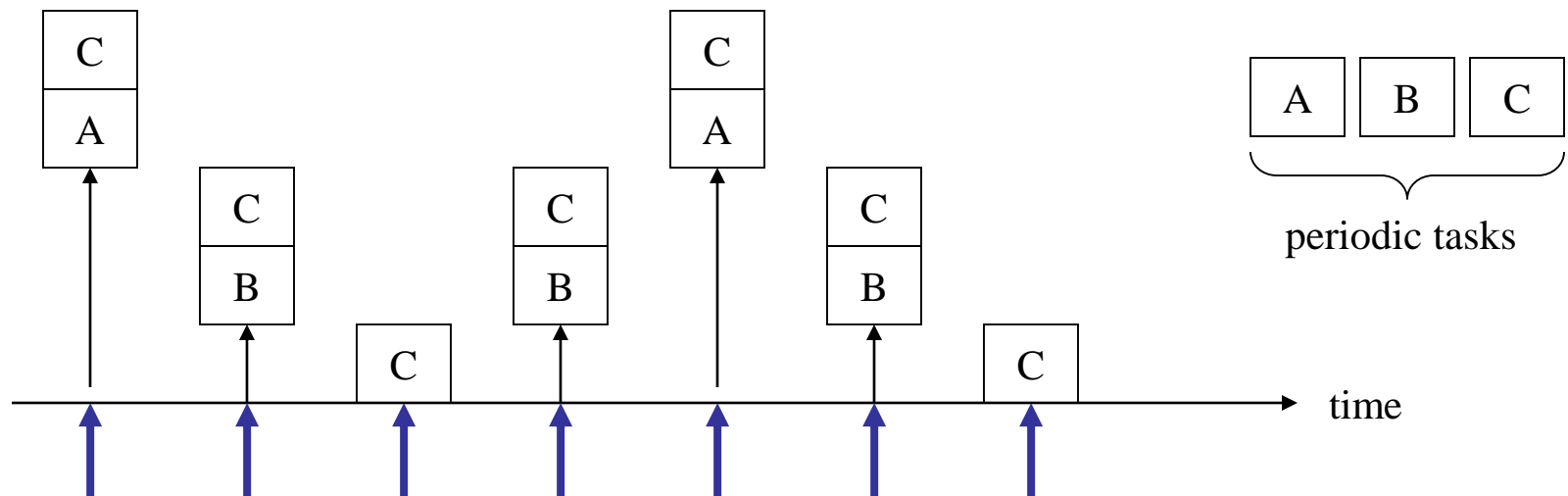




Sound Control

Outline

- How to control the buzzer (蜂鳴器) to play required sound?
- How to generate a signal with required frequency?
 - play a sound with fixed frequency
- How to play a song?
 - the frequency changes with time

