EE2016: Experiment 9
Group:19

Sparsh Gupta	Charan Srikkanth	Kaushik Iyer
EE23B117	EE23B127	EE23B135

August 14, 2024

1 Introduction

In this experiment, we perform digital to analog conversion using the LPC2148 microcontroller (both the LPC2148 microcontroller and the LPC2378 microcontroller are based on ARM-7).Hence we use LPC2148.

2 Objectives

The primary objectives of this experiment are:

1. To generate a sine wave on the oscilloscope
2. Find the maximum amplitude possible.
3. Find out the resolution of the DAC
4. Repeat the same experiment for triangular wave and staircase waveform on the oscilloscope

3 The DAC on the LPC2148

The LPC2148 microcontroller provides built-in DAC functionality. To enable and use the DAC:

- **DAC Configuration:** The `PINSEL1` register controls the configuration of pin P0.25 for DAC output. Setting bits 19:18 of `PINSEL1` to `10` enables the DAC.
- The `DACR` register (32-bit) holds the digital value to be converted. The value is placed in bits 15:6, while bit 16 is used to enable bias control.

4 Oscilloscope Setup

A Tektronix digital storage oscilloscope is used to observe the waveforms generated by the DAC. The oscilloscope displays the signal with time on the X-axis and voltage on the Y-axis.

5 Tasks

5.1 Task 1: Sine wave

```
#include <lpc214x.h>
#include <stdint.h>

void delay_ms(uint16_t j)
```

```
{
    uint16_t x, i;
    for (i = 0; i < j; i++)
    {
        for (x = 0; x < 6; x++)
            ; /* loop to generate 1 milisecond delay with Cclk = 60MHz */
    }
}

int main(void)
{
    // Initialize DAC
    PINSEL1 = 1 << 19; /* P0.25 as DAC output */
        int mode = 3;

    while (1)
    {
        if (mode == 3)
        { // sin
            DACR = (1 << 16) | (512 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (591 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (665 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (742 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (808 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (873 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (926 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (968 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (998 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (1017 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (1023 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (1017 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (998 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (968 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (926 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (873 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (808 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (742 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (665 << 6);
```

```
            delay_ms(10);
            DACR = (1 << 16) | (591 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (512 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (436 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (359 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (282 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (216 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (211 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (151 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (97 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (55 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (25 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (6 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (0 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (6 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (25 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (55 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (97 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (151 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (211 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (216 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (282 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (359 << 6);
            delay_ms(10);
            DACR = (1 << 16) | (436 << 6);
            delay_ms(10);

        }
    }
}
```

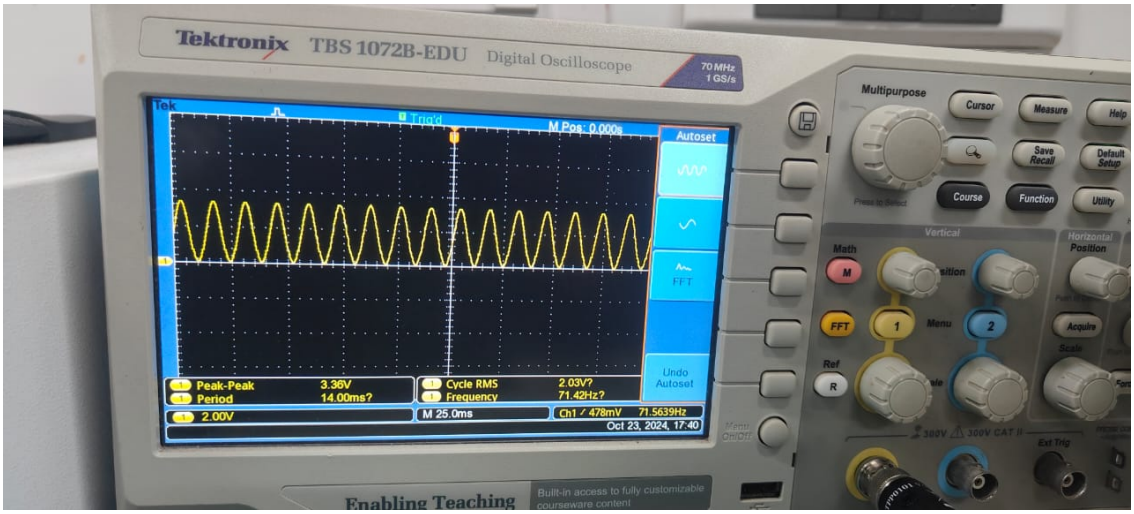

Figure 1: OUTPUT for sine wave

5.1.1 Output

5.2 Task 2: Maximum Amplitude

Since max output voltage is 3.3 V the max value on the oscilloscope is 3.3V,and the lowest value is 0V . Max peak-to-peak of any wave generated using lpc is 3.3V. **Max Amplitude possible is 1.65V**

