

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 μ vision :

- 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 $\frac{V_{max}}{No.ofbits}$

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 PINSEL1 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.

Code - Task-1

```

1  #include "LPC214x.h"
2  #define M_PI 3.14159265359
3  #define ITERATIONS 1000
4  #define FACTOR 0x3FF
5  #define DAC_BIAS 0x00010000
6
7
8  void DACInit( void )
9  {
10     /* setup the related pin to DAC output */
11     PINSEL1 &= 0xFFFF3FFFF;
12     PINSEL1 |= 0x00080000;      /* set p0.25 to DAC output */
13     return;
14 }
15 void mydelay(int x)
16 {
17     int j,k;
18     for(j=0;j<=x;j++)
19     {
20         for(k=0;k<=0xFF;k++);
21     }
22 }
23
24 int main()
25 {
26     // main routine
27     DACInit();
28     while(1)
29     {
30         val = 512 << 6;
31         DACR = (val) | DAC_BIAS;
32         mydelay(0x0f);
33
34         val = 591 << 6;
35         DACR = (val) | DAC_BIAS;
36         mydelay(0x0f);
37
38         val = 665 << 6;
39         DACR = (val) | DAC_BIAS;
40         mydelay(0x0f);
41
42         val = 742 << 6;
43         DACR = (val) | DAC_BIAS;
44         mydelay(0x0f);
45
46         val = 808 << 6;
47         DACR = (val) | DAC_BIAS;
48         mydelay(0x0f);
49
50         val = 873 << 6;
51         DACR = (val) | DAC_BIAS;
52         mydelay(0x0f);
53
54         val = 926 << 6;
55         DACR = (val) | DAC_BIAS;
56         mydelay(0x0f);
57
58         val = 968 << 6;

```

```

59     DACR = (val) | DAC_BIAS;
60     mydelay(0x0f);
61
62     val = 998 << 6;
63     DACR = (val) | DAC_BIAS;
64     mydelay(0x0f);
65
66     val = 1017 << 6;
67     DACR = (val) | DAC_BIAS;
68     mydelay(0x0f);
69
70     val = 1023 << 6;
71     DACR = (val) | DAC_BIAS;
72     mydelay(0x0f);
73
74     val = 1017 << 6;
75     DACR = (val) | DAC_BIAS;
76     mydelay(0x0f);
77
78     val = 998 << 6;
79     DACR = (val) | DAC_BIAS;
80     mydelay(0x0f);
81
82     val = 968 << 6;
83     DACR = (val) | DAC_BIAS;
84     mydelay(0x0f);
85
86     val = 926 << 6;
87     DACR = (val) | DAC_BIAS;
88     mydelay(0x0f);
89
90     val = 873 << 6;
91     DACR = (val) | DAC_BIAS;
92     mydelay(0x0f);
93
94     val = 808 << 6;
95     DACR = (val) | DAC_BIAS;
96     mydelay(0x0f);
97
98     val = 742 << 6;
99     DACR = (val) | DAC_BIAS;
100    mydelay(0x0f);
101
102    val = 665 << 6;
103    DACR = (val) | DAC_BIAS;
104    mydelay(0x0f);
105
106    val = 591 << 6;
107    DACR = (val) | DAC_BIAS;
108    mydelay(0x0f);
109
110    val = 512 << 6;
111    DACR = (val) | DAC_BIAS;
112    mydelay(0x0f);
113
114    val = 436 << 6;
115    DACR = (val) | DAC_BIAS;
116    mydelay(0x0f);
117
118    val = 359 << 6;
119    DACR = (val) | DAC_BIAS;
120    mydelay(0x0f);
121

