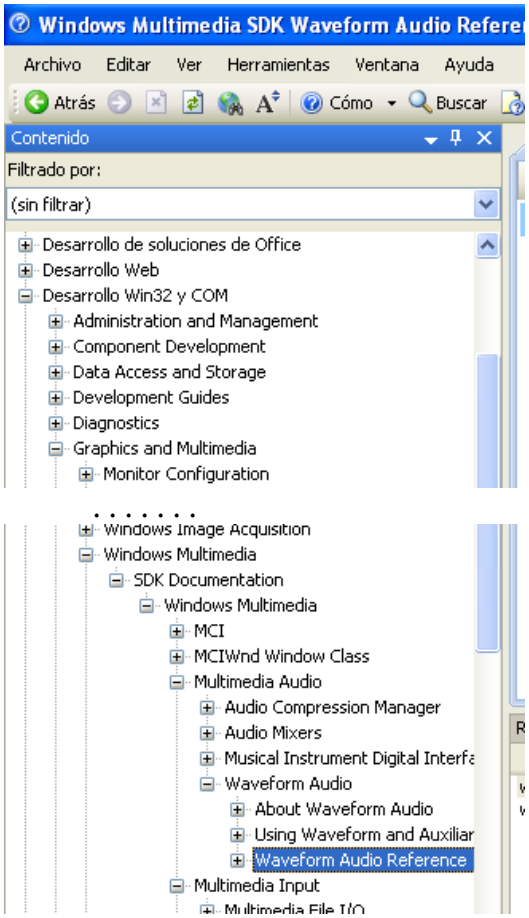
ENTRADA DE SONIDO EN WINDOWS

A continuación se describen los pasos a realizar para leer datos de la entrada de micrófono en Windows.
Para estos dispositivos la E/S no puede realizarse mediante los métodos estándar (fopen, fread, etc.) dado que no aseguran la operación en tiempo real necesaria para que el sonido sea tratado correctamente.
Windows provee una librería específica para estos dispositivos, que se llama Waveform Audio. La ayuda detallada sobre la programación de esta librería se puede encontrar en Visual Studio 2008:



A continuación se describe de forma resumida el funcionamiento de esta librería.

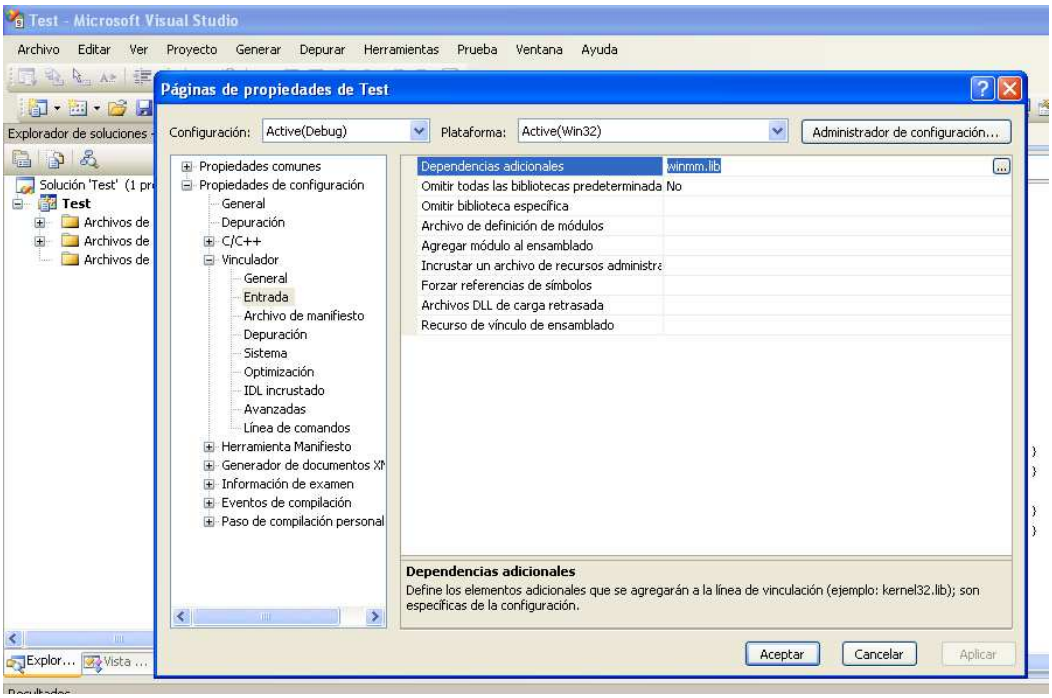
Archivos a incluir para utilizar la librería:

```
#include <windows.h>
#include <mmsystem.h>
```

