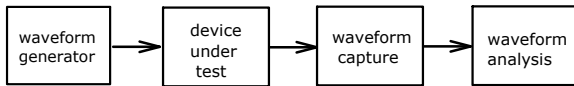


Generating Test Signals

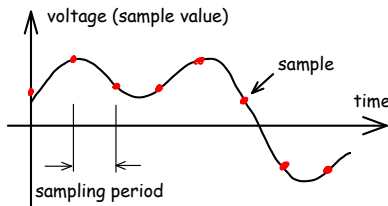
Introduction

One way to test a channel or communication system is to apply a known waveform as the input and analyze the output:



This approach is useful when you need to test a device or system for which there is no specialized test equipment. In this lab you will use an Arbitrary Waveform Generator (AWG) to generate a test waveform.

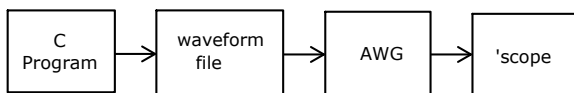
As the name implies, an AWG is able to generate an arbitrary waveform because it allows you to specify the output waveform as a sequence of sample values. These sample values are integers whose value is proportional to the voltage of the waveform at equally-spaced intervals of time:



You will write and run a C program that will compute the sample values and write them to a file.

You will transfer this file to the AWG using a USB “flash” drive and then use the front panel controls to configure the output voltage range and the sample period.

Finally, you will measure and record this analog waveform on an oscilloscope to verify that it is correct.

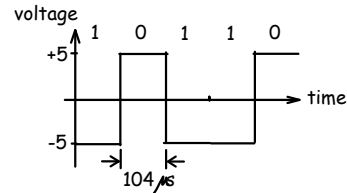


Analog Waveform Specification

The analog waveform output by the AWG should consist of a voltage that can be either +5 V or -5 V and is

held steady for intervals of 208 μ s. Each interval corresponds to one bit of data. For a ‘1’ bit the voltage should have a voltage of -5 V and for a ‘0’ it should have a voltage of +5 V.

Here’s an example of the waveform corresponding to the sequence of bits 1, 0, 1, 1, 0:

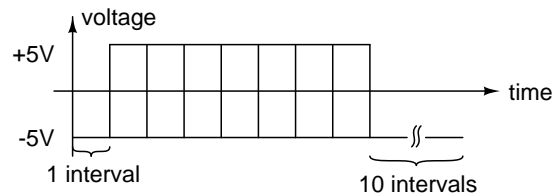


Your waveform must encode a 7-bit binary number that is the sum of all the digits of your student ID. For example, if your ID number is A00123456 the number you would output is $1 + 2 + 3 + 4 + 5 + 6 = 21$ (hex 0x15, binary 0010101). This number may not match that of others doing the lab.

The bits of the number should be output in *least significant bit first* (lsb-first) bit order. For the binary value 0010101, the output will begin with 101..., *not* 001...

Note that both the bit order and the signal polarity are probably the opposite of what you were expecting.

In addition, one positive voltage interval (equivalent to a ‘0’ bit) should be added before the data bits and ten negative voltage intervals (equivalent to ten ‘1’ bits) should be added after the data bits:



The AWG will continuously repeat your waveform and these additional “start” and “stop” bits will make it possible to identify the start and end of your waveform.

Digital Waveform Specification

To obtain the analog output described above we must load the AWG with a file containing the required sample values. This file should contain a sequence of 16-bit (2 byte) values, each value corresponding to one sample.

Each 16-bit sample values should be 16383 (0x3fff) for the positive voltage levels or 0 (0x00000) for the negative levels. You will configure the AWG so that a sample value of 0 produces a voltage of -5 V and a voltage of 16383 produces a voltage of +5 V (see Appendix A).

In order to generate a waveform that looks like pulses (rather than triangles), you should write the same sample value 10 times for each bit. The AWG period will be set to 1/10th of the desired bit duration, or equivalently, the sample rate will be set to a frequency that is 10 times the bit rate.

Your C program should create a file containing the desired sequence of sample values. Use the `fopen()` and `putc()` function calls to create and write to the file.

Note that your program does not generate the waveform when it runs. Instead, the waveform samples written by your program are loaded into the AWG which will be configured to output the samples at the appropriate sample rate and voltages.

The pseudo-code shown below might help if you aren't sure how to structure your program.

