EE2016: Microprocessers Lab - Assignment 9 Report

Aadarsh Ramachandiran (EE23B001), Aditya Kartik (EE23B003), Rohita Datta (EE23B065)

November 3, 2024

Abstract

This is a detailed report on an assignment to write a C program to facilitate digital to analog conversion of signals using an LPC2148 microcontroller and realising the outputs in an oscilloscope. The report contains a brief of the concepts, the code and explanation and the observations made in the lab.

1 Introduction

This assignment involves programming an LPC2148 mcirocontroller in C to act as a digital to analog converter and realising the output waveforms such as sinusoidal, DC, triangular and a stair case on an oscilloscope.

A Digital to Analog Converter (DAC) is a device commonly used to generate analog waveforms particularly useful in applications such as audio/music, video, etc. It works by taking in a binary input number and outputs the corresponding Voltage value in analog domain as output. For example, if it can consider 10-bit binary values and the voltage capacity is 3.3 V, then, some value less than 10-bits correspond to a voltage of (value/1023) * 3.3 V. The values are sampled at some rate and that corresponds to some noise in the analog output.

1.1 DAC on the LPC2148

The LPC2148 provides "built-in" DAC support. The built-in DAC is configured by appropriate setting of the pin function select register 1 (PINSEL1). In particular, bits 19:18 of the PINSEL1 register are used to enable the DAC. These bits should be 10 for the DAC to be powered on and active. For the conversion itself, the 32-bit DACR register is used. It is a read/write register. The digital value (10 bits) to be converted to analog form is written in bits 15:6 of this register.

1.2 Setting up the oscilloscope

A Tektronix digital storage oscilloscope was used for the experiment. The X-axis of the oscilloscope represents time while the Y -axis represents voltage. Before beginning, we first check oscilloscope functionality. For this purpose, the oscilloscope provides a calibration output of 5V. The probe has a Bayonet Neill Concelman (BNC) plug at one end that should be connected to the socket on channel 1 on the oscilloscope. The other end of the probe has a black crocodile clip that should be connected to the ground near the "probe. comp". In addition, the long black syringe-like live pin is connected to the "prob. comp" (on the oscilloscope). Press Autoset on the oscilloscope to see a 5V 1KHz square waveform. This ensures the oscilloscope is working.

1.3 Connecting the LPC2148 board to the Oscilloscope

The green 4-pin connector on the right bottom of the LPC2148 development board provides the DAC output. The top pin is ground. The next lower pin is DAC output. We connect the oscilloscope probes appropriately.

---

2 Design

2.1 Sine Waveform

2.1.1 Description

We set up and run a simple digital-to-analog (D-to-A) conversion experiment on the LPC2148 microcontroller. The main aim is to generate a sine wave output using the on-chip Digital-to-Analog Converter (DAC).

2.1.2 Code

```
// Include necessary headers
#include "LPC214x.h"
#include "stdint.h"
#include "math.h"
#define PI 3.14159

// configure the DAC
void DACInit(void)
{
        PINSEL1 |= (0b1 << 19);
        PINSEL1 &= ~(0b1<<18);
        return;
}


void delay_ms(uint16_t j)
{
    uint16_t x, i;
    for(i=0; i<j; i++)
    {
        for(x=0; x<60000; x++);    /* loop to generate 1 millisecond
            delay with Cclk = 60MHz */
    }
}

// Main routine
int main (void)
{
    uint16_t value;
    int i;
    i = 0;

    DACInit();

    while(1)
    {
        while(i != 32)
        {
            value = (uint16_t)(1023 * (sin(PI * i / 32) + 1));
            DACR |=  (0b1 << 16);
                  DACR = (0x3FF << 6) & (value << 6);
            delay_ms(1);
            i++;
        }
        i = 0;
```

```
    }
    return 0;
}
```

2.1.3 Explanation

- **Headers and Constants**: The code includes `stdint.h` for fixed-width integers, `math.h` for mathematical functions, and defines `PI` as 3.14159 for sine calculations.
- **DAC Initialization (`DACInit`)**: Configures pin `P0.25` for DAC output by setting and clearing specific bits in the `PINSEL1` register.
- **Delay Function (`delay_ms`)**: Provides an approximate delay in milliseconds using nested loops, assuming a 60 MHz clock frequency.
- **Main Loop**: - Calculates sine wave values by scaling `sin()` results to fit the DAC's 10-bit range (0–1023). - Shifts `value` left by 6 before assigning to `DACR` to fit within DAC bit positions (bits 0–5 are reserved). - Resets the index after 32 samples, creating a looped sine waveform output.

