```
//-----------------------------------------------------------------------------
// Distance.c
//-----------------------------------------------------------------------------
// Author: Baylor Electromechanical Systems
//
// Operates on an external 18.432 MHz oscillator.
//
// Target: Cygnal Educational Development Board / C8051F020
// Tool chain: KEIL C51 6.03 / KEIL EVAL C51
//
// Utilizes external microphone and speaker devices and measures the distance
// between them. Output is display on the LCD.
//

//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------

#include <c8051f020.h>              // SFR declarations
#include <stdio.h>

//-----------------------------------------------------------------------------
// 16-bit SFR Definitions for 'F02x
//-----------------------------------------------------------------------------

sfr16 DP       = 0x82;              // data pointer
sfr16 TMR3RL   = 0x92;              // Timer3 reload value
sfr16 TMR3     = 0x94;              // Timer3 counter
sfr16 ADC0     = 0xbe;              // ADC0 data
sfr16 ADC0GT   = 0xc4;              // ADC0 greater than window
sfr16 ADC0LT   = 0xc6;              // ADC0 less than window
sfr16 RCAP2    = 0xca;              // Timer2 capture/reload
sfr16 T2       = 0xcc;              // Timer2
sfr16 RCAP4    = 0xe4;              // Timer4 capture/reload
sfr16 T4       = 0xf4;              // Timer4
sfr16 DAC0     = 0xd2;              // DAC0 data
sfr16 DAC1     = 0xd5;              // DAC1 data

//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------

void main (void);
void SYSCLK_Init (void);
void PORT_Init (void);
void UART0_Init (void);
void Timer3_Init (int counts);
void ADC1_Init (void);
void ADC1_ISR (void);
void Timer2_Init (void);
void delay (int millisec);
void Timer4_Init (int counts);
void Timer4_ISR (void);


//-----------------------------------------------------------------------------
// Global CONSTANTS
```

```
//-----------------------------------------------------------------------------

#define SYSCLK          18432000        // SYSCLK frequency in Hz

#define BAUDRATE        9600            // Baud rate of UART in bps

#define SAMPLE_RATE_DAC 80000L          // DAC sampling rate in Hz

#define PHASE_PRECISION 65536           // range of phase accumulator
#define SAMPLE_RATE  120000L            // Sample frequency in Hz (ADC)

typedef union lng {                     // access a long variable as two
      long Long;                        // 16-bit integer values
      int Int[2];
   } lng;



//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------



unsigned int phase_add = 1000 * PHASE_PRECISION / SAMPLE_RATE_DAC;

unsigned int amplitude = 0;       // start out with amplitude of 0
                                  // see the Timer 4 ISR



// a full cycle, 16-bit, 2's complement sine wave lookup table
int code SINE_TABLE[256] = {

