AUDIOMATICA

AN-025
APPLICATION NOTE

CLIO 13/14 SINUSOIDAL FILE STRUCTURE WITH IMPORT EXAMPLES IN SCILAB

by Daniele Ponteggia - dp@audiomatica.com

INTRODUCTION

The release CLIO 13/14 changed the file structure of Sinusoidal files with the introduction of THD+N.

Our previous application note 001 is still valid for Sinusoidal files from CLIO version 10 up to 12.5. Files saved with version 13 and 14 can be still opened using the previous Scilab code but the THD+N data, if available, is not read.

This application note updates the data structure.

CLIO 13/14 SINUSOIDAL FILE STRUCTURE

The Sinusoidal file structure is composed of an header section of a fixed length and data sections which length is dependent on measurement settings. The SinData section will always be present while RBData and THDData sections are present only if the relative calculation options are active during the measurement.

The following table report in details the Sinusoidal file fields, rows with gray background are optional.

Position (Bytes)	Field	Type	Length (Bytes)	Notes
0	Undocumented Space	char	28	
28	RelBackComp	unsigned int	4	Sets the lowest compatible release (This file structure applies to RelBackComp=1000)
32	Undocumented Space	char	758	
790	Channel	char	1	0 – ChA, 1 – ChB, 2 – ChA&B
791	Undocumented Space	char	22	
813	CHAUnit	char	1	CHA Y Unit, see TLevUnit definition
814	Undocumented Space	char	54	
868	THDFlag	char	1	THDData present: 0 – No, 1 – Yes
869	RBFlag	char	1	Rub and Buzz Data present: 0 – No, 1 – Yes
870	CHBUnit	char	1	CHB Y Unit, see below TLevUnit definition for details
871	Undocumented Space	char	14	
885	THDNFlag	char	1	THD+N Data present: 0 – No, 1 – Yes

Position (Bytes)	Field	Type	Length (Bytes)	Notes
886	Undocumented Space	char	70	
956	NumOfFreq	unsigned int	4	How many frequency points
960	SinData	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	Sinusoidal response data
+20*Nu-mOfFreq	RBData	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	Present only if RBFlag=1
+20*Nu-mOfFreq	THDData	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	Present only THDFlag=1
+20*Nu-mOfFreq	2nd Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	3rd Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	4th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	5th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	6th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	7th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	8th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	9th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	10th Armonic	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	
+20*Nu-mOfFreq	THD+NData	Array [1..NumOfFreq] of sinstep	20*NumOfFreq	Present only THDNFlag=1

The sinstep data type is defined as:

```
struct complex
{
float re;
float im;
};

struct sinstep
{
float freq;
complex valA;
complex valB;
};
```

The TlevUnit is of a set type:

```
TlevUnit=(Vrms,dBV,dBu,dBspl,dBRel,Ohm,Deg,ms,dB,Perc,dBmet,dBms2,dBPa,dB
PaV,dBms);
```

Value	Shown Unit in CLIO	Saved Data Unit
0	Vrms	V
1	dBV	V
2	dBu	V
3	dBspl	Pa
4	dBrel	V
5	Ohm	Ohm
6	Deg	
7	ms	
8	dB	
9	%	
10	dBmet	m
11	dBm/s2	m/s2
12	dBPa	
13	dBPa/V	
14	dBm/s	m/s

