

525.446  
DSP Hardware Laboratory  
Assignment #2

## Fixed Point Direct Digital Synthesizer

### Introduction

For this assignment you will create a direct digital synthesizer (DDS) using the TIC6713 DSK. A DDS is a method for digitally creating an accurate digital representation of an arbitrary periodic waveform (our arbitrary waveform will be a sine wave). The digital signal is then reconstructed with a high-speed digital to analog (D/A) converter. Because the waveform is synthesized digitally, it can easily support fine tuning resolution as well as no settling time constraints. For this reason, the DDS technique is a useful component for both frequency and phase modulation.

A concise introduction to DDS can be found in the IEEE article *Direct Digital Synthesis: A Tool for Periodic Wave Generation* available on the class website. A brief overview will be included here.

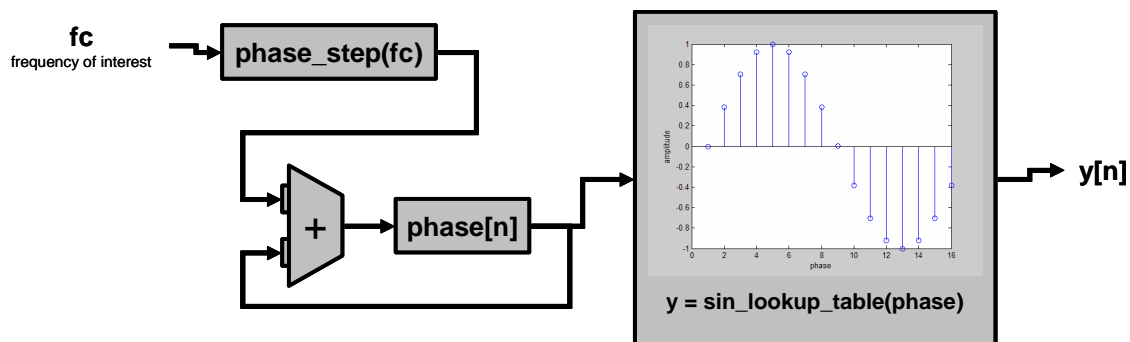


Figure 1: Block Diagram of Typical DDS System

Above is a block diagram of a typical DDS system. In general, a DDS operates by generating each sample sequentially from a lookup table that contains one cycle of the waveform of interest. Different frequencies are generated by stepping through the lookup table at different rates. To do this, the synthesizer maintains a phase value in a register following the phase accumulator (this is the value  $\text{phase}[n]$  above). For each sample, this phase value is updated by adding a phase step to it, where the step is dependent on the desired frequency of the output waveform. At each sample, the current phase value is used as an index into a stored lookup table (generally the number of elements in the table is less than can be indexed by the phase accumulator, so just the appropriate number of top bits of the phase accumulator are used). The output of the lookup table is passed through a D/A to create the synthesized wave. Note that when generating a sine wave, the lookup table generally only has one quarter of the sine wave stored to minimize the amount of memory used. Other, more advanced techniques exist to minimize the amount of memory required to store the waveform.

### Simulation

Before implementing the DDS on the TIC6713 DSK, develop a MATLAB simulation called `dds_sim.m`. This simulation should allow the frequency, number of output points, sample rate, and number of points in the lookup table to be varied. As a start, use the following code snippet:

```
%DDS_SIM Script to simulate a direct digital synthesizer (DDS).  
% DDS_SIM simulates a DDS, with the frequency (f), number of output  
% points (n), sample rate (fs), and look-up table length (N) all
```

```

% variable settings.

f      = 440;
n      = 48000*10;

fs     = 48000;
N      = 32;      % phase accumulator length (num bits)
P      =          % table length (num bits)
M      = 16;      % word length (num bits)

.
.
%generate table and output waveform "y" here
.
.

%plot the results
fInd = linspace(0,fs-fs/n,n);
figure
win = hann(n);
win = win(:);
plot(fInd, 20*log10(abs(fft(y.*win))), 'r');
xlabel('Frequency (Hz)');
ylabel('dB')

```

The code above also performs a discrete Fourier transform (DFT) of the synthesized waveform (note that the frequency resolution of the transform is governed by the number of synthesized points generated).

Perform the following experiment:

From your simulation, determine the number of elements in the lookup table required to get better than 50 dB SFDR. Does this make sense, given formula (9) in *Direct Digital Synthesis: A Tool for Periodic Wave Generation*?

Once you have determined the number of elements required, create a plot of the output spectrum for a synthesized wave at 100 and 3000 Hz (you will need to overlay these plots with actual results below). Note the spurs in the output. These are due to harmonics (created by the discrete steps between points from the lookup table as well as the rounding of the phase index). For more information on these effects, see the reference given above.

### DSP Implementation

Implement the DDS using polling data transfers.

Generally a DDS is implemented using a fixed point implementation (that is, **phase[n]** is stored as a fixed point number). Thus, despite the fact that the C6713 is a floating point processor, we will maintain the phase value as a fixed point number similar to the implementation described in *Direct Digital Synthesis: A Tool for Periodic Wave Generation*.

Use the DIP switches to switch between four output frequencies (we will use the DDS for future labs at various frequencies). Light LEDs 0 and 1 to correspond to which switches are pressed:

- DIP Switches 1/0 = 00, 100 Hz
- DIP Switches 1/0 = 01, 3000 Hz
- DIP Switches 1/0 = 10, 440 Hz
- DIP Switches 1/0 = 11, 440.5 Hz

The sine waveform should be generated from the phase accumulator in two ways:

1. The “real” DDS way, using a sine lookup table created at the beginning of the program.
2. By calling the sine function for the stored phase value each time a sample is sent to the output function.

When using the sine lookup table method, create the table prior to starting the waveform generation. The sine table should be the shortest length required (power of 2) to get 50 dB SFDR.

Light LED3 when calling the output\_sample() function. Ensure that you are not doing any unnecessary calculations when the LED is lit. For example, do not recalculate the phase step each time through your input/output loop.

Perform the following three simple experiments:

1. Record a 1 second sample of two frequencies (100 & 3000 Hz) and plot the magnitude of the DFT in dB. Estimate the SFDR (signal to noise-and-distortion – this is just the difference between the synthesized frequency peak and the next highest peak) for each set of recorded data. Overlay the recorded results with simulation results generated in MATLAB. Normalize the plots so that the peaks are at the same value, 0 dB.
2. Create two frequencies about .5Hz apart. Record, analyze, and show if it is possible with your implementation. From the reference, what is the smallest difference our implementation should be able to create?
3. Using LED3, calculate how long it takes to generate a sample using the sine table. Also calculate how long it takes to generate a sample using the sine function. If you want to come up with a clever way to compare times without using a scope, feel free.

Note that you can use the MATLAB function `wavrecord` to record data from the sound card for processing in MATLAB (use “help wavrecord” for more information).

### Grading

To allow us to grade your implementation, please do the following:

- Turn in a hardcopy of the MATLAB simulation and the C code for the DSP implementation of the DDS. Make sure you have answered the question in the **Simulation** and **DSP Implementation** sections.
- Turn in a hardcopy of the plots requested in “Simulation” and “DSP Implementation” above. There should be three plots.
- Demonstrate the operation of the DDS to an instructor.

**525.446**

**DSP Hardware Laboratory**

**Assignment #2, Grade Sheet**

**Fixed Point Direct Digital Synthesizer**

**DDS, Lab2 ( \_\_\_\_/100)**

( \_\_\_\_/10) **MATLAB simulation code**

( \_\_\_\_/10) **Correct table length chosen for SFDR. "Makes sense" answered.**

( \_\_\_\_/15) **DDS implementation**

( \_\_\_\_/05) **Output chosen from DIP switches, LEDs lit appropriately**

( \_\_\_\_/10) **Output created from sin() and DDS**

( \_\_\_\_/10) **Sine look-up table created in DDS**

( \_\_\_\_/10) **Analysis #1 (100 Hz and 3000 Hz, overlaid w/ MATLAB, SFDR estimated)**

( \_\_\_\_/10) **Analysis #2 (2 frequencies separated by .5 Hz, minimum separation possible)**

( \_\_\_\_/10) **Analysis #3 (Difference in timing)**

( \_\_\_\_/10) **Well written, well documented code**