David Bozarth
SSU Engineering Science
Summer 2007

DSP Project – Mini Piano Keyboard – answers to questions

II. Procedures

2. a) The time duration between the first and 48th samples is (48 – 1) sample * (1 / 48000) s / sample = **979 μs**

b) With sampling rate 48000 sample / s, and one period being 48 samples, the period of the signal is **1 ms** and the frequency is **1 KHz**.

c) **R = f * N** … sampling rate (sample / s) = frequency (period / s) * samples per period (sample / period)

d) In order to play the signal for 10 s, change the outer for-loop limit from 5000 (ms) to 10000 (ms).

```
        /* Generate a 1KHz sine wave for 5 seconds */
        //for (msec = 0; msec < 5000; msec++)
        for (msec = 0; msec < 10000; msec++)
  {
```

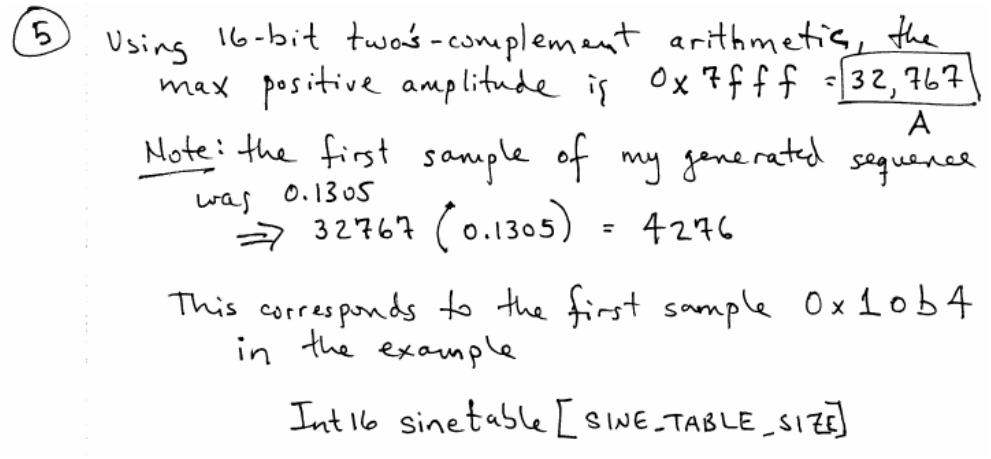
3. (no question)

4. Write a Matlab program to generate one period of samples.

```
function [X] = sin_samples(Rs, f, psi, A)
% Returns the vector of sample magnitudes of a sinusoidal time series.
% Rs is the sampling rate in samples per second.
% f is the signal frequency.
% psi is the phase of the signal with respect to the first sample.
% A is the signal amplitude.
%
Ts = 1 / Rs;        % Time duration of one sample.
T  = 1 / f;         % Signal period.
t  = [0:Ts:(T-Ts)]; % Independent variable.

% Obtain the return value.
X = A * sin(2*pi()*f*t + psi);

% end of program
```

Test:

```
>> sin_samples(48000, 1000, 0, 1)

ans =

  Columns 1 through 11

         0    0.1305    0.2588    0.3827    0.5000    0.6088    0.7071    0.7934    0.8660    0.9239
0.9659

  Columns 12 through 22

    0.9914    1.0000    0.9914    0.9659    0.9239    0.8660    0.7934    0.7071    0.6088    0.5000
0.3827

  Columns 23 through 33

    0.2588    0.1305    0.0000   -0.1305   -0.2588   -0.3827   -0.5000   -0.6088   -0.7071   -0.7934   -
0.8660

  Columns 34 through 44

   -0.9239   -0.9659   -0.9914   -1.0000   -0.9914   -0.9659   -0.9239   -0.8660   -0.7934   -0.7071   -
0.6088

  Columns 45 through 48

   -0.5000   -0.3827   -0.2588   -0.1305
```



