Interpreting D3 XML files

Mark Johnson, markjohnson@st-andrews.ac.uk
12 April 2011

Each .dtg format file recorded by a D3 device produces an XML (.xml suffix) file when it is uncompressed using d3read. The XML file contains metadata, i.e., information about the recording but no actual data. It should not be necessary normally to look at the XML files but they can be useful to verify how and when a recording was made. XML is a markup language and is used to gather information in a self-documented way. An XML file can be read as a text file in a word processor or notepad but usually opens in a browser if you double click on it. Ultimately, a D3 stylesheet will be available to format the .xml file contents making it easier to interpret. In the meantime, this document provides a brief example of what can be found in the XML files.

XML uses nested fields to gather information. A field is opened with a declaration like:
<EVENT>
This opens a field called EVENT which may contain data, more fields, or nothing. A field is closed by a declaration </EVENT>. There must be a closing declaration for each field and the opening-closing declarations delimit all of the information associated with that field. If the field contains nothing, it can be opened and closed in a single declaration like <EVENT />. A field-opening declaration can also contain one or more attributes which is another way to gather information, e.g.,
<EVENT TIME="2011,1,6,16,37,51">
Here the attribute TIME is assigned the string 2011,1,6,16,37,51 which is the date in a 6-element vector.

All D3 XML files open with a standard header and then declare a D3 field which contains the information in the file:

```
<?xml version="1.0" encoding="US-ASCII"?>
<D3>
```

The D3 field contains some general information about the file and programs that produced the data. First is the name and version of the program that was used to uncompress the .dtg file:

```
<HOST PROG="d3read.exe" VERSION="1.1" />
```

Then comes the name of the .dtg file that was processed:

```
<INPUTFILE NAME="C:\d3\data\test010.dtg" />
```

Next is the unique identifier of the D3 device that made the recording. This could be used, for example, to find a calibration value for the device or to report a malfunction:

```
<DEVID> 2551,a351,1403,121a,7192 </DEVID>
```

Then comes three fields that define where in the FLASH memory array on the device the recording was stored. These are expressed as physical chip, block. NBLOCKS is the number of memory blocks in the recording. These data are not useful for much except perhaps to make sure you offloaded all of the

recordings in a deployment:

<STARTADDR> 2,3536 </STARTADDR>
<ENDADDR> 3,109 </ENDADDR>
<NBLOCKS> 670 </NBLOCKS>

The next field explains when and why the recording was made. The time is not definitive - a more precise start time for the recording will be given later. The START STATE possibilities are:
NEW - this is the first recording in a deployment or after powering up.
REOPEN - this is a continuation recording - the previous recording was terminated, probably because the file was getting too big. This is done to keep the .dtg files to a manageable size and also to prevent the .wav files generated from the .dtg files from having too many samples and so being unreadable.

<EVENT TIME="2011,1,6,16,37,51">
<START STATE="REOPEN" />
</EVENT>

Now comes a sequence of CFG fields. These collectively define how the D3 device was programmed to record data. Each processing module generates its own CFG field with specific information about how it was configured. The unique ID number of the processor instance is given as well as the ID of the preceding processor module that it is connected to, allowing you to reconstruct the entire processing chain.

The first CFG field here is for a log file generator. This processor just accepts text messages and stores them in memory. When the .dtg file is unpacked, a file with the suffix specified in the SUFFIX field (i.e., .log) will be generated containing any data that was sent to the log. The FORMAT field specifies if the file will have a free text format (FORMAT none) or a comma-separated-variable format.

<CFG TIME="2011,1,6,16,37,51" ID="1" FTYPE="txt">
<PROC> LOG </PROC>
<SUFFIX> log </SUFFIX>
<FORMAT> none </FORMAT>
</CFG>

The next CFG field is for a decimator. This is part of one of the audio processing chains and demonstrates that the CFG entries can come in any order in the XML file. This decimator has an ID number of 2 and receives data from a processor with ID 3 (yet to be defined). The decimation factor is 2 and a 36 length FIR filter is used.

<CFG TIME="2011,1,6,16,37,51" ID="2">
<SRC ID="3" />
<PROC> DECM </PROC>
<DF> 2 </DF>
<NF> 36 </NF>
</CFG>

The next CFG field is the source of audio data for the previous decimator. It is an audio (i.e., ADC) input module sampling at 240kHz. The CHANBITMAP defines which hydrophone channels are being converted by this module - a bitmap of 5 means channels 1 and 3, i.e., this is a 2-channel source

module. Each audio sample is represented by 16 bits. We now know that the output sampling rate of module 2 will be 240/2=120kHz (2 is the decimation factor of module 2).

```
<CFG TIME="2011,1,6,16,37,51" ID="3">
<PROC> AUDIO </PROC>
<CHANBITMAP> 5 </CHANBITMAP>
<BAND> MF2 </BAND>
<FS UNIT="Hz"> 240000 </FS>
<NBITS> 16 </NBITS>
</CFG>
```