   0x0000, 0x0324, 0x0647, 0x096a, 0x0c8b, 0x0fab, 0x12c8, 0x15e2,
   0x18f8, 0x1c0b, 0x1f19, 0x2223, 0x2528, 0x2826, 0x2b1f, 0x2e11,
   0x30fb, 0x33de, 0x36ba, 0x398c, 0x3c56, 0x3f17, 0x41ce, 0x447a,
   0x471c, 0x49b4, 0x4c3f, 0x4ebf, 0x5133, 0x539b, 0x55f5, 0x5842,
   0x5a82, 0x5cb4, 0x5ed7, 0x60ec, 0x62f2, 0x64e8, 0x66cf, 0x68a6,
   0x6a6d, 0x6c24, 0x6dca, 0x6f5f, 0x70e2, 0x7255, 0x73b5, 0x7504,
   0x7641, 0x776c, 0x7884, 0x798a, 0x7a7d, 0x7b5d, 0x7c29, 0x7ce3,
   0x7d8a, 0x7e1d, 0x7e9d, 0x7f09, 0x7f62, 0x7fa7, 0x7fd8, 0x7ff6,
   0x7fff, 0x7ff6, 0x7fd8, 0x7fa7, 0x7f62, 0x7f09, 0x7e9d, 0x7e1d,
   0x7d8a, 0x7ce3, 0x7c29, 0x7b5d, 0x7a7d, 0x798a, 0x7884, 0x776c,
   0x7641, 0x7504, 0x73b5, 0x7255, 0x70e2, 0x6f5f, 0x6dca, 0x6c24,
   0x6a6d, 0x68a6, 0x66cf, 0x64e8, 0x62f2, 0x60ec, 0x5ed7, 0x5cb4,
   0x5a82, 0x5842, 0x55f5, 0x539b, 0x5133, 0x4ebf, 0x4c3f, 0x49b4,
   0x471c, 0x447a, 0x41ce, 0x3f17, 0x3c56, 0x398c, 0x36ba, 0x33de,
   0x30fb, 0x2e11, 0x2b1f, 0x2826, 0x2528, 0x2223, 0x1f19, 0x1c0b,
   0x18f8, 0x15e2, 0x12c8, 0x0fab, 0x0c8b, 0x096a, 0x0647, 0x0324,
   0x0000, 0xfcdc, 0xf9b9, 0xf696, 0xf375, 0xf055, 0xed38, 0xea1e,
   0xe708, 0xe3f5, 0xe0e7, 0xdddd, 0xdad8, 0xd7da, 0xd4e1, 0xd1ef,
   0xcf05, 0xcc22, 0xc946, 0xc674, 0xc3aa, 0xc0e9, 0xbe32, 0xbb86,
   0xb8e4, 0xb64c, 0xb3c1, 0xb141, 0xaecd, 0xac65, 0xaa0b, 0xa7be,
   0xa57e, 0xa34c, 0xa129, 0x9f14, 0x9d0e, 0x9b18, 0x9931, 0x975a,
   0x9593, 0x93dc, 0x9236, 0x90a1, 0x8f1e, 0x8dab, 0x8c4b, 0x8afc,
   0x89bf, 0x8894, 0x877c, 0x8676, 0x8583, 0x84a3, 0x83d7, 0x831d,
   0x8276, 0x81e3, 0x8163, 0x80f7, 0x809e, 0x8059, 0x8028, 0x800a,
   0x8000, 0x800a, 0x8028, 0x8059, 0x809e, 0x80f7, 0x8163, 0x81e3,
   0x8276, 0x831d, 0x83d7, 0x84a3, 0x8583, 0x8676, 0x877c, 0x8894,
   0x89bf, 0x8afc, 0x8c4b, 0x8dab, 0x8f1e, 0x90a1, 0x9236, 0x93dc,
   0x9593, 0x975a, 0x9931, 0x9b18, 0x9d0e, 0x9f14, 0xa129, 0xa34c,
   0xa57e, 0xa7be, 0xaa0b, 0xac65, 0xaecd, 0xb141, 0xb3c1, 0xb64c,
   0xb8e4, 0xbb86, 0xbe32, 0xc0e9, 0xc3aa, 0xc674, 0xc946, 0xcc22,
   0xcf05, 0xd1ef, 0xd4e1, 0xd7da, 0xdad8, 0xdddd, 0xe0e7, 0xe3f5,
   0xe708, 0xea1e, 0xed38, 0xf055, 0xf375, 0xf696, 0xf9b9, 0xfcdc,
};




//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------

void main (void) {

      int i,j,dec;
      long Time, distacc;
      int max, temp;


      WDTCN = 0xde;                    // Disable watchdog timer
      WDTCN = 0xad;

      SYSCLK_Init ();
      PORT_Init ();

            // initializations for wave generation
      REF0CN = 0x03;                   // enable internal VREF generator
      DAC1CN = 0x97;                   // enable DAC1 in left-justified mode

      Timer4_Init(SYSCLK/SAMPLE_RATE_DAC);
                                    // using Timer4 as update scheduler
                                    // initialize T4 to update DAC1
                                    // after (SYSCLK cycles)/sample have
                                    // passed.


