# DAC Programming

ARM Cortex-M3 LPC1768 incorporate a 10-bit Digital to Analog Converter that provides buffered analog output. LPC1768 DAC has only 1 output pin, referred to as **AOUT**. The analog voltage at the output of this pin is given as:

$$\mathbf{V_{AOUT} = (VALUE * (V_{REFP}-V_{REFN})/1024) + V_{REFN}}$$

When we have $V_{REFN} = 0$:

$$\mathbf{V_{AOUT} = VALUE * V_{REF}/1024}$$

where VALUE is the 10-bit digital value which is to be converted into its analog counterpart and $V_{REF}$ is the input reference voltage.

## Pins relating to LPC1768 DAC block:

| Pin | Description |
|---|---|
| **AOUT (P0.26)** | Analog Output pin. Provides the converted Analog signal. Set Bits[21:20] in PINSEL1 register to [10] ($2^{nd}$ alternate function) to enable this function. |

### DAC Registers in ARM Cortex-M3 LPC1768:

The DAC module in ARM LPC1768 has 3 registers viz. DACR, DACCTRL, DACCNTVAL. Out of these DACCTRL, DACCNTVAL are related with DMA operation, and only DACR is used in the DAC programming. Hence DACCTRL, DACCNTVAL are not discussed here which are not included in the syllabus.

Also note that the DAC doesn't have a power control bit in PCONP register. Simply select the AOUT alternate function for pin P0.26 using PINSEL1 register to enable DAC output.

### The DACR register in LPC1768

The field containing bits [15:6] is used to feed a digital value which needs to be converted and bit 16 is used to select settling time. The bit significance is as shown below:

1. **Bit[5:0]:** Reserved.
2. **Bit[15:6] – VALUE:** After a new VALUE is written to this field, given settling time selected using BIAS has elapsed, we get the converted analog voltage at the output. The formula for analog voltage at AOUT pin is as shown above.
3. **Bit[16] – BIAS:** Setting this bit to 0 selects settling time of 1us max with max current consumption of 700uA at max 1Mhz update rate. Setting it to 1 will select settling time of 2.5us but with reduced max current consumption of 300uA at max 400Khz update rate.
4. **Bits[31:17]**: Reserved

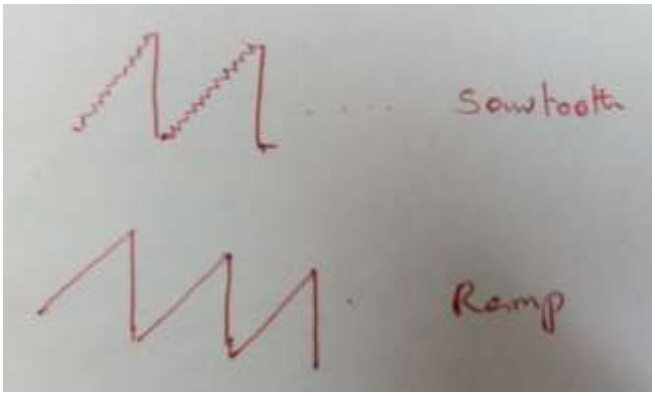
**ARM Cortex-M3 LPC1768 DAC example**

Let us see a basic LPC1768 DAC example. In this DAC example, we will be changing the output from 0V to $V_{REFP}$ ($V_{REFN}$=0V) and then falling back to 0V in steps of 10ms. Vrefp = 3.3 V. This DAC program will basically output a sawtooth waveform. We will be using BIAS = 0 i.e. settling time of 1us. You can connect an Oscilloscope or a Multimeter between P0.26 and GND to check the changing analog output. Since the output is buffered you can drive an LED from $A_{OUT}$ but it won't glow until it reaches its forward bias voltage of around 1.7 Volts. So, keep this in mind when checking for analog output using an LED.

The digital values from 0 to 1023 are sent to DACR with 10 millisecond delay between each value. So the converted analog voltage will be increasing in steps up to the maximum value of 3.3 V corresponding to 1023. These analog voltages are observed on a CRO as sawtooth. This process repeats continuously to get a sawtooth waveform. If there is no delay between the values, then we get ramp waveform.

```
#include <lpc17xx.h>

void delayMS(unsigned int milliseconds);

int main(void)

{

unsigned int value=0; //Binary value used for Digital to Analog Conversion

LPC_PINCON->PINSEL1 |= (1<<21); //Select AOUT function for P0.26 , bits[21:20] = [10]

while(1)

        {

                if(value > 1023) value=0; //For 10-bit DAC max-value is 2^10 - 1 = 1023

                LPC_DAC->DACR = (value<<6);

                delayMS(9999); //for 10 millisecond delay

                value++;

        }

}

void delayMS(unsigned int count) //Using Timer0

{
```

```
        LPC_TIM0->CTCR = 0x0; //Timer mode
        LPC_TIM0->PR = 2; //Increment TC at every 3 pclk
        LPC_TIM0->TCR = 0x02; //Reset Timer
        LPC_TIM0->TCR = 0x01; //Enable timer
        while(LPC_TIM0->TC < count); //wait until timer counter reaches the desired delay
        LPC_TIM0->TCR = 0x00; //Disable timer
}
```

