```
/*-------------------------------------------------------------------------------+
|                                                                               |
|                                DTMF Receiver                                  |
|                                                                               |
| Name: DTMFReceiver.c                                                          |
| Purpose:                                                                      |
|   DTMF stands for dual tone multiple frequency. When you press a button on a  |
|   telephone keypad, the telephone will generate two sine waves with different |
|   frequency at the same time, the audible sounds that you hear. The program start |
|   to initalize the transceiver, then set for receiveing the DTMF tone. When valid |
|   tone is received, it will print the tone digit to LCD and RS232 port. If no tone|
|   is received, print nothing out, and wait. The program demonstration the useage |
|   of DTMF receiver.                                                           |
|                                                                               |
| By Dillian Wong, last modify data 27/1/04                                     |
+-------------------------------------------------------------------------------*/

/*____________ I N C L U D E S _____________________________________________*/

#include <reg51.h>
#include <stdio.h>
#include <ctype.h>                  //use toint()
#include "MT8880C\configMT8880C.h"
#include "LCDDriver\LCDConfig.h"

sbit toneLamp = P3^7;
#define toneLampOn  toneLamp=0;
#define toneLampOff toneLamp=1;
/*_____________ M A I N  P R O G R A M _____________________________________*/

// com port with 9600 baud with crystal 11.0592MHz.
void init_uart(void)
{
    SCON  = 0x50;
    TMOD |= 0x20;
    TH1   = 253;
    TR1   = 1;
    TI    = 1;
}

/*
    main program to implement MT8880C DTMF Transceiver
*/
void main(void)
{
    char chDigit;       // tone digit
    int i=0;

    init_uart();        // utilize rs232 port
    turnOnLCD();        // turn on LCD
    turnOnRS232();      // turn on RS232
    toneLampOff;

    MT8880C_Reset();    //software reset.
    LCD_Locate(1);
    printf("\nDTMF Decoder");
    printf("\nReceived Tone : ");
    while(1){
        chDigit = MT8880C_ReceiveDTMF_tone();
        if(chDigit!='N')            // if no valid tone, 'N' will return
        {
            toneLampOn;
            printf("%c",chDigit);
            for(i=0;i<4000;i++);
            toneLampOff;
        }
    }
}
```

```
/*-------------------------------------------------------------------------------+
|                                                                                |
|                    MITEL Integrated DTMF Transceiver (MT8880C)                 |
|                                                                                |
| Name: MT8880C.c                                                                |
| Purpose:                                                                       |
|   DTMF stands for dual tone multiple frequency. When you press a button on a   |
|   telephone keypad, the telephone will generate two sine waves with different  |
|   frequency at the same time, the audible sounds that you hear. This program   |
|   is used for the access and control the DTMF Transceiver, to transmit tone, or |
|   recevice tone from telephone line. software reset must be implement, before any |
|   transceive function.                                                         |
|                                                                                |
| By Dillian Wong, last modify data 27/1/04                                      |
+-------------------------------------------------------------------------------*/

/*_____________ I N C L U D E S _____________________________________________*/

#include <reg51.h>
#include <stdio.h>
#include "configMT8880C.h"

/*_____________ M A C R O _____________________________________________*/

/* set or clr of important PIN */
#define setCS CS=1;while (CS!=1); // chip select, clr for select
#define clrCS CS=0;
#define setRS RS=1;while (RS!=1); // register select, set for select
#define clrRS RS=0;
#define setRW RW=1;while (RW!=1); // read write, set for read, clr for write
#define clrRW RW=0;

/*_____________ P R O T O C O L _____________________________________________*/

void MT8880C_Reset(void);                       // RESET
void MT8880C_SetForTransmit_tone(void);         // TRANSMIT
void MT8880C_TransDTMF_tone(char ch);
char MT8880C_ReceiveDTMF_tone(void);            // RECEIVE
void MT8880C_ReadFromStatusRegister(void);
void MT8880C_ReadFromReceiveDataRegister(void);
int digit2ToneNo(char ch);                      // CONVERTION
char toneNo2Digit(int num);


/*_____________ DTMF transceiver iitialization_____________________________*/

/*
    A software reset must be included at the beginning of all programs to initialize the
    control registers after power up. The initialization procedure should be implemented
    100ms after power up.
