Electronic Design

By RICARDO JIMENEZ, Imperial Valley College, Imperial, Calif., and GABRIEL LEE **ÁLVAREZ**, Instituto Tecnológico de Mexicali, B.C. México.

Microcontroller Sends Voltage and Frequency via Low-Cost Modules

In this Idea for Design, voltage and frequency can be sent wirelessly using PIC microcontrollers.

This project pairs a set of 8-bit microcontrollers with a 433-MHz industrial, scientific and medical (ISM) band transmitter/receiver. The FS1000A RF transmitter has a range of up to 200 m; the XY-MK-5V RF receiver operates at 5 V and uses only 4 mA *(Fig. 1)*. These modules are readily available and used in projects with platforms like Arduinos.

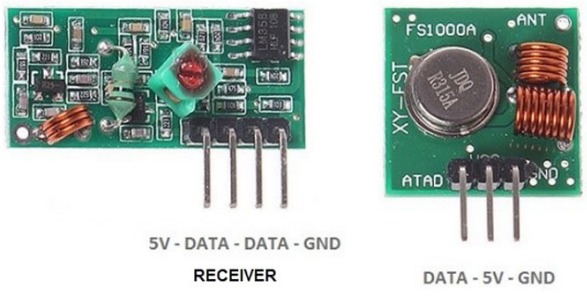
The projects uses Microchip PIC microcontrollers including the 16F1619 and 16F1614. These utilize the radio modules employing the on-chip EUSART (Enhanced Universal Synchronous/Asynchronous Receiver Transmitter) interfaces. On the transmitter side, we use the PIC's analog and digital interfaces to read a voltage and a frequency source within the range of 0 to 4.99 V dc and 0 to 65.5 kHz, respectively.

The logic diagram for the transmitter *(Fig. 2)* includes an LCD display to provide feedback. *Listing 1* has the code for the 16F1619.

The receiver side *(Fig. 3)* also includes a pair of LCD displays. The displays aren't needed in an application, but they're handy for debugging. *Listing 2* has the code for the 16F1614.

Transmission in ASCII Code for Letter "A"

Start Bit	LSB Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	MSB Bit 7	Stop Bit
0	1	0	0	0	0	0	1	0	1



1. The FS1000A RF transmitter (right) has a range of up to 200 m. The XY-MK-5V RF (left) receiver operates at 5 V and uses only 4 mA.

Serial Communication

First, we will touch on the serial communication support that's tied to the wireless modules. To establish communication, it's necessary to have a starting bit for a period of time to alert the receiver that a data package is about to be transmitted. This forces the receiver clock to start synchronization with a 0 bit. Then each bit is sent individually, starting with the LSB bit through the MSB bit *(see table)*. Each bit has the same period. Once all bits are transmitted, it must wait for the Stop bit to indicate end of transmission. This is achieved with a High logic, where the communication ends.