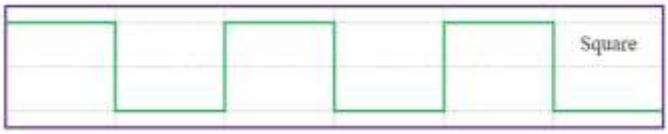



```
//To generate triangular wave:
#include <lpc17xx.h>
int main(void)
{
      unsigned int temp, i; //Binary value used for Digital to Analog Conversion
      LPC_PINCON->PINSEL1 |= (1<<21); //Select AOUT function for P0.26 , bits[21:20] = [10]
      while(1)
         {
            for(i=0;i!=1023;i++)    //keep on incrementing value from 00 till 1023 in step of 1
         {
             temp=i;
             LPC_DAC->DACR = (temp<<6);
```

```
        }
          for(i=1023; i!=0;i--) //keep on decrementing value from 1023 till 00 in step of 1
        {
         temp=i;
         LPC_DAC->DACR = (temp<<6);

       }

   }

}
```

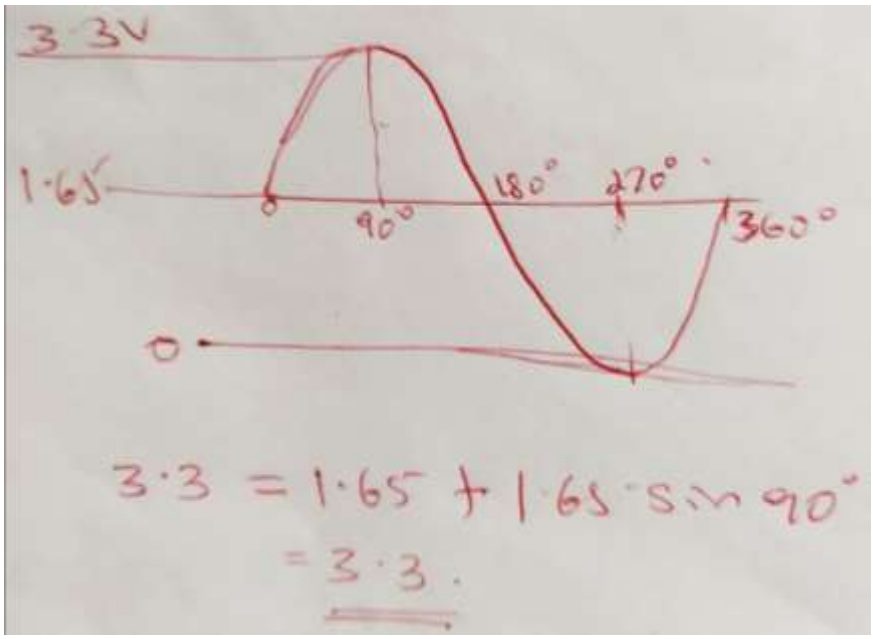

```
//To generate square wave

#include <lpc17xx.h>

void delayMS(unsigned int milliseconds);

int main(void)

{

    LPC_PINCON->PINSEL1 |= (1<<21); //Select AOUT function for P0.26 , bits[21:20] = [10]

     while(1)

        {

                LPC_DAC->DACR = (1023<<6);

                delayMS(9999); //for 10 millisecond delay

                LPC_DAC->DACR = (0<<6);

                delayMS(9999); //for 10 millisecond delay

        }

}

void delayMS(unsigned int count) //Using Timer0
```

```
{
        LPC_TIM0->CTCR = 0x0; //Timer mode
        LPC_TIM0->PR = 2; //Increment TC at every 3 pclk
        LPC_TIM0->TCR = 0x02; //Reset Timer
        LPC_TIM0->TCR = 0x01; //Enable timer
        while(LPC_TIM0->TC < count); //wait until timer counter reaches the desired delay
        LPC_TIM0->TCR = 0x00; //Disable timer
}
```




//To generate sinewave:

Find values for sine function which is given by Y = A+Asin(θ) where, A=1.65V because we are generating unipolar sinewave. Let the swing be from 0V to 3.3V and hence 3.3/2= 1.65V is the mid value.




We can assign 0v with 0, mid value 1.65V with 1023/2 = 511 and 3.3V with 1023. Considering 10 samples to construct one full cycle of sine wave each sample differ with its adjacent sample by an angle of 360/10 = 36 degree/sample.

Example:
for θ=36'
Y = A + Asin(θ)

= 1.65 + 1.65 * sin(36)

= 2.62V

for 3.3V ----------> 1023

then 2.62V ----------> ?

= (1023 * 2.62)/3.3

= 812

Similarly find all the 10 values for 72", 108'. 144', 180' and so on till 360'.

Convert these digital samples into analog values and observe the waveform on the CRO. The more the no. of samples the smother the waveform.

```
#include <lpc17xx.h>

#include <math.h>

#define pi 3.14

int main(void)

{

	unsigned int i, temp;

	float rad, value;

	LPC_PINCON->PINSEL1 |= (1<<21); //Select AOUT function for P0.26, bits[21:20] = [10]

	   while(1)

	   {

	      for(i=0;i<=360;i=i+9)    //keep on incrementing value from 00 till 360 in step of 9. 40

	                               //samples for one cycle

	    {

	       rad = sin(i*(pi/180));

	       value = 1.65 + 1.65*rad;

	       temp = (value*1023) /3.3;
```

```
            LPC_DAC->DACR = (temp<<6);
          }
        }
}
```