The next CFG field defines a loss-less compressor attached to the output of the decimator module 2. It uses our X3 compression protocol and the parameters of the compressor are specified (these are only interesting if you want to explore how well the compressor is working).

```
<CFG TIME="2011,1,6,16,37,51" ID="4" FTYPE="X3">
<SRC ID="2" />
<BLKLEN> 16 </BLKLEN>
<NCHS> 2 </NCHS>
<CODE> 1 </CODE>
<FILTER> 1 </FILTER>
<NBITS> 16 </NBITS>
<THRESH> 10 </THRESH>
<CONTIG> 1 </CONTIG>
</CFG>
```

The final CFG field in this audio processing chain defines how the data is stored in memory and how it should be handled by the d3read program. A WAV format file with suffix .wav will be generated containing 2-channel audio data sampled at 120kHz.

```
<CFG TIME="2011,1,6,16,37,51" ID="5" FTYPE="wav" CODEC="4">
<SRC ID="4" />
<FS> 120000 </FS>
<NBITS> 16 </NBITS>
<NCHS> 2 </NCHS>
<SUFFIX> wav </SUFFIX>
</CFG>
```

We now know the full processing chain for the MF channels. It goes:

Module ID	Type	Function
3	AUDIO	raw audio data input at 240kHz, audio channels 1 and 3
2	DECM	decimate by 2 to get a sampling rate of 120kHz
4	X3	loss-less audio compressor
5	WAV	stores compressed audio to memory

The next set of CFG fields define the processing chain for the other audio channel. It is a LF channel and is sampled at 80kHz by module 6. It is then decimated by 5 in module 7. It is then compressed using the X3 algorithm in module 8 and finally stored to memory by module 9. When the LF data is unpacked by d3read, it will generate a WAV format file with a suffix .lwv.

<CFG TIME="2011,1,6,16,37,51" ID="6">
<PROC> AUDIO </PROC>
<CHANBITMAP> 2 </CHANBITMAP>
<BAND> LF1 </BAND>
<FS UNIT="Hz"> 80000 </FS>
<NBITS> 16 </NBITS>
</CFG>

<CFG TIME="2011,1,6,16,37,51" ID="7">
<SRC ID="6" />
<PROC> DECM </PROC>
<DF> 5 </DF>
<NF> 36 </NF>
</CFG>

<CFG TIME="2011,1,6,16,37,51" ID="8" FTYPE="X3">
<SRC ID="7" />
<BLKLEN> 16 </BLKLEN>
<NCHS> 1 </NCHS>
<CODE> 1 </CODE>
<FILTER> 1 </FILTER>
<NBITS> 16 </NBITS>
<THRESH> 10 </THRESH>
<CONTIG> 1 </CONTIG>
</CFG>

<CFG TIME="2011,1,6,16,37,51" ID="9" FTYPE="wav" CODEC="8">
<SRC ID="8" />
<FS> 16000 </FS>
<NBITS> 16 </NBITS>
<NCHS> 1 </NCHS>
<SUFFIX> lwv </SUFFIX>
</CFG>

The last set of fields define some important times in the recording. The first two fields provide the exact device time for the first audio sample in each processing chain. The first sample of the .wav (MF) chain was recorded at 16:37:51.128756 on 6 Jan. 2011. The first sample of the .lwv (LF) chain was recorded at 16:37:51.065114 on the same day. There will be a CUE field for each processing chain.

<CUE TIME="2011,1,6,16,37,51" ID="5" SUFFIX="wav" SAMPLE="0"> 0.128756 </CUE>
<CUE TIME="2011,1,6,16,37,51" ID="9" SUFFIX="lwv" SAMPLE="0"> 0.065114 </CUE>

EVENT fields are used to notify changes in parameters. Here they are used to report when the audio mute function was enabled. The mute function bypasses the hydrophones with a capacitor of the same value to enable measurement of system noise. A MUTE="1" event means that the hydrophone is bypassed. The mute state is usually only maintained for a few seconds (in this case, 10s) and then the hydrophones are re-connected to the input. All audio channels are affected by a mute and a mute is usually done every hour or so to keep track of the self-noise.

<EVENT TIME="2011,1,6,17,21,33">
<AUDIO MUTE="1" />
</EVENT>

<EVENT TIME="2011,1,6,17,21,43">
<AUDIO MUTE="0" />
</EVENT>

The final event signals the end of this recording. The recording was ended with a REOPEN state meaning that it was automatically ended to keep the file size down. There should be a subsequent recording with a filename that is 1 larger than this one, i.e., C:\d3\data\test011.dtg. The next recording should start exactly where this one finished and there should be no samples lost. The duration of this recording can be worked out by comparing the CUE time to the end EVENT time - it is about 1 hour and 16 minutes.

<EVENT TIME="2011,1,6,17,54,5">
<END STATE="REOPEN" />
</EVENT>

The XML file ends by closing the D3 field.

</D3>