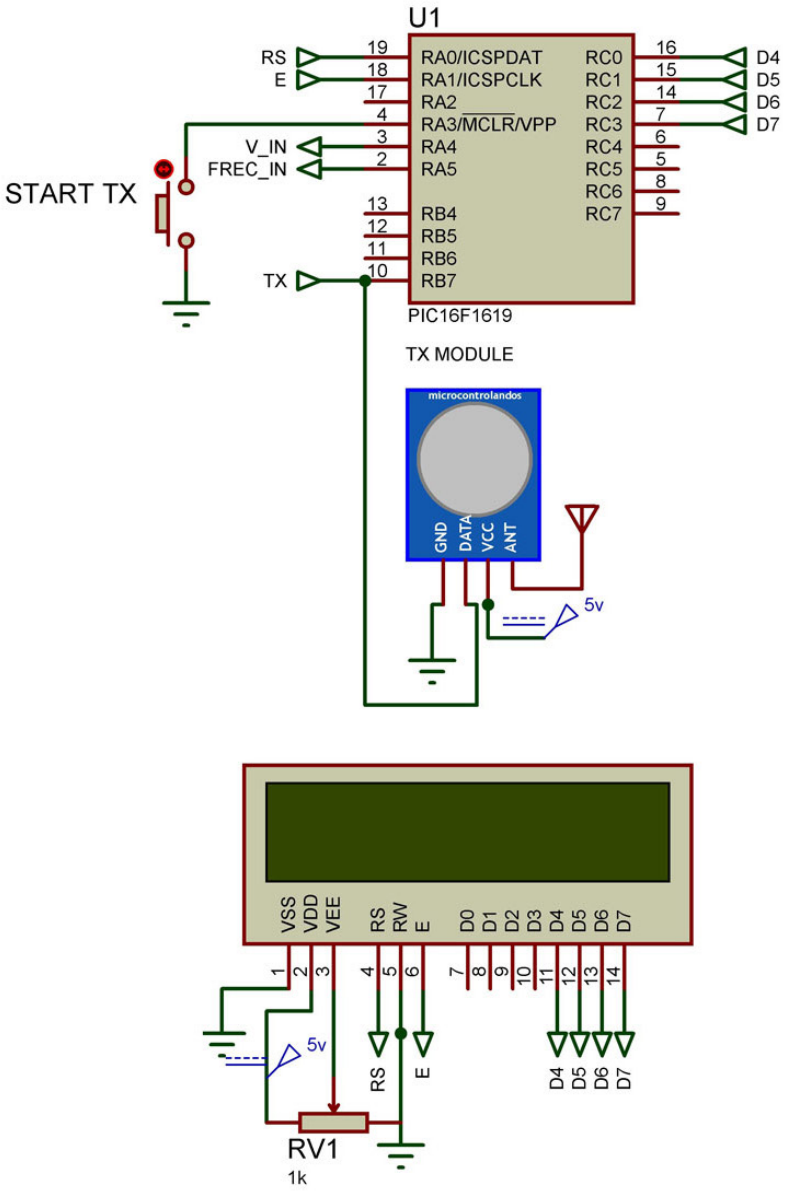
To transmit the alphanumeric character "A," whose ASCII code is 0b01000001 *(Fig. 4)*, the bits are organized as shown in the table. Each bit has a period determined by the transmission speed (baud rate), which can vary from 115.2 kb/s to 200 bits/s. The time for each bit is given by:

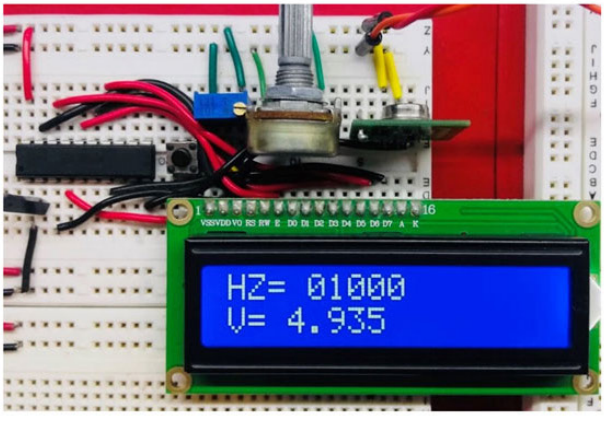
$$T = \frac{1}{Baud\ Rate}$$

To transmit a voltage, we will read it using the 10-bit analog-to-digital converter (ADC), which by default has its reference voltage connected to 5 V. This defines the ADC resolution:

$$V_{bit} = \frac{V_{ref+}*(bit\ read)}{2^{10}} = \frac{5v*1\ bit}{1024} = 4.882mV$$

To perform the binary to decimal conversion, we use this code:





2. The transmit module is tied to the PIC16F1619 microcontroller. The voltage and frequency inputs are not shown. Our transmitter prototype uses a potentiometer to adjust the voltage and frequency inputs.

VINBCD = VIN*4887 ; multiplying by RESlsb = 4.8887
VR = div32 1000 ; perform 16-bit division

where VIN is the binary voltage read by the ADC, and VR is the voltage result.

