CMPE 121L: Microprocessor System Design Lab

Fall 2016

Lab Exercise 2

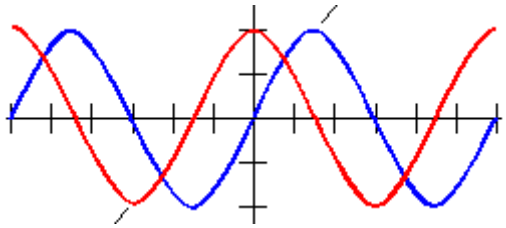
Check-off Due in lab section, week of October 17, 2016

Report due: Within 3 days after checkoff due

The purpose of this exercise is to learn how to use the DMA Hub in the PSoC 5 system. In the first part you will design a dual-channel waveform generator with controllable phase. In the second part you will use the DMA hub to perform a memory-to-memory block transfer with byte transposition, and contrast this with using a software loop to perform the same block transfer.

Part 1a: Dual-Channel Waveform Generator

This is an extension of Example 3 in the Cypress Application Note *An Introduction to DMA* (link posted on the class Web site). You will use a lookup table stored in flash memory to create two sine waves on two output pins with a variable phase difference between them, as shown below.



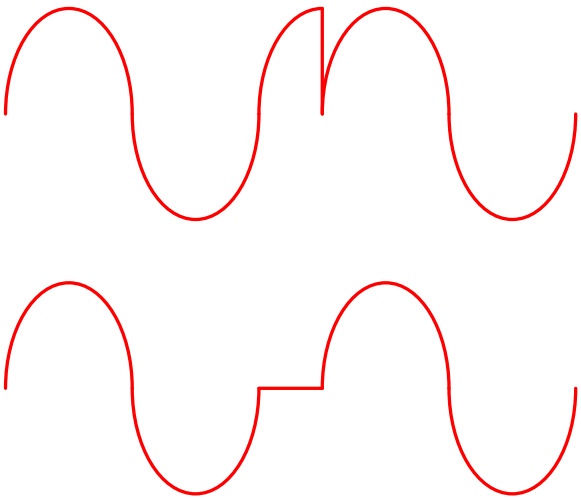
The blue wave can be generated by following Example 3. The red wave has the same amplitude and frequency, but you should be able to vary its phase with respect to the blue wave from 0 to 360 degrees by turning the potentiometer supplied in the parts kit. The phase shift value in degrees should also be displayed on the 16x2 LCD display.

Here are some hints on how to proceed.

1. First, read the application note carefully and try out some of the examples on the board. You should also read the appendices, which describe how to set up the DMA channels and program them.
2. You can use the same 128-entry lookup table and 8-bit DACs to generate the sine waves. A single lookup table can drive both waveform outputs, but you will need a separate DMA channel for each. The key is to use the Transaction Descriptors cleverly.
3. Avoid using periodic interrupts, as they consume CPU cycles.

Part 1b: Dual-Channel Waveform Generator with Controlled Phase Change

This is a refinement of Part 1a. The objective is to change the phase of the red sine wave only at its zero-crossing points. That is, if the potentiometer setting is changed, the system should wait until a zero-crossing point (you may choose any zero-crossing point) to shift the waveform. This avoids glitches in the output. This is illustrated in the figure bellow:



In the example above, the phase of the sine wave is being shifted by 90 degrees. In the top wave, the phase shift occurred at a non zero-crossing point, resulting in a glitch. In the bottom wave, there is no such glitch, which is the desired behavior.

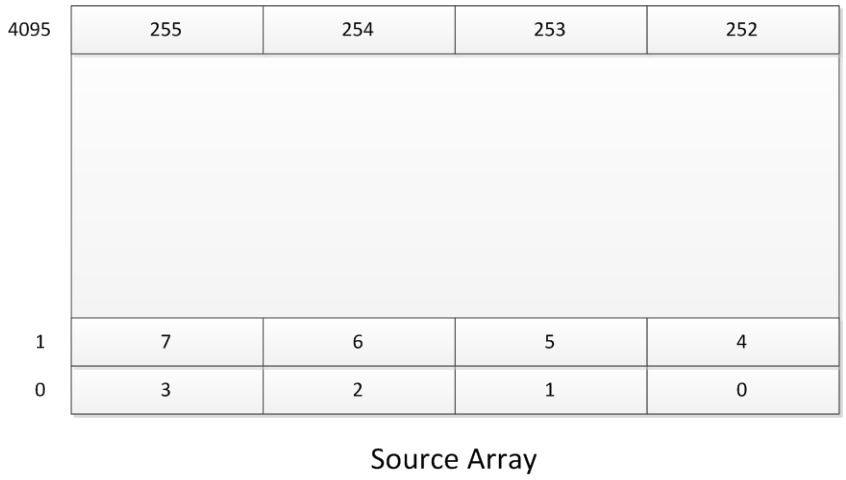
You can switch the phase at any zero-crossing point. Note that a phase change of +x degrees is the same as a phase change of (360-x) degrees, and the exact cycle in which the phase change occurs is not important.

If you set the frequency of the wave very low (under 100 Hz), you can actually see the phase changes on the oscilloscope and confirm that they occur at zero-crossing points.

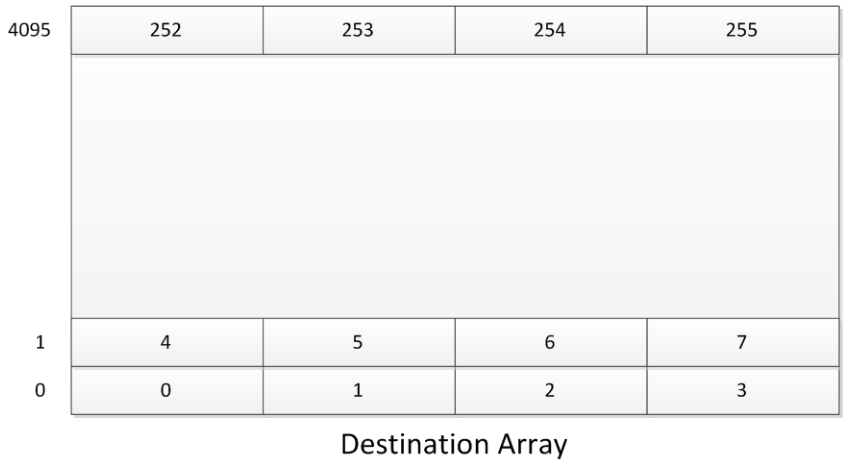
You can use interrupts for this part, but you should disable them when you don't need them (an interrupt at every zero-crossing point when the phase is not changing will consume too many CPU cycles).

Part 2: Memory-to-Memory Block Transfer

In this second part, you will use the DMA hub to transfer a block of data from one part of the RAM to another.



Source Array



Destination Array

The format of the block of data is shown above. The size of the block is 4096x4 = 16,384 bytes, and the individual byte locations are filled with an increasing pattern. When the program is run, it should copy the values into a destination array in the format shown.

Steps:

1. Write a program to create the source array in RAM and fill it with an increasing pattern of byte values (modulo 256).
2. Clear the destination array by filling with zeroes.
3. Configure a DMA channel to move the data in byte-transposed order to the destination array, and interrupt the main program when the DMA transfer is complete.
4. In the ISR, use a software loop to compare the source and destination arrays (after transposing the bytes) and verify that the transfer was done correctly.
5. Measure the time taken for the block transfer using a timer block as accurately as possible.

6. Now use a software loop to perform the same transfer done by the DMA Hub, and use a timer to measure the time taken. Compare the two times and determine the speedup for the DMA transfer over the program-controlled transfer. Explain why they are different.

What to submit?

- Schematics and code for all parts
- Descriptions of your designs
- Results
- You should also demonstrate the working designs to your TA.