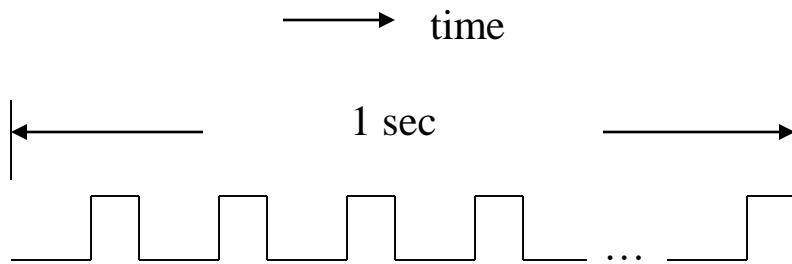




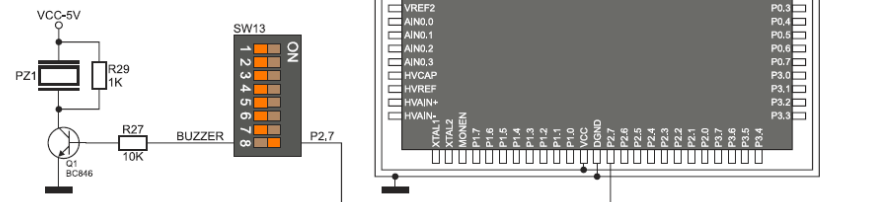
How to control the buzzer

How to control the buzzer

- control the sound by the frequency f of the input signal to the buzzer



f = number of pulses in one second





Appendix: frequency of tones

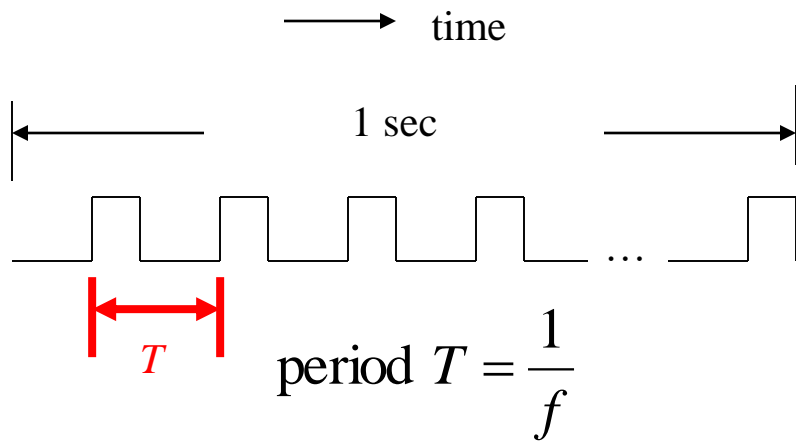
表 26-1 C 調各音階之頻率表

音階		DO	RE	MI	FA	SO	LA	SI
高音	簡符	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$	$\dot{6}$	$\dot{7}$
	頻率(Hz)	522	587	659	700	784	880	988
中音	簡符	1	2	3	4	5	6	7
	頻率(Hz)	262	294	330	349	392	440	494
低音	簡符	$\underset{\cdot}{1}$	$\underset{\cdot}{2}$	$\underset{\cdot}{3}$	$\underset{\cdot}{4}$	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$
	頻率(Hz)	131	147	165	175	196	220	247

Generating a signal with given frequency



How to send out regular pulses with given frequency

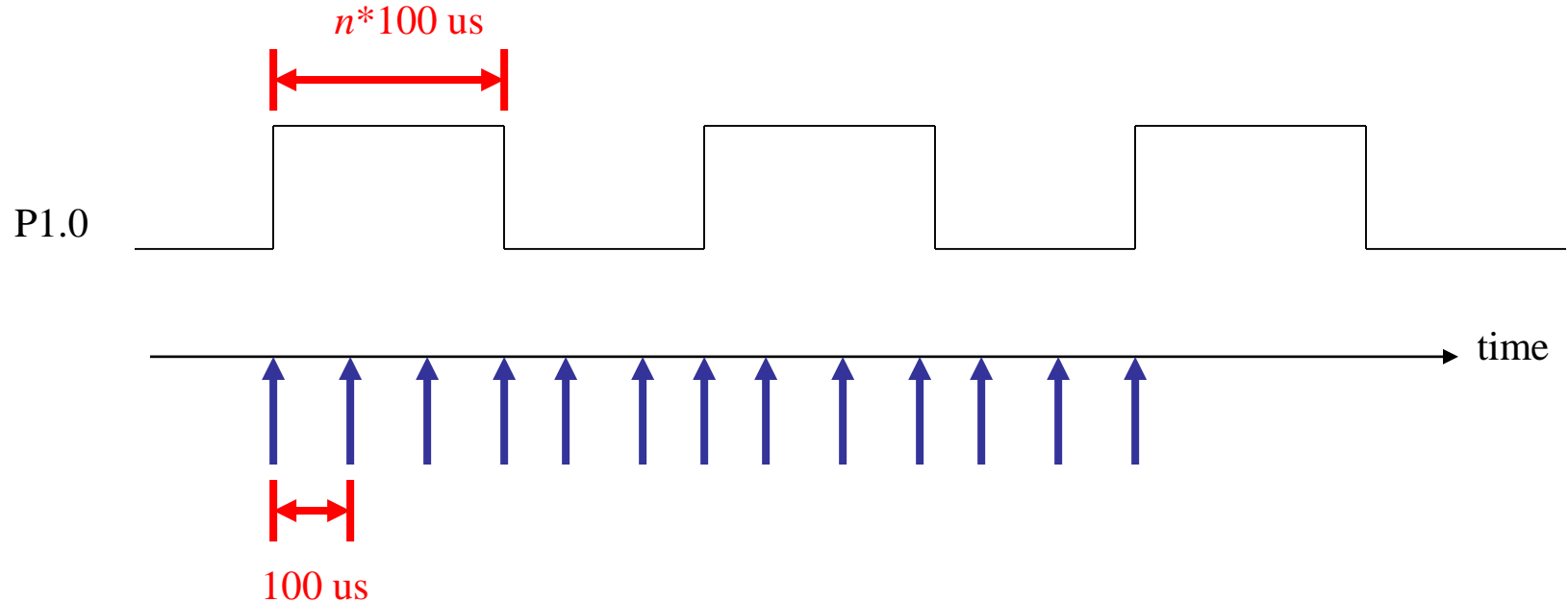


```
main () {  
    while (1) {  
        status = ~status;  
        P2.7 = status;  
        delay (N);  
    }  
}
```