      UART0_Init ();
      Timer3_Init (SYSCLK/SAMPLE_RATE);         // initialize Timer3 to overflow
at
                                                // sample rate
      Timer2_Init ();                           // init Timer 2
      ADC1_Init ();                             // init ADC
      ADC1CN |= 0x80;                           // enable ADC1
      EA = 1;                         // Enable global interrupts
      putchar (254);                            // LCD command
      putchar (0x01);                           // clear LCD
      dec = 4;                                  // decimate number (to average 4
values)
      distacc=0;                                // init accumulator
      while(1){
            amplitude = 5 * 655;    // set amplitude
```

```
                max = 0;                        // dump max
                  delay (100);                  // wait for settle
                  for (i=0;i<= 15;i++)
                        for (j=0; j<=200;j++)
                        {
                        if (ADC1 > max )        // if adc value is greater than
max
                              max = ADC1;             // adc becomes new max
                        }

                  temp = max >> 4;
                  max = max + temp;             // Increase max by x%
                  T2=0;                               // dump Timer2

                  amplitude =0;                       // stop sine wave
                  DAC1=0x8000 ^ 32767;          // Generate impulse
                  DAC1=0x8000 ^ 32767;          // Generate impulse

                  TR2=1;                                    // Start Timer 2
                  while (ADC1 <= max);
                  TR2=0;                                    // Stop Timer 2
                Time = T2;                            // Time = Timer 2 value
                  Time = Time / 2000;
                  Time = Time * 7703;
                  Time = Time / 5000;
                  Time = (( Time * 250) - 1177 ) / 250;
                  distacc += Time +1;                 // calculate distance and
add to acc
                  dec--;                                    // decrement dec

                  if (dec ==0)                        // when dec = 0 take avg
of 4 values
                  {
                        temp = distacc >> 2;    // shift to divide by 4
                        printf ("Distance is \n %d in \n",temp);
                        dec = 4;                      // reset dec
                        distacc = 0;                  // dump distacc
                        delay (500);                  // wait
                        putchar (254);                // LCD command
                        putchar (0x01);               // clear LCD
                  }

                  delay (30);                         // wait

   } // end while(1)

} // end main


//-----------------------------------------------------------------------------
// Init Routines
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
```

```
// This routine initializes the system clock to use a 18.432MHz crystal
// as its clock source.
//
void SYSCLK_Init (void)
{
   int i;                             // delay counter

   OSCXCN = 0x67;                     // start external oscillator with
                                      // 18.432MHz crystal

   for (i=0; i < 256; i++) ;          // Wait for osc. to start up

   while (!(OSCXCN & 0x80)) ;         // Wait for crystal osc. to settle

   OSCICN = 0x88;                     // select external oscillator as SYSCLK
                                      // source and enable missing clock
                                      // detector

}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
   XBR0     = 0x04;                   // Enable UART0
   XBR1     = 0x00;
   XBR2     = 0x40;                   // Enable crossbar and weak pull-up
   P0MDOUT |= 0x01;                   // Set TX0 pin to push-pull

   P1MDIN = 0xFC;                                 // Input configuration for P1
                                                       // Set P1.0 as Analog
Input A1.0
   P1MDOUT = 0x00;
   P1 = 0xFF;

}

//-----------------------------------------------------------------------------
// Timer2_Init
//-----------------------------------------------------------------------------
// This routine initializes Timer2 in auto-reload mode to generate interrupts
// at intervals specified in <counts>.
//
void Timer2_Init (void)
{


                   T2CON |= 0x04;         // start Timer2
                   T2 = 0;
                   CKCON |=0x20;                // Timer2 uses sysclk
}

//-----------------------------------------------------------------------------
```

```
// Timer3_Init
//-----------------------------------------------------------------------------
//
// Configure Timer3 to auto-reload at interval specified by <counts> (no
// interrupt generated) using SYSCLK as its time base.
//
void Timer3_Init (int counts)
{
   TMR3CN = 0x02;                      // Stop Timer3; Clear TF3;
                                       // use SYSCLK as timebase
   TMR3RL  = -counts;                  // Init reload values
   TMR3    = 0xffff;                   // set to reload immediately
   EIE2   &= ~0x01;                    // disable Timer3 interrupts
   TMR3CN |= 0x04;                     // start Timer3
}

//-----------------------------------------------------------------------------
// Timer4_Init
//-----------------------------------------------------------------------------
// This routine initializes Timer4 in auto-reload mode to generate interrupts
// at intervals specified in <counts>.