Your program will probably make use of the following C functions which are declared in the `<stdio.h>` header file:

- `fopen()` - to open a file
- `putc()` - to write a byte to the file

You may need to look up the documentation for these two functions. Open the file in "write binary" ("wb") mode, and use a file name extension of `.raf` (e.g. "waveform.raf").

For example:

```
#include <stdio.h>
// ...
FILE *f ;
// ...
f = fopen("waveform.raf", "wb") ;
// ...
putc(0xff, f) ;
putc(0x3f, f) ;
// ...
```

You can use the shift and masking C operators to extract the values of individual bits from an integer. For example, to test if the value of the i 'th bit of the integer n is non-zero (i.e. 1) you can use the expression: $(n \gg i) \& 1$.

You must write the sample values to the `.raf` file in little-endian byte order. For example to write the 16-bit value `0x3fff` you should use `putc(0xff, f)`; followed by `putc(0x3f, f)`; where `f` is the pointer returned by `fopen()` as shown above.

To avoid duplicating code you may want to write your program using functions. For example, you might want to use a function that writes the samples required for one bit as shown in the pseudo-code below.

```
open the output file

write a '0' bit

for each bit in your number (bit 0...6)
  if that bit is '1'
    write a '1' bit
  else
    write a '0' bit
end

do 10 times:
  write a '1' bit

function to write a bit:
  for each sample (sample 0...9)
    if we are writing a '1' bit:
      write 0x0000 (as two bytes)
    else
      write 0x3fff (as two bytes)
    end
  end
end
```

Procedure

The lab PCs should have the `tcc` and `Pelles C` compilers installed. The instructions below are for using the Notepad++ editor and `tcc` (tiny C compiler). These are small free programs you can run on any Windows PC.

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 a flash or network drive so that you don't lose your work if the PC reboots.

Start a command prompt (also known as a command interpreter or "shell") (Windows -> 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 "ELEX 3525\lab2"`. Try to avoid using spaces in file or folder names. If the file or folder names contain special characters or spaces you must use quotes as shown above.

Type the command:

```
tcc -run filename.c
```

to compile and run your program using the Tiny C Compiler (`tcc`) where `filename` is the name of your C file (e.g. `tcc -run wavegen.c`). If there are any error messages, correct the errors in your program and repeat the command.

You can use the arrow keys to repeat (up-arrow) and edit (left- and right-arrow) previous commands. You can use the tab key to auto-complete partial file names.

Once your program has run and created the `.raf` file you should check that the contents of the file are correct. Run the HexEdit utility and open the `.raf` file. The program will display the values of the bytes in the file in hexadecimal format. Check that the values and byte order are what you expected. Hexedit "locks" the file so exit it before running `tcc` again.

Take a screen capture that shows the hex value of the first 8 samples (16 bytes) (use `Start -> Snipping Tool`) and save it for use in your report.

To view the waveform you can use Audacity, an audio waveform editor. Run `Start -> Audacity` to start the program. Since your waveform file contains only the sample values and is not in an 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 (Signed¹ 16-bit Little-Endian), number of channels (1, Mono) and the sampling rate you assumed when creating the file.

Check the displayed waveform to make sure the waveform shape is correct. Take a screen capture that shows the time scale, the sample rate, sample format and the waveform and save it for use in your report.

¹Audacity can't import unsigned values. But sample values will appear positive since the two most-significant bits are zero.

Copy your `.raf` file to your USB flash drive and load it into the AWG using the instructions in Appendix B below.

Connect the 'scope's Channel 1 probe to the alligator clips on the cable connected to the AWG Channel 1 output. Use the oscilloscope to check for a ± 5 V waveform on the AWG output. Capture a sufficiently-long portion of the waveform and adjust the horizontal scale and position to show your complete waveform.

Note: Each time the 'scope is turned on you must select CH1 -> Probe Ratio -> 10X to get correct voltage scaling. Unfortunately, there is no way to make this the default.

Set up the 'scope's decoding feature as follows to check your waveform. Use the "Decode" button to bring up the decode menu. Select "RS-232" decoding with TX as Channel 1. Set the baud rate appropriately, a threshold voltage of 0V, LSB order, 7 data bits, no check (parity) bit and decimal display format. If you've done everything correctly you should see "your" number displayed.

Have the instructor check that your waveform as displayed on the 'scope is correct to get credit for completing the lab. Capture the 'scope screen to your flash drive by pressing the orange button with the printer icon so you can include the screen capture in your report.