6.

```
 >> S = sin_samples(48000, 1000, 0, 32767); format short e; S

S =

  Columns 1 through 8

           0  4.2770e+003  8.4807e+003  1.2539e+004  1.6383e+004  1.9947e+004  2.3170e+004  2.5996e+004

  Columns 9 through 16

  2.8377e+004  3.0273e+004  3.1650e+004  3.2487e+004  3.2767e+004  3.2487e+004  3.1650e+004  3.0273e+004

  Columns 17 through 24

  2.8377e+004  2.5996e+004  2.3170e+004  1.9947e+004  1.6384e+004  1.2539e+004  8.4807e+003  4.2770e+003

  Columns 25 through 32

  4.0128e-012 -4.2770e+003 -8.4807e+003 -1.2539e+004 -1.6383e+004 -1.9947e+004 -2.3170e+004 -2.5996e+004

  Columns 33 through 40

 -2.8377e+004 -3.0273e+004 -3.1650e+004 -3.2487e+004 -3.2767e+004 -3.2487e+004 -3.1650e+004 -3.0273e+004

  Columns 41 through 48

 -2.8377e+004 -2.5996e+004 -2.3170e+004 -1.9947e+004 -1.6384e+004 -1.2539e+004 -8.4807e+003 -4.2770e+003
```

** There is a one-bit difference in the step size as rendered by Matlab using A=32767 (4277) vs. using A=1 (4276).

The number of samples is 48, which matches exercise 2B.

```
>> stem(S, '.');
```

7.

```
>> Sx = signeddec2hex(S, 16);

hex_string =

0x0000,0x10B4,0x2120,0x30FB,0x3FFF,0x4DEB,0x5A81,0x658B,0x6ED9,0x7640,0x7BA2,0x7EE6,0x7FFF,0x7EE6,0x7BA2,0x76
40,0x6ED9,0x658B,0x5A81,0x4DEB,0x3FFF,0x30FB,0x2120,0x10B4,0x0000,0xEF4B,0xDEDF,0xCF04,0xC000,0xB214,0xA57E,0
x9A74,0x9126,0x89BF,0x845D,0x8119,0x8001,0x8119,0x845D,0x89BF,0x9126,0x9A74,0xA57E,0xB214,0xC000,0xCF04,0xDED
F,0xEF4B,
```

Very similar to Int16 sinetable[SINE_TABLE_SIZE].

(8) If a tone with frequency f lasts for t seconds, how many periods of the signal are played during that time span?

$$\left(f \ \frac{\text{cycles}}{s}\right) \cdot \left(t \ s\right) = \boxed{f \cdot t \ \text{cycles}}$$

The limit of loop_idx is DURATION * FREQ. Duration is a constant that can be adjusted for the processing platform – in this case, the DSK board. Modifying DURATION will lengthen or shorten the duration of each tone generated when a key is pressed.

