Based on the sound driver provided by the LeJOS project.

Expected usage is to declare a File object representing a wav file and import it as follows:

```
Beep : NXT.Audio.Wav.File;
pragma Import (C, Beep, "beep_wav_start");
```

You can then play the file via procedure Play, where "My_Volume" is some allowed volume value:

```
Play (Beep, My_Volume);
```

The wav file must be specified to the linker to satisfy the pragma Import. The base name of the file must correspond to the first part of the name used in the Import pragma. For the example above, the file name specified to the linker would have a base name of "beep_wav". The file name extension would not be ".wav" however, because the wav file must be converted to ELF format and must have a specific symbol defined. To do the conversion you can use the following in a makefile, or perform the indicated steps manually. The conversion need only be done once. The resulting new file will have an extension of ".owav" when using the steps below, but the specific extension is not important as long as that same name is specified to the linker.

```
arm-eabi-objcopy -I binary -O elf32-littlearm -B arm \ 
   --redefine-sym _binary_hello_wav_start=hello_wav_start \ 
   --redefine-sym _binary_hello_wav_end=hello_wav_end \ 
   --redefine-sym _binary_hello_wav_size=hello_wav_size \ 
   hello.wav hello.owav
```

Note how the symbols are altered. That is how the name string used in the pragma Import is defined.
package NXT.Audio.Wav is

   type File is limited private;
   -- A "wav" file. Currently we support only 8-bit PCM data formats.

   procedure Play (This : File; Volume : Allowed_Volume);

   Invalid_Format : exception;

private

   type File is limited null record;
   -- The type must be actually limited (or tagged) in the full view because
   -- it must be passed by reference for the sake of taking the address of
   -- parameters of the type when passed to subprograms.
   --
   -- A wav file is a contiguous region of memory that contains subregions
   -- referred to as "chunks". Each chunk is a descriptor that contains
   -- specific data. The size of these data, and thus the chunk itself, vary
   -- with the kind of chunk. Some kinds of chunk are mandatory, others are
   -- optional. In addition, the required order of the chunks is only
   -- partially defined. Therefore, we cannot easily define a single type that
   -- statically represents all the possible content of a wav file. Hence we
   -- declare a minimal chunk representation and use address arithmetic to
   -- access the chunks within a file.
   --
   -- for File'Alignment use Unsigned_32'Alignment;
   -- We map the first chunk in a file to the address of the file itself. The
   -- alignment clause ensures the alignment of objects of type File will be
   -- sufficient for the mapped "chunk" objects.

end NXT.Audio.Wav;