5.3 Task 3: DAC Resolution and DC Voltage Generation

The DAC has 10-bit resolution, and peak to peak is 3.3V. Hence, resolution is $3.3V/1024 = 0.003V$

5.3.1 DC voltage Generation

```
#include <lpc214x.h>
#include <stdint.h>

void delay_ms(uint16_t j)
{
    uint16_t x, i;
    for (i = 0; i < j; i++)
    {
        for (x = 0; x < 6; x++)
            ; /* loop to generate 1 milisecond delay with Cclk = 60MHz */
    }
}

int main(void)
{
    // Initialize DAC
    PINSEL1 = 1 << 19; /* P0.25 as DAC output */
        int mode = 3;
 while (1)
    {
        if (mode == 3)
        { // DC voltage of 1.25 V
```

```
        DACR = (1 << 16) | (388 << 6);

        }
```

5.4 Task 4: Triangular Wave Generation

```
#include <lpc214x.h>
#include <stdint.h>

void delay_ms(uint16_t j)
{
    uint16_t x, i;
    for (i = 0; i < j; i++)
    {
        for (x = 0; x < 6; x++)
            ; /* loop to generate 1 milisecond delay with Cclk = 60MHz */
    }
}

int main(void)
{
    // Initialize DAC
    PINSEL1 = 1 << 19; /* P0.25 as DAC output */
        int mode = 0;
 while (1)
    {
        if (mode == 0)
        { // triangle
            for (int i = 0; i < 1023; i++)
            {
                DACR = (1 << 16) | (i << 6);
                delay_ms(1);
            }
            for (int i = 1023; i >= 0; i--)
            {
                DACR = (1 << 16) | (i << 6);
                delay_ms(1);
            }
        }
```

5.4.1 Output

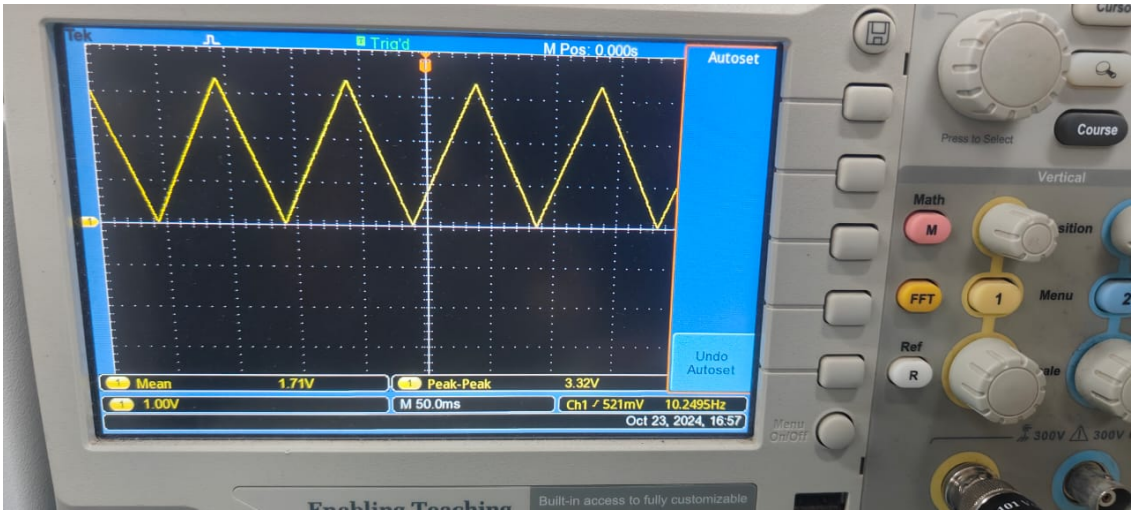

Figure 2: OUTPUT

5.5 Task 5: Staircase Waveform Generation

```c
#include <lpc214x.h>
#include <stdint.h>

void delay_ms(uint16_t j)
{
    uint16_t x, i;
    for (i = 0; i < j; i++)
    {
        for (x = 0; x < 6; x++)
            ; /* loop to generate 1 milisecond delay with Cclk = 60MHz */
    }
}

int main(void)
{
    // Initialize DAC
    PINSEL1 = 1 << 19; /* P0.25 as DAC output */
        int mode = 2;
 while (1)
    {
        else if (mode == 2)
        { // staircase
            for (int i = 0; i < 1023; i+=100)
            {
                DACR = (1 << 16) | (i << 6);
                delay_ms(100);
            }
        }
```