9.

```
>> Rs = 48000; psi = 0; A = 2^15 - 1;
>> f = 262; C4 = sin_samples(Rs, f, psi, A); C4x = signeddec2hex(C4, 16);

hex_string =

0x0000,0x0463,0x08C5,0x0D25,0x1180,0x15D7,0x1A27,0x1E6F,0x22AD,0x26E2,0x2B0A,0x2F26,0x3333,0x3731,0x3B1F,0x3E
FA,0x42C3,0x4677,0x4A17,0x4DA0,0x5111,0x546B,0x57AB,0x5AD0,0x5DDA,0x60C8,0x6399,0x664B,0x68DF,0x6B54,0x6DA8,0
x6FDB,0x71EC,0x73DB,0x75A7,0x7750,0x78D5,0x7A35,0x7B71,0x7C87,0x7D78,0x7E43,0x7EE8,0x7F67,0x7FC0,0x7FF2,0x7FF
E,0x7FE3,0x7FA1,0x7F3A,0x7EAB,0x7DF7,0x7D1D,0x7C1D,0x7AF8,0x79AD,0x783F,0x76AB,0x74F5,0x731B,0x711E,0x6EFF,0x
6CBF,0x6A5E,0x67DD,0x653D,0x627E,0x5FA2,0x5CA9,0x5993,0x5663,0x5319,0x4FB6,0x4C3B,0x48A9,0x4501,0x4145,0x3D74
,0x3992,0x359E,0x3199,0x2D86,0x2965,0x2538,0x20FF,0x1CBD,0x1871,0x141F,0x0FC6,0x0B69,0x0708,0x02A5,0xFE41,0xF
9DE,0xF57D,0xF11F,0xECC5,0xE871,0xE424,0xDFE0,0xDBA5,0xD775,0xD352,0xCF3C,0xCB34,0xC73D,0xC356,0xBF82,0xBBC1,
0xB815,0xB47E,0xB0FE,0xAD96,0xAA47,0xA712,0xA3F7,0xA0F8,0x9E15,0x9B51,0x98AA,0x9623,0x93BB,0x9174,0x8F4E,0x8D
4A,0x8B69,0x89AB,0x8811,0x869A,0x8548,0x841B,0x8314,0x8232,0x8176,0x80E0,0x8070,0x8027,0x8004,0x8007,0x8032,0
x8082,0x80F9,0x8197,0x825A,0x8343,0x8452,0x8586,0x86DF,0x885C,0x89FD,0x8BC2,0x8DAA,0x8FB5,0x91E1,0x942E,0x969
C,0x9929,0x9BD6,0x9EA0,0xA188,0xA48C,0xA7AC,0xAAE7,0xAE3B,0xB1A7,0xB52C,0xB8C7,0xBC77,0xC03B,0xC413,0xC7FD,0x
CBF8,0xD002,0xD41B,0xD841,0xDC73,0xE0B0,0xE4F6,0xE944,0xED99,0xF1F4,0xF653,0xFAB5,

>> f = 330; E4 = sin_samples(Rs, f, psi, A); E4x = signeddec2hex(E4, 16);

hex_string =

0x0000,0x0586,0x0B0B,0x108A,0x1601,0x1B6E,0x20CD,0x261D,0x2B5B,0x3084,0x3596,0x3A8E,0x3F6A,0x4428,0x48C6,0x4D
40,0x5196,0x55C5,0x59CB,0x5DA6,0x6154,0x64D3,0x6823,0x6B41,0x6E2B,0x70E1,0x7361,0x75AA,0x77BB,0x7993,0x7B30,0
x7C93,0x7DBA,0x7EA5,0x7F54,0x7FC6,0x7FFA,0x7FF2,0x7FAD,0x7F2A,0x7E6B,0x7D70,0x7C38,0x7AC5,0x7918,0x7731,0x751
0,0x72B8,0x7029,0x6D65,0x6A6C,0x6741,0x63E4,0x6057,0x5C9D,0x58B6,0x54A5,0x506B,0x4C0B,0x4787,0x42E0,0x3E1A,0x
3936,0x3436,0x2F1E,0x29EF,0x24AC,0x1F58,0x19F4,0x1484,0x0F0B,0x098A,0x0405,0xFE7D,0xF8F7,0xF374,0xEDF7,0xE882
,0xE319,0xDDBD,0xD872,0xD33A,0xCE17,0xC90C,0xC41B,0xBF47,0xBA91,0xB5FD,0xB18C,0xAD41,0xA91D,0xA523,0xA154,0x9
