Sleep #include<windows.h>

The **Sleep** function suspends the execution for specified milliseconds.

```
void Sleep( unsigned int unMilliseconds );
```

Example:

```
#include <iostream.h>
#include <windows.h>
int main()
{
    cout << "Start\nWait 3 seconds\n";
    cout.flush();   // Output ostream before next instruction
    Sleep(3000);    // Wait for 3 Seconds
    cout << "Wait 7.5 more seconds\n";
    cout.flush();   // Output ostream before next instruction
    Sleep(7500);    // Wait for 7.5 Seconds
    cout << "Bye\n\n";
    return 0;
}
```

Beep #include<windows.h>

The **Beep** function generates simple tones on the speaker. The function is synchronous; it does not return control to its caller until the sound finishes.

```
boolean Beep(
  unsigned int unFreq,     // sound frequency, in hertz
  unsigned int unDuration // sound duration, in milliseconds  );
```

Parameters

unFreq **Windows NT+:** Specifies the frequency, in hertz, of the sound.
unDuration **Windows NT+:** Specifies the duration, in milliseconds, of the sound.
Windows 95: The **Beep** function ignores the *dwFreq* and *dwDuration* parameters.

Return Values

If the function succeeds, the return value is nonzero else return value is zero.

Example:

```
#include <windows.h>
int main()
{
       int nFreq=400, nDuration=1000;
       Beep(nFreq, nDuration);
       Beep(1000, 500);
       Beep(800, 2000);
       return 0;
}
```

MessageBeep

The **MessageBeep** function plays a waveform sound. The waveform sound for each sound type is identified by an entry in the [sounds] section of the registry.

```
BOOL MessageBeep(
  UINT uType   // sound type
);
```

Parameters

uType

Specifies the sound type, as identified by an entry in the [sounds] section of the registry. This parameter can be one of the following values:

Value	Sound
0xFFFFFFFF	Standard beep using the computer speaker
MB_ICONASTERISK	SystemAsterisk
MB_ICONEXCLAMATION	SystemExclamation
MB_ICONHAND	SystemHand
MB_ICONQUESTION	SystemQuestion
MB_OK	SystemDefault

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

Remarks

After queuing the sound, the **MessageBeep** function returns control to the calling function and plays the sound asynchronously.

If it cannot play the specified alert sound, **MessageBeep** attempts to play the system default sound. If it cannot play the system default sound, the function produces a standard beep sound through the computer speaker.

The user can disable the warning beep by using the Sound Control Panel application.

sndPlaySound #include<windows.h>

The **sndPlaySound** function plays a waveform sound specified either by the filename.
If you get link errors, the winmm.lib file will have to be added to project. (See Notes)

```
boolean sndPlaySound( char[] szFileName, int nSound );
```

Parameters

szFileName

A string that specifies the name of a waveform-audio file.
If NULL, any currently playing sound is stopped.

nSound

Flags for playing the sound. The following values are defined:

`SND_ASYNC`

Sound is played asynchronously and function returns after beginning the sound. To terminate, call **sndPlaySound** with ***szFileName*** set to NULL.

`SND_LOOP`

Sound plays repeatedly until **sndPlaySound** is called again with *szFileName* parameter set to NULL. Must also specify SND_ASYNC flag to loop sounds.

`SND_NODEFAULT`

If the sound cannot be found, function returns without playing the default sound.

`SND_NOSTOP`

If a sound is currently playing, the function immediately returns FALSE, without playing the requested sound.