TIMER1 takes care of reading frequencies and is configured to read the number of pulses in one second. After that, the PIC transmits and receives those readings. With the data ready, the instruction HSEROUT sends the serial data. The code is always waiting to be activated by the user. Once activated, it sends a control variable to the receiving unit with the instruction HSEROUT, which is sent at a speed of 2400 bits/s. This variable indicates that data transmission is starting:

HSEROUT ["BZ0",10]; SEND ACTIVATION INSTRUCTION

For the transmitter code, it's necessary to read a voltage and frequency with the microcontroller's ADC and TIMER1, respectively. The maximum frequency is:

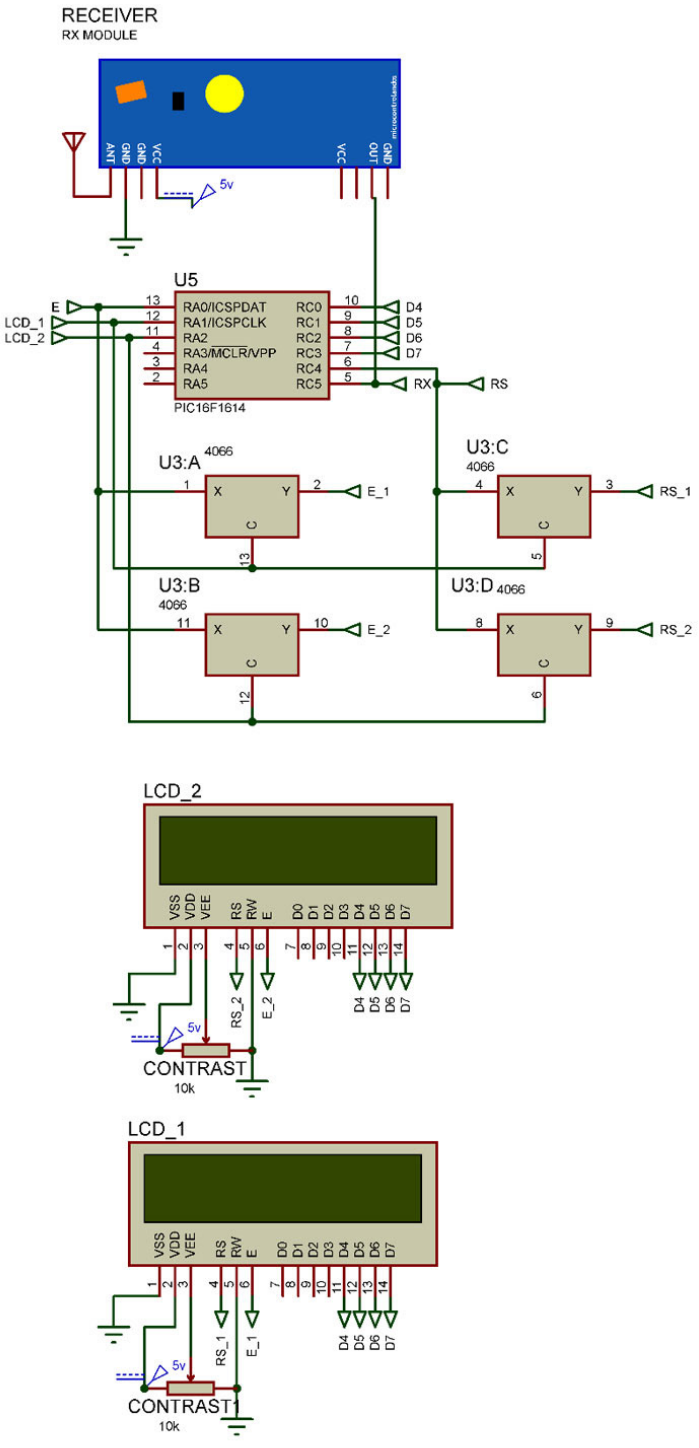
$$F_{max} = 2^{16} - 1 = 65535$$

Then the code proceeds to obtain each decimal digit with the command DIG, and each digit is sent at 2400 bits/s with the serial data with the instruction Hserout. Two digits in ASCII code are inserted at the beginning of each data package, so that the receive identifies which data is receiving.