DB2,0x9A40,0x96FD,0x93ED,0x9111,0x8E6A,0x8BF8,0x89BF,0x87BD,0x85F6,0x8468,0x8316,0x81FF,0x8124,0x8086,0x8025,
0x8001,0x801A,0x8070,0x8103,0x81D3,0x82DE,0x8426,0x85A9,0x8766,0x895D,0x8B8D,0x8DF4,0x9092,0x9364,0x966B,0x99
A4,0x9D0E,0xA0A8,0xA46E,0xA861,0xAC7D,0xB0C2,0xB52C,0xB9BA,0xBE69,0xC338,0xC824,0xCD2A,0xD249,0xD77D,0xDCC5,0
xE21E,0xE785,0xECF8,0xF274,0xF7F6,

>> f = 392; G4 = sin_samples(Rs, f, psi, A); G4x = signeddec2hex(G4, 16);

hex_string =

0x0000,0x0690,0x0D1C,0x13A0,0x1A16,0x207A,0x26C9,0x2CFE,0x3314,0x3908,0x3ED5,0x4478,0x49ED,0x4F30,0x543E,0x59
12,0x5DAB,0x6205,0x661D,0x69F0,0x6D7B,0x70BD,0x73B3,0x765A,0x78B2,0x7AB9,0x7C6D,0x7DCD,0x7ED9,0x7F8E,0x7FEE,0
x7FF8,0x7FAB,0x7F09,0x7E11,0x7CC4,0x7B22,0x792E,0x76E8,0x7452,0x716E,0x6E3D,0x6AC2,0x66FF,0x62F6,0x5EAB,0x5A2
0,0x5558,0x5057,0x4B20,0x45B6,0x401D,0x3A59,0x346D,0x2E5E,0x2830,0x21E7,0x1B87,0x1514,0x0E94,0x0809,0x0179,0x
FAE8,0xF45A,0xEDD5,0xE75B,0xE0F2,0xDA9E,0xD464,0xCE46,0xC84A,0xC274,0xBCC7,0xB748,0xB1F9,0xACDF,0xA7FD,0xA356
,0x9EEE,0x9AC7,0x96E5,0x9349,0x8FF7,0x8CF0,0x8A36,0x87CC,0x85B3,0x83EC,0x8279,0x815B,0x8092,0x801F,0x8002,0x8
03B,0x80CA,0x81AF,0x82E9,0x8478,0x865A,0x888D,0x8B11,0x8DE4,0x9104,0x946F,0x9822,0x9C1B,0xA058,0xA4D5,0xA98F,
0xAE83,0xB3AF,0xB90E,0xBE9D,0xC458,0xCA3B,0xD042,0xD669,0xDCAD,0xE308,0xE977,0xEFF5,0xF67D,

>> f = 494; B4 = sin_samples(Rs, f, psi, A); B4x = signeddec2hex(B4, 16);

hex_string =

0x0000,0x0845,0x1081,0x18AC,0x20BD,0x28AA,0x306C,0x37FA,0x3F4C,0x465B,0x4D1E,0x538F,0x59A6,0x5F5D,0x64AE,0x69
94,0x6E08,0x7207,0x758C,0x7893,0x7B19,0x7D1B,0x7E97,0x7F8C,0x7FF9,0x7FDC,0x7F37,0x7E09,0x7C55,0x7A1B,0x775F,0
x7423,0x706B,0x6C3B,0x6797,0x6284,0x5D07,0x5727,0x50EA,0x4A56,0x4372,0x3C47,0x34DA,0x2D36,0x2561,0x1D64,0x154
7,0x0D14,0x04D2,0xFC8C,0xF449,0xEC13,0xE3F2,0xDBEF,0xD413,0xCC66,0xC4F0,0xBDB9,0xB6C9,0xB027,0xA9DB,0xA3EB,0x
9E5D,0x9938,0x9481,0x903D,0x8C71,0x8920,0x864F,0x8400,0x8235,0x80F1,0x8035,0x8002,0x8057,0x8136,0x829B,0x8488
,0x86F8,0x89EA,0x8D5A,0x9145,0x95A6,0x9A79,0x9FB9,0xA55F,0xAB67,0xB1C9,0xB87F,0xBF82,0xC6C9,0xCE4E,0xD608,0xD
DEF,0xE5FA,0xEE21,0xF65B,
```