- It's difficult to fine-tune delay count N to match $T/2$ (if you use a for-loop)
- This method is not recommended!

The Problem

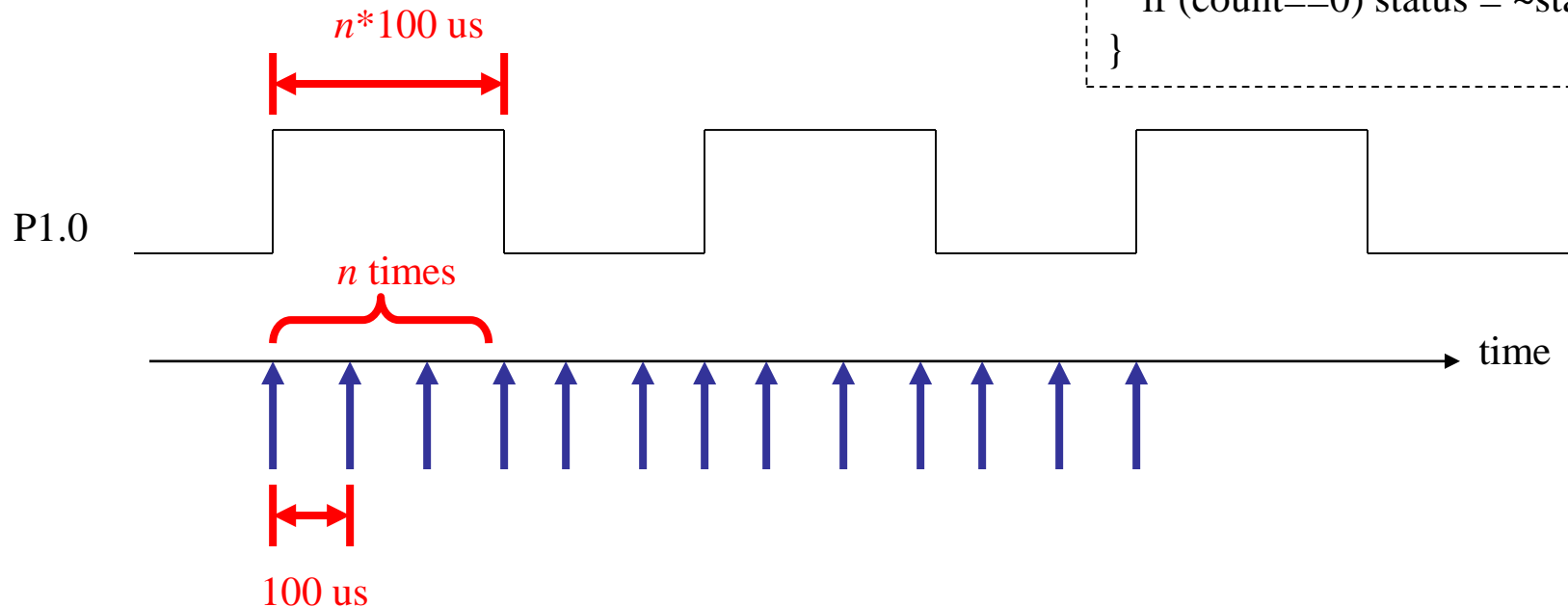
- Frequency generation using timer interrupt
 - Suppose: the period for timer interrupt is **100us**
 - Question: How to generate a signal with **half-period** $n*100\text{us}$?



