As with all programs accessing the Windows API we must trap errors and close all windows associated with the program so that a cleanup operation can be implemented. One of the first decisions that has to be made is what audio format should be used. For example, the sampling rate and number of channels must be decided. For a higher ideal frequency, we must choose a higher sampling rate. Unfortunately, there is always a trade-off as we increase the sampling rate to achieve higher audio frequency more coding becomes necessary. "Normally you should choose the lowest sampling rate suitable for your application, remembering that it needs to be at least double the highest audio frequency in which you are interested" (according to the Nyquist criterion).

First code must be written to set up our audio format while including wave input device as follows.

```
 340 DIM Format{wFormatTag{l&,h&}, nChannels{l&,h&}, nSamplesPerSec%, \  350 \
       nAvgBytesPerSec%, nBlockAlign{l&,h&}, wBitsPerSample{l&,h&}, \  360 \
       cbSize{l&,h&}}  370  380 Format.wFormatTag.l& = 1 : REM WAVE_FORMAT_PCM
 390 Format.nChannels.l& = 1  : REM Monaural  400 Format.nSamplesPerSec% = 44100  410
Format.wBitsPerSample.l& = 16  420 Format.nBlockAlign.l& = Format.nChannels.l& *
Format.wBitsPerSample.l& / 8  430 Format.nAvgBytesPerSec% = Format.nSamplesPerSec% *
Format.nBlockAlign.l&  440  450 _WAVE_MAPPER = -1  460 SYS "waveInOpen",
^WaveIn%, _WAVE_MAPPER, Format{}, 0, 0, 0 TO ret%  470 IF ret% ERROR 100,
"waveInOpen failed: "+STR$~ret%
```

For this project, I chose a sampling rate of 44100 Hz. This range is more adequate for the project. 44100 Hz was chosen because most audio input may be as low as 11025 HZ even when stating a rating of 44100 Hz. The next logical step in the creation of the code is to create and initialize a few buffers. In this project, I created three and all are reused cyclically. The first buffer is set up for inputting the sampled sound, the second buffer is actually being processed by the program, while the third buffer was used as a spare. In order to avoid loss of data, the program has to be set up to process the input of data quickly. To minimize latency, it is best to increase the amount of buffers This works best rather than making larger buffers while working out the fluctuations in this dicotomy. Here is the code used for buffering note there are three buffers with 1024 samples at 44100 Hz. Thus, we can expect a latency of at least 24 ms.

```
 530 nBuffers% = 3  540 SamplesPerBuffer% = 1024  550 BytesPerBuffer% =
SamplesPerBuffer% * Format.nBlockAlign.l&  560  570 DIM _WAVEHDR{lpData%,
dwBufferLength%, dwBytesRecorded%, dwUser%, \  580 \           dwFlags%, dwLoops%,
lpNext%, Reserved%}  590  600 DIM Headers{(nBuffers%-1)} = _WAVEHDR{}  610  620
FOR buff% = 0 TO nBuffers%-1  630   DIM buffer% BytesPerBuffer% - 1  640     650
```

```
  Headers{(buff%)}.lpData% = buffer%  660   Headers{(buff%)}.dwBufferLength% =
BytesPerBuffer%  670     680   SYS "waveInPrepareHeader", WaveIn%, Headers{(buff%)},
DIM(_WAVEHDR{}) TO ret%  690   IF ret% ERROR 100, "waveInPrepareHeader failed:
"+STR$~ret%  700     710   SYS "waveInAddBuffer", WaveIn%, Headers{(buff%)},
DIM(_WAVEHDR{}) TO ret%  720   IF ret% ERROR 100, "waveInAddBuffer failed:
"+STR$~ret%  730 NEXT
```

Please note that in this code I have allocated the buffers using BASIC's heap, However I could have used the Windows API to allocate the memory. After preparing a way to input real-time audio capture the audio should be set up.

```
 820 SYS "waveInStart", WaveIn% TO ret%  830 IF ret% ERROR 100, "waveInStart failed:
"+STR$~ret%
```

In order to make sure the program can keep up with the received audio implementation, the next code decides if the buffers need to process.

```
 870 _WHDR_DONE = 1  880 REPEAT  890   FOR buff% = 0 TO nBuffers%-1  900     IF
Headers{(buff%)}.dwFlags% AND _WHDR_DONE THEN  910
     PROCprocessbuffer(Headers{(buff%)}.lpData%, SamplesPerBuffer%)  920
     Headers{(buff%)}.dwFlags% AND= NOT _WHDR_DONE  930       SYS
"waveInAddBuffer", WaveIn%, Headers{(buff%)}, DIM(_WAVEHDR{})  940     ENDIF  950
  NEXT  960   SYS "Sleep", 1  970 UNTIL FALSE
```

Processing the audio data is the most important aspect of the project. However, I prefer to keep this process proprietary. Each audio sample consists of a signed 16-bit value in the range -32768 to +32767. Lastly, a standard cleanup routine should be implemented. Note the missing line numbers in the code seqments indicate rededacted proprietary code.