10.

```
/*
 *  ======== tone.c ========
 *
 *  20080218: This is the instructor version 'keyboard.c' copied over 'tone.c'
 *  20080218: Modified to use 4 tones forming a Cmaj7 chord. D.Bozarth
 *
 *  This example uses the AIC23 codec module of the 5510 DSK Board Support
 *  Library to generate a 1KHz sine wave on the audio outputs for 5 seconds.
 *  The sine wave data is pre-calculated in an array called sinetable.  The
 *  codec operates at 48KHz by default.  Since the sine wave table has 48
 *  entries per period, each pass through the inner loop takes 1 millisecond.
 *  5000 passes through the inner loop takes 5 seconds.
 *
 *  Please see the 5510 DSK help file under Software/Examples for more
 *  detailed information.
 */

/*
 *  DSP/BIOS is configured using the DSP/BIOS configuration tool.  Settings
 *  for this example are stored in a configuration file called tone.cdb.  At
 *  compile time, Code Composer will auto-generate DSP/BIOS related files
 *  based on these settings.  A header file called tonecfg.h contains the
 *  results of the autogeneration and must be included for proper operation.
 *  The name of the file is taken from tone.cdb and adding cfg.h.
 */
#include "tonecfg.h"

/*
 *  The 5510 DSK Board Support Library is divided into several modules, each
 *  of which has its own include file.  The file dsk5510.h must be included
 *  in every program that uses the BSL.  This example also includes
 *  dsk5510_aic23.h because it uses the AIC23 codec module.
 */
#include "dsk5510.h"
#include "dsk5510_aic23.h"
#include "dsk5510_led.h"
#include "dsk5510_dip.h"

/* Sampling rate */
#define SAMPLE_RATE 48000
#define FREQ_0 262   // C4
#define FREQ_1 330   // E4
#define FREQ_2 392   // G4
#define FREQ_3 494   // B4

/* freq = 262 Hz */
#define SINE_TABLE_SIZE_0 (Uint16)SAMPLE_RATE/FREQ_0
/* freq = 330 Hz */
#define SINE_TABLE_SIZE_1 (Uint16)SAMPLE_RATE/FREQ_1
/* freq = 392 Hz */
#define SINE_TABLE_SIZE_2 (Uint16)SAMPLE_RATE/FREQ_2
/* freq = 494 Hz */
#define SINE_TABLE_SIZE_3 (Uint16)SAMPLE_RATE/FREQ_3

/* there are 4 DIP switches */
#define N_switch 4
/* the duration of each key push is 1 second */
#define DURATION 0.4


/* Codec configuration settings */
DSK5510_AIC23_Config config = {
    0x0017, // 0 DSK5510_AIC23_LEFTINVOL  Left line input channel volume
    0x0017, // 1 DSK5510_AIC23_RIGHTINVOL Right line input channel volume
    0x00d8, // 2 DSK5510_AIC23_LEFTHPVOL  Left channel headphone volume
    0x00d8, // 3 DSK5510_AIC23_RIGHTHPVOL Right channel headphone volume
    0x0011, // 4 DSK5510_AIC23_ANAPATH    Analog audio path control
    0x0000, // 5 DSK5510_AIC23_DIGPATH    Digital audio path control
    0x0000, // 6 DSK5510_AIC23_POWERDOWN  Power down control
    0x0043, // 7 DSK5510_AIC23_DIGIF      Digital audio interface format
    0x0081, // 8 DSK5510_AIC23_SAMPLERATE Sample rate control
    0x0001  // 9 DSK5510_AIC23_DIGACT     Digital interface activation
};

/* Pre-generated sine wave data, 16-bit signed samples */
Int16 sinetable_0[SINE_TABLE_SIZE_0] = {

0x0000,0x0463,0x08C5,0x0D25,0x1181,0x15D7,0x1A27,0x1E6F,0x22AE,0x26E2,0x2B0B,0x2F26,0x3334,0x3732,0x3B1F,0x3E
FB,0x42C3,0x4678,0x4A17,0x4DA0,0x5112,0x546B,0x57AB,0x5AD1,0x5DDB,0x60C9,0x6399,0x664C,0x68E0,0x6B54,0x6DA8,0
x6FDB,0x71ED,0x73DC,0x75A8,0x7751,0x78D6,0x7A36,0x7B72,0x7C88,0x7D79,0x7E44,0x7EE9,0x7F68,0x7FC1,0x7FF3,0x7FF
F,0x7FE4,0x7FA2,0x7F3B,0x7EAC,0x7DF8,0x7D1E,0x7C1E,0x7AF9,0x79AE,0x7840,0x76AC,0x74F6,0x731B,0x711F,0x6F00,0x
6CC0,0x6A5F,0x67DE,0x653E,0x627F,0x5FA2,0x5CA9,0x5994,0x5664,0x531A,0x4FB7,0x4C3C,0x48AA,0x4502,0x4145,0x3D75
,0x3992,0x359E,0x319A,0x2D86,0x2965,0x2538,0x20FF,0x1CBD,0x1872,0x141F,0x0FC6,0x0B69,0x0708,0x02A5,0xFE41,0xF
