```
import sys
import numpy as np
import struct
import os
import math
import scipy.io.wavfile

"""  follow documentation on the km3net wiki on the format of the info word and data

large_UDP_buffer_size = 8972
small_UDP_buffer_size = 6724
TAES = 1413563731

common_header_dt   = np.dtype([('DataType',   np.uint32),
                               ('RunNumber',  np.uint32),
                               ('UDPSeq',     np.uint32),
                               ('TimeStamp',  np.uint32),
                               ('TimeStamp2', np.uint32),
                               ('DOMid',      np.uint32),
                               ('DOMStatus0', np.uint32),
                               ('DOMStatus1', np.uint32),
                               ('DOMStatus2', np.uint32),
                               ('DOMStatus3', np.uint32)]).newbyteorder('>')
first_acoustic_dt  = np.dtype([('Infoword1', np.uint16),
                               ('Infoword2', np.uint16),
                               ('InfoWord3', np.uint16)]).newbyteorder('>')
subseq_acoustic_dt  = np.dtype([('Audioword', np.uint32)]).newbyteorder('>')

N_words0 = (large_UDP_buffer_size - common_header_dt.itemsize
            - first_acoustic_dt.itemsize)/subseq_acoustic_dt.itemsize
N_words1 = (large_UDP_buffer_size - common_header_dt.itemsize)/subseq_acoustic_dt.it
N_words2 = (small_UDP_buffer_size - common_header_dt.itemsize)/subseq_acoustic_dt.it

def find_start_of_run(filename):
    with open(filename, "rb") as f:
        while True:
            word = struct.unpack('>I',f.read(4))[0]
            if word == TAES:
                return f.tell() - struct.calcsize('>I')
    return False

def get_bin(x, n):
    return format(x, 'b').zfill(n)

def AudioSpecs(infoword):
    bitstring = get_bin(infoword[0][0], 16) + \
        get_bin(infoword[0][1], 16) +  \
        get_bin(infoword[0][2], 16)

    if  int(bitstring[0]) != 1:
        sys.exit("Wrong audio info word")

    Nchannels = (2,1)[int(bitstring[1:3], 2) > 0 ]
    Nbits     = (12, 16, 24)[int(bitstring[3:5], 2)] # no bits either 12, 16 or 24
    sampling  = int(bitstring[8:16],2) # frequency = sampling/128
    N_clocks  = int(bitstring[16:],2)  # timing info
    return Nchannels, Nbits, sampling, N_clocks

def AudioBuffer(audiowords):
    return map(AudioData, audiowords)
```

```python
def AudioData(audioword):
    # get audio specs from the info word, documentation on the km3net wiki
    thisword = audioword[0]
    bitstring = get_bin(thisword, 32)
    if int(bitstring[0]) != 0:
        sys.exit("Wrong audio data word")

    aes3_flag = int(bitstring[5], 2) # marks the beginning of a new AES3 frame
    nadc = 24
    start_ch1 = 8
    stop_ch1  = start_ch1 + nadc + 1
    ch1  = twos_complement(int(bitstring[start_ch1: stop_ch1], 2), len(bitstring[sta
    return ch1

def twos_complement(input_value, num_bits):
    '''Calculates a two's complement integer from the given input value's bits'''
    mask = 2**(num_bits - 1)
    return -(input_value & mask) + (input_value & ~mask)


def main():
    for fn in os.listdir('.'):
        if ".bin" in fn:
            print (fn)
            filename = fn
            byte_offset = find_start_of_run(filename)
            if byte_offset:
                f = open(filename, "rb")
                f.seek(byte_offset, os.SEEK_SET)
            else:
                sys.exit("Cannot find start of file")

            data = np.array(0)

            thistime = 0
            delta_time = 0
            Time = 0
            while True:
                common_header_data = np.fromfile(f, dtype=common_header_dt, count=1)
                if not common_header_data:
                    break
                elif common_header_data["UDPSeq"][0] == 0 and common_header_data["Da
                    infoword  = np.fromfile(f, dtype=first_acoustic_dt, count=1)
                    datawords = np.fromfile(f, dtype=subseq_acoustic_dt, count=N_wor
                    extraword = np.fromfile(f, dtype=np.uint32, count=2)  # seems tc
                    halfword  = np.fromfile(f, dtype=np.uint16, count=1)
                    thistime = common_header_data["TimeStamp"][0] + common_header_da
                    delta_time = 0
                elif 0 < common_header_data["UDPSeq"][0] < 8 and common_header_data[
                    wordhalf  = np.fromfile(f, dtype=np.uint16, count=1)
                    datawords = np.fromfile(f, dtype=subseq_acoustic_dt, count=N_wor
                    extraword = np.fromfile(f, dtype=np.uint32, count=2)  # seems tc
                    halfword  = np.fromfile(f, dtype=np.uint16, count=1)
                elif common_header_data["UDPSeq"][0] == 8 and common_header_data["Da
                    wordhalf  = np.fromfile(f, dtype=np.uint16, count=1)
                    datawords = np.fromfile(f, dtype=subseq_acoustic_dt, count=N_wor
                    if datawords[-1][0] == 0:
                        datawords = datawords[:-2]
                    elif int(datawords[-1][0]) == 203620352:   # some kind of traile
```

```
                        datawords = datawords[:-1]
                        extraword = np.fromfile(f, dtype=np.uint32, count=1)
                    else:
                        sys.exit("Cannot read this format...")
                else:
                    sys.exit("Cannot read this format.")
                delta_time = delta_time + len(datawords)*(128./25e6)
                Time = thistime + delta_time

                data = np.append(data, AudioBuffer(datawords))

            # saving the data
            path = fn
            basename = os.path.splitext(os.path.basename(path))[0]
            if "wav" in sys.argv[1:]:
                scipy.io.wavfile.write(basename+'.wav',195300, data.astype(np.dtype(
            else:
                np.savetxt(basename+'.dat', data, fmt="%d")

if __name__ == "__main__":
    sys.exit(main())
```