MICROPROCESSOR LAB EXPERIMENT 9

GROUP - 18
Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

October 2024

Introduction :

- In this experiment, we will perform digital to analog conversion using the LPC2148 micro-controller (the LPC2148 microcontroller as well as the LPC2378 microcontroller are based on ARM-7).
- The aim of this experiment is to learn about Digital to Analog Converter and use it to display various plots.
- ARM stands for **Advanced RISC Machine** which is bit more advanced than AVR.
- We are going to use LPC-2148 as a microcontroller to perform this.

ARM and Keil μvison :

- ARM architectures are the most common electronic design in the world.
- It is known for its speed.
- Using an ARM architecture gives hardware designers more control over their designs and performance, as well as more control over their supply chains.
- In this experiment , we use **LPC2148** microcontroller for operations.
- The tasks that we are going to perform here will help us know the basic operations in DAC.

Brief introduction to DAC

- DAC helps us convert a series of digital signals to continuously varying analog signal.
- For example , to represent sine wave , we can discretize the amplitudes at different positions and pass it to DAC to generate a smooth,continuous sine wave.
- The way how it works is it will intake a specific number of bits which is a digital input and converts it to a analog signal based on the minimum and maximum voltage that dac is designed to.
- For example , the DAC in LPC-2148 works at maximum 3.3V and takes 10 bits of input.
- We define the minimum voltage value that can be given with the digital input as resolution which is $\boxed{\frac{V_{max}}{No.of bits}}$

Difficulties faced while compiling it in C language

- While writing C code and burning it to LPC-2148 , we came across multiple issues.
- The key issue that we faced was **Arrays don't work in LPC-2148**.
- One another issue that we faced was even **Float** datatype was not defined which restricted us from using sine function given in math library.
- The wierdest issue that we faced was , we tried to write a function that operates on switch cases and returns value and we inferred that the function can take only **atmost 4 cases**.
- Due to these issues , we just hardcoded all the sine values and passed it to DAC.

Instructions to write code

- There are two things to be noted while giving the values.
- 19:18 bits of PINSSEL1 must be 10.
- Therefore we defined a DACinit() function to ensure that the above condition is satisfied by using OR operation.
- One another constraint is that 0-5 bits are reserved , so we right shift the value by 6 bits to satisfy this condition.

Tasks 1 & 2

Code - Task-1

```
#include "LPC214x.h"
#define M_PI 3.14159265359
#define ITERATIONS 1000
#define FACTOR 0x3FF
#define DAC_BIAS 0x00010000


void DACInit( void )
{
    /* setup the related pin to DAC output */
        PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000;         /* set p0.25 to DAC output */
    return;
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}

int main()
{
// main routine
        DACInit();
    while(1)
    {
            val = 512 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 591 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 665 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 742 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 808 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 873 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 926 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 968 << 6;
```

```
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 998 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 1017 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 1023 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 1017 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 998 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 968 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 926 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 873 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 808 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 742 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 665 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 591 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 512 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 436 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 359 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

```

```
        val = 282 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 216 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 211 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 151 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 97 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 55 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 25 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 6 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 0 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 6 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 25 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 55 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 97 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 151 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 211 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);

        val = 216 << 6;
        DACR = (val) | DAC_BIAS;
        mydelay(0x0f);
```

```
185
186         val = 282 << 6;
187         DACR = (val) | DAC_BIAS;
188         mydelay(0x0f);
189
190         val = 359 << 6;
191         DACR = (val) | DAC_BIAS;
192         mydelay(0x0f);
193
194         val = 436 << 6;
195         DACR = (val) | DAC_BIAS;
196         mydelay(0x0f);
197
198     }
199         return 0;
200 }
```

Explanation

- To generate the sine wave, we step through 42 different values of sine in one period and manually assign the corresponding voltage. Refer to Observation and Difficulties faced for the reason.
- Before sending the signal, we need to initialize the DAC, we accomplish this by setting bits 19:18 of the PINSEL1 register. 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

Task-2

The maximum amplitude of the DAC module on LPC2148 is 3.3V according to the datasheet. This corresponds to a value of 0x3FF in code.

Task 3 & 4

Code - Task-3

```c
#include "LPC214x.h"
#define M_PI 3.14159265359
#define ITERATIONS 1000
#define FACTOR 0x3FF
#define DAC_BIAS 0x00010000

void DACInit( void )
{
    /* setup the related pin to DAC output */
        PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000;        /* set p0.25 to DAC output */
    return;
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}

int main()
{
// main routine
        int val = 387;
        DACInit();
while(1)
{
        DACR=(val<<6) | DAC_BIAS;
        mydelay(0x0f);
}
        return 0;
}
```

Explanation :

- The resolution of the DAC is
$$\frac{3.3}{2^{10}} = 3.22mV$$
- To generate a DC Voltage of $1.25V$, the corresponding value in the DAC register should be $(387)_{10} = (110000011)_2$

Code - Task-4

```c
#include "LPC214x.h"
#define M_PI 3.14159265359
#define ITERATIONS 1000
#define FACTOR 0x3FF
#define DAC_BIAS 0x00010000

void DACInit( void )
{
    /* setup the related pin to DAC output */
        PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000;        /* set p0.25 to DAC output */
    return;
```

```
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}

int main()
{
// main routine
        int val=0;
        int dx=100;
        int i=0;
        DACInit();
while(1)
{
        DACR=(val<<6) | DAC_BIAS;
        mydelay(0x0f);
        val+=dx;
        if(val>0x3FF-100 || val < 100)
        {
                dx=-dx;
        }
}
        return 0;
}

```

Explanation :

- the frequency of the wave depends on the frequency of the processor, assuming 25 MHz the frequency of the triangular pulse is around $306Hz$
- The amplitude of the wave is $3.3V$

Task-5

Code - Task-5

```c
#include "LPC214x.h"
#define M_PI 3.14159265359
#define ITERATIONS 1000
#define FACTOR 0x3FF
#define DAC_BIAS 0x00010000

void DACInit( void )
{
    /* setup the related pin to DAC output */
        PINSEL1 &= 0xFFF3FFFF;
    PINSEL1 |= 0x00080000;         /* set p0.25 to DAC output */
    return;
}
void mydelay(int x)
{
int j,k;
for(j=0;j<=x;j++)
{
    for(k=0;k<=0xFF;k++);
}
}

int main()
{
// main routine
        int val=0;
        int dx=100;
        int i=0;
        DACInit();
while(1)
{
        DACR=(val<<6) | DAC_BIAS;
        mydelay(0x0f);
        val+=dx;
        val %= 739;
}
        return 0;
}

```

Explanation :

- the value 739 corresponds to a $2.38V$ max amplitude
- the frequency of the staircase is around $730Hz$