

```

1. #include <asf.h>
2. #include <math.h>
3.
4. void wait(int t);
5. void configure_dac(void);
6. void configure_dac_clock(void);
7.
8. Dac *DAC_Ptr = (Dac *)0x42004800UL; //initialize the DAC pointer
9. //Gclk *Gclk_Ptr = (Gclk *)0x40000C00U;
10.
11. void makeNoise(double scale, int length, Dac *DAC_Ptr);
12.
13. void wait(int t){
14.     int counter = 0;
15.
16.     while(counter<t){
17.         counter++;
18.     }
19. }
20.
21. void configure_dac_clock(void)
22. {
23.     /* Turn on the digital interface clock */
24.     system_apb_clock_set_mask(SYSTEM_CLOCK_APB_APBC, PM_APBCMASK_DAC);
25.
26.     /* Configure GCLK channel and enable clock */
27.     struct system_gclk_chan_config gclk_chan_conf;
28.     system_gclk_chan_get_config_defaults(&gclk_chan_conf);
29.     gclk_chan_conf.source_generator = GCLK_GENERATOR_0;
30.     system_gclk_chan_set_config(DAC_GCLK_ID, &gclk_chan_conf);
31.     system_gclk_chan_enable(DAC_GCLK_ID);
32.     //GCLK->GENCTRL.reg=
33.     //GCLK->GENDIV.reg=0b00000000000000001100000000; // Div 4

```

```

34. //GCLK->GENDIV.reg=0b000000000000001000000000; // Div 3
35. //GCLK->GENDIV.reg=0b000000000000000100000000; // Div 2
36. }
37.
38. void configure_dac(void)
39. {
40. //set pin as output for the dac
41. Port *ports = PORT_INSTS;
42. PortGroup *por = &(amp;ports->Group[0]);
43.
44. por->DIRSET.reg = PORT_PA02|PORT_PA13;
45. por->OUTSET.reg = PORT_PA02|PORT_PA13;
46. por->PINCFG[13].bit.DRVSTR = 0x1;
47.
48. por->PINCFG[2].bit.PMUXEN = 0x1; //set to correct pin configuration
49. por->PMUX[1].bit.PMUXE = 0x1; //set to correct peripheral
50.
51. while (DAC_Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {} /* Wait until the
    synchronization is complete */
52.
53. DAC_Ptr->CTRLB.reg = 0b01000000; /* Set reference voltage with CTRLB */
54.
55. while (DAC_Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {} /* Wait until the
    synchronization is complete */
56.
57. DAC_Ptr->CTRLA.reg = 0b00000010; /* Enable the module with CTRLA */
58. DAC_Ptr->CTRLB.reg = 0b01000001;
59.
60. }
61.
62. void makeNoise(double scale, int length, Dac *Ptr){
63. int counter = 0;
64. int forever = 0;
65. int k = 0;

```

```

66. int range = 363;
67. int rScale1 = range * scale;
68. if(length>9){
69.     forever = 1;
70. }
71. length = length * 111111;
72. int sineArray1[rScale1];
73.
74. for(int i=0;i<rScale1;i++){           //Populate an array with current sine array
75.     sineArray1[i]=(sin(((i*(0.9917/scale))/180)*3.14159265)*511) + 511; //Conversion
    to radian and scaled to go between 0-3.3V
76. }
77. if(forever==0){
78.     while(k<length){
79.         if(counter == (rScale1)){
80.             counter = 0;
81.         }
82.         while (Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {} /* Wait until the
    synchronization is complete */
83.
84.         Ptr->DATA.reg = sineArray1[counter]; /* Write the new value to the DAC
    DATA register */
85.
86.         counter++;
87.         k++;
88.     }
89. }
90. if(forever==1){
91.     while(1){
92.         if(counter == (rScale1)){
93.             counter = 0;
94.         }
95.         while (Ptr->STATUS.reg & DAC_STATUS_SYNCBUSY) {} /* Wait until the
    synchronization is complete */

```

```

96.
97.          Ptr->DATA.reg = sineArray1[counter]; /* Write the new value to the DAC
          DATA register */
98.
99.          counter++;
100.         }
101.     }
102.
103. }
104.
105. int main(void)
106. {
107.     system_init();
108.     configure_dac_clock();
109.     configure_dac();
110.
111.     while(1){
112.
113.         //makeNoise(2.05,9,DAC_Ptr); //A
114.         //makeNoise(1.85,9,DAC_Ptr); //B
115.         //makeNoise(1.75,9,DAC_Ptr); //C
116.         //makeNoise(1.56,9,DAC_Ptr); //D
117.         //makeNoise(1.4,9,DAC_Ptr);      //E
118.         //makeNoise(1.31,9,DAC_Ptr); //F
119.         //makeNoise(1.17,9,DAC_Ptr); //G
120.
121.         makeNoise(1,10,DAC_Ptr);    //makeNoise(Frequency Scaler, Length,
          DAC_Ptr)
122.
          //Note, if length is
          10 or greater, length is infinite.
123.
          //As frequency scaler
          increases, frequency decreases.
124.
125.

```

```

126.          //makeNoise(2.05,1,DAC_Ptr);          //If code is uncommented,
    will play "happy birthday"
127.          //makeNoise(2.05,1,DAC_Ptr);
128.          //makeNoise(1.85,2,DAC_Ptr);
129.          //makeNoise(2.05,1,DAC_Ptr);
130.          //makeNoise(1.56,2,DAC_Ptr);
131.          //makeNoise(1.64,3,DAC_Ptr);
132.  //
133.          //makeNoise(2.05,1,DAC_Ptr);
134.          //makeNoise(2.05,1,DAC_Ptr);
135.          //makeNoise(1.85,2,DAC_Ptr);
136.          //makeNoise(2.05,1,DAC_Ptr);
137.          //makeNoise(1.4,2,DAC_Ptr);
138.          //makeNoise(1.56,3,DAC_Ptr);
139.  //
140.          //makeNoise(2.05,1,DAC_Ptr);
141.          //makeNoise(2.05,1,DAC_Ptr);
142.          //makeNoise(1.0,2,DAC_Ptr);
143.          //makeNoise(1.22,2,DAC_Ptr);
144.          //makeNoise(1.56,1,DAC_Ptr);
145.          //makeNoise(1.72,1,DAC_Ptr);
146.          //makeNoise(1.85,2,DAC_Ptr);
147.          //
148.          //makeNoise(1.17,1,DAC_Ptr);
149.          //makeNoise(1.17,1,DAC_Ptr);
150.          //makeNoise(1.22,2,DAC_Ptr);
151.          //makeNoise(1.56,1,DAC_Ptr);
152.          //makeNoise(1.4,2,DAC_Ptr);
153.          //makeNoise(1.56,3,DAC_Ptr);
154.
155.
156.    }
157.
158.    return 0;

```

159. }

160.

161.