```c
#include <asf.h>
#include <math.h>

void wait(int t);
void configure_dac(void);
void configure_dac_clock(void);

Dac *DAC_Ptr = (Dac *)0x42004800UL;   //initialize the DAC pointer
//Gclk *Gclk_Ptr = (Gclk *)0x40000C00U;

void makeNoise(double scale, int length, Dac *DAC_Ptr);

void wait(int t){
 int counter = 0;

 while(counter<t){
        counter++;
 }
}

void configure_dac_clock(void)
{
 /* Turn on the digital interface clock */
 system_apb_clock_set_mask(SYSTEM_CLOCK_APB_APBC, PM_APBCMASK_DAC);

 /* Configure GCLK channel and enable clock */
 struct system_gclk_chan_config gclk_chan_conf;
 system_gclk_chan_get_config_defaults(&gclk_chan_conf);
 gclk_chan_conf.source_generator = GCLK_GENERATOR_0;
 system_gclk_chan_set_config(DAC_GCLK_ID, &gclk_chan_conf);
 system_gclk_chan_enable(DAC_GCLK_ID);
 //GCLK->GENCTRL.reg=
 //GCLK->GENDIV.reg=0b000000000000001100000000; // Div 4
 //GCLK->GENDIV.reg=0b000000000000001000000000; // Div 3
 //GCLK->GENDIV.reg=0b000000000000000100000000; // Div 2
}

void configure_dac(void)
{
 //set pin as output for the dac
 Port *ports = PORT_INSTS;
 PortGroup *por = &(ports->Group[0]);

 por->DIRSET.reg = PORT_PA02|PORT_PA13;
 por->OUTSET.reg = PORT_PA02|PORT_PA13;
 por->PINCFG[13].bit.DRVSTR = 0x1;

 por->PINCFG[2].bit.PMUXEN = 0x1;    //set to correct pin configuration
 por->PMUX[1].bit.PMUXE = 0x1;       //set to correct peripheral

 while (DAC_Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {}      /* Wait until the synchronization is complete */

 DAC_Ptr->CTRLB.reg = 0b01000000;    /* Set reference voltage with CTRLB */

 while (DAC_Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {}      /* Wait until the synchronization is complete */

 DAC_Ptr->CTRLA.reg = 0b00000010;    /* Enable the module with CTRLA */
 DAC_Ptr->CTRLB.reg = 0b01000001;

}

void makeNoise(double scale, int length, Dac *Ptr){
 int counter = 0;
 int forever = 0;
 int k = 0;
 int range = 363;
 int rScale1 = range * scale;
 if(length>9){
        forever = 1;
 }
 length = length * 111111;
 int sineArray1[rScale1];

 for(int i=0;i<rScale1;i++){         //Populate an array with current sine array
        sineArray1[i]=(sin(((i*(0.9917/scale))/180)*3.14159265)*511) + 511; //Conversion to radian and scaled to go between 0-3.3V
 }
 if(forever==0){
        while(k<length){
               if(counter == (rScale1)){
                      counter = 0;
               }
               while (Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {}   /* Wait until the synchronization is complete */

               Ptr->DATA.reg = sineArray1[counter]; /* Write the new value to the DAC DATA register */

               counter++;
               k++;
        }
 }
 if(forever==1){
        while(1){
               if(counter == (rScale1)){
                      counter = 0;
               }
               while (Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {}   /* Wait until the synchronization is complete */
```

```

            Ptr->DATA.reg = sineArray1[counter]; /* Write the new value to the DAC DATA register */

            counter++;
            }
    }

}

int main(void)
{
    system_init();
    configure_dac_clock();
    configure_dac();

    while(1){

            //makeNoise(2.05,9,DAC_Ptr); //A
            //makeNoise(1.85,9,DAC_Ptr); //B
            //makeNoise(1.75,9,DAC_Ptr); //C
            //makeNoise(1.56,9,DAC_Ptr); //D
            //makeNoise(1.4,9,DAC_Ptr);         //E
            //makeNoise(1.31,9,DAC_Ptr); //F
            //makeNoise(1.17,9,DAC_Ptr); //G

            makeNoise(1,10,DAC_Ptr);     //makeNoise(Frequency Scaler, Length, DAC_Ptr)
                                                                //Note, if length is 10 or greater, length is infinite.
                                                                //As frequency scaler increases, frequency decreases.


```

```
                //makeNoise(2.05,1,DAC_Ptr);                //If code is uncommented, will play "happy birthday"
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(1.85,2,DAC_Ptr);
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(1.56,2,DAC_Ptr);
                //makeNoise(1.64,3,DAC_Ptr);
    //
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(1.85,2,DAC_Ptr);
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(1.4,2,DAC_Ptr);
                //makeNoise(1.56,3,DAC_Ptr);
        //
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(2.05,1,DAC_Ptr);
                //makeNoise(1.0,2,DAC_Ptr);
                //makeNoise(1.22,2,DAC_Ptr);
                //makeNoise(1.56,1,DAC_Ptr);
                //makeNoise(1.72,1,DAC_Ptr);
                //makeNoise(1.85,2,DAC_Ptr);
                        //
                //makeNoise(1.17,1,DAC_Ptr);
                //makeNoise(1.17,1,DAC_Ptr);
                //makeNoise(1.22,2,DAC_Ptr);
                //makeNoise(1.56,1,DAC_Ptr);
                //makeNoise(1.4,2,DAC_Ptr);
                //makeNoise(1.56,3,DAC_Ptr);


        }

        return 0;
```

```
}


```