*/
void MT8880C_Reset(void)
{
    int i;

    setCS;      //1) read status register
    setRS;setRW;
    clrCS;
    for(i=0;i<3;i++);
    setCS;

    setCS;      //2) write to control register
    setRS;clrRW;b3=0;b2=0;b1=0;b0=0;
    clrCS;
    for(i=0;i<3;i++);
    setCS;

    setCS;      //3) write to control register
    setRS;clrRW;b3=0;b2=0;b1=0;b0=0;
    clrCS;
    for(i=0;i<3;i++);
    setCS;

    setCS;      //4) write to control register
    setRS;clrRW;b3=1;b2=0;b1=0;b0=0;
    clrCS;
    for(i=0;i<3;i++);
    setCS;

    setCS;      //5) write to control register
```

```
    setRS;clrRW;b3=0;b2=0;b1=0;b0=0;
    clrCS;
    for(i=0;i<3;i++);
    setCS;

    setCS;      //6) read status register
    setRS;setRW;
    clrCS;
    for(i=0;i<3;i++);
    setCS;
}


/*_____________ DTMF tone transmitter function_____________________________________*/

/*
    setting before transmit a tone.
*/
void MT8880C_SetForTransmit_tone(void)
{
    int m;

    /*tone out, DTMF, IRQ, select control regiser B*/
    setCS;                  //write to control register A
    setRS;  clrRW;  b3=1;   b2=1;   b1=0;   b0=1;
    clrCS;                  //toggle chip select to fatch the input data
    for(m=0;m<3;m++);
    setCS;

    /*burst mode*/
    setCS;                  //write to control register B
    setRS;  clrRW;  b3=0;   b2=0;   b1=0;   b0=0;
    clrCS;                  //toggle chip select to fatch the input data
    for(m=0;m<3;m++);
    setCS;
}

/*
    Transmit a DTMF tone, ch is the tone digit (0-9,*,#,A-D)
*/
void MT8880C_TransDTMF_tone(char ch)
{
    int m;
    int i=0,j=0,k=0,l=0,digit;

    digit=digit2ToneNo(ch); //get the interger that represent the tone

    i=(digit&0x0008)/8;     //calculate the b3,b2,b1,b0
    j=(digit&0x0004)/4;
    k=(digit&0x0002)/2;
    l=(digit&0x0001)/1;

    setCS;                  //write to transmit data register
    clrRS;  clrRW;  b3=i;   b2=j;   b1=k;   b0=l;
    clrCS;                  //toggle chip select to fatch the input data
    for(m=0;m<3;m++);
    setCS;
}


/*_____________ DTMF tone receiver function_____________________________________*/

/*
    if DTMF tone received, return the received DTMF tones, else return 'N' indicate
    no DTMF tone received.
*/
char MT8880C_ReceiveDTMF_tone(void)
{
    int i=0,j=0,k=0,l=0,num=0;
    char ch;

    MT8880C_ReadFromStatusRegister();           //code for read status
    if(b2==1)                                   //if bit2 is set, a DTMF tone has been
    {                                           //received, in which case....
        MT8880C_ReadFromReceiveDataRegister();  //code for read data
        i=b3;j=b2;k=b1;l=b0;
        num=i*8+j*4+k*2+l;                      //convert 4 binary to integer
        ch=toneNo2Digit(num);                   //get the relate digit character
        return ch;
    }
```

```
    else
        return 'N';
}

/*
    read the status register
*/
void MT8880C_ReadFromStatusRegister(void)
{
    int i=0;
    /*Read the register*/
    setCS;                  //disable chip select
    setRS;  setRW;          //select register, select read
    setCS;                  //toggle chip select to fatch the input data
    for(i=0;i<30;i++);
    clrCS;                  //chip select
}

/*
    read the received data register
*/
void MT8880C_ReadFromReceiveDataRegister(void)
{
    int i=0;
    /*Read the register*/
    setCS;
    clrRS;  setRW;          //not register, select read
    setCS;                  //toggle chip select to fatch the input data
    for(i=0;i<30;i++);
    clrCS;
}


/*_____________ tone Number & DTMF digit Convertion_____________________________________*/

/*
    return the interger that represent the digal number or character.
    1-9,0,*,#,A-D.