2.2 DC waveform

2.2.1 Description

We modify the DACR value to generate a constant voltage of 1.25V.

2.2.2 Code

```
// C program to analyze a DC signal in DAC
#include "LPC214x.h"
#include "stdint.h"
#include "math.h"

// configure the DAC
void DACInit(void)
{
        PINSEL1 |= (0b1 << 19);
        PINSEL1 &= ~(0b1<<18);
        return;
}
// main routine
int main (void)
{
        int value;
        DACInit();
        while(1)
        {
                // Vout = VALUE/VAL_MAX * Vref
                // Resolution = 3.32 / 1023 V
                // 1.25 = VALUE / 1023 * 3.32
                value = 385;  // FOr 1.25 V
                DACR |=  (0b1 << 16);
                DACR = ( 0x3FF << 6) & (value<<6);
        }
        return 0;
}
```

2.2.3 Explanation

- **Voltage Calculation for 1.25V Output**:
  - The reference voltage (V_{ref}) is assumed to be 3.32V, and the DAC is 10-bit, meaning it has 1024 levels.
  - To generate 1.25V, the program calculates the required DAC value using:

$$\text{DAC Value} = \frac{1.25 \times 1023}{3.3} \approx 385$$

  - `value = 385` is then shifted by 6 bits (to account for the lower 6 reserved bits) and stored in `DACR`.
- **Outputting the DC Signal**: The main loop assigns `DACR` with the calculated value, holding a constant voltage output. The statement `DACR = (0x3FF << 6) & (value << 6)` uses bitwise operations to fit `value` in the correct DAC bits.

This code produces a steady DC output of approximately 1.25V.

2.3 Triangular Waveform

2.3.1 Description

We increment the voltage in steps in a loop then decrement it, producing a triangular waveform.

2.3.2 Code

```
// C program to generate a triangle waveform in DAC
#include "LPC214x.h"
#include "stdint.h"

// configure the DAC
void DACInit(void)
{
        PINSEL1 |= (0b1 << 19);
        PINSEL1 &= ~(0b1<<18);
        return;
}

void delay_ms(int j)
{
        int x,i;
        for(i=0;i<j;i++)
        {
                for(x=0; x<6000; x++);     // loop to generate 1
                    milisecond delay with Cclk = 60MHz
        }
}

// main routine
int main (void)
{
        int value;
        int i;
        i = 0;
        DACInit();
while(1)
        {
```

```
                while(i !=32)
                {
                        value = (1023 *i)/32;
                        DACR |=  (0b1 << 16);
                        DACR = ( 0x3FF << 6) & (value<<6);
                        delay_ms(1);
                        i++;
                }
                i = 0;
                while(i !=32)
                {
                        value = (1023 *(32-i))/32;
                        DACR |=  (0b1 << 16);
                        DACR = ( 0x3FF << 6) & (value<<6);
                        delay_ms(1);
                        i++;
                }
                i = 0;
        }
return 0;
}
```

2.3.3 Explanation

Loop for Triangle Waveform:

- **Rising Ramp**: A loop increments `i` from 0 to 31. Each step sets the DAC value as $\frac{1023 \times i}{32}$, outputting a steadily increasing voltage.
- **Falling Ramp**: After the rising ramp, `i` resets, and the next loop decrements the DAC value from 1023 to 0, producing the falling edge.
- **Continuous Waveform**: The rising and falling ramps repeat, creating a continuous triangle waveform.

2.4 Staircase Waveform

2.4.1 Description

We increase the voltage in steps, from 0 to 2.38V.