9DE,0xF57D,0xF11F,0xECC5,0xE871,0xE424,0xDFE0,0xDBA5,0xD775,0xD351,0xCF3B,0xCB34,0xC73C,0xC356,0xBF81,0xBBC1,
0xB814,0xB47E,0xB0FE,0xAD96,0xAA46,0xA711,0xA3F6,0xA0F7,0x9E15,0x9B50,0x98A9,0x9622,0x93BA,0x9173,0x8F4D,0x8D
```

```
4A,0x8B68,0x89AA,0x8810,0x8699,0x8547,0x841A,0x8313,0x8231,0x8175,0x80DF,0x806F,0x8026,0x8003,0x8006,0x8031,0
x8081,0x80F8,0x8196,0x8259,0x8342,0x8451,0x8585,0x86DE,0x885B,0x89FD,0x8BC1,0x8DA9,0x8FB4,0x91E0,0x942D,0x969
B,0x9928,0x9BD5,0x9E9F,0xA187,0xA48C,0xA7AC,0xAAE6,0xAE3A,0xB1A7,0xB52B,0xB8C6,0xBC76,0xC03B,0xC413,0xC7FD,0x
CBF7,0xD002,0xD41B,0xD841,0xDC73,0xE0AF,0xE4F6,0xE944,0xED99,0xF1F4,0xF653,0xFAB4,
};

Int16 sinetable_1[SINE_TABLE_SIZE_1] = {
       0x0000,0x0586,0x0B0B,0x108A,0x1601,0x1B6E,0x20CD,0x261D,0x2B5B,0x3084,0x3596,0x3A8E,0x3F6A,0x4428,0x4
8C6,0x4D40,0x5196,0x55C5,0x59CB,0x5DA6,0x6154,0x64D3,0x6823,0x6B41,0x6E2B,0x70E1,0x7361,0x75AA,0x77BB,0x7993,
0x7B30,0x7C93,0x7DBA,0x7EA5,0x7F54,0x7FC6,0x7FFA,0x7FF2,0x7FAD,0x7F2A,0x7E6B,0x7D70,0x7C38,0x7AC5,0x7918,0x77
31,0x7510,0x72B8,0x7029,0x6D65,0x6A6C,0x6741,0x63E4,0x6057,0x5C9D,0x58B6,0x54A5,0x506B,0x4C0B,0x4787,0x42E0,0
x3E1A,0x3936,0x3436,0x2F1E,0x29EF,0x24AC,0x1F58,0x19F4,0x1484,0x0F0B,0x098A,0x0405,0xFE7D,0xF8F7,0xF374,0xEDF
7,0xE882,0xE319,0xDDBD,0xD872,0xD33A,0xCE17,0xC90C,0xC41B,0xBF47,0xBA91,0xB5FD,0xB18C,0xAD41,0xA91D,0xA523,0x
A154,0x9DB2,0x9A40,0x96FD,0x93ED,0x9111,0x8E6A,0x8BF8,0x89BF,0x87BD,0x85F6,0x8468,0x8316,0x81FF,0x8124,0x8086
,0x8025,0x8001,0x801A,0x8070,0x8103,0x81D3,0x82DE,0x8426,0x85A9,0x8766,0x895D,0x8B8D,0x8DF4,0x9092,0x9364,0x9
66B,0x99A4,0x9D0E,0xA0A8,0xA46E,0xA861,0xAC7D,0xB0C2,0xB52C,0xB9BA,0xBE69,0xC338,0xC824,0xCD2A,0xD249,0xD77D,
0xDCC5,0xE21E,0xE785,0xECF8,0xF274,0xF7F6
};

Int16 sinetable_2[SINE_TABLE_SIZE_2] = {

0x0000,0x0690,0x0D1C,0x13A0,0x1A16,0x207A,0x26C9,0x2CFE,0x3314,0x3908,0x3ED5,0x4478,0x49ED,0x4F30,0x543E,0x59
12,0x5DAB,0x6205,0x661D,0x69F0,0x6D7B,0x70BD,0x73B3,0x765A,0x78B2,0x7AB9,0x7C6D,0x7DCD,0x7ED9,0x7F8E,0x7FEE,0
x7FF8,0x7FAB,0x7F09,0x7E11,0x7CC4,0x7B22,0x792E,0x76E8,0x7452,0x716E,0x6E3D,0x6AC2,0x66FF,0x62F6,0x5EAB,0x5A2
0,0x5558,0x5057,0x4B20,0x45B6,0x401D,0x3A59,0x346D,0x2E5E,0x2830,0x21E7,0x1B87,0x1514,0x0E94,0x0809,0x0179,0x
FAE8,0xF45A,0xEDD5,0xE75B,0xE0F2,0xDA9E,0xD464,0xCE46,0xC84A,0xC274,0xBCC7,0xB748,0xB1F9,0xACDF,0xA7FD,0xA356
,0x9EEE,0x9AC7,0x96E5,0x9349,0x8FF7,0x8CF0,0x8A36,0x87CC,0x85B3,0x83EC,0x8279,0x815B,0x8092,0x801F,0x8002,0x8
03B,0x80CA,0x81AF,0x82E9,0x8478,0x865A,0x888D,0x8B11,0x8DE4,0x9104,0x946F,0x9822,0x9C1B,0xA058,0xA4D5,0xA98F,
0xAE83,0xB3AF,0xB90E,0xBE9D,0xC458,0xCA3B,0xD042,0xD669,0xDCAD,0xE308,0xE977,0xEFF5,0xF67D,
};

Int16 sinetable_3[SINE_TABLE_SIZE_3] = {

0x0000,0x0845,0x1081,0x18AC,0x20BD,0x28AA,0x306C,0x37FA,0x3F4C,0x465B,0x4D1E,0x538F,0x59A6,0x5F5D,0x64AE,0x69
94,0x6E08,0x7207,0x758C,0x7893,0x7B19,0x7D1B,0x7E97,0x7F8C,0x7FF9,0x7FDC,0x7F37,0x7E09,0x7C55,0x7A1B,0x775F,0
x7423,0x706B,0x6C3B,0x6797,0x6284,0x5D07,0x5727,0x50EA,0x4A56,0x4372,0x3C47,0x34DA,0x2D36,0x2561,0x1D64,0x154
7,0x0D14,0x04D2,0xFC8C,0xF449,0xEC13,0xE3F2,0xDBEF,0xD413,0xCC66,0xC4F0,0xBDB9,0xB6C9,0xB027,0xA9DB,0xA3EB,0x
9E5D,0x9938,0x9481,0x903D,0x8C71,0x8920,0x864F,0x8400,0x8235,0x80F1,0x8035,0x8002,0x8057,0x8136,0x829B,0x8488
,0x86F8,0x89EA,0x8D5A,0x9145,0x95A6,0x9A79,0x9FB9,0xA55F,0xAB67,0xB1C9,0xB87F,0xBF82,0xC6C9,0xCE4E,0xD608,0xD
DEF,0xE5FA,0xEE21,0xF65B,
};



/*
 *  main() - Main code routine, initializes BSL and generates tone
 */

void main()
{
    DSK5510_AIC23_CodecHandle hCodec;
    Int16 loop_idx, sample;
       Uint8 DIP_flag;


    /* Initialize the board support library, must be called first */
