```
#######################-TIMERS-########################

TxCONbits.TCKPS = ?;        // K scaler
PRx = ?;
TMRx = 0;
TxCONbits.TON = 1;

while (IFS?bits.TxIF == 0);   // polling
IFS?bits.TxIF = 0;

IPC?bits.TxIP = 2;          // Interrupt
IEC?bits.TxIE = 1;
IFS?bits.TxIF = 0;

OC1CONbits.OCM = 6;         // PWM
OC1CONbits.OCTSEL = ?;
OC1RS = ?;
OC1CONbits.ON = 1;

EnableInterrupts();
```

```
int freq = 1 + (ADC1BUF0 * 4) / 1023;
int delayMs = 1000 / freq;

#######################-DELAY-########################

void delay(unsigned int ms) {
  resetCoreTimer();
  while (readCoreTimer() < 20000 * ms);
}
```

```
#######################-DISPLAY-########################

// RB8-RB14 (segments) & RD5-RD6 (displays) as outputs
TRISB &= 0x80FF;
TRISD &= 0xFF9F;

unsigned char toBcd(unsigned char value) {
  return ((value / 10) << 4) + (value % 10);
}

void send2displays(unsigned char value) {
  static const char disp7Scodes[16] = {0x3F, 0x06, 0x58, 0x4F, 0x66,
0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
  static char displayFlag = 0;

  int digit_low = toBcd(value) & 0x0F;
  int digit_high = toBcd(value) >> 4;

  if (displayFlag == 0) {
    LATD = (LATD & 0xFF9F) | 0x0020;
    LATB = (LATB & 0x80FF) | (disp7Scodes[digit_low] << 8);
  } else {
    LATD = (LATD & 0xFF9F) | 0x0040;
    LATB = (LATB & 0x80FF) | (disp7Scodes[digit_high] << 8);
  }
  displayFlag = !displayFlag;

}
```

```
#######################-ADC-########################

  TRISBbits.TRISB4 = 1;
  AD1PCFGbits.PCFG4= 0;
  AD1CON1bits.SSRC = 7;
  AD1CON1bits.CLRASAM = 1;
  AD1CON3bits.SAMC = 16;
  AD1CON2bits.SMPI = N-1;
  AD1CHSbits.CH0SA = 4;
  AD1CON1bits.ON = 1;

  IPC6bits.AD1IP = 1;           // Interrupt
  IFS1bits.AD1IF = 0;
  IEC1bits.AD1IE = 1;

  EnableInterrupts();

  ### while

  AD1CON1bits.ASAM = 1;

  while( IFS1bits.AD1IF == 0 );     // polling
  IFS1bits.AD1IF = 0;

  int *p = (int *)(&ADC1BUF0);
  int media = 0;
  for(; p <= (int *)(&ADC1BUFF); p++=4 ) {
    media += *p;
  }
  media /= SAMPLES;

  V = (media * 33 + 511) / 1023;



void _int_(VECTOR) isr_adc(void) {   // VECTOR number page 74-76
  (...)
  IFS1bits.AD1IF = 0;
}
```

```
#######################-UART-########################

void configureUART2(void) {
  // Configure UART2:
  // 1 - Configure BaudRate Generator
  // BRGH: High Baud Rate Enable bit
  // 1 = High - Speed mode – 4x baud clock enabled
  // 0 = Standard Speed mode – 16x baud clock enabled
  U2MODEbits.BRGH = 0;

  U2BRG = ((PBCLK + 8 * 115200) / (16 * 115200)) - 1;

  // 2 – Configure number of data bits, parity and number of stop bits
  // (see U2MODE register)
  U2MODEbits.PDSEL = 0;
  // PDSEL<1 :0> : Parity and Data Selection bits
  // 11 = 9 - bit data, no parity
  // 10 = 8 - bit data, odd parity
  // 01 = 8 - bit data, even parity
  // 00 = 8 - bit data, no parity
  U2MODEbits.STSEL = 0;
  // STSEL:
  // Stop Selection bit
  // 1 = 2 Stop bits
  // 0 = 1 Stop bit

  // 3 – Enable the transmitter and receiver modules (see register U2STA)
  U2STAbits.URXEN = 1;
  U2STAbits.UTXEN = 1;

  // 4 – Enable UART2 (see register U2MODE)
  U2MODEbits.ON = 1;
}
```

```
#####-UART receiver Interrupt
  IEC1bits.U2RXIE = 1;
  IEC1bits.U2TXIE = 0;
  IPC8bits.U2IP = 2;
  U2STAbits.URXISEL = 0;

  IFS1bits.U2RXIF = 0;

  EnableInterrupts();

#######  -UART transmitter Interrupt
  IEC1bits.U2TXIE = 0;
  IEC1bits.U2RXIE = 0;
  IPC8bits.U2IP = 2;
  U2STAbits.UTXISEL = 0;

  IFS1bits.U2TXIF = 0;

  EnableInterrupts();

#######  -UART transmitter polling
void putc (char byte) {
  while (U2STAbits.UTXBF == 1);

  U2TXREG = byte;
}

void putstr (char *str) {
  while(*str){
    putc(*str++);
  }
}

#######  -UART receiver polling

char getc(char byte) {
  while (U2STAbits.URXDA == 0);

  return U2RXREG;
}
```