*/
int digit2ToneNo(char ch)
{
    int digit;
    switch(ch)
    {   //digit     toneNo
        case '1':   digit=1;    break;
        case '2':   digit=2;    break;
        case '3':   digit=3;    break;
        case '4':   digit=4;    break;
        case '5':   digit=5;    break;
        case '6':   digit=6;    break;
        case '7':   digit=7;    break;
        case '8':   digit=8;    break;
        case '9':   digit=9;    break;
        case '0':   digit=10;   break;
        case '*':   digit=11;   break;
        case '#':   digit=12;   break;
        case 'A':   digit=13;   break;
        case 'B':   digit=14;   break;
        case 'C':   digit=15;   break;
        case 'D':   digit=0;    break;
        default:    digit=0;    break;
    }
    return digit;
}

/*
    convert the tone number that represent the digit.
*/
char toneNo2Digit(int num)
{
    char ch;
    switch(num)
    {   //toneNo    digit
        case 1:     ch='1';     break;
        case 2:     ch='2';     break;
        case 3:     ch='3';     break;
        case 4:     ch='4';     break;
        case 5:     ch='5';     break;
        case 6:     ch='6';     break;
        case 7:     ch='7';     break;
```

```
        case 8:     ch='8';     break;
        case 9:     ch='9';     break;
        case 10:    ch='0';     break;
        case 11:    ch='*';     break;
        case 12:    ch='#';     break;
        case 13:    ch='A';     break;
        case 14:    ch='B';     break;
        case 15:    ch='C';     break;
        case 0:     ch='D';     break;
    }
    return ch;
}
```

```
/*
  Name :  configMT8880C.h

  Header file for MT8880C.c, notice that the header file name sometime cannot same
  as the source file name.
*/


/*_____________ H A R D W A R E _____________________________________________*/

sbit CS = P1^5; // control
sbit RS = P1^4;
sbit RW = P1^7;
sbit b3 = P1^3; // data
sbit b2 = P1^2;
sbit b1 = P1^1;
sbit b0 = P1^0;

/*_____________ P R O T O C O L _____________________________________________*/
void MT8880C_Reset(void);                       // initialize MT8880C before encode
                                                // or decode DTMF tone
void MT8880C_SetForTransmit_tone(void);         // setting before transmit DTMF tone
void MT8880C_TransDTMF_tone(char ch);           // transmit a tone
char MT8880C_ReceiveDTMF_tone(void);            // receive a tone
```

```
/*-------------------------------------------------------------------------------+
|                                                                                |
|                          LCD 16x2 Driver (HD44780)                             |
|                                                                                |
| Name: Driver_Lcd16x2.c                                                         |
| Purpose:                                                                       |
|   This is the function code used to modify the standard c function "printf" by |
|   replacing the function "putchar" to suit the LCD driver HD44780. By adding this |
|   file, LCD panel with a chip HD44780 can print out a string using the standard c |
|   function "printf". \n, \r, and roll down will be included.                   |
|                                                                                |
| By Dillian Wong, last modify data 19/9/03                                      |
+-------------------------------------------------------------------------------*/

/*_____________ I N C L U D E S _____________________________________________*/

#include <reg51.h>
#include <stdio.h>
#include "LCDConfig.h"
unsigned char flag=0;

/*_____________ M A C R O S _____________________________________________*/

#define DISP_BUSY   0x80
#define DISP_FUNC   0x38
#define DISP_ENTRY  0x06
#define DISP_CNTL   0x08
#define DISP_ON     0x04
#define DISP_CURSOR 0x02
#define DISP_CLEAR  0x01
#define DISP_HOME   0x02
#define DISP_POS    0x80
#define DISP_LINE2  0x40

/*_____________ P R O T O C O L _____________________________________________*/

void disp_cmd(unsigned char);
void disp_init(void);
unsigned char disp_read(void);
void disp_write(unsigned char);
unsigned char disp_read_char(void);
void disp_copyLine2toLine1(void);
void disp_clrLine2(void);
char LCD_putchar (char c);
void LCD_Locate(int p);
void LCD_Clr(int s, int e);

/*_____________ B A S I C _____________________________________________*/

/*
    writes a given command to the LCD panel and waits to assure that the command
    was completed by the panel.
*/
void disp_cmd(unsigned char cmd) {
    DISPDATA=cmd;
    REGSEL=0;
    RDWR=0;
    ENABLE=1;
    ENABLE=0;
    while (disp_read() & DISP_BUSY);               // wait for the Lcd
}

/*
    sends the correct data sequence to the display to initialize it for use.
