```c
/**
  Generated Main Source File

  Company:
    Microchip Technology Inc.

  File Name:
    main.c

  Summary:
    This is the main file generated using PIC10 / PIC12 / PIC16 / PIC18 MCUs

  Description:
    This header file provides implementations for driver APIs for all modules
selected in the GUI.
    Generation Information :
        Product Revision  :  PIC10 / PIC12 / PIC16 / PIC18 MCUs - 1.81.8
        Device            :  PIC16F1619
        Driver Version    :  2.00
*/

/*
    (c) 2018 Microchip Technology Inc. and its subsidiaries.

    Subject to your compliance with these terms, you may use Microchip software
and any
    derivatives exclusively with Microchip products. It is your responsibility
to comply with third party
    license terms applicable to your use of third party software (including open
source software) that
    may accompany Microchip software.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY
    IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS
    FOR A PARTICULAR PURPOSE.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP
    HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO
    THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL
    CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT
    OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS
    SOFTWARE.
*/

#include "mcc_generated_files/mcc.h"
#include "stdlib.h"
#include "string.h"


/*
                         Main application
```

```
 *
 * *
 * This program is the Digimodes interface to function with the HF
 * Transceiver for the HF digital modes.
 *
 * It monitors the DTR/RTS signal, as generated by the laptop for Tx and when
 * detected, sends a 50 ms pulse to the Tx switch
 * in the transceiver to switch all the latching relays to Tx.
 *
 * When the DTR/RTS signal goes to zero, it sends another 50 ms pulse but
 * this time to the Rx switch which will switch the latching relays to Rx.
 *
 * It runs in an endless loop, averaging the ADC output over the last period.
 *
 * High level flow:
 *
 * Initialize the transceiver to Rx, a known starting point.
 *
 * In an infinite loop:
 *
 * Check DTR/RTS, if in Tx:
 *
 * Is the transceiver in Tx or Rx mode?
 *
 * If in Rx mode:

 * Issue a "Tx" signal to the Tx port and start a timer.
 *    After 50 ms, take the signal back to zero.
 *    Write the Tx audio level to the LCD screen
 * Otherwise do nothing
 *
 * Else:
 *
 * If in Tx mode:
 *
 * Issue a "Rx" signal to the Rx port and start a timer.
 *    After 50 ms, take the signal back to zero.
 *    Write "Rx" to the LCD screen
 * Otherwise do nothing
 *
 */
int Txmode, Rxmode;
long unsigned int j, accum;
unsigned char level;

//
#define Acc_in PORTCbits.RC3
#define Tx_out PORTAbits.RA1
#define Rx_out PORTAbits.RA2
//
//make life easier with the LCD display
//
#define LCD_RS        PORTBbits.RB7        // Register select
#define LCD_EN        PORTCbits.RC1        // Enable
```

```
#define LCD_D4 PORTCbits.RC4        // Data bits
#define LCD_D5 PORTBbits.RB6        // Data bits
#define LCD_D6 PORTCbits.RC6        // Data bits
#define LCD_D7 PORTCbits.RC7        // Data bits

#define        LCD_STROBE        ((LCD_EN = 1),(LCD_EN=0))

unsigned char c;
const char s;
unsigned char *buf;
char buf1[6] = {0}; //ensure the string is terminated with null
//
   // LCD display routines
    //
    /* write a byte to the LCD in 4 bit mode */

void
lcd_write(unsigned char c)
{
        if(c & 0x80) LCD_D7=1; else LCD_D7=0;
        if(c & 0x40) LCD_D6=1; else LCD_D6=0;
        if(c & 0x20) LCD_D5=1; else LCD_D5=0;
        if(c & 0x10) LCD_D4=1; else LCD_D4=0;
        //LCD_STROBE;
           LCD_EN = 1;
           __delay_us(20);
           LCD_EN=0;
        if(c & 0x08) LCD_D7=1; else LCD_D7=0;
        if(c & 0x04) LCD_D6=1; else LCD_D6=0;
        if(c & 0x02) LCD_D5=1; else LCD_D5=0;
        if(c & 0x01) LCD_D4=1; else LCD_D4=0;
        //LCD_STROBE;
           LCD_EN = 1;
           __delay_us(20);

           LCD_EN=0;
        __delay_ms(4);
}

/*
 *         Clear and home the LCD
 */

void
lcd_clear(void)
{
        LCD_RS = 0;

        lcd_write(0x1);
        __delay_ms(5);
}

/* write a string of chars to the LCD */
```

```
void
lcd_puts(const char * s)
{
        LCD_RS = 1;        // write characters

        while(*s) lcd_write(*s++);
}

/* write one character to the LCD */

void
lcd_putch(unsigned char c)
{
        LCD_RS = 1;        // write characters

        lcd_write(c);
}

/* initialise the LCD - put into 4 bit mode */

void
lcd_initt(void)
{
        LCD_RS = 0;        // write control bytes

        __delay_ms(100);// power on delay

        LCD_D4 = 1;        // init!
