DICOM Waveform Generator

Budhi Kristianto [a], Chung Wen-Yaw [a] (鍾文耀), Tsai Yuh-Show [b] (蔡育秀)
[a] *Department of Electronic Engineering*
[b] *Department of Biomedical Engineering*
Chung-Yuan Christian University, Chung-Li, Taiwan, R.O.C. 32023

**Correspondence:Budhi Kristianto, yuhshow@cycu.edu.tw*

Abstract

The authors developed a DICOM Waveform Generator to implement the DICOM Waveform Standard. The application can generate DICOM waveform file from streamed signal data in a text file. The goal is to facilitate interchange of waveforms, support for audio waveforms, ECG waveforms, hemodynamic signals and Holter in medical imaging context.

The initial results show that we have successfully generated DICOM file with the waveform content.

Since raw signal data of the waveform comes in several types such as binary data, text file, and so on, an application with the capability to transform the data into DICOM format is needed as the solution.

***Keywords**: DICOM, waveform, VR, data element*

1. Introduction

In the medical domain, there is a standard imaging exchange system called DICOM. DICOM stands for Digital Imaging and Communications in Medicine. DICOM is widely accepted and implemented in hospitals around the world, especially in the radiology department to files radiology system results such as MRI images, CT images, and so on.

DICOM is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a network communications protocol. The communication protocol is an application protocol that uses TCP/IP to communicate between systems. DICOM files can be exchanged between two entities that are capable of receiving image and patient data in DICOM format. The National Electrical Manufacturers Association (NEMA) holds the copyright to this standard. It was developed by the DICOM Standards Committee, whose members are also partly members of NEMA [1].

DICOM enables the integration of scanners, servers, workstations, printers, and network hardware from multiple manufacturers into a picture archiving and communication system. The different devices come with DICOM conformance statements which clearly state the DICOM classes they support. DICOM has been widely adopted by hospitals and is making inroads in smaller applications like dentists' and doctors' offices.

For the communication and storage of medical waveforms, DICOM Work Group 1 developed the Waveform Standard (DICOM Supplement 30) in 2000, which addressed the robust interchange of waveforms, including support for audio waveforms, ECG waveforms, hemodynamic signals and Holter in medical imaging context [2].

2. DICOM File Format

The DICOM File Format is described by the American College of Radiology (ACR) and National Electrical Manufacturers Association (NEMA) in PS3.10 specification "Media Storage and File Format for Media Interchange", of the DICOM Standard. An illustration of the basic file structure can be seen below:

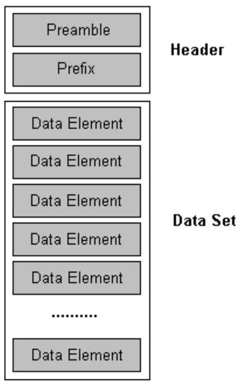


Figure 1. DICOM file format

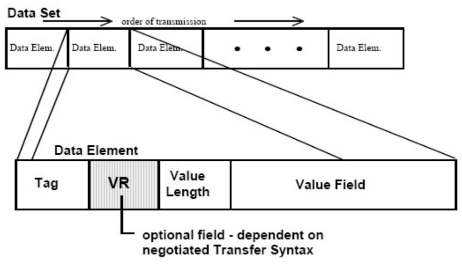


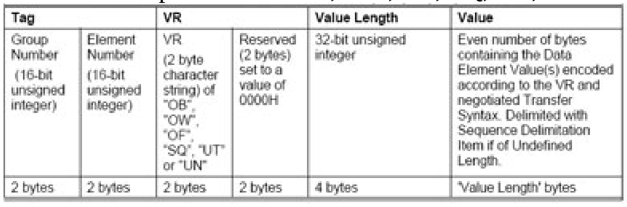
Figure 2. DICOM data element structure

A DICOM file has .dcm as its file extension. Each .dcm file consists of a header and a data set. The header started with 128 bytes with 0x00 for the value, and followed by 4 bytes prefix with “DICM” for the value. The data set consists of a stream of data elements which are consist of tag value, value representation (VR) code, value length, and value field.

DICOM uses two methods to handle the bit store mechanism which is named Transfer Syntax. The first is DICOM implicit VR, the other is DICOM explicit VR. DICOM also recognizes well known byte ordering, both little endian and big endian.

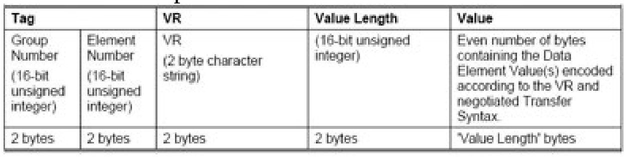
The differences between explicit VR and implicit VR are based on its VR value, which are can be seen as follow:

Table 1. Explicit VR of OB, OW, OF, SQ, UT, UN



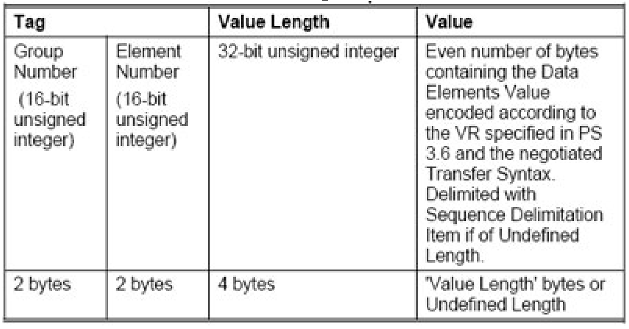
Tag		VR		Value Length	Value
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	VR (2 byte character string) of "OB", "OW", "OF", "SQ", "UT" or "UN"	Reserved (2 bytes) set to a value of 0000H	32-bit unsigned integer	Even number of bytes containing the Data Element Value(s) encoded according to the VR and negotiated Transfer Syntax. Delimited with Sequence Delimitