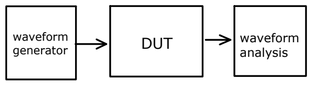
Generating Test Signals

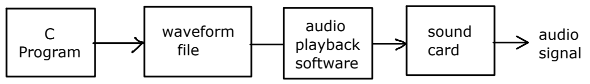
Introduction

Testing communication systems often requires generating test waveforms that are input to a system or component (the DUT or "Device Under Test") in order to see how it responds.



In this lab you will first write a C program that computes the voltages of a test waveform. When these voltages are supplied to a digital-to-analog (D/A) converter (DAC) they will create an analog test signal.

The regularly-spaced (in time) sample values generated by running your program will be written to a disk file. You will then "play" the file through a PC's analog speaker output. You will observe the resulting analog test signal on an oscilloscope.



The technique demonstrated in this lab is useful because it can be used to test any device or system, even those for which no specialized test equipment exists.

In this lab we will be using a PC's sound card as the D/A converter. This works well for audio-frequency signals. In later labs we will use an Arbitrary Waveform Generator (AWG).

Procedure

Use a text editor such as Notepad or Notepad++ to create and edit a C-language program file. When working in the lab, it is recommended that you save files on your own network drive (typically `H:`) so that you don't lose it if the PC reboots.

Start a command prompt (also known as a command interpreter or "shell") (`All Programs -> Accessories-> Command Prompt`). The command interpreter will execute the commands you type after you press Enter.

Type the command `h:` to change to the drive where your files are stored (assuming they are in the 'h' drive). Use the command `cd` *folder-name* to Change Directory to the folder (directory) where your C file is stored. For example, `cd ELEX3525/lab1`. Avoid using spaces or special characters in the file and folder names.

Type the command:

```
tcc -run filename.c >lab1.out
```

to compile and run your program using the Tiny C Compiler (`tcc`). Note the > character. It causes the standard output of your program to be written to the file whose name is given after > (`lab1.out` in this example) rather than the console. If there are any error messages, correct the errors in your program and repeat the tcc command.

Audacity is an audio editor that can also output waveforms through the PC's audio output DAC. Run `All Programs -> Audacity` to start the program. Since your waveform file contains only the sample values and is not in any standard audio file format, use `File -> Import -> Raw Data` to read the waveform sample values from the file. You will have to specify the file and format. Choose the same format, number of channels and sampling rate as you assumed when creating the audio file (see below).

Check the displayed waveform to make sure the waveform polarity and values are correct. Take a screen capture that shows the time scale, the sample rate, sample format and the waveform (use `All Programs -> Accessories -> Snipping Tool`) and save it to your `H:` drive for use in your report.

Select `Transport -> Loop Play` to repeatedly output the file to the PC's audio output.

Connect an audio cable to the headphone output on the front of the computer case. Clip the 'scope probe to the contacts on the plug on the other end of the cable. Adjust the 'scope vertical gain, horizontal scale and trigger levels to view the waveform (it may not be stable).

Stop Audacity playback and switch the 'scope to single-sweep mode. Play the waveform once to obtain a display of a single waveform. Set up RS-232 decoding as described below to check your waveform.

Have the instructor check and record that your waveform as displayed on the 'scope is correct. Capture the 'scope screen to a flash drive to include in your report.

Waveform Specifications

Your C program should output a waveform composed of a sequence of pulses, each pulse having a duration of 833 μs (1200 pulses/second).

Each pulse should have a level of either the maximum or the minimum allowed sample value. The actual output voltages can't be determined since the audio DAC output levels and and amplifier gain are not calibrated. For 8-bit signed values the maximum value is +127 and the minimum value is -128.

Your program must generate pulses that encode a 7-bit binary number that is the sum of all the digits of your student ID. For example, if your ID number is A00123456 the number you would output is 21 (binary 0010101). This number may or may not match that of anyone else doing the lab.

The value should be output in little-endian order (least significant bit first). A positive pulse should be used to represent a '0' and a negative pulse for a '1'. This is a common encoding although both the bit order and the signal polarity are the opposite of what you might expect. For the above example you would output pulses with polarities -+-+-++).

Output an extra high pulse (equivalent to an extra '0' value) before the pulses corresponding to your number to make it possible to trigger at the start of the pulse sequence. Output four trailing low pulses (equivalent to four extra '1' values) after your number so you can easily see where the pulse train ends. Thus the waveform you write to the file should consist of a total of 12 pulses: an initial trigger or 'start' pulse, seven data pulses encoding your specific number and four 'stop' pulses.

Assume a sampling rate of 9600 Hz. Calculate the corresponding sample period and the number of samples you need to output to create each 833 μs-long pulse.

Hints

- `#include <stdio.h>` and use `putc()` or `putchar()` to write bytes to the standard output. If you don't redirect the standard output, the characters corresponding to the sample values (+127 and -128) will be displayed on the command prompt window.
- declare the variables you write as 'signed char' if you want the byte values to be encoded in signed two's-complement notation (-128 being encoded as a byte with its bits set to 10000000 and +127 being 01111111).
- use a for loop (possibly within a function) to output the required number of samples per pulse
- you can use the "Decode" menu on the the 'scope to have the 'scope decode waveform. Select RS-232 decoding with TX as Channel 1, the baud rate set to 1200 and a threshold voltage of 0V. Use LSB order, 7 data bits, no check (parity) bit.

Pre-Lab

Submit the following to the appropriate drop box:

- the sum of the digits in your student number.
- compute the sample period (time between samples) and the number of samples per pulse.
- a C program that generates the required waveform. Save the program on your `H:` drive or bring it to the lab on your flash drive.

Post-Lab Report

Submit a report that includes:

- the cover page information as described in the course information handout
- the calculation of your unique number in decimal and the conversion to bits and voltage levels. Use a table showing the pulse number from 1 to 12, the bit value (0 or 1) and the signal polarity (+ or -).
- the Audacity and 'scope screen captures
- the program listing with all errors corrected (use a monospaced font such as Courier for the program listing so your indentation is correct – you can use copy an paste-with-formatting from Notepad++)
- export the report to a PDF format file and submit it via the appropriate course web site dropbox