//
void Timer4_Init (int counts)
{
   T4CON = 0;                       // STOP timer; set to auto-reload mode
   CKCON |= 0x40;                   // T4M = '1'; Timer4 counts SYSCLKs
   RCAP4 = -counts;                 // set reload value
   T4 = RCAP4;
   EIE2 |= 0x04;                    // enable Timer4 interrupts
   T4CON |= 0x04;                   // start Timer4
}

//-----------------------------------------------------------------------------
// UART0_Init
//-----------------------------------------------------------------------------
//
// Configure the UART0 using Timer1, for <baudrate> and 8-N-1.
//
void UART0_Init (void)
{
   SCON0   = 0x50;                  // SCON0: mode 1, 8-bit UART, enable RX
   TMOD    = 0x20;                  // TMOD: timer 1, mode 2, 8-bit reload
   TH1    = -(SYSCLK/BAUDRATE/16);  // set Timer1 reload value for baudrate
   TR1    = 1;                      // start Timer1
   CKCON |= 0x10;                   // Timer1 uses SYSCLK as time base
   PCON  |= 0x80;                   // SMOD0 = 1
   TI0    = 1;                      // Indicate TX0 ready
}


//-----------------------------------------------------------------------------
// ADC1_Init
//-----------------------------------------------------------------------------
//
// Configure ADC1 to use Timer3 overflows as conversion source, to
// generate an interrupt on conversion complete, and to use left-justified
// output mode.  Enables ADC end of conversion interrupt. Leaves ADC disabled.
```

```
//
void ADC1_Init (void)
{
    AMX1SL = 0x00;  // AMUX1 Channel Select Register
                              // Select Analog Input A1.0
    ADC1CF = 0x10;  // ADC1 Configuration Register
    ADC1CN = 0x82;  // ADC1 Control Register
                              // convert on Timer3 overflow
  //  EIE2 |= 0x08;     // enable ADC1 interrupts


}


//-----------------------------------------------------------------------------
// Interrupt Handlers
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// Timer4_ISR -- Wave Generator
//-----------------------------------------------------------------------------
//
// This ISR is called on Timer4 overflows.  Timer4 is set to auto-reload mode
// and is used to schedule the DAC output sample rate in this example.
// Note that the value that is written to DAC1 during this ISR call is
// actually transferred to DAC1 at the next Timer4 overflow.
//
void Timer4_ISR (void) interrupt 16 using 3
{

   static unsigned phase_acc = 0;  // holds phase accumulator

   int temp1;                      // the temporary value that passes
                                   // through 3 stages before being written
                                   // to DAC1

   int code *table_ptr;            // pointer to the lookup table

   lng temporary_long;             // holds the result of a 16-bit multiply

   T4CON &= ~0x80;                 // clear T4 overflow flag

   table_ptr = SINE_TABLE;

   phase_acc += phase_add;         // increment phase accumulator


   // set the value of <temp1> to the next output of DAC1 at full-scale
   // amplitude; the  rails are +32767, -32768


                  // sine wave generation
            temp1 = *(table_ptr + (phase_acc >> 8));



   // Adjust the Gain
   temporary_long.Long = (long) ((long)temp1 * (long)amplitude);


   temp1 = temporary_long.Int[0];  // same as temporary_long >> 16


   // Add a DC bias to make the rails 0 to 65535
   // Note: the XOR with 0x8000 translates the bipolar quantity into
   // a unipolar quantity.

         DAC1 = 0x8000 ^ temp1;

}

//-----------------------------------------------------------------------------
// ADC1_ISR
//-----------------------------------------------------------------------------
// ADC1 end-of-conversion ISR
//

void ADC1_ISR (void) interrupt 15
{

   ADC1CN &= ~0x20;                             // clear ADC conversion complete
                                                            // indicator
}

//-----------------------------------------------------------------------------
// Delay
//-----------------------------------------------------------------------------
// delay routine
//
void delay (int millisec)
{
      int i,j;
      for (j=0;j<=20;j++)
         for (i=0;i<=millisec;i++);
}
```