The goals of this exercise are to study:
   (1) hierarchical design of embedded system software,
   (2) real-time operations, coordinated by programmable timers.
   (3) digital-to-analog conversion,
   (4) serial device interfaces (I²C in this case),

You are to design a musical “tone generator” for the DISCOVERY board, with button-selectable frequencies. These tones should be heard via the audio mini jack on the board (most standard speakers, “ear buds”, headphones, or similar listening device can be plugged into this jack.)

Project Specifications
1. Tones are to be produced by generating sinusoidal signals at the output of the microcontroller’s built-in 12-bit digital-to-analog converter (DAC). These sinusoids are to be produced by an algorithm, such as a series approximation. (Not table lookup.)
2. The DAC output (via GPIO pin PA4), which is connected to analog inputs AIN1A/AIN1B of the Cirrus CS43L22 Audio DAC, is to be directed to the headphone (HP) jack on the board via the “pass-through mode” of the CS43L22.
3. The tone frequencies are to be those of the eight notes of a musical C-major scale (C-D-E-F-G-A-B-C). (You will need to look up these frequencies.)
4. The sampling frequency is to be 20KHz. This information plus the tone frequencies will be used to generate the data samples to be converted by the DAC.
5. No tone should be generated immediately after resetting the board.
6. The first press of the user button should initiate generation of the low C tone.
7. Subsequent presses of the button should change the tone to the next higher note in the scale. (Only one change per button press.) After the high C tone, continue again from the low C tone.
8. If the button is pressed twice within a one-second interval, generation of tones should stop. Generation of tones should begin again on the next button press, with the last tone that was generated.
9. LEDs 4-3-5 (in that order) are to display the current tone number (binary numbers 000-111 correspond to tones from low C to high C.)
10. LED 6 is to blink at a rate that is proportional to the tone being generated. Select blinking rates that can be distinguished visually.

   LED3 (orange) = I/O port pin PD13
   LED4 (green) = I/O port pin PD12
   LED5 (red)   = I/O port pin PD14
   LED6 (blue)  = I/O port pin PD15
   User button (blue) = I/O port pin PA0

(Continue on next page for assignment Parts 1 & 2.)
PART 1 – Generation of data samples.

Design and program an algorithm, such as a series approximation, to generate 12-bit data samples at the required sampling rate for each selected tone frequency. (A lookup table may not be used.) Ideally, this should be one algorithm, with the tone frequency and sample rate as parameters. For testing purposes, you may change the initialization of the System Tick Timer to generate interrupts at a frequency of 20KHz, which is the required sampling frequency.

Test the algorithm by writing each sample to a global variable and displaying that variable in the debugger Logic Analyzer window. (This should show the sinusoidal signal at the tone frequency.) The document “Cortex-M4 training for STM32F4-Discovery board using ARM Keil MDK toolkit” (especially sections 15 and 19) describes the use of Keil Logic Analyzer to graphically display variable values and signals. This document is on the course web site.

Submit your source program and printouts of the Logic Analyzer window, showing waveforms of two different tones, corresponding to the musical notes E and A.

PART 2 – Generation of tones at the headphones.

1. Use one of the microcontroller’s general-purpose programmable timers, other than the system tick timer, to produce the required 20KHz data sampling frequency (i.e. generate new data samples every 1/20KHz = 0.05msec.)
2. Convert the 12-bit data samples to analog form with the microcontroller’s on-chip digital to analog converter (DAC). (Not the DAC on the CS43L22.)
3. Initialize the CS43L22 to “pass through” the analog signal from the output of the microcontroller DAC to the headphones.
4. Implement the rest of the button press and LED display requirements.

Testing and submitting the final program:

1. As before, the program can be tested in RAM or in flash memory. The final version should be programmed into the flash memory of the board, so that the program can be demonstrated without being connected to the Keil debugger. Bring your programmed board to my office for a demonstration.

2. Write a short report, briefly describing how you addressed/implemented each of the project requirements. Attach a printout of your source program to this report, and submit the report both in hard copy and electronic form (email).