**Rochester Institute of Technology**

**SWEN 563/CMPE 663/EEEE 663**

**Project 4 – Signal Generator with EKG**

Design and implement a 2-channel digital signal generator (like the one we measured for Project 1) using the STM32L476 Nucleo board. You will write an embedded stand-alone program that demonstrates the capabilities of the DAC driven under DMA and interrupt control by the microcontroller. The program must output different waveforms at various frequencies and levels as instructed by a user from a console. This is a hardware / software project - you will need to use both the Nucleo and an oscilloscope to complete and demonstrate the project.

**Problem Statement:**

* The microcontroller must generate several periodic selectable waveform outputs as illustrated below.
* The period of the waveform shall be settable by the user between 0.5 Hz <-> 10 kHz.
* The program must generate a clean output waveform without glitches per the user specification. An interrupt must be used to provide isochronous DAC updates up to 100 kHz (10 µs / sample).
* User interface shall accept the following commands:

**> gen channel type freq minv maxv noise**

Where

* **gen** is the name of the command to be executed. Note there will be additional commands added in Project 5.
* **channel** is either 1 or 2.
* **type** is the type of waveform (s=sine, r=rectangle (square), t=triangle, a=arbitrary (EKG, see below).
* **freq** is the frequency of the waveform in Hz, (0.5Hz to 100kHz). Note: if the freq==0, you will produce a DC voltage as given by minv.
* **minv** & **maxv** are the min and max voltages in the output waveform.
* **noise** is the number of bits of random noise to be added by the DAC (0-12).

Thus, a 1kHz sine wave ranging from 0.5 to 3.0V with 3 bits of noise on channel 1 would be specified by the user over the console as:

**gen 1 S 1000 0.5 3.0 3**

And a DC value of 2.5V in channel 2 would be specified by the user over the console as:

**gen 2 S 0 2.5 0.0 0**

| Waveform type | “R” for a rectangle   (square wave) |  |
| --- | --- | --- |
| “T” for triangle |  |
| “S” for sine |  |
| “A” for arbitrary |  |
| Frequency | In Hz (-.5-10000) | This is the number of cycles of the selected waveform to be output per second. Or 0 for DC. |
| Minimum voltage | 0 – 3.3V | Sets the minimum value of the output waveform. |
| Maximum voltage | 0 - 3.3V | Sets the maximum value of the output waveform. Note that it is expected that the maximum voltage is greater than the minimum voltage, and both are between 0V-3.3V. |

**Metrics:**

To monitor the performance of the device, display the following information to the console when changed by the user:

* The waveform type
* The signal frequency (waveforms per second)
* The update frequency (how fast you change the output, e.g. the DAC rate)
* The number of samples in 1 period of the waveform
* The minimum and maximum DAC codes in the waveform (used to produce the min/max voltages as specified as the user).

**Design Constraints:**

* You must build this project using an RTOS with queues and tasks. A receive task will accept user commands and place them in a receive queue. A command processing task will pull commands from the receive queue and execute them.
* Be sure to describe your selected architecture in your lab report. It is recommended that you fully define the architecture before you begin coding and implementation.
* You may use the CubeMX software to generate the initial project skeleton; enable the use of the FreeRTOS, UART2 device, DAC, DMA, interrupts, timers etc. to save time.

**Arbitrary Waveform**

Oftentimes, it is useful to be able to capture a periodic signal found in the ‘real world’, and be able to play it back in a repeated fashion. This affords the engineer the opportunity to develop real time signal processing of the signal. An example is given below of an EKG signal. You should be able to play this signal to get a periodic waveform output (typically 0.5 Hz to 3 Hz).

uint16\_t ekg[] = {

1690, 1680, 1680, 1669, 1648, 1648, 1648, 1680, 1680, 1690, 1680, 1680, 1680, 1680, 1680, 1658,

1690, 1690, 1712, 1690, 1690, 1680, 1669, 1669, 1680, 1690, 1690, 1680, 1669, 1669, 1669, 1680,

1680, 1680, 1669, 1669, 1680, 1690, 1690, 1680, 1690, 1690, 1680, 1690, 1690, 1712, 1680, 1680,

1658, 1648, 1648, 1648, 1669, 1669, 1680, 1690, 1690, 1701, 1680, 1680, 1669, 1680, 1680, 1680,

1701, 1701, 1701, 1690, 1690, 1701, 1712, 1712, 1722, 1712, 1712, 1690, 1669, 1669, 1680, 1690,

1690, 1690, 1733, 1733, 1765, 1776, 1861, 1882, 1936, 1936, 1968, 1989, 1989, 2032, 2053, 2053,

2085, 2149, 2069, 2080, 2058, 2058, 1930, 1930, 1845, 1824, 1792, 1872, 1840, 1754, 1754, 1722,

1680, 1680, 1680, 1637, 1637, 1637, 1637, 1637, 1626, 1648, 1648, 1637, 1605, 1605, 1616, 1680,

1680, 1765, 1776, 1861, 2042, 2106, 2021, 1776, 2480, 2400, 2176, 1632, 1637, 1360, 933, 928,

1962, 1962, 2042, 2149, 3141, 3141, 2320, 1200, 1200, 1392, 1669, 1669, 1658, 1701, 1701, 1701,

1701, 1701, 1722, 1690, 1690, 1690, 1680, 1680, 1690, 1690, 1690, 1669, 1669, 1669, 1701, 1733,

1733, 1754, 1744, 1744, 1733, 1733, 1733, 1722, 1765, 1765, 1765, 1733, 1733, 1733, 1722, 1722,

1701, 1690, 1690, 1701, 1690, 1690, 1701, 1701, 1701, 1701, 1722, 1722, 1712, 1722, 1722, 1733,

1733, 1733, 1733, 1712, 1712, 1712, 1733, 1733, 1733, 1733, 1733, 1733, 1744, 1744, 1744, 1744,

1744, 1744, 1733, 1733, 1722, 1722, 1722, 1722, 1722, 1722, 1733, 1722, 1722, 1722, 1722, 1722,

1701, 1669, 1669, 1680, 1690, 1690, 1690, 1701, 1701, 1712, 1712, 1712, 1690, 1669, 1669, 1680,

};

**Report:**

* Explain how your system works. Include a block diagram.
* Explain the architecture you settled on, and what engineering constraints led you to that architecture. Describe and explain the need for and use of interrupts and DMA.
* Discuss the difference between the computed waveform sampling interval (leading to the size of the DAC buffer) and the DAC update interval work together to produce the specified waveform.

In addition to the demonstration of your project, a brief report is required to describe your design. As to the final project report, refer to the Report Specifications for the required content and format.

Be sure to include one run of your output in your report. Please submit your report as either a Word compatible document or a PDF document in the project dropbox. Do not put it inside a zip file or other archive.

Your source code (the .c files you created along with any .h files) must be included in your electronic submission. Files generated by the compiler do not need to be submitted.

**Due Dates:**

Demonstration and collateral reports / code are due prior to the last lab/class.