HSEROUT ["BZ",DEC A,10];
HSEROUT ["AZ",VD[3],VD[2],VD[1],VD[0],10]; send serial data
HSEROUT ["CZ",H[4],H[3],H[2],H[1],H[0],10];

When the receiver gets the data "BZ," it represents the activity control in the serial port. The data package AZ represents the Voltage, while the data package CZ represents Frequency.

When the Micro receives the stop instruction



with the pushbutton, it transmits a "1" in the control data package, indicating end of transmission.

The receiver waits for the control variable BZ with the instructions HSERIN and WAIT, which waits specifically for the data "BZ". Because that RF data transmission has external noise, after receiving the data BZ, the next data is stored in a variable to manipulate as follows:

HSERIN 10,MAIN1,[WAIT("BZ"), STR A\1]; wait one second to receive instruction

When the second micro receives the control variable, it starts the data input where the instructions HSERIN wait for ASCII codes in each package that's transmitted. If it doesn't receive any data, the code jumps to the next instruction as follows:

HSERIN 10,JUMP1,[WAIT ("BZ"), STR A\1]
JUMP1: HSERIN 100,JUMP2,[WAIT ("AZ"), STR VD\4]
JUMP2: HSERIN 10,HERE, [WAIT ("CZ"), STR H\5]

When the data is received, it's transferred to two LCD displays. By using a CMOS switch HCF4066, it's possible to control the Enable and R/W functions in each LCD. Two bits in the microcontroller select which LCD will be operating as shown in the following code:

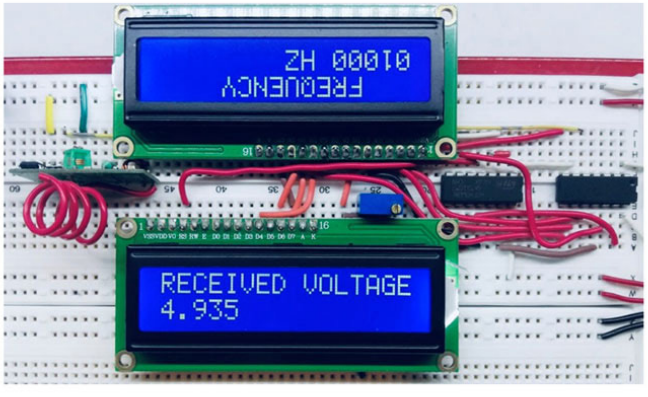
```
LCD1 = 1      ; ENABLE LCD1
LCD2 = 1      ; ENABLE LCD2
```

The receiver PIC microcontroller drives two 16X2 LCD displays controlled by a quad CMOS switch—HCF4066—configured as multiplexer to select which LCD will receive data. In this case, the first LCD shows Voltage, and the second one shows Frequency.

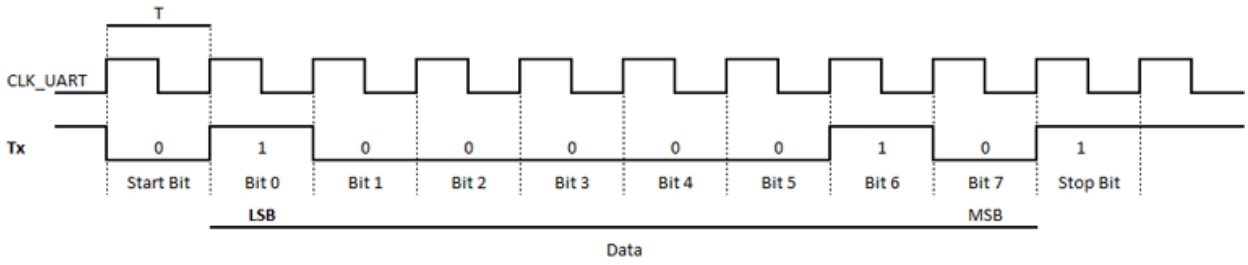
For the receiver, we use the instruction HSERIN, which receives two ASCII characters, and then saves the respective data of both characters.