        LCD_D5 = 1; //
        //LCD_STROBE;
           LCD_EN = 1;
           __delay_us(20);
           LCD_EN=0;
        __delay_ms(10);

        //LCD_STROBE;        // init!
           LCD_EN = 1;
           __delay_us(20);
           LCD_EN=0;
        __delay_ms(2);

        //LCD_STROBE;        // init!
           LCD_EN = 1;
           __delay_us(20);
           LCD_EN=0;
        __delay_ms(5);

        LCD_D4 = 0;        // set 4 bit mode
        //LCD_STROBE;
           LCD_EN = 1;
           __delay_us(20);
           LCD_EN=0;
        __delay_ms(2);
        lcd_write(0x20);// 4 bit mode, 1/16 duty, 5x8 font, 1line
        //lcd_write(0x28);// 4 bit mode, 1/16 duty, 5x8 font, 2lines
        lcd_write(0x0C);// display on
        lcd_write(0x06);// entry mode advance cursor
        lcd_write(0x01);// clear display and reset cursor
           __delay_ms(50);
}
//


void main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    // When using interrupts, you need to set the Global and Peripheral
Interrupt Enable bits
    // Use the following macros to:

    // Enable the Global Interrupts
    INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    INTERRUPT_PeripheralInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    //Initialize the LCD
    lcd_initt();
    //
    // Give the transceiver time to settle
    //
    __delay_ms(5000);
    //
    // Initialize the transceiver to Rx
    //
    Rxmode = 1;
    Txmode = 0;
    Rx_out = 0;
    Tx_out = 0;
    //
    //
    Rx_out = 1;
    __delay_ms(50);
    Rx_out = 0;
    //
    //Write general message to LCD
    lcd_puts("DIGIMODE DISPLAY");
    //
```

```
while (1)
{
    //
 if (Acc_in == 1)
        {
        //in Rx mode? Switch to Tx
        if (Rxmode == 1)
            {Txmode = 1;
             Rxmode = 0;
            // switch transceiver to Tx mode
             Tx_out = 1;
             __delay_ms(50);
             Tx_out = 0;
            }
//
// initialize for ADC sampling
    //
    j = 100;  //number of samples
    accum = 0;
    //
    while (j)
    {

              // take adc sample

        ADC_GetConversion(Audio_in);
        //
        j--;
        accum = accum + ADRESH;

    /*
     * The ADC will return a value between 0 and 255.
     * But this is half cycle sine wave.
     * The average value will be more like the rms value. */

             // determine Tx level
             level = '9';
             if (accum<2500) level = '8';
             if (accum<2000) level = '7';
             if (accum<1750) level = '6';
             if (accum<1500) level = '5';
             if (accum<1250) level = '4';
             if (accum<1000) level = '3';
             if (accum<750) level = '2';
             if (accum<710) level = '1';
             if (accum<500) level = '0';
             //write Tx level to LCD
             lcd_puts("TX AUDIO LEVEL ");
             lcd_putch(level);
    }
    }
    // else the PC is in Rx mode
    else {
            // in Tx mode? The output switch to Rx is done in the ISR
            if (Txmode == 1)
                {
                Txmode = 0;
                Rxmode = 1;
                // Write Rx to LCD
                lcd_clear(); // The LCD unit did not seem to be ready...
                lcd_puts("  RX MODE       ");
                }
           }


         }

}


// End of File
```