`SND_SYNC`

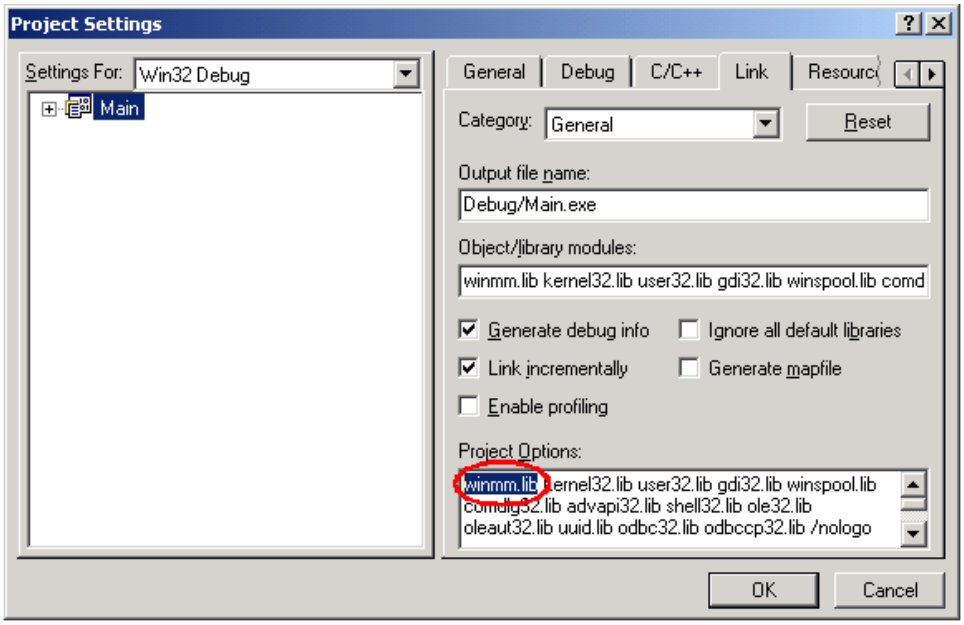
Sound is played synchronously and function does not return until the sound ends.

Return Values

Returns TRUE if successful or FALSE otherwise.

Remarks

- If the specified sound cannot be found, **sndPlaySound** plays the system default sound.
- If you get link errors, make sure that your compiler knows where to find **`winmm.lib`**.
  Add **`winmm.lib`** file to the project by going to Project | Settings | Link tab.



Example:

```
#include <iostream.h>
#include <windows.h>
int main()
{
// If you get link errors, make sure that your compiler knows
// where to find winmm.lib in th Project file

     cout << "Playing the sounds ";
     cout.flush();  // Required to output line now
     sndPlaySound("forest.wav", SND_SYNC);
     sndPlaySound("baloney.wav", SND_SYNC);
     sndPlaySound("forest.wav", SND_SYNC);
     sndPlaySound("44mag.wav",SND_SYNC);
     // Creates loop
     sndPlaySound("forest.wav", SND_ASYNC | SND_LOOP);
     Sleep(1500); // Delay for 1.5 Second
     sndPlaySound("44mag.wav",SND_SYNC);
     sndPlaySound("44mag.wav",SND_SYNC);
     sndPlaySound("44mag.wav",SND_SYNC);
     sndPlaySound("forest.wav", SND_ASYNC | SND_LOOP);
     Sleep(5000);   // Delay for 5 Second
     sndPlaySound(NULL, SND_SYNC); // End Sound
     cout << "\n\n\nDone\n";
     return 0;
}
```

Piano Musical Notes and Their Frequencies

Note name	Frequency	
A0	27.500	
B0	30.868	29.135
C1	32.703	
D1	36.708	34.648
E1	41.203	38.891
F1	43.654	
G1	48.999	46.249
A1	55.000	51.913
B1	61.735	58.270
C2	65.406	
D2	73.416	69.296
E2	82.407	77.782
F2	87.307	
G2	97.999	92.499
A2	110.00	103.83
B2	123.47	116.54
C3	130.81	
D3	146.83	138.59
E3	164.81	155.56
F3	174.61	
G3	196.00	185.00
A3	220.00	207.65
B3	246.94	233.08
C4	**261.63**	
D4	293.67	277.18
E4	329.63	311.13
F4	349.23	
G4	392.00	369.99
A4	**440.00**	415.30
B4	493.88	466.16
C5	523.25	
D5	587.33	554.37
E5	659.26	622.25
F5	698.46	
G5	783.99	739.99
A5	880.00	830.61
B5	987.77	932.33
C6	1046.5	
D6	1174.7	1108.7
E6	1318.5	1244.5
F6	1396.9	
G6	1568.0	1480.0
A6	1760.0	1661.2
B6	1975.5	1864.7
C7	2093.0	
D7	2349.3	2217.5
E7	2637.0	2489.0
F7	2793.0	
G7	3136.0	2960.0
A7	3520.0	3322.4
B7	3951.1	3729.3
C8	4186.0