2.4.2 Code

```
    // C program to generate staircase waveform in DAC
#include "LPC214x.h"
#include "stdint.h"


// configure the DAC
void DACInit(void)
{
        PINSEL1 |= (0b1 << 19);
        PINSEL1 &= ~(0b1<<18);
        return;
}

void delay_ms(int j)
{
```

```
        int x,i;
        for(i=0;i<j;i++)
        {
                for(x=0; x<6000; x++);     /* loop to generate 1
                    milisecond delay with Cclk = 60MHz */
        }
}
// main routine
int main (void)
{
        int value;
        int i;
        i = 0;
        DACInit();
        while(1)
                {
                        while(i !=16)
                        {
                                value = ((1023/3.3 * 2.38) *i)/16; //
                                    Max Amplitude = 2.38 V
                                DACR |=  (0b1 << 16);
                                DACR = ( 0x3FF << 6) & (value<<6);
                                delay_ms(1);
                                i++;
                        }
                        i = 0;
                }
        return 0;
}
```

2.4.3 Explanation

Main Loop:

- Increments `i` from 0 to 15 to create 16 steps.
- Calculates the DAC output as value $= \left(\frac{1023}{3.3} \times 2.38\right) \times \frac{i}{16}$ for a max amplitude of 2.38 V.
- Outputs the value to the DAC, generating a staircase waveform.

3 Implementation

1. **Connect LPC2148 to Digital Oscilloscope**: The green 4-pin connector on the right bottom of the LPC2148 development board provides the DAC output. The top pin is ground. The next lower pin is DAC output. The ground pin is connected to the black crocodile clip at the end of one of the probes from the oscilloscope. The other end of the probe is connected to the top pin.
2. **Powering the LPC2148**: Connect the adapter to the power source and the other end to the LPC2148 microcontroller to power the board.
3. **Install Keil µVision**: Ensure Keil µVision is installed. If not, download it from the Keil website and install it.
4. **Open Keil µVision**: Launch the Keil µVision IDE.
5. **Create a New Project**: Go to **Project** → **New µVision Project**, choose a folder to save your project, and give it a name.

6. **Add Source Files**: Go to **File** → **New** and write your ARM assembly code. Save it and add it to the project by right-clicking the project in the **Project window** → **Add Files to Group**.
7. **Generate Hex File**: Right-click **Target 1** and in the **Output** section check the **Create HEX File** box.
8. **Compile the Code**: Click **Project** → **Build Target** or press **F7** to compile the code.
9. **Flash Magic**: Open **Flash Magic** application and select device as LPC2148. Set the **Baudrate** to 19200 and browse the computer files to add the hex file of our project. Click the **Start** button to flash it onto the LPC2148 microcontroller which is in turn connected to the digital oscilloscope.
10. **Autoset the wave**: Press the autoset button on the digital oscilloscope to fit the wave appropriately and then use cursor 1 and cursor 2 to make necessary measurements (in this case, measuring the voltage amplitude and frequency of the wave). Feel free to use the Run/Stop button to play/pause the wave.

4 Observation

4.1 Displaying sine wave on the Oscilloscope

We observe the sine waveform on the oscilloscope after using the Autoset feature. The values can be scaled to scale the waveform as needed.

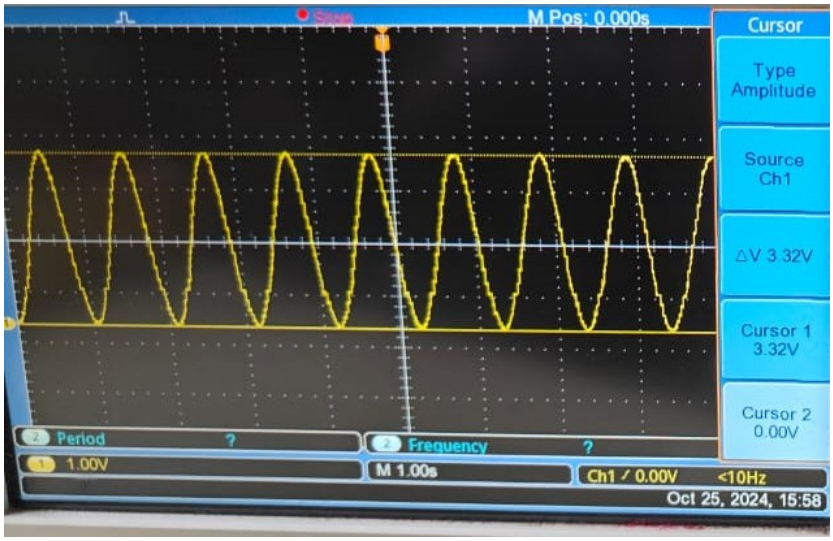

Figure 1: Sine Wave

The maximum amplitude is 3.32 V as you can see from the cursors.