Ricardo Jimenez, holds a Master's degree in Electronics Engineering. He is the author of several Lab Practices Notebooks on PIC Microcontrollers applied to Instrumentation.

Gabriel Lee Alvarez is pursuing his Electronics Engineering degree at Instituto Tecnologico de Mexicali (ITM).



3. The receiver is based on a PIC16F1614 that handles incoming messages to display on the LCD displays. For the receiver side, we have the voltage and frequency displayed on different LCD displays.



4. This timing diagram shows an ASCII character A (0b10000010) using the usual asynchronous process with a start bit and the trailing stop bit.

Listing 1: Code for the PIC16F1619 Used as a transmitter

```
; WIRELESS VOLTMETER/FREQ RF RS232 TRANSMITER
; PBP3 COMPILER from melabs.com
; Authors: Ricardo Jimenez and Gabriel Lee Alvarez
; © July 13, 2020
; PIC16F1619
; TRANSMITTER PIC16F1619
 #CONFIG
   __config _CONFIG1, _FOSC_INTOSC & _PWRTE_ON & _MCLRE_OFF & _CP_OFF & _BOREN_ON & _CLKOUTEN_OFF
   __config _CONFIG2, _WRT_OFF & _PPS1WAY_OFF & _ZCD_OFF & _PLLEN_OFF & _STVREN_ON & _BORV_LO & _LVP_OFF
   __config _CONFIG3, _WDTCPS_WDTCPS4 & _WDTE_ON & _WDTCWS_WDTCWS100 & _WDTCCS_LFINTOSC
 #ENDCONFIG
DEFINE OSC 16;
OSCCON = %01111010; intenal Osc set to 16 MHZ
OSCTUNE = 0
OSCSTAT = %00011111;
TRISA = %00111110; RA0 AS A OUTPUTS, RA1:RA2:RA3:RA4:RA5 AS A INPUTS
ANSELA = %000100; RA0:RA1:RA4:RA5 AS DIGITAL,RA2:RA3 AS ANALOG,
TRISC = %100000;RC0:RC1:RC2:RC3:RC4 AS OUTPUTS, RC5 AS INPUTS
TRISC=0; Clearing PORTC
TRISB = 0; Clearing PORTB
ANSELB = 0; PORTB set as digital
;PPSLOCK=0;
;ANSEL PULL-UP resistors disabled
WPUA = 0; AC = 0;
WPUC = 0;          PULL-UPS DISABLED
OPTION_REG.7 = 0; PULL-UPS ENABLED

WPUA.3 = 1; PULL-UP IN RA3 ENABLED
T1CON = %10000101; TMR1 ENABLED
ADCON0 = %00001111;AN3 IS ENABLED
ADCON1 = %10000000; FOSC/2, VDD
;---------------------------------
DEFINE HSER_RCSTA 90h; RX MODULE IS ENABLED
DEFINE HSER_TXSTA 20h; TX MODULE IS ENABLED
DEFINE HSER_BAUD 2400; BAUD RATE IS 2400
rc1sta.7 = 1; SERIAL COMUNICATION IS ENABLED
RB7PPS = %10010;

;--LCD CONFIGURATION ----------------------
DEFINE LCD_DREG PORTC ' PORTC is LCD data port
DEFINE LCD_DBIT 0 ' PORTC.0 is the data LSB
DEFINE LCD_RSREG PORTA ' RS is connected to PORTA.0
DEFINE LCD_RSBIT 0
DEFINE LCD_EREG PORTA ' E is connected to PORTA.1
DEFINE LCD_EBIT 1
DEFINE LCD_BITS 4 ' 4  data line
DEFINE LCD_LINES 2 '  2-line display
DEFINE LCD_COMMANDUS 1500 ' Use 1500uS command delay
DEFINE LCD_DATAUS 44 ' Use 44uS data delay
;;-----------------------------------------------------

LCDOUT $FE,$28;  $28 FUNCTION SET, 4 BITS
LCDOUT $FE,$10;  $10 SHIFT DISPLAY
LCDOUT $FE,$0C;  $0C DISPLAY ON
LCDOUT $FE,$06;  $06 ENTRY MODE SET
;---------------------------------------
;CREATING ALIAS
;TX VAR PORTC.0;;
PB VAR PORTA.3
;----