5.5.1 Output

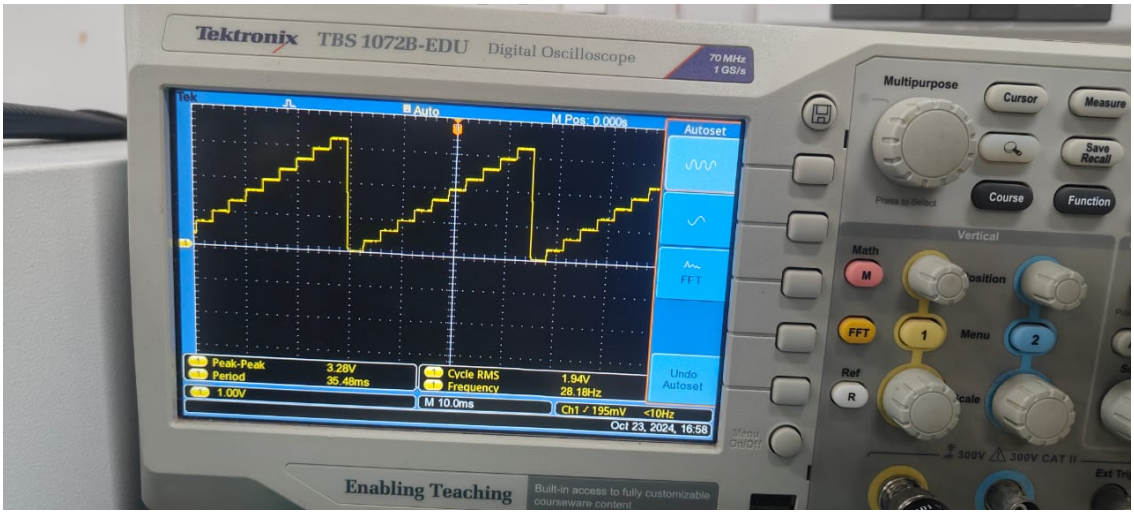

Figure 3: OUTPUT

6 BONUS

Square wave generated

6.0.1 Output

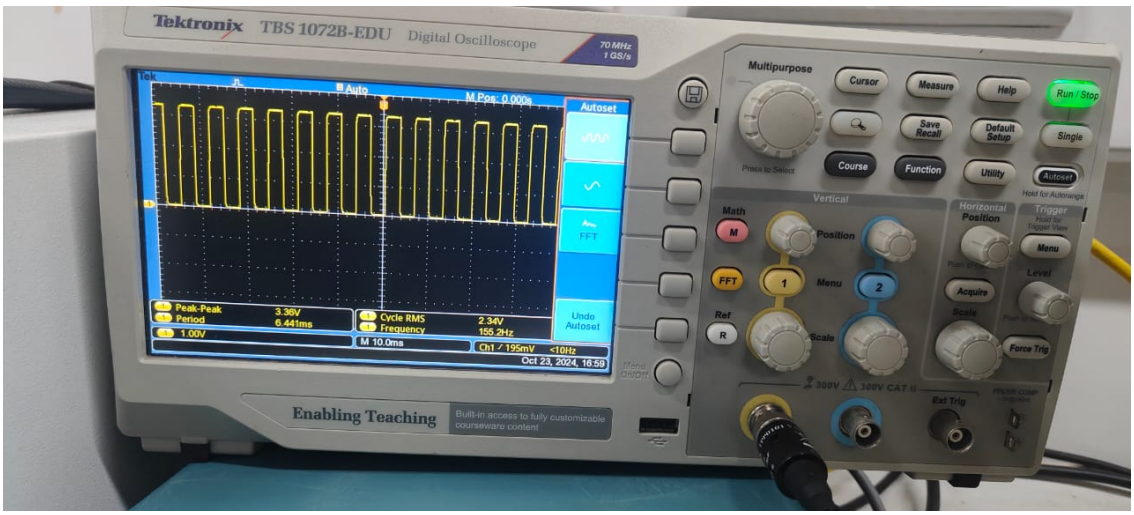

Figure 4: OUTPUT

7 Conclusion

The experiment successfully demonstrated the generation of multiple waveforms using the DAC of the LPC2148 microcontroller. These waveforms were observed on the oscilloscope, illustrating the practical application of digital-to-analog conversion in embedded systems.

8 Comments

1. We initially faced issues due to a noisy board, since the board kept giving us noisy sine wave, We later realised that there is a reset button on the board to actually refreshes the code to run.

2. Another issue we faced was that the machine didn't support arrays or floats. since both arrays and floats were not there in our disposal we had to hardcode all the values to our sine wave.