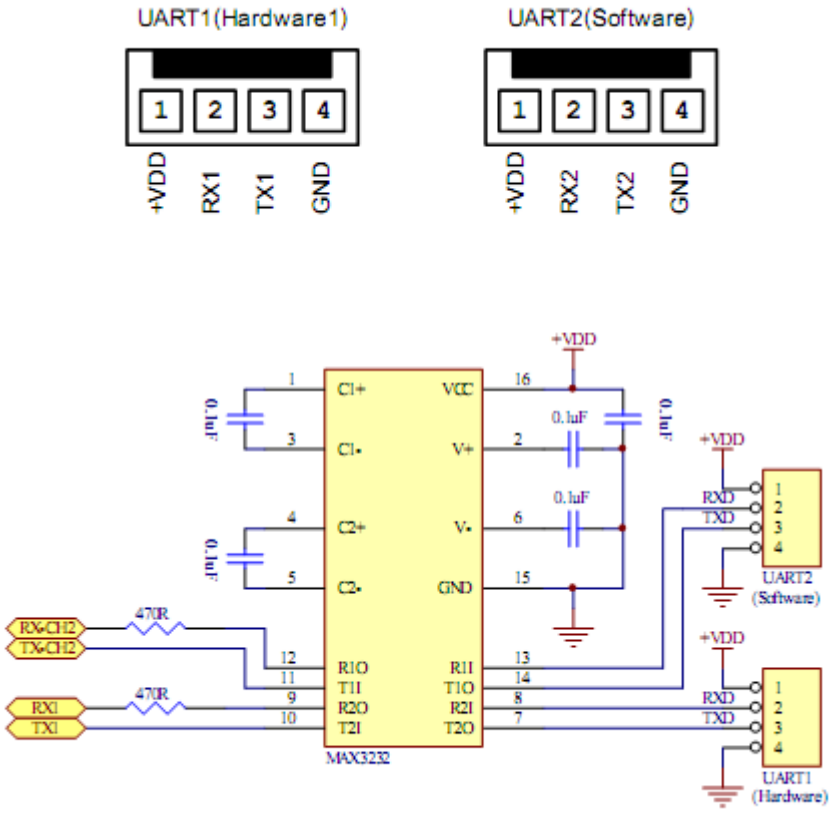
PORT-RE[0..2]



PORT-RE[0..2]: It is Signal from RE0…RE2 of MCU. All 3 Signals are directly interfaced from Pin of MCU.

RS232

PORT RS232: It is Signal RS232 that has converted Signal Level by MAX3232 completely. In case of PIC18F46K80, there are 2 channels of Hardware UART; it uses Pin RC6(TX1),RC7(RX1) and RD6(TX2),RD7(RX2) for connection. However, Circuit of board is designed to use only 1 channel of Hardware UART; it is interfaced with Circuit RS232 Line Driver that is UART1(Hardware UART1). Another one channel of Hardware UART(Hardware UART2:RD6/RD7) is floated; so, it is independent to use this signal for various applications as required. For example, it is used to be RS422/RS485 or it is used to be UART Communication with other devices directly in the format of TTL Level. Moreover, it adds Circuit Line Driver for UART2(Software UART) into the circuit of board. The second channel uses RC0(TX2) and RC1(RX2) with Jumper to connect/disconnect Signal UART2 as required. Signal of RS232 in each channel arranges Connector to be CPA-4PIN(RS232) as shown in the picture below;



It uses Cable RS232 to connect Signal RS232 between ComPort of computer PC and Connector RS232 of Board ET-BASE PIC40/46K80(ICSP) as follows;