;---VARIABLES
X VAR byte[4];
Y VAR WORD;
VIN VAR WORD;
VINBCD var word
H VAR BYTE[5];
TMR VAR WORD;
VD var byte[4];
```

```
A VAR BIT
IN var byte;
OUT VAR BYTE;
;--------
A = 1
for x = 0 to 5
vd[x] = “0”;
h[x] = “0”;
next x
;send a message to the LCD
LCDOUT $FE,$80,”FREQUENCY METER-”;
LCDOUT $FE,$C0,”--VOLTMETER RF--”;
PAUSE 2000;
TMR = 0;
T1CON.0 =0; T1CON input Frequency DISABLED
MAIN

LCDOUT $FE,$80,”FREQUENCY METER-”;
LCDOUT $FE,$C0,”--VOLTMETER RF--”;
;---------
IF PB = 0 THEN; wait for Push button to go Low
   PAUSE 150
   A = 0;              CONTROL VARIABLE IS 0
   FOR X= 0 TO 3;
            HSEROUT [“BZ0”,10]; SEND ACTIVATION
INSTRUCTION
   NEXT X;
ENDIF;

IF A = 0 THEN; START QUESTION FOR CONTROL
variable
   LCDOUT $FE,$80,”   START      “;
   LCDOUT $FE,$C0,”    TX       “;
   PAUSE 1000;
   TMR1L = 0;CLEAR TIMER
   TMR1H = 0;
   T1CON.0 = 1;  ENABLES TIMER

   STAY:;
      T1CON.0=1;
      PAUSE 1000
      T1CON.0=0; TIMER DISABLED
      GOSUB ADC;
      GOSUB H_DEC;      GO TO h_DEC AND RETURN;
        GOSUB SHOW_LCD;  GO TO SHOW_LCD AND
RETURN
      GOSUB SEND;

      if pb = 0 then; wait for Push button to go high
               A = 1;      CONTROL variable is 1, END
TRANSMISSION
          PAUSE 200;
      ENDIF;

   IF A = 0 THEN STAY; If CONTROL IS “0” GO TO STAY
   A = 1;;                CONTROL VARIABLE IS “1”
   FOR X = 0 TO 2;
     GOSUB SEND
   NEXT X
   LCDOUT $FE,$80,”   END    “;
   LCDOUT $FE,$C0,”    TX    “;
   PAUSE 1000
ENDIF

goto main;
ADC:;
ADCON0.1 = 1;            ENABLE ADC MODULE
HERE1: IF ADCON0.1 = 1 THEN HERE1;  CONVER-
SION in progress
VIN.BYTE0 = ADRESL;   SAVE LOWER REGISTER OF
THE ADC IN VARIABLE VIN
VIN.BYTE1 = ADRESH;   SAVE HIGHER REGISTER OF
THE ADC IN VARIABLE VIN
disable;                 DISABLE INTERRUPTS
VINBCD = VIN*4887;       MULTIPLYING BY RESLSB
= 4.8887
VIN = div32 1000;        PERFORM 16-BIT DIVISION
enable;                  ENABLE INTERRUPTS
FOR X = 0 TO 3;          START LOOP
IN = VIN DIG X;          GET DIGIT X
LOOKUP IN,[“0123456789ABCDEFG”],OUT;   DIGITS
DECODING
VD[X] = OUT;             SAVE DECODED DIGITS
NEXT X;
return;

SHOW_LCD:;  display LABEL on LCD
    LCDOUT $FE,$80,”HZ= “,H[4],H[3],H[2],H[1],H[0],”
“;
   LCDOUT $FE,$C0,”V= “,VD[3],”.”,VD[2],VD[1],VD[0],”
“;
RETURN;

SEND:; LABEL SEND;
   HSEROUT [“BZ”,DEC A,10];
   HSEROUT [“AZ”,VD[3],VD[2],VD[1],VD[0],10]; SEND
SERIAL data
   HSEROUT [“CZ”,H[4],H[3],H[2],H[1],H[0],10];

RETURN;

H_DEC:; LABEL h_DEC;
TMR.BYTE0 = TMR1L ; OBTAIN LOWER REGISTER OF
TIMER1
```

```
TMR.BYTE1 = TMR1H; OBTAIN HIGHER REGISTER OF TIMER1
FOR X = 0 TO 4;   START LOOPS
    IN = TMR DIG X;          OBTAIN DIGITS
   LOOKUP IN,["0123456789"],OUT; DECODING DIGITS
    H[X] = OUT;  SAVE DIGITS
NEXT X; NEXT LOOPS
TMR1L = 0; CLEAR TIMER
TMR1H = 0;
RETURN;
END;
```

