```c
/*************************************************************************************
                     DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
                                   IMPERIAL COLLEGE LONDON

                        EE 3.19: Real Time Digital Signal Processing
                             Dr Paul Mitcheson and Daniel Harvey

                          LAB 2: Learning C and Sinewave Generation

                               ********* S I N E . C **********

                  Demonstrates outputing data from the DSK's audio port.
                Used for extending knowledge of C and using look up tables.

 *************************************************************************************
                  Updated for use on 6713 DSK by Danny Harvey: May-Aug 06/Dec 07/Oct 09
                  CCS V4 updates Sept 10
 ************************************************************************************/
/*
 *  Initialy this example uses the AIC23 codec module of the 6713 DSK Board Support
 *  Library to generate a 1KHz sine wave using a simple digital filter.
 *  You should modify the code to generate a sine of variable frequency.
 */
/**************************** Pre-processor statements ******************************/

//  Included so program can make use of DSP/BIOS configuration tool.
#include "dsp_bios_cfg.h"

/* The file dsk6713.h must be included in every program that uses the BSL.  This
   example also includes dsk6713_aic23.h because it uses the
   AIC23 codec module (audio interface). */
#include "dsk6713.h"
#include "dsk6713_aic23.h"

// math library (trig functions)
#include <math.h>

// Some functions to help with configuring hardware
#include "helper_functions_polling.h"


// PI defined here for use in your code
#define PI 3.141592653589793

//sine generation look up table size
#define SINE_TABLE_SIZE 256



/******************************* Global declarations ********************************/

/* Audio port configuration settings: these values set registers in the AIC23 audio
   interface to configure it. See TI doc SLWS106D 3-3 to 3-10 for more info. */
DSK6713_AIC23_Config Config = { \
             /**********************************************************************/
             /*   REGISTER             FUNCTION                     SETTINGS         */
             /**********************************************************************/\
    0x0017,  /* 0 LEFTINVOL  Left line input channel volume  0dB                   */\
    0x0017,  /* 1 RIGHTINVOL Right line input channel volume 0dB                   */\
    0x01f9,  /* 2 LEFTHPVOL  Left channel headphone volume   0dB                   */\
    0x01f9,  /* 3 RIGHTHPVOL Right channel headphone volume  0dB                   */\
    0x0011,  /* 4 ANAPATH    Analog audio path control       DAC on, Mic boost 20dB*/\
    0x0000,  /* 5 DIGPATH    Digital audio path control      All Filters off       */\
    0x0000,  /* 6 DPOWERDOWN Power down control              All Hardware on       */\
    0x004f,  /* 7 DIGIF      Digital audio interface format  32 bit                */\
    0x008d,  /* 8 SAMPLERATE Sample rate control             8 KHZ                 */\
    0x0001   /* 9 DIGACT     Digital interface activation    On                    */\
             /**********************************************************************/
};


// Codec handle:- a variable used to identify audio interface
```

```c
DSK6713_AIC23_CodecHandle H_Codec;

/* Sampling frequency in HZ. Must only be set to 8000, 16000, 24000
32000, 44100 (CD standard), 48000 or 96000  */
int sampling_freq = 8000;


// Array of data used by sinegen to generate sine. These are the initial values.
float y[3] = {0,0,0};


float a0 = 1.4142; // coefficients for difference equation
float b0 = 0.707;

// Holds the value of the current sample
float sample;

/* Left and right audio channel gain values, calculated to be less than signed 32 bit
 maximum value. */
Int32 L_Gain = 2100000000;
Int32 R_Gain = 2100000000;


/* Use this variable in your code to set the frequency of your sine wave
   be carefull that you do not set it above the current nyquist frequency! */
float sine_freq = 1000.0;

float table[SINE_TABLE_SIZE];
float x = 0;

/******************************** Function prototypes ********************************/
void init_hardware(void);
float sinegen(void);
void sine_init(void);
/********************************** Main routine ************************************/
void main()
{

    // initialize board and the audio port
    init_hardware();
    sine_init();
    // Loop endlessley generating a sine wave
   while(1)
    {
        // Calculate next sample
        sample = sinegen();
        /* Send a sample to the audio port if it is ready to transmit.
           Note: DSK6713_AIC23_write() returns false if the port if is not ready */

        //  send to LEFT channel (poll until ready)
        while (!DSK6713_AIC23_write(H_Codec, ((Int32)(sample * L_Gain))))
        {};
        // send same sample to RIGHT channel (poll until ready)
        while (!DSK6713_AIC23_write(H_Codec, ((Int32)(sample * R_Gain))))
        {};

        // Set the sampling frequency. This function updates the frequency only if it
        // has changed. Frequency set must be one of the supported sampling freq.
        set_samp_freq(&sampling_freq, Config, &H_Codec);

    }

}

/******************************** init_hardware() ************************************/
void init_hardware()
{
    // Initialize the board support library, must be called first
    DSK6713_init();

    // Start the codec using the settings defined above in config
    H_Codec = DSK6713_AIC23_openCodec(0, &Config);
```

```c
    /* Defines number of bits in word used by MSBSP for communications with AIC23
     NOTE: this must match the bit resolution set in in the AIC23 */
    MCBSP_FSETS(XCR1, XWDLEN1, 32BIT);

    /* Set the sampling frequency of the audio port. Must only be set to a supported
       frequency (8000/16000/24000/32000/44100/48000/96000) */

    DSK6713_AIC23_setFreq(H_Codec, get_sampling_handle(&sampling_freq));

}

/********************************** sinegen() ***************************************/

float sinegen(void)
{

    // x is global float variable
    float jump;                                         //gap to next sample in lookup table

    //0.5 deals with integer cast truncation
    jump = (SINE_TABLE_SIZE*sine_freq/sampling_freq)+0.5;
    x += jump;                                          //increment x by jump
    x = (int)x%SINE_TABLE_SIZE;                         //wrap round lookup table

                                                        //return (x);
     return(table[(int)x]);

}
void sine_init(void){
    int i;
    for(i=0; i<SINE_TABLE_SIZE; i++){
        table[i]=sin(i*2*PI/SINE_TABLE_SIZE);
    }
}
```