*/
void disp_init(void) {

    while (disp_read() & DISP_BUSY);
    disp_cmd(DISP_FUNC);                           // set the display for an 8
                                                   // bit bus, 2 display lines,
                                                   // and a 5x7 dot font
    disp_cmd(DISP_ENTRY);                          // set the character entry
                                                   // mode to increment display
                                                   // address for each
                                                   // character, but not to scroll
    disp_cmd(DISP_CNTL | DISP_ON);                 // turn the display on, cursor off
    disp_cmd(DISP_CLEAR);                          // clear the display
}

/*
```

```
    reads from the LCD panel status register.
*/
unsigned char disp_read(void) {
    unsigned char value;
    DISPDATA=0xFF;
    REGSEL=0;
    RDWR=1;
    ENABLE=1;
    value=DISPDATA;
    ENABLE=0;
    return(value);
}

/*
    writes a data byte to the LCD panel.
*/
void disp_write(unsigned char value) {
    DISPDATA=value;
    REGSEL=1;
    RDWR=0;
    ENABLE=1;
    ENABLE=0;
}

/*
    read data from CF or DD Ram
*/
unsigned char disp_read_char(void) {
    unsigned char value;
    DISPDATA=0xFF;
    REGSEL=1;
    RDWR=1;
    ENABLE=1;
    value=DISPDATA;
    ENABLE=0;
    return(value);
}

/*
    reads data from DD Ram in line2, and write char of these datas in line1
*/
void disp_copyLine2toLine1(void){
    unsigned int i;
    unsigned char k[16];
    disp_cmd(DISP_POS | DISP_LINE2);
    for(i=0;i<16;i++){
        k[i]=disp_read_char();
        while (disp_read() & DISP_BUSY);
    }
    disp_cmd(DISP_HOME);
    for(i=0;i<16;i++){
        disp_write(k[i]);
        while (disp_read() & DISP_BUSY);
    }
}

/*
    clear the display of Line2
*/
void disp_clrLine2(void){
    unsigned int i;
    disp_cmd(DISP_POS | DISP_LINE2);
    for(i=0;i<16;i++){
        disp_write(' ');
        while (disp_read() & DISP_BUSY);
    }
}



/*_____________ A D V A N C E _____________________________________________________*/

/*
    clear the DD ram data of LCD from position s to position e
*/
void LCD_Clr(int s, int e)
{
    int i;
    LCD_Locate(s);
```

```
    for(i=0;i<=(e-s);i++){
        disp_write(' ');
        while (disp_read() & DISP_BUSY);
    }
}

/*
    loacate the position of LCD
*/
void LCD_Locate(int p)
{
    /*
    HD44760 16x2 character LCD display
    the following .no is variable flag refer to
    +-------------------------------------------------+
    |  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 |
    | 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
    +-------------------------------------------------+

    HD44760 16x2 character LCD display
    the following DDRam Address
    +-------------------------------------------------------------------------------+
    | 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F |
    | 0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F |
    +-------------------------------------------------------------------------------+
    */

    if(p<=15){
        disp_cmd(DISP_POS | p);
        flag = p;
    }
    else if(p>15 && p<=31){
        disp_cmd(DISP_POS | (p+0x30));
        flag=p;
    }
    else if(p>31){
        disp_cmd(DISP_POS | 0x00);
        flag = 0;
    }
}



/*************************************************************
Function:    putchar
Description: This routine replaces the standard putchar
             function. Its job is to redirect output to the LCD panel.
Parameters:  c - char. This is h\next character to write to the
             display.
Returns:     The character just written.
