```
/*Date: 21/02/2021
                                Variable Frequency Sine Wave Generator

Submitted BY:
    Pratyush Jaiswal (18EE30021)
    Nuruddin Jiruwala (18EE30029)



Logic Used:
        Base Frequency(Highest Frequency): F_CPU/(256*resolution)
        Prescaler: 1    (Since, no prescaling)
        From the base frequency and frequency Factor we can get the required frequency using the below formula:
            Frequency: (Base Frequency/Freqfactor)
        Frequency Factor and number of ramps will get the get the same value from the sine lookup table so the
        wave gets stretched and the frequency gets decreased to the required value.

        The number of cycles, a particular wave can be outputted can be precalculated form the product of
        Time for which the wave is being generated and the frequency of that wave(from the definition of Frequency).*/


#define duty R17                        ; value of sineLookUp at the current index of resolution count
#define resolutionCount R19             ; Temporary resolution count for being compared with resolution (0 to resolution-1)
#define tempCycleCount R20              ; Temporary cycle count for being compared with currCycle (0 to currCycle-1)

#define currFreqFactor R21              ; Frequency of the wave at index
#define currCycle R22                   ; Total number of cycles for which the current wave with current index
#define index R23                       ; For storing the current wave index
#define count R18                       ; Compare with currFreqFactor


.ORG 0x0
    JMP MAIN
.ORG 0x20
    JMP overflow_isr

MAIN:

    ; Initializing the stack
    LDI R16,HIGH(RAMEND)
    OUT SPH,R16
    LDI R16,LOW(RAMEND)
    OUT SPL,R16
```

```
44
45      ; Initializing stack is done
46      clr tempCycleCount
47      CALL loadLookUp                         ; loading the Sine LookUp
          Table for the first time
48
49      SBI DDRD,6                              ; Setting PD6 as output
50
51      LDI duty, 63                             ; just for initialisation
           sake
52      OUT OCR0A,duty                              ; Loading Timer0 with
          127
53
54      ; clearing all the temporary counters
55      clr count
56      clr resolutionCount
57      clr tempCycleCount
58
59      ; Loading first wave by calling loadwave
60      call loadWave
61
62      LDI R16,(1<<WGM01)  | (1<<WGM00) |
          (1<<COM0A1)                       ;Setting timer mode to fast PWM
63      OUT TCCR0A, R16
64
65      LDI R16, (1<<CS00)                      ;Start Timer0 - prescaler
          = (no prescaling)
66      OUT TCCR0B, R16
67
68      LDI R16, (1<<TOIE0)                     ;Setting the ISR vector
69      STS TIMSK0, R16                         ;Enable Timer0 compare
          match interrupt
70
71      SEI                                     ;Enable global interrupts
72
73  Again:
74      CP tempCycleCount, currCycle            ; comparing the tempCycleCount
           with the number of cycles assigned for the current wave(index)
75      brne Repeat
76      call loadWave                           ; loading the next wave
          (incrementing the index)
77
78      Repeat:
79          cp count, currFreqFactor            ; checking the number of
              number of ramps required for current angle value in the Sine
              LookUpTable
80          brne Repeat2                        ; if not equal, then don't
              change the current PWM value
81          LPM duty, Z+                        ; if equalt, then load with
              the next value in sine LookUp Table
82          clr count                           ; Reset number of ramps
83          inc resolutionCount                 ; increase index in sineLookUp
               Table
```

```
84          Repeat2:
85              out OCR0A, duty                 ; Give output PWM
86              cpi resolutionCount, resolution ; checking if the one cycle of current wave is completed
87              brne Again                      ; if not equal then complete the current cycle
88              call loadLookUp                 ; if equal then load the new cycle by calling LookUp Table again,
89                                              ; and resetting resolutionCount and incrementing tempCycleCount
90      JMP Again                               ; Repeat the phenomena infintite times
91
92  ; For getting a new wave form the Freqfactor
93  /*Here we used only one dataframe as we could access memory through Z only, two functions could be
94  implemented and Z be pushed onto the stack multiple times to achieve this using 2 dataframes,
95  but this method felt simpler and the values are taken pairwise i.e. freqfactor and number of cycles*/
96  loadWave:
97      ; clear the parameters for a new wave
98      clr tempCycleCount
99      clr count
100
101     ; Pushing Z indirect register onto Stack (it is used for loading SineLookUp that is why I have to push it)
102     PUSH ZL
103     PUSH ZH
104
105     ; Loading with the Freqfactor data
106     LDI ZL, LOW(2*freqFactor)
107     LDI ZH, HIGH(2*freqFactor)
108
109
110     CPI index, freqCount        ; comapring and checking if the total waves have been outputted
111     brne updateWave             ; if not equal then go for the remaining waves
112     CLR index                   ; if equal then start from the beginning of the data
113
114     ; Loading parameters for the wave at current index
115     updateWave:
116         add ZL, index
117         LDI R16, 0
118         adc ZH, R16
119         LPM currFreqFactor, Z+
120         LPM currCycle, Z+
121
122         ; incrementing index twice because the two parameters (currFreqFactor and currCycle) for a wave are stored sequentially
123         inc index
```

```
124         inc index
125
126         ; Popping out the Z register from the Stack for lookUpTable
127         POP ZH
128         POP ZL
129         ret
130
131 ; for loading SineLookUp Table
132 loadLookUp:
133     clr resolutionCount         ; since a new cycle is starting, reset the
           resolutionCount(index of lookUpTable)
134     ldi ZL, LOW(2*sineLookUp)
135     ldi ZH, HIGH(2*sineLookUp)
136     inc tempCycleCount  ; incrementing the current number of cycles
137     ret
138
139 overflow_isr:
140     inc count           ; incrementing the number of ramps
141     reti
142
143 ; for storing the LookUpTable of a Sine Wave with 64 resolution
144 sineLookUp:
145         .DB 128, 140, 153, 165, 177, 188, 199, 209, 219, 227, 235, 241,
              246, 250, 253, 255, 255, 254, 252, 248, 244, 238, 231, 223, 214,
              204, 194, 183, 171, 159, 147, 134, 121, 108, 96, 84, 72, 61, 51,
              41, 32, 24, 17, 11, 7, 3, 1, 0, 0, 2, 5, 9, 14, 20, 28, 36, 46,
              56, 67, 78, 90, 102, 115, 127
146         .EQU resolution = 63
147
148 ; for storing the frequencies(Frequency Factor) and Number of Cycles
149 freqFactor:
150         .DB 20, 4, 2, 6, 10, 2, 4, 10
151         .EQU freqCount = 8
152
153
154 /*For the Proteus Simulation:
155     We are taking the output of the PWM from PD6 clipping the output with
          1V(For getting it into the range of -1 to 1 as it
156     was being generated from 0 to 2 inside the microcontoller) and after
          that passing through a Low Pass Filter for getting the Sine Wave
          from generated PWM.
157
158     A speaker is also attached for getting the audio version of the signal
           being generated after being amplified through an inverting
          Amplifier
159     because of the threshold voltage of the speaker*/
```