Listing 2: Code for the PIC16F1614 Working as a RF Receiver

```
; PBP3 COMPILER from melabs.com
; Authors: Ricardo Jimenez and Gabriel Lee Alvarez
; © July 13, 2020
; PIC16F1614 for the Receiver module
;Include "modedefs.bas"; include library
OSCCON = %01111010; 16 MHZ
OSCTUNE = 0;
OSCSTAT = %00011111;  PLL IS OFF,HFINTOSC AND MFINTOSC IS READY

DEFINE OSC 16;        CLOCK SET TO 16MHZ
TRISA = %0000000;     ALL PINS ARE OUTPUTS
ANSELA = %000000;     ALL PINS ARE DIGITAL
WPUA = 0;             INTERN PULL-UPS DISABLED
TRISC = %100000;      RC5 INPUT, RC0:RC4 OUTPUTS
ANSELC = 0;           ALL PINS ARE DIGITAL
WPUC = 0;             INTERN PULL-UPS IS DISABLED
OPTION_REG.7 = 0;     WEAK PULL-UPS ENABLED BY INDIVIDUAL WPUX
;----UART-HSERIN CONFIGURATION------
RXPPS = %10101;             EUSART CR/RX PORTC.5
DEFINE HSER_RCSTA 90h;   RECEIVER ENABLED
DEFINE HSER_TXSTA 20h;   TRANSMITTER ENABLED
DEFINE HSER_BAUD 2400;   2400 BAUD RATE
DEFINE HSER_CLROERR 1;  CLEAR OVERRUN ERROR
rc1sta.7 = 1;                SERIAL PORT IS ENABLED
;--LCD CONFIGURATION ----------------------
DEFINE LCD_DREG PORTC;  PORTC IS A DATA PORT
DEFINE LCD_DBIT 0 ;                RC0 IS THE LSB, RC1,RC2,RC3 ARE MSB
DEFINE LCD_RSREG PORTC ' RS IT IS IN PORTC
DEFINE LCD_RSBIT 4;          RS IN RC4
DEFINE LCD_EREG PORTA ;  EN IS IN PORTA
DEFINE LCD_EBIT 0;           RA0 IS EN
DEFINE LCD_BITS 4 ' 4 ;       IMES
DEFINE LCD_LINES 2 '          It is a 2-line display
DEFINE LCD_COMMANDUS 1500 ; Use 1500uS command delay
DEFINE LCD_DATAUS 44 ' Use 44uS data delay
;;-----------------------------------------------------
;----SET ALIAS--------
PB VAR PORTA.3
LCD1 VAR PORTA.1;
LCD2 VAR PORTA.2;
;----CREATE VARIABLES-----
X VAR byte;
VIN VAR WORD;
VIN2 var word
VD var byte[4];
Y VAR BYTE;
H VAR BYTE[5];
A VAR BYTE;
R VAR BYTE[7]
IN var byte;
OUT VAR BYTE;
;----
;-CLEAR VARIABLES--
FOR Y = 0 TO 5;
VD[Y] = "0"
H[Y] = "0"
NEXT Y;
X =0;

;---INITIALIZE LCD--------------------
LCD1 = 1;  ENABLE LCD1
LCD2 = 0;  DISABLE LCD2