    DSK5510_init();

       // Initialize LED and DIP switches
       DSK5510_LED_init();
       DSK5510_DIP_init();

    /* Start the codec */
    hCodec = DSK5510_AIC23_openCodec(0, &config);



    /* Generate a 1KHz sine wave for 5 seconds */

       while (1)
       {
              if     (DSK5510_DIP_get(0) == 0)
                     DIP_flag = 0;
              else if (DSK5510_DIP_get(1) == 0)
                     DIP_flag = 1;
              else if (DSK5510_DIP_get(2) == 0)
                     DIP_flag = 2;
              else if (DSK5510_DIP_get(3) == 0)
                     DIP_flag = 3;
              else
                     DIP_flag = 5;

              switch (DIP_flag)
              {
                     case 0:
                            DSK5510_LED_on(0);
                            for (loop_idx = 0; loop_idx < DURATION*FREQ_0; loop_idx++)
                        {
```

```
                        for (sample = 0; sample < SINE_TABLE_SIZE_0; sample++)
                        {
                            /* Send a sample to the left channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_0[sample]));

                            /* Send a sample to the right channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_0[sample]));
                        }
                    }

                        DSK5510_LED_off(0);

                    break;
                    case 1:
                        DSK5510_LED_on(1);

                        for (loop_idx = 0; loop_idx < DURATION*FREQ_1; loop_idx++)
                    {
                        for (sample = 0; sample < SINE_TABLE_SIZE_1; sample++)
                        {
                            /* Send a sample to the left channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_1[sample]));

                            /* Send a sample to the right channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_1[sample]));
                        }
                    }

                        DSK5510_LED_off(1);

                    break;
                    case 2:
                        DSK5510_LED_on(2);

                        for (loop_idx = 0; loop_idx < DURATION*FREQ_2; loop_idx++)
                    {
                        for (sample = 0; sample < SINE_TABLE_SIZE_2; sample++)
                        {
                            /* Send a sample to the left channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_2[sample]));

                            /* Send a sample to the right channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_2[sample]));
                        }
                    }

                        DSK5510_LED_off(2);

                    break;
                    case 3:
                        DSK5510_LED_on(3);

                        for (loop_idx = 0; loop_idx < DURATION*FREQ_3; loop_idx++)
                    {
                        for (sample = 0; sample < SINE_TABLE_SIZE_3; sample++)
                        {
                            /* Send a sample to the left channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_3[sample]));

                            /* Send a sample to the right channel */
                            while (!DSK5510_AIC23_write16(hCodec, sinetable_3[sample]));
                        }
                    }

                        DSK5510_LED_off(3);

                    break;
                    default:
                        printf("default\n");

                    break;
                }

            }


    /* Close the codec */
/*    DSK5510_AIC23_closeCodec(hCodec);*/
}
```