Pre-Lab Assignment

Submit the following to the appropriate drop box before the start of the lab:

- The cover page information as described in the course information handout.
- The calculation of the sum of the digits in your student number.
- A table showing the conversion of this number to bits and voltage levels. Show the bit number (0, 1, 2, ...), the bit value (0 or 1) and the signal voltage (+5 or -5).
- A drawing of the waveform you expect to see on the 'scope. Show each bit, not necessarily each sample. Label the voltage and time axes.
- Compute the sample period (time between samples) based on the bit period and the number of

samples per bit specified above. Compute the corresponding sample rate that you will need to configure into the AWG.

- A listing of a C program that generates the required waveform.

Remember to save the program on your network drive or bring it to the lab on your flash drive.

Program listings should use a mono-spaced font (e.g. Courier) and single line spacing. A simple way to format listings is to open the file with Notepad++, select all text (Control-A), use Plugins -> NppExport -> Copy RTF to clipboard and then paste (Control-V) the formatted text into a word processor.

Post-Lab Report

Submit a report that includes:

- The cover page information as described in the course information handout.
- The HexEdit, Audacity and 'scope screen captures.
- The C program listing with all errors corrected.

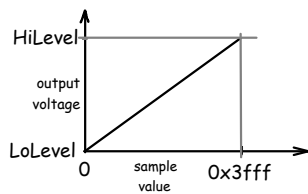
Convert your report to a PDF format file and submit it via the appropriate course web site dropbox.

Appendix A - AWG File Format

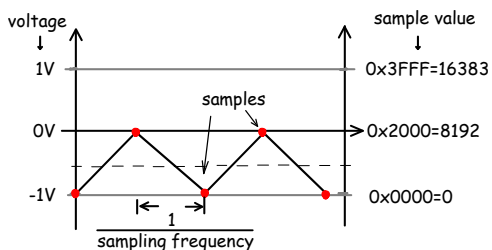
The Rigol DG1000Z AWG reads files in the `.raf` (Rigol AWG File) binary file format as documented on page 2-75 of the user manual.

An `.raf` file consists of a sequence of 16-bit samples. Each sample's value must be an unsigned 14-bit integer (i.e. one whose value is between 0 and 16383 ($0x3fff \equiv 2^{14} - 1$)).

The relationship between the sample value and the output voltage is a linear function. A sample value of 0 will result in an output voltage equal to the configured `LoLevel` voltage. A sample value of 16383 will result in an output voltage equal to the configured `HiLevel` voltage. Intermediate sample values give intermediate voltage levels:



For example, if the AWG was configured for a `LoLevel` of -1 V , a `HiLevel` of $+1\text{ V}$ and the `.raf` file contained values alternating between 0 and 8192 – half-way between the low and high levels – then the output analog waveform would alternate between levels of -1 V and 0 V as shown in the following figure:



The samples values in the `.raf` file must be in “little-endian” byte order – the least-significant byte of each two-byte word should appear first. The contents of the file for the waveform above would be as shown in the following hex dump:

```

00: 00 01 02 03 04 05 06 07
10: 00 00 00 20 00 00 00 20

```

You will have to configure the desired `HiLevel`, `LoLevel` and sampling rate as in the following appendix.

Appendix B - Configuring The AWG

- Press `Utility` `Set To Default` `OK` to reset the AWG to its default settings.
- Press `Arb` to select Arbitrary waveform output.
- Press `Arb Mode` until `SRate` (sample rate mode) is selected. Note: to access this and some of the following buttons you may need to press `▽` (gray key on the bottom) to toggle between two sets of menus.
- Plug your USB drive into the USB socket on the front of the AWG. You should see the message `USB device detected` and the USB icon (🔌) will appear on the display.
- Press `Select Waveform` `Stored Wforms`.
- Press `File Type` `Arb File`
- Press `Browser` until `Dir` is selected.
- Use the knob to highlight drive `D:` (the USB drive).
- Press `Browser` until `File` is selected.
- Use the knob to highlight your `.raf` file.
- Press `Read` and you should see the message `Completed! Arb Wave data have been read.`
- Press `Arb` to return to the `Arb` menu.
- Press `Srate` and enter the sample rate on the keyboard. Terminate the entry by selecting the appropriate units.
- Press `Ampl` until `HiLevel` is highlighted.
- Enter the voltage that should be output for the maximum sample value (`0x3ff`).
- Press `Offset` until `LoLevel` is highlighted.
- Enter the voltage that should be output for the minimum sample value (`0x000`).
- Press `Output1` to turn on the Channel 1 output.

Note: Keep these two appendices; you will need them for future labs.