

EECS 388  
LABORATORY EXERCISE II

---

# **SPEAKERBUZZ**

---

September 22, 2018

Benjamin Streit

## LABORATORY OVERVIEW

This laboratory involves the implementation of a new task to be run on the Tiva TM4C1294 evaluation board with the TI Audio BoosterPack add-on evaluation board. This new task will generate a tone using the Digital to Analog (DAC) and speaker that is housed on the TI Audio BoosterPack evaluation board.

## TONE GENERATION TASK

The structure of the tone generation task, named `Task_Speakerbuzz.c`, is as follows:

```
1 // Include DAC drivers
2 // Define state variables
3 // Initialize the DAC
4 // Initialize state value
5 // Enter endless while loop
6     // If state variable is high, toggle state variables and write state to DAC
7     // Else, toggle state variables and write state to DAC
8     // Delay
9     // Loop again
```

In more detail: The DAC drivers are included using `#include "Drivers/EECS388_DAC.h"`. State variables defined are `DAC_State` as a `uint32_t`, and `high` as a `bool`. The DAC is then initialized using the call `EECS388_DAC_Initialization()`; and `high` is set to `false`. Now the loop is entered, and both state variables are toggled depending on the current value of `high`, and the newly toggled `DAC_State` value is written to the DAC using the call `EECS388_WriteDAC( DAC_State )`. Afterwards, a delay is introduced using the call `vTaskDelay( ( x * configTICK_RATE_HZ ) / 10000 )`; where the value of `x` is determined and explained in the following Analysis section. Finally, the loop begins again.

In addition to the writing of `Task_Speakerbuzz.c`, this function had to be included in the `EECS_388_Program_Base_Fa18.c` file. This was done by declaring the function outside the `int main( void )` function, and within the function using a `xTaskCreate()` call.

## ANALYSIS

For this laboratory exercise, exploration of the relationship between the speaker buzz frequency requested by the TA, and the task delay the `Task_Speakerbuzz.c` is to employ. Say

## CODE SCREENSHOTS

The image displays two screenshots of the TI-ARM-PROJECTS IDE, showing C code for a TI-ARM-PROJECTS project.

**Top Screenshot:** The code is in the file `Task_Speakerbuzz.c`. It includes headers for `driverlib/gpio.h`, `driverlib/processor.h`, `drivers/uartstdio.h`, `FreeRTOS.h`, and `task.h`. It defines a task `Task_Blink_LED_Porth1` and a task `Task_Speakerbuzz`. The `main` function creates tasks for `Task_Blink_LED_Porth1`, `Task_Speakerbuzz`, `Task_ReportData`, and `Task_ReportSysTickCount`. It then starts the FreeRTOS scheduler.

```

16 #include "driverlib/gpio.h"
17 #include "driverlib/processor.h"
18 #include "drivers/uartstdio.h"
19 #include "FreeRTOS.h"
20 #include "task.h"
21 #include "stdio.h"
22
23 extern void Task_Blink_LED_Porth1(void *pParameters);
24 extern void Task_Speakerbuzz(void *pParameters);
25 extern void Task_ReportData(void *pParameters);
26 extern void Task_ReportSysTickCount(void *pParameters);
27
28 int main(void) {
29     Processor_Initiation();
30     UARTStdio_Initiation();
31
32     // Create a task to blink LED, Porth1
33     // TaskCreate( Task_Blink_LED_Porth1, "Blink", 12, NULL, 1, NULL );
34
35     // Create a task to change speaker frequency
36     // TaskCreate( Task_Speakerbuzz, "Speakerbuzz", 32, NULL, 1, NULL );
37
38     // Create a task to report data.
39     // TaskCreate( Task_ReportData, "ReportData", 512, NULL, 1, NULL );
40
41     // Create a task to report SysTickCount
42     // TaskCreate( Task_ReportSysTickCount, "ReportSysTick", 512, NULL, 1, NULL );
43
44     vTaskPrint( "FreeRTOS Starting!\n" );
45
46     // Start FreeRTOS Task Scheduler
47     vTaskStartScheduler();
48
49     while ( 1 ) {
50     }
51 }

```

**Bottom Screenshot:** The code is in the file `EC3388_DAC.c`. It includes headers for `inc/hw_types.h`, `inc/hw_uart.h`, `stddef.h`, `stdbool.h`, `stdint.h`, `stdarg.h`, `driverlib/sysctl.h`, `driverlib/gpio.h`, `driverlib/adc.h`, `FreeRTOS.h`, and `task.h`. It defines a task `Task_EC3388_DAC`. The `main` function initializes the EC3388 DAC interface and starts the task.

```

13 #include "inc/hw_types.h"
14 #include "inc/hw_uart.h"
15 #include "stddef.h"
16 #include "stdbool.h"
17 #include "stdint.h"
18 #include "stdarg.h"
19 #include "driverlib/sysctl.h"
20 #include "driverlib/gpio.h"
21 #include "driverlib/adc.h"
22 #include "FreeRTOS.h"
23 #include "task.h"
24
25 extern void Task_EC3388_DAC(void *pParameters);
26
27 int main(void) {
28     uint32_t DAC_State;
29     bool high;
30
31     // Initialize the EC3388 DAC interface.
32     EC3388_DAC_Initiation();
33
34     // Set boolean value
35     high = false;
36
37     while ( 1 ) {
38         if ( high ) {
39             // Set DAC value
40             DAC_State = 0x0000;
41             EC3388_WriteDAC( DAC_State );
42             high = false;
43         } else {
44             // Set DAC value
45             DAC_State = 0x3FFF;
46             EC3388_WriteDAC( DAC_State );
47             high = true;
48         }
49         vTaskDelay( ( 20 * configTICK_RATE_HZ ) / 1000 );
50     }
51 }

```