Figure 2: Sine Waveform Frequency

We can also observe that the frequency is about 925.9mHz from the cursors.

4.2 Displaying DC wave on the Oscilloscope

We observe a flat horizontal waveform on the oscilloscope as expected for a DC (constant) voltage.

This code produces a steady DC output of approximately 1.25V.

4.3 Displaying Triangular waveform on Oscilloscope

We observe the triangular waveform on the oscilloscope after using the Autoset feature.

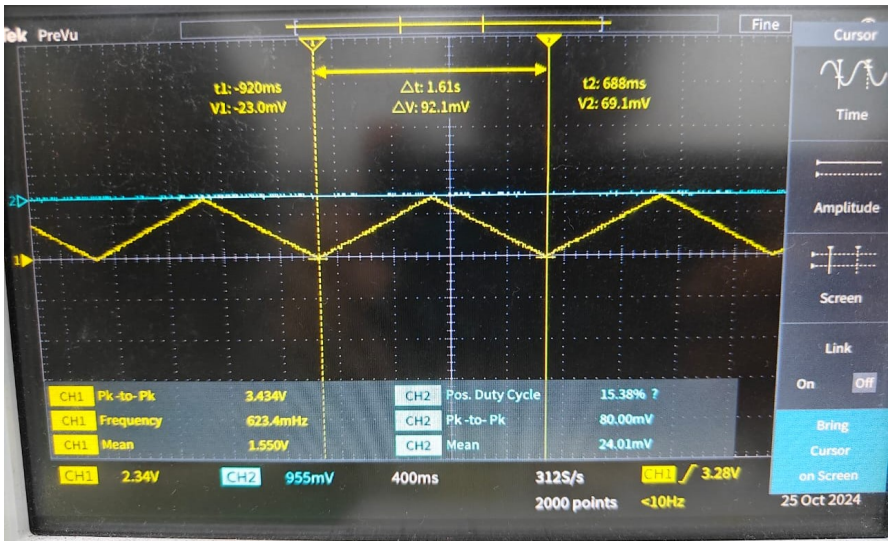

Figure 3: Triangular Waveform

As we can observe from the oscilloscope, the peak voltage is around 3.434V and frequency is 623.4mHz

4.4 Displaying Staircase waveform on Oscilloscope

We observe the staircase waveform on the oscilloscope after using the Autoset feature.

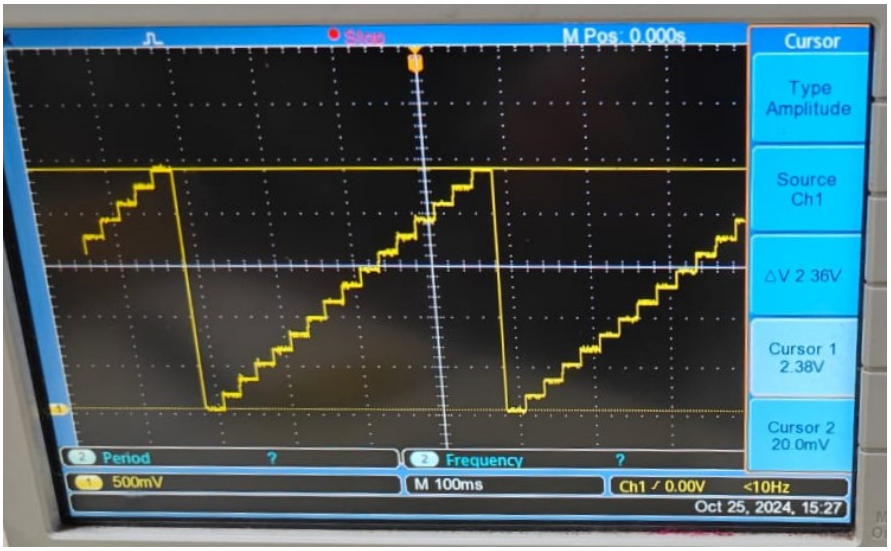

Figure 4: Staircase Waveform

The resultant waveform has a voltage ampiltude of 2.38 V with 16 steps.

---

5 Conclusion

The experiment successfully demonstrated the use of LPC2148 as a DAC to realise various waveforms on an oscilloscope. Correlating the sampling rate of the digital values in the program and the corresponding analog difference helps in understanding how analog signals are generated in today's digital era. The code for programming the ARM board for DAC on various waveforms can be found here.