Archivos de librería a añadir al proyecto:

```
winmm.lib
```

Para añadir un archivo de librería, seleccionar Proyecto, botón derecho, Propiedades, y buscar la patilla Propiedades de Configuración → Vinculador → Entrada. En ella, añadir el archivo de librería deseado a Dependencias Adicionales:



Programación de la librería para entrada de sonido en programas de tipo consola

Se necesita un identificador de dispositivo, que en este caso es de tipo `HWAVEIN` (tipo de datos definido en `mmsystem.h`).

Se debe abrir el stream que conecta con el dispositivo micrófono mediante la función `waveInOpen()` (ver ayuda en Visual C++). Esta función devolverá el id. de dispositivo.

La función `waveInOpen()` devolverá un código de error si no puede abrirse el dispositivo.

Los parámetros que requiere `waveInOpen()` serán (ver ayuda):

```
phwi = puntero al identificador de tipo HWAVEIN  (valor
          devuelto).
uDeviceID = WAVE_MAPPER.
pwfx =    puntero a una estructura de tipo WAVEFORMATEX con
          la configuración para el dispositivo. Los
          contenidos de la estructura son los siguientes
          (ver ayuda de WAVEFORMATEX):
          wFormatTag =       WAVE_FORMAT_PCM
          nChannels =        1
          nSamplesPerSec   =  Frecuencia de muestreo a
                              utilizar en Hz (8000 – 22500)
          wBitsPerSample = 16 (valores -32768 a 32767)
          nBlockAlign =      wBitsPerSample*nChannels/8
          nAvgBytesPerSec =  nSamplesPerSec*nBlockAlign
          cbSize             = 0;

dwCallback = dirección de una función que será llamada
          cuando se haya leído un buffer, utilizando un
          molde (DWORD). Ver ayuda de waveInProc para el
          formato de esta función.
```

```
dwCallbackInstance = NULL
fdwOpen = CALLBACK_FUNCTION
```

Una vez abierto correctamente el dispositivo, se debe preparar algún buffer para recibir datos del mismo. La función para prepararlo es `waveInPrepareHeader()`, que recibe 3 parámetros (ver ayuda):

- `hwi` = identificador de tipo HWAVEIN devuelto por la función `waveInOpen()`.
- `LPWAVEHDR` = puntero a una estructura WAVEHDR con los datos del buffer. Los contenidos de esta estructura son los siguientes (ver ayuda de WAVEHDR):
  - `lpData` = Puntero a una zona de memoria donde se dejarán los datos una vez leídos. En esa zona de memoria deben haberse reservado al menos el nº de bytes indicados en el siguiente parámetro
  - `dwBufferLength` = tamaño en bytes del buffer anterior
  - resto de parámetros = 0 (son rellenados por la función)
- `cbwh` = tamaño de la estructura WAVEHDR (utilizar sizeof)

A continuación, se llama a la función `waveInStart()` (ver ayuda) para que empiece la adquisición de la tarjeta de sonido. Dicha adquisición comienza, pero no se transfiere al buffer hasta que se diga específicamente con la función `waveInAddBuffer()`. A esta función se le pasan como parámetros el identificador hwi, el puntero a la estructura de buffer preparada anteriormente y el tamaño de la misma. Lo que hace esta función es que los siguientes datos que lleguen, hasta completar el tamaño indicado en el campo `dwBufferLength`, sean almacenados en el buffer indicado en el campo `dwData`.
La función `waveInAddBuffer()` no espera a que se llene el buffer con los datos, sino que retorna inmediatamente. Cuando el buffer se haya llenado será llamada de forma automática la función que se indicó en el parámetro dwCallback de `waveInOpen()`. Es en esta función callback donde se debe hacer el procesamiento del buffer y, normalmente, volver a llamar a `waveInAddBuffer()` para que se vaya llenando otro.

Cuando ya no se desee más entrada, llamar a las funciones `waveInReset()` y `waveInClose()` para cerrar la conexión con el dispositivo micrófono.

Importante: comprobar el valor de salida de cada una de las funciones para ver que no ha habido error.