The method

- just reverse the signal every n times the timer ISR is called

```
int n;  
int count=0;  
char status;  
  
main ()  
{  
    while(1) P2.7 = status;  
}  
  
Timer_ISR ()  
{  
    count = (count+1)%n;  
    if (count==0) status = ~status;  
}
```





How to play a song?

the frequency changes with time



The method

- Store the 歌譜 as an array of **half-period**
 - $\text{Song_Table}[i]$ = the half-period of the i -th tone
- Use two timer interrupts
 - Timer0_ISR: to control the signal frequency by half-period n
 - Timer1_ISR: change the half-period (n) periodically
 - $n = \text{Song_Table}[i++]$



The method

```
int Song_Table[3];

main ()
{
    //setup timer interrupts

    //initialize
    Song_Table [0] = Half_Period (Do);
    Song_Table [1] = Half_Period (Re);
    Song_Table [2] = Half_Period (Mi);
    ...
    while (1) P2.7=status;
}
```

```
int n;
int count;
char status;

Timer0_ISR () //to control signal frequency
{
    count = (count+1)%n;
    if (count==0) status=~status;
}

int i;

Timer1_ISR () //to change the tone
{
    //counting for 1 second...
    n = Song_Table [i];
    i = (i+1)%3;
}
```



The method

```
int Song_Table[3];

main ()
{
    //setup timer interrupts
```

```
    //initialize
```

```
    Song_Table [0] = Half_Period (Do);
    Song_Table [1] = Half_Period (Re);
    Song_Table [2] = Half_Period (Mi);
```

```
    ...
```

```
    while (1) P2 = status;
```

```
}
```

Half period of each tone is
stored in an array

```
int n;
int count;
char status;
```

```
Timer0_ISR () //to control signal frequency
```

```
{
    count = (count+1)%n;
    if (count==0) status=~status;
}
```

```
int i;
```

```
Timer1_ISR () //to change the tone
```

```
{
    //counting for 1 second...
    n = Song_Table [i];
    i = (i+1)%3;
}
```



The method

```
int Song_Table[3];

main ()
{
    //setup timer interrupts

    //initialize
    Song_Table [0] = Half_Period (Do);
    Song_Table [1] = Half_Period (Re);
    Song_Table [2] = Half_Period (Mi);
    ...
    while (1) P2.7=status;
}
```

```
int n;
int count;
char status;
```

Frequency generator
according to half-period n

```
Timer0_ISR () //to control signal frequency
{
    count = (count+1)%n;
    if (count==0) status=~status;
}
```

```
int i;
```

```
Timer1_ISR () //to change the tone
{
    //counting for 1 second...
    n = Song_Table [i];
    i = (i+1)%3;
}
```

The method

```
int Song_Table[3];
```

```
main ()
```

```
{
```

```
    //setup timer interrupts
```

```
    //initialize
```

```
    Song_Table [0] = Half_Period (Do);
```

```
    Song_Table [1] = Half_Period (Re);
```

```
    Song_Table [2] = Half_Period (Mi);
```

```
    ...
```

```
    while (1) P2.7=status;
```

```
}
```

changes half-period n of the generated signal every second

```
int n;
```

```
int count;
```

```
char status;
```

Frequency generator according to half-period n

```
Timer0_ISR () //to control signal frequency
```

```
{
```

```
    count = (count+1)%n;
```

```
    if (count==0) status=~status;
```

```
}
```

```
int i;
```

```
Timer1_ISR () //to change the tone
```

```
{
```

```
    //counting for 1 second...
```

```
    n = Song_Table [i];
```

```
    i = (i+1)%3;
```

```
}
```