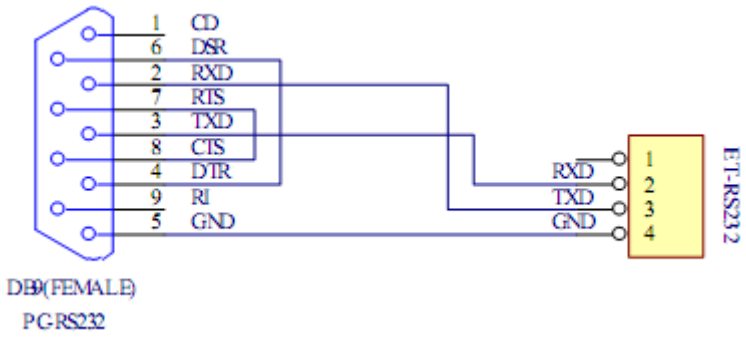
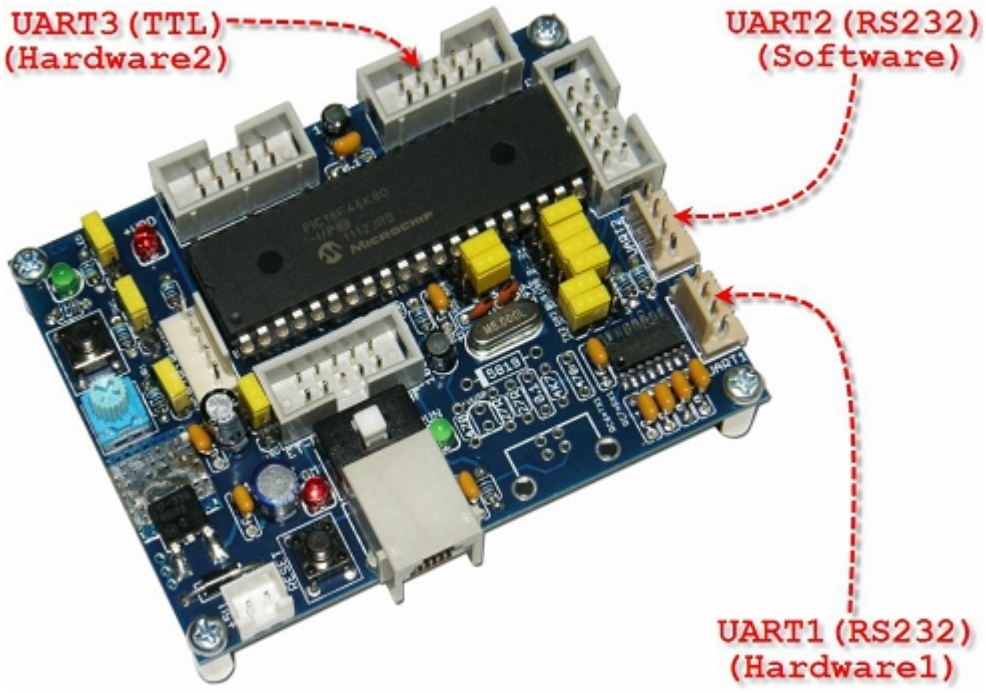


Figure shows circuit of CABLE RS232.



NOTE: When using UART2(Software UART) to write program, user should consider the capability of Compiler to see if it can support the Library that is Software UART. If using PIC CSS Compiler to compile the language, it can use the capability of Software UART easily. If user does not require using Pin RC0/RC1 to be UART2, user can set the function of both pins to be GPIO by setting Jumper as required. Because Circuit Driver in the part of UART2 is designed to be flexible structure; moreover, there is Jumper RC0/TX2 and RC1/RX2 to connect/disconnect signals as required according to the preferable objectives.

```
/*************************************************/
/* Demo Program For ET-BASE PIC40/46K80 UART Demo */
/* MCU Control  : PIC18F46K80                     */
/*              : Run 32MHz(X-TAL 8.00 MHz +PLL)  */
/* +VDD Power   : +5V/+3V3 Operate Voltage        */
/* Function     : Demo UART1,UART2 Echo Test      */
/*************************************************/
#include <18F46K80.h>
#include <stdlib.h>

// Fuses: PIC18F46K80(CCS Compiler)
// Fuses: LP,XT,HS,RC,INTRC_IO,ECL,ECM,ECH,NOWDT,WDT_SW,WDT_NOSL,WDT
// Fuses: PUT,NOPUT,NOMCLR,MCLR,PROTECT,NOPROTECT,CPD,NOCPD,NOBROWNOUT
// Fuses: BROWNOUT_SW,BROWNOUT_NOSL,BROWNOUT,CLKOUT,NOCLKOUT,NOIESO
// Fuses: IESO,NOFCMEN,FCMEN,WRT,WRT_EECON1000,WRT_EECON200,NOWRT
// Fuses: VCAP_A0,VCAP_A5,VCAP_A6,NOVCAP,PLL_SW,PLL,NOSTVREN,STVREN
// Fuses: BORV25,BORV19,DEBUG,NODEBUG,NOLVP,LVP
#fuses   HS,PLL,NOWDT,NOPROTECT,NOLVP            // X-TAL 8MHz + PLL(x4)
#use     delay(clock=32000000)                   // 32.00 MHz

/* 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)
```