```

```

122     val = 282 << 6;
123     DACR = (val) | DAC_BIAS;
124     mydelay(0x0f);
125
126     val = 216 << 6;
127     DACR = (val) | DAC_BIAS;
128     mydelay(0x0f);
129
130     val = 211 << 6;
131     DACR = (val) | DAC_BIAS;
132     mydelay(0x0f);
133
134     val = 151 << 6;
135     DACR = (val) | DAC_BIAS;
136     mydelay(0x0f);
137
138     val = 97 << 6;
139     DACR = (val) | DAC_BIAS;
140     mydelay(0x0f);
141
142     val = 55 << 6;
143     DACR = (val) | DAC_BIAS;
144     mydelay(0x0f);
145
146     val = 25 << 6;
147     DACR = (val) | DAC_BIAS;
148     mydelay(0x0f);
149
150     val = 6 << 6;
151     DACR = (val) | DAC_BIAS;
152     mydelay(0x0f);
153
154     val = 0 << 6;
155     DACR = (val) | DAC_BIAS;
156     mydelay(0x0f);
157
158     val = 6 << 6;
159     DACR = (val) | DAC_BIAS;
160     mydelay(0x0f);
161
162     val = 25 << 6;
163     DACR = (val) | DAC_BIAS;
164     mydelay(0x0f);
165
166     val = 55 << 6;
167     DACR = (val) | DAC_BIAS;
168     mydelay(0x0f);
169
170     val = 97 << 6;
171     DACR = (val) | DAC_BIAS;
172     mydelay(0x0f);
173
174     val = 151 << 6;
175     DACR = (val) | DAC_BIAS;
176     mydelay(0x0f);
177
178     val = 211 << 6;
179     DACR = (val) | DAC_BIAS;
180     mydelay(0x0f);
181
182     val = 216 << 6;
183     DACR = (val) | DAC_BIAS;
184     mydelay(0x0f);

```

```

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

Code - Task-3

```

1  #include "LPC214x.h"
2  #define M_PI 3.14159265359
3  #define ITERATIONS 1000
4  #define FACTOR 0x3FF
5  #define DAC_BIAS 0x00010000
6
7  void DACInit( void )
8  {
9      /* setup the related pin to DAC output */
10     PINSEL1 &= 0xFFFF3FFFF;
11     PINSEL1 |= 0x00080000;      /* set p0.25 to DAC output */
12     return;
13 }
14 void mydelay(int x)
15 {
16     int j,k;
17     for(j=0;j<=x;j++)
18     {
19         for(k=0;k<=0xFF;k++);
20     }
21 }
22
23 int main()
24 {
25     // main routine
26     int val = 387;
27     DACInit();
28     while(1)
29     {
30         DACR=(val<<6) | DAC_BIAS;
31         mydelay(0x0f);
32     }
33     return 0;
34 }

```

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

```

1  #include "LPC214x.h"
2  #define M_PI 3.14159265359
3  #define ITERATIONS 1000
4  #define FACTOR 0x3FF
5  #define DAC_BIAS 0x00010000
6
7  void DACInit( void )
8  {
9      /* setup the related pin to DAC output */
10     PINSEL1 &= 0xFFFF3FFFF;
11     PINSEL1 |= 0x00080000;      /* set p0.25 to DAC output */
12     return;

```

```

13 }
14 void mydelay(int x)
15 {
16     int j,k;
17     for(j=0;j<=x;j++)
18     {
19         for(k=0;k<=0xFF;k++);
20     }
21 }
22
23 int main()
24 {
25     // main routine
26     int val=0;
27     int dx=100;
28     int i=0;
29     DACInit();
30     while(1)
31     {
32         DACR=(val<<6) | DAC_BIAS;
33         mydelay(0x0f);
34         val+=dx;
35         if(val>0x3FF-100 || val < 100)
36         {
37             dx=-dx;
38         }
39     }
40     return 0;
41 }
42

```

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

```
1  #include "LPC214x.h"
2  #define M_PI 3.14159265359
3  #define ITERATIONS 1000
4  #define FACTOR 0x3FF
5  #define DAC_BIAS 0x00010000
6
7  void DACInit( void )
8  {
9      /* setup the related pin to DAC output */
10     PINSEL1 &= 0xFFFF3FFFF;
11     PINSEL1 |= 0x00080000;      /* set p0.25 to DAC output */
12     return;
13 }
14 void mydelay(int x)
15 {
16     int j,k;
17     for(j=0;j<=x;j++)
18     {
19         for(k=0;k<=0xFF;k++);
20     }
21 }
22
23 int main()
24 {
25     // main routine
26     int val=0;
27     int dx=100;
28     int i=0;
29     DACInit();
30     while(1)
31     {
32         DACR=(val<<6) | DAC_BIAS;
33         mydelay(0x0f);
34         val+=dx;
35         val %= 739;
36     }
37     return 0;
38 }
39
```

Explanation :

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