PAUSE 10;
LCDOUT $FE,$28;  $28 FUNCTION SET, 4 BITS
LCDOUT $FE,$10;  $10 SHIFT DISPLAY
LCDOUT $FE,$0C;  $0C DISPLAY ON
LCDOUT $FE,$06;  $06 ENTRY MODE SET
;---
LCD1 = 0;    DISABLE LCD1
LCD2 = 1;    ENABLE LCD2
PAUSE 10;
LCDOUT $FE,$28;  $28 FUNCTION SET, 4 BITS
LCDOUT $FE,$10;  $10 SHIFT DISPLAY
LCDOUT $FE,$0C;  $0C DISPLAY ON
LCDOUT $FE,$06;  $06 ENTRY MODE SET
;-------------------------------------
LCD1 = 1;  ENABLE LCD1
LCD2 =1;   ENABLE LCD2
PAUSE 1;
LCDOUT $FE,$80,"---VOLTMETER----";
LCDOUT $FE,$C0,"--RF RECEIVER---";

PAUSE 1000
MAIN:
```

```
  LCD1 = 1;  ENABLE LCD1
  LCD2 = 0;  DISABLE LCD2
  PAUSE 1;
  LCDOUT $FE,$80,"----WAITING-----";
  LCDOUT $FE,$C0,"----VOLTAGE-----";

  LCD1 = 0;  ENABLE LCD1
  LCD2 = 1;  DISABLE LCD2
  PAUSE 1;
  LCDOUT $FE,$80,"----WAITING-----";
  LCDOUT $FE,$C0,"---FREQUENCY----";
  LCD1 = 1;  ENABLE LCD1
  LCD2 = 1;  ENABLE LCD2
  ;ON INTERRUPT GOTO READ_UART
  MAIN1:
  HSERIN 10,MAIN1,[WAIT("BZ"),  STR A\1]; WAIT 1S
TO RECEIVE INSTRUCTION

  if a = "0" then
    LCD1 = 1;  ENABLE  LCD1
    LCD2 = 1;  ENABLE  LCD2
    PAUSE 1;
    LCDOUT $FE,$80,"  STARTING  ";
    LCDOUT $FE,$C0,"    RX      ";
    PAUSE 1000;
    HERE_START:
    HSERIN 10,JUMP1,[WAIT ("BZ"), STR A\1]
    JUMP1:HSERIN 100,JUMP2,[WAIT ("AZ"), STR VD\4]
    JUMP2:HSERIN 10,HERE, [WAIT ("CZ"), STR H\5]
    HERE:;
    LCD1 = 1;   ENABLE LCD1
    LCD2 = 0;   DISABLE LCD2
    PAUSE 2;
    LCDOUT $FE,$80,"RECEIVED VOLTAGE";
    LCDOUT $FE,$C0,VD[0],"",VD[1],VD[2],VD[3],"      ";
    HERE3:; LABEL HERE3
    LCD1 = 0;  DISABLE LCD1;
    LCD2 = 1;  ENABLE LCD2
    PAUSE 2;
    LCDOUT $FE,$80,"  FREQUENCY  ";
    LCDOUT $FE,$C0,H[0],H[1],H[2],H[3],H[4]," HZ     ";
    ;ENABLE
    LCD1 = 1;  ENABLE LCD1;
    LCD2 = 1;  ENABLE LCD2;
   ;ENABLE INTERRUPT
   if a = "0" then HERE_START
    LCD1 = 1;  ENABLE LCD1
    LCD2 = 1;  ENABLE LCD2
    PAUSE 1
    LCDOUT $FE,$80,"    END     ";
    LCDOUT $FE,$C0,"    RX      ";
    PAUSE 1000 ;
  endif
  goto main;
  END
```