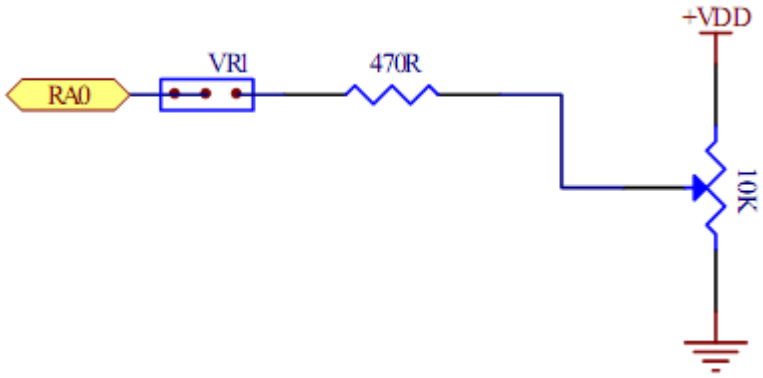
```
/****************/
/* Main Program */
/****************/
void main()
{
  char rx_buff;

  //Start-Up UART1
  fprintf(CH1,"\n\n\n\r");
  fprintf(CH1,"Demo UART1 ET-BASE PIC40/46K80(ICSP)\n\r");
  fprintf(CH1,"Run 32.00 MHz(Internal 8 MHz + PLL)\n\r");
  fprintf(CH1,"UART1>");

  //Start-Up UART2
  fprintf(CH2,"\n\n\n\r");
  fprintf(CH2,"Demo UART2 ET-BASE PIC40/46K80(ICSP)\n\r");
  fprintf(CH2,"Run 32.00 MHz(Internal 8 MHz + PLL)\n\r");
  fprintf(CH2,"UART2>");

  while(true)
  {
    //Verify & Echo UART1
    if(kbhit(CH1))
    {
      rx_buff = fgetc(CH1);
      if (rx_buff == 0x0D)
      {
        fprintf(CH1,"\n\r");
        fprintf(CH1,"ET-BASE PIC40/46K80(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 PIC40/46K80(ICSP)\n\r");
        fprintf(CH2,"UART2>");
      }
      else
      {
        fputc(rx_buff,CH2);                // Echo Received Characters
      }
    }

  }

}
```

This example shows Code to communicate with C Language UART (PICC CCS Compiler).

Adjustable VR

This VR1 is Adjustable Resistor to adjust voltage level between +VDD and GND to create the voltage level for testing the operation of Analog Input (ADC). The adjustable voltage of VR1 is connected to Pin RA0 of MCU and there is Jumper to connect/disconnect Signal; so, user can set this Jumper for using this adjustable VR1 to test the operation independently.



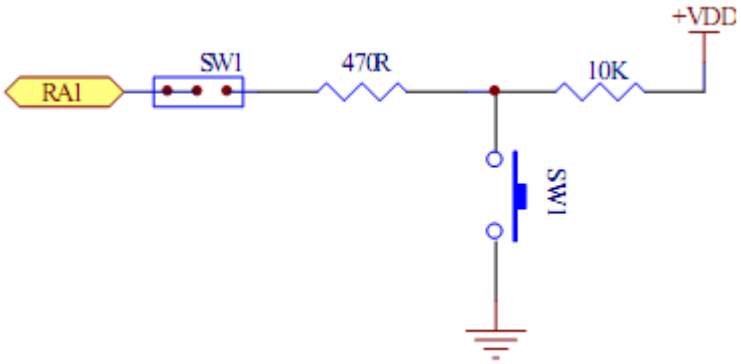
In this case, user can program Pin RA0 of PIC18F46K80 to run many functions. If user requires programming it to be function ADC(AN0), it always sets Bit ANSEL0 of Register ANCON0 to be "1" first and then set Initial the operation of ADC in other parts as required, otherwise user cannot use Pin RA0 to be ADC as shown in the example below;