SCILAB IMPORT EXAMPLE

We report here a very simple example of import function for CLIO 13/14 Sinusoidal files in Scilab language.

```
function [Ch,CHAUnit,CHBUnit,SinFreq,SinData,RNBFlag,RNBFreq,RNBData,THD-
Flag,THDFreq,THDData,THDNFlag,THDNFreq,THDNData]=loadsin(filename)

  [fd]=mopen(filename);

  skip=mget(28,'uc',fd);

  RelBackComp=mget(1,'ui',fd);

  if RelBackComp<1000 then
    mclose(fd);
    error('File not compatible');
  end;

  skip=mget(740,'uc',fd);
  skip=mget(18,'uc',fd);
```

```
Ch=mget(1,'uc',fd);

skip=mget(22,'uc',fd);

CHAUnit=mget(1,'uc',fd);

skip=mget(54,'uc',fd);

THDFlag=mget(1,'uc',fd);
RNBFlag=mget(1,'uc',fd);

CHBUnit=mget(1,'uc',fd);

skip=mget(85,'uc',fd);

NumOfFreq=mget(1,'ui',fd);

for i=1:NumOfFreq
  SinFreq(1,i)=mget(1,'f',fd);
  SinRe(1,i)=mget(1,'f',fd);
  SinIm(1,i)=mget(1,'f',fd);
  SinRe(2,i)=mget(1,'f',fd);
  SinIm(2,i)=mget(1,'f',fd);
end
SinData=complex(SinRe,SinIm);

if RNBFlag==1 then
  for i=1:NumOfFreq
    RNBFreq(1,i)=mget(1,'f',fd);
    RNBRe(1,i)=mget(1,'f',fd);
    RNBIm(1,i)=mget(1,'f',fd);
    RNBRe(2,i)=mget(1,'f',fd);
    RNBIm(2,i)=mget(1,'f',fd);
  end
  RNBData=complex(RNBRe,RNBIm);
else
  RNBFreq=[];
  RNBData=[];
end

if THDFlag==1 then
  for j=1:10
    for i=1:NumOfFreq
      //THDFreq(1,i,j)=mget(1,'f',fd);
      AppoFreq(1,i)=mget(1,'f',fd);
      THDRe(1,i)=mget(1,'f',fd);
      THDIm(1,i)=mget(1,'f',fd);
      THDRe(2,i)=mget(1,'f',fd);
      THDIm(2,i)=mget(1,'f',fd);
    end
    THDFreq(:,:,j)=AppoFreq;
    THDData(:,:,j)=complex(THDRe,THDIm);
    end
  else
    THDFreq=[];
    THDData=[];
  end

if THDNFlag==1 then
    for i=1:NumOfFreq
      THDNFreq(1,i)=mget(1,'f',fd);
      THDNRe(1,i)=mget(1,'f',fd);
      THDNIm(1,i)=mget(1,'f',fd);
      THDNRe(2,i)=mget(1,'f',fd);
      THDNIm(2,i)=mget(1,'f',fd);
    end
    THDNData=complex(THDNRe,THDNIm);
  else
    THDNFreq=[];
    THDNData=[];
  end

  mclose(fd);
endfunction
```

The function can be called from the console:

```
-->[Ch,CHAUnit,CHBUnit,SinFreq,SinData,RNBFlag,RNBFreq,RNBData,THDFlag,
   THDFreq,THDData,THDNFlag,THDNFreq,THDNData]=loadsin('filename.sin');
```

and return a series of vector and matrices in function of the data available into the file.

`SinData` is a (2,NumOfFreq) complex vector where (1,:) is the channel A response and (2,:) is the channel B response.

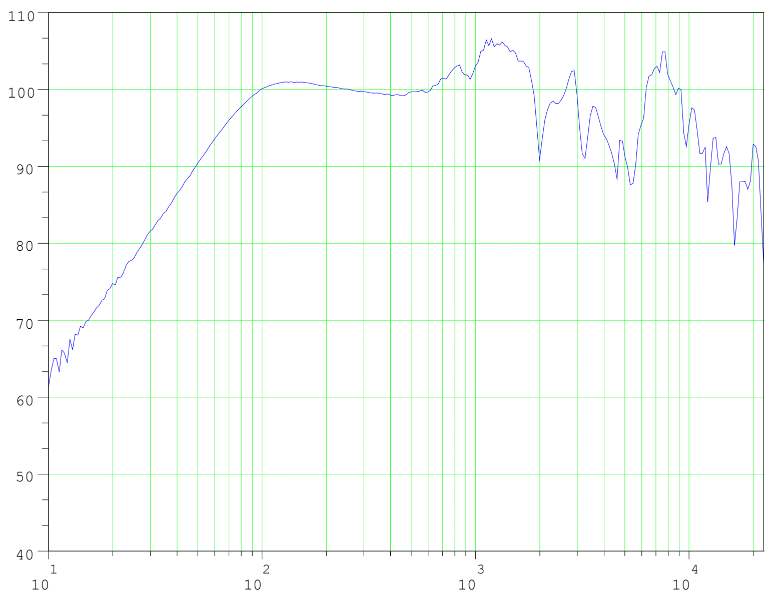
If present `THDData` is a (2,NumOfFreq,10) complex matrix. (1,:,1) is the THD of the channel A, (2,:,1) is the THD of the channel B, (1,:,2) is the second harmonic response of the channel A and so on...

As an example we can import the sinusoidal measurement of the acoustical response of a loudspeaker:

```
-->[Ch,CHAUnit,CHBUnit,SinFreq,SinData,RNBFlag,RNBFreq,RNBData,THDFlag,
THDFreq,THDData,THDNFlag,THDNFreq,THDNData]=loadsin('example_lspk.sin');
```

We can plot the measured sinusoidal SPL response (channel A), since the data is stored in Pa and in complex format it is necessary to calculate the modulus and convert to dB SPL:

```
-->plot2d(SinFreq(1,:),20.*log10(abs(SinData(1,:)+1e-35)
   +94,logflag="ln",style=2);
```



We can also add to the plot the second harmonic distortion:

```
-->plot2d(THDFreq(1,:,2),20.*log10(abs(THDData(1,:,2)+1e-35))
   +94,logflag="ln",style=5);
```