Subject: Re: Write data to .WAV file?
Posted by hahn on Mon, 19 Jan 1998 08:00:00 GMT
View Forum Message <> Reply to Message

lauth@linmpi.mpg.de_spamstop (Uwe Lauth) wrote:

> Hello all
>
> Does anyone have an IDL routine which writes an array of data into a .WAV file?
>
> Regards,
> Uwe

RIFF wav files are a bit trickly to handle because they are organized in chunks
and the sequence of the chunks are not fixed. If you restrict yourself to
process the data chunk only and assume that it is the first chunk after the
header processing is fairly easy.

To process a RIFF WAV file, you open the file in IDL and read the header.
Here is a data structure I use to read and write uncompressed .wav files:

```
WAV_HDR = { WAVE_HEADER,    $
          ID1:'****',     LoF:0L,      ID2:'********',     LoH:0L, $
          FMT:0,          NChan:0,     SpSc:0L,            DpSc:0L,   $
          BypSa:0,        BipSa:0,     ID3:'****',         LoD:0L    }

; ID1       the letters "RIFF"
; LOF       bytes from here (i.e. from byte 8) to end of file
; ID2       the letters "WAVE" followed by "fmt "
;           "fmt " is the beginning of the format descriptor block
; LoH       Length of Header (here 16 Bytes format descriptor) that follows
;           ---- start of format header ----
; FMT       format tag, 1 = PCM, 101 = ADPCM
; NChan     channels (1 = Mono, 2 = Stereo ...)
; SpSc      samples per second
; DpSc      data bytes per second = samples per second * bytes per sample
; BypSa     bytes per sample = channels * (bits per sample / 8)
; BipSa     bits per sample
;           ---- end of format header ----
;           Now watch for "data": Other blocks will be skipped.
; ID3       the letters "data". This is the beginning of the data block
; LoD       data length, i.e. the number of bytes of the actual data.
;           ---- start of sampled data ----
;           for multi-channel samples 1st byte is channel 1, 2nd byte is
;           channel 2, 3rd byte channel 3 (or if 2 channels, channel 1 again)
;           Amplitude (y-values) from -127 to 127 or -32767 to 32767.
;           ---- end of sampled data ----
;           Other chunks may follow. Don't play 'em
```

;
Now pick those attribute you may need to know, i.e. sample rate, number of bits per sample, and use this information to allocate byte arrays or integer arrays. Then go on and read the data. All is done in binary. Note that the byte order is Intel (least significant byte first).

Here comes a piece of IDL code to perform a consistency check on the header. It forces a sampling rate of 48,000 sampes/sec, 16 bit per sample and 2 channels.

```
pro repwav, dsn, sps = sps
;
; Prozedur zum Reparieren des Kopfes einer Wave-Datei. Der Parameter sps
;       gibt die Samplingrate an. Default ist 48000.
;       Der Parameter  dsn  ist der Dateiname. Wenn nicht angegeben, wird
;       Pickfile aufgerufen.
;
@c:\musik\wavhdr.inc

if n_elements(sps) eq 0 then  new_sps = 48000  else  new_sps = sps

if dsn eq '?' then  $
   in_dsn = Pickfile ( Path = 'd:\musik\data', Filter = '*.wav' ) $
   else $
   in_dsn = dsn

If In_dsn eq '' Then Return

In_hdr  = { Wave_Header }   ; Kopf der Eingabe-Datei deklarieren
In_hdr  = Wav_HDR           ;                 und initialisieren.

; Datei oeffnen, den Kopf lesen und auf dem Bildschirm anzeigen.

OpenU, iunit, In_dsn, /get_lun

In_dcb = fstat ( iunit )

ReadU, iunit, In_hdr

print, In_hdr



In_hdr.id1 = 'RIFF'
In_hdr.id2 = 'WAVEfmt '
In_hdr.id3 = 'data'

if In_hdr.LoF ne ( In_dcb.size - 8 ) then $
```

```
   In_hdr.LoF = In_dcb.size - 8

In_hdr.LoH = 16
In_hdr.fmt    = 1
In_hdr.NChan  = 2
In_hdr.SpSC   = new_sps
In_hdr.BipSa  = 16
In_hdr.BypSa  = In_hdr.NChan * In_hdr.BipSa / 8
In_hdr.DpSC   = In_hdr.spsc * In_hdr.bypsa

In_hdr.LoD = In_hdr.LoF - In_hdr.LoH - 20

ReadU, iunit, ampl16
help, In_hdr, /stru
; plot, ampl16

Point_Lun, iunit, 0

WriteU, iunit, In_HDR

; Datei schliessen, Kanal freigeben

free_lun, iunit

dsn = in_dsn

end
```