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

  //Initial RA0 = ADC(AN0)
  ANCON0bits.ANSEL0=1;  // RA0 = Analog Function
  TRISAbits.TRISA0=1;   // RA0 = Analog Input
  ADCON2bits.ADFM=0;    // Result Format 0 = Left justified
  ADCON2bits.ACQT=7;    // Acquisition time 7 = 20TAD
  ADCON2bits.ADCS=6;    // Clock conversion bits 6= FOSC/64
  ADCON1bits.VCFG0=0;   // Vref+ = AVdd
  ADCON1bits.VCFG1=0;   // Vref+ = AVdd
  ADCON1bits.VNCFG=0;   // Vref- = AVss
  ADCON1bits.CHSN=0;    // Select ADC Negative Channel = AVSS
  ADCON0bits.CHS=0;     // ADC Channel Input = AN0
  ADCON0bits.ADON=1;    // Turn on ADC
  …
  ADCON0bits.GO = 1;                // Start Conversion
  while(ADCON0bits.NOT_DONE == 1);  // wait for it to complete
  Result = ADRESH;                  // return high byte of result
}
```

This example shows Code (PIC-C18 Compiler) for Initial ADC.

Circuit SW1

This SW1 is Circuit Push-Button Switch to create Signal LOGIC "0" and "1" to test the operation of LOGIC INPUT. For example, it is used to detect the value of pressing Switch. If the Switch is released, it is Logic "1"; on the other hand, if the Switch is pressed, it is Logic "0". The Signal Logic from this circuit is connected to Pin RA1 of MCU; moreover, there is Jumper to connect/disconnect Signal Logic by pressing this Switch SW1 independently.



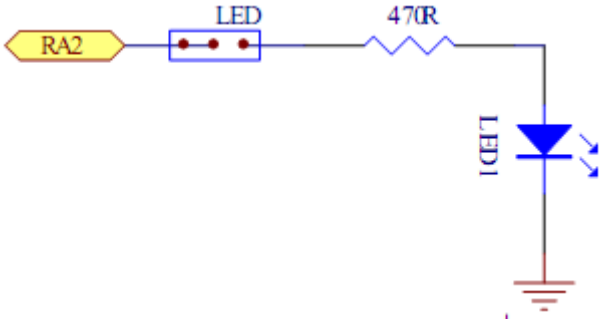
In this case, user can program Pin RA1 of PIC18F46K80 to run many functions. If user requires using it to be SW, it always sets Bit ANSEL1 of Register ANCCON0 to be "0" first and then set Initial the operation of RA1 in other parts as required, otherwise user cannot use Pin RA1 to be Digital Input as shown in the example below;

```
#define sw1                PORTAbits.RA1
…
void main(void)
{
  …
  //Initial RA1 = Digital Input(SW1)
  ANCON0bits.ANSEL1=0;   // RA1 = Digital Function
  TRISAbits.TRISA1=1;    // RA1 = Digital Input
  …
  if(sw1 != 1)
  {
    …                    // SW1 Press(0)
  }
  else
  {
    …                    // SW1 Release(1)
  }
  …
}
```

This example shows Code (PIC-C18 Compiler) for Initial SW1.

Circuit LED

LED is Circuit LOGIC Display that displays the logic result to user. It is used with Signal Logic Output; if it receives the Signal Logic "1", it makes LED ON; on the other hand, it receives the Signal Logic "0", it makes LED OFF. The Signal Logic that is used to drive the LED Display in this circuit is connected from Pin RA2 of MCU; moreover, there is Jumper to connect/disconnect Signal Logic from Pin RA2 to drive this LED independently.



In this case, user can program Pin RA2 of PIC18F46K80 to run many functions. If user requires using it to be LED, it always sets Bit ANSEL2 of Register ANCON0 to be "0" first and then set Initial the operation of RA2 in other parts as required, otherwise user cannot use Pin RA2 to be Digital Output as shown in the example below;