*************************************************************/
char LCD_putchar(char c) {
    /*
    HD44760 16x2 character LCD display
    the following .no is variable flag refer to
    +-------------------------------------------------+
    |  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 |
    | 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
    +-------------------------------------------------+
    */

    if (flag <=15){
        // standard format for the nextline
        if(c=='\n'){
                disp_clrLine2();
                disp_cmd(DISP_POS | DISP_LINE2);
                flag=16;
                return(c);
        }
        // standard format for return the line
        if(c=='\r'){
                disp_cmd(DISP_HOME);
                flag=1;
                return(c);
        }

    }

    if(flag==16){
            disp_cmd(DISP_POS | DISP_LINE2);
```

```
    }

    if(flag>=16){
        // standard format for the nextline
        if(c=='\n'){
                disp_copyLine2toLine1();
                disp_clrLine2();
                disp_cmd(DISP_POS | DISP_LINE2);
                flag=16;
                return(c);
        }
        // standard format for return the line
        if(c=='\r'){
                disp_cmd(DISP_POS | DISP_LINE2);
                flag=16;
                return(c);
        }
    }

    if (flag>31){
        disp_copyLine2toLine1();
        disp_clrLine2();
        disp_cmd(DISP_POS | DISP_LINE2);
        flag=16;
    }


    disp_write(c);                          // write the character to
                                            // the display
    while (disp_read() & DISP_BUSY);        // wait for the display
    flag++;
    return(c);

}
```

```
/*
    Name : LcdAccess.c
    purpose:


*/


/*_____________ I N C L U D E S _____________________________________________________*/

#include <reg51.h>
#include <stdio.h>
#include "LCDConfig.h"

/*_____________ M A C R O S _____________________________________________________*/

#define true    1
#define false   0

/*_____________ P R O T O C O L _____________________________________________________*/

char standard_putchar (char c);
char LCD_putchar(char c);
void disp_init(void);

/*_____________ V A R I A B L I E S _____________________________________________________*/

bit isRS232 = true;
bit isLCD = false;


/*_____________ F U N C T I O N _____________________________________________________*/


/*
    replace the standard putchar by adding a extra instruction.
*/
char putchar(char c)
{
    char ch;
    if(isRS232)
        ch = standard_putchar(c);
    if(isLCD)
        ch = LCD_putchar(c);
    return c;
}

/*
    flag used to indicate the status of display. Whether the lcd, com port should display the data
    out or not.
*/
void turnOnLCD(void)    {   disp_init(); isLCD=true;        }   // indicate lcd on
void turnOffLCD(void)   {   isLCD=false;    }                   // indicate lcd off
void turnOnRS232(void)  {   isRS232=true;   }                   // indicate com port on
void turnOffRS232(void) {   isRS232=false;  }                   // indicate com port off
bit  isLCDOn(void)      {   return isLCD;   }                   // indicate where LCD is on or off
bit  isRS232On(void)    {   return isRS232; }                   // indicate where RS232 is on of off.
```

```
/*
    Name : standardPutChar.c
    purpose:
            standard putchar function of keil C
*/


/*_____________ I N C L U D E S _____________________________________________*/

#include <reg51.h>


/*_____________ M A C R O S _____________________________________________*/

#define XON  0x11
#define XOFF 0x13

/*_____________ F U N C T I O N _____________________________________________*/


/*
    standard putchar function of keil C
*/
char standard_putchar (char c)  {

  if (c == '\n')  {
    if (RI)  {
      if (SBUF == XOFF)  {
        do  {
          RI = 0;
          while (!RI);
        }
        while (SBUF != XON);
        RI = 0;
      }
    }
    while (!TI);
    TI = 0;
    SBUF = 0x0d;                        /* output CR  */
  }
  if (RI)  {
    if (SBUF == XOFF)  {
      do  {
        RI = 0;
        while (!RI);
      }
      while (SBUF != XON);
      RI = 0;
    }
  }
  while (!TI);
  TI = 0;
  return (SBUF = c);
}
```

```
/*
  Name :  LCDConfig.h

  Header file for Driver_Lcd16x2.c, LCDAccess.c.

*/


/*_____________ H A R D W A R E _____________________________________________*/

sbit ENABLE   =  P0^2;                    // display enable output (EN)
sbit RDWR     =  P0^1;                    // display access mode output (RW)
sbit REGSEL   =  P0^0;                    // display register select output (DI)
sfr  DISPDATA =  0xA0;                    // data bus to the display panel,P2

/*_____________ P R O T O C O L _____________________________________________*/

// LCDAccess.c
void turnOnLCD(void);                     // turn on LCD, default is off
void turnOffLCD(void);                    // turn off LCD
void turnOnRS232(void);                   // turn on RS232 port
void turnOffRS232(void);                  // turn on RS232 port
bit  isLCDOn(void);                       // indicate where LCD is on or off
bit  isRS232On(void);                     // indicate where RS232 is on of off.

// Driver_Lcd16x2.c
void LCD_Locate(int p);                   // location the DD ram position of LCD
void LCD_Clr(int s, int e);               // clear the DD ram data from position s
                                          // to e.
```