```
…
#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)
  ANCON0bits.ANSEL2=0;   // RA2 = Digital Function
  TRISAbits.TRISA2=0;    // RA2 = Digital Output
  …
  LED1_On();             // ON LED
  LED1_Off();            // OFF LED
  LED1_Toggle();         // Toggle LED
  …
}
```

This example shows Code (PIC-C18 Compiler) for Initial LED.

ICSP

ICSP is Connector RJ11 to interface with Program Developer Kit in the series of PIC and it arranges the connector according the ICSP standard of MICROCHIPS such as ICD2, ICD3, Pickit2 or Pickit3. It is compatible with Program Developer kit of MICROCHIPS or equivalence such as ET-PGMPIC USB (be equivalent to PICKit2) or ET-PGMPIC PK3 (be equivalent to PICKit3), or ET-ICDX (be equivalent to ICD2). When user requires using any device version, user should consider the MCU number mainly because each MCU number supports different programmer device and it has different capability as mentioned above; however, it only arranges pin for programming MCU in the same standard. The circuit of board in this part has Switch to connect/disconnect Signal of RB6, RB7, and MCLR for interfacing with either Programmer/Debugger or Normal Run; moreover, there is LED to display the current position of LED that is running. If it shifts the Switch to the side of Programmer/Debugger, the red LED of PGM is in the status ON; on the other hand, if it shifts the Switch to the side of Normal Run, the green LED of RUN is in the status of ON. It arranges pins according to the ICSP standard as follows;

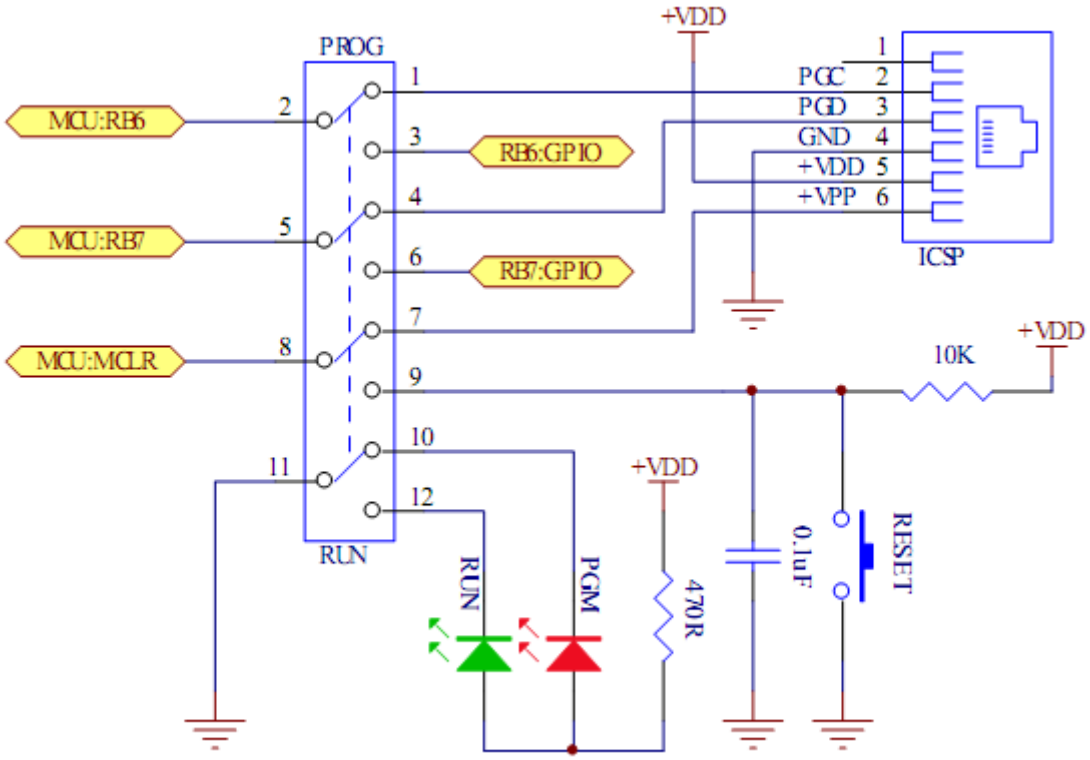
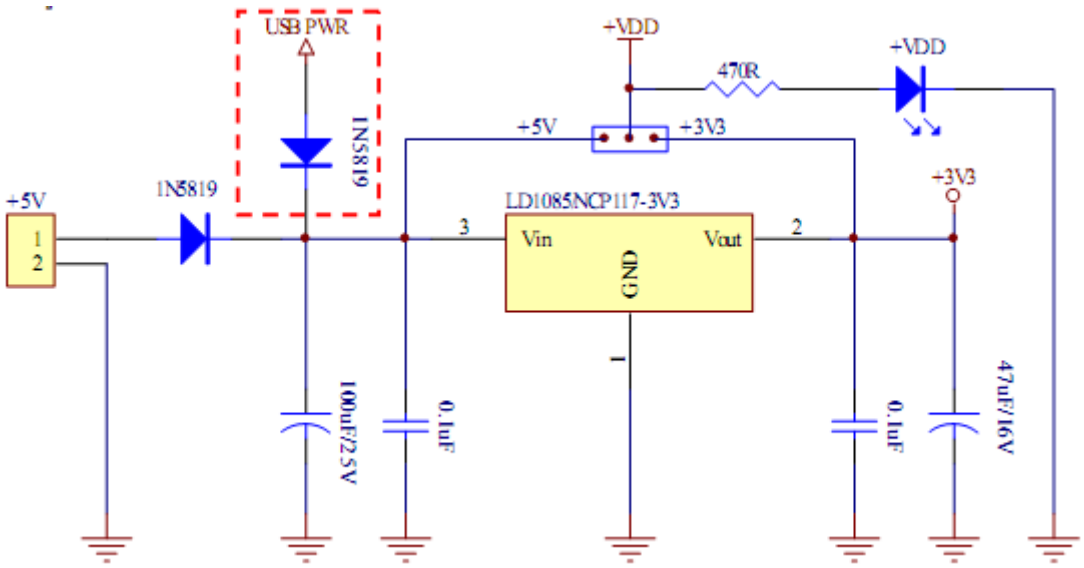


Figure shows structure of circuit in the part of connecting ICSP of Board ET-BASE PIC40/46K80(ICSP).

POWER SUPPLY

The Power Supply of this Board is compatible with external +5VDC. There is a Circuit Regulate 3.3V/1A internal board with Jumper to choose the voltage types either to be +5VDC or +3V3 that is voltage for MCU(+VDD) and Circuit I/O internal board.

It uses Jumper 3V3/+5V to choose the voltage level +VDD for MCU internal board. When user requires choosing and setting any +VDD, user should consider the objectives and the connected device; in this case, MCU No.PIC18F46K80 can operate well with the voltage level in the range of 1.8V to 5.5V. There are 2 voltage ranges on board that can be chosen; +5V and +3.3V. User always remembers that when user has chosen any voltage level of Power Supply for board, the Signal Logic for interfacing with external device does not exceed the level from Power Supply. For example, if using 3.3V Power Supply, the connected Signal must be 3.3V as well; on the other hand, if connecting with 5V Power Supply, it makes the MCU damaged. So, user must be careful to choose and setup the voltage level for MCU.


USB PWR
1N5819
+VDD
470R
+VDD
+5V
+3V3
+5V
1N5819
+3V3
LD1085NCP117-3V3
3
Vin
Vout
2
GND
1
100uF/25V
0.1uF
0.1uF
47uF/16V

How to setup Configuration for Board ET-BASE PIC40/46K80(ICSP)

```
//ETT:ET-BASE PIC40/46K80(PIC18F46K80)
#pragma config RETEN    = ON        // Ultra low-power regulator is Enabled
#pragma config INTOSCSEL = LOW      // LF-INTOSC in Low-power mode
#pragma config SOSCSEL  = LOW       // Low Power SOSC circuit selected
#pragma config XINST    = OFF       // Disabled Extended Instruction Set
#pragma config FOSC     = INTIO1    // Internal RC,CLKOUT function on OSC2
#pragma config PLLCFG   = ON        // Oscillator multiplied by 4
#pragma config FCMEN    = 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 in hardware
#pragma config BORV     = 0         // VBOR set to 3.0V V nominal
#pragma config BORPWR   = LOW       // BORMV set to low power level
#pragma config WDTEN    = OFF       // Watch dog timer is always disabled.
#pragma config WDTPS    = 1048576   // 1:1048576
#pragma config CANMX    = PORTB     // ECAN TX,RX are located on RB2,RB3
#pragma config MSSPMSK  = MSK7      // 7 Bit address masking mode
#pragma config MCIRE    = ON        // MCLR Enabled, RG5 Disabled
#pragma config STVREN   = ON        // Enabled Stack Overflow Reset
#pragma config BBSIZ    = BB2K      // 2K word Boot Block size

//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       //Config(300000-3000FFh) not protected
#pragma config WRTD     = OFF       //Data EEPROM not 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 PIC18F46K80 Oscillator : Run 64MHz from Internal 16MHz + PLL(4)
  OSCCONbits.IRCF=7;     // HF-INTOSC output frequency is used (16 MHz)
  OSCCONbits.OSTS=0;     // Oscillator Start-up Timer(OST)
  OSCCONbits.HFIOFS=1;   // HF-INTOSC oscillator frequency is stable
  OSCCONbits.SCS=0;      // HF-INTOSC with PLL
  OSCTUNEbits.PLLEN=1;   // x4 PLL enabled = 64MHz

  ...
}
```

This is example of setup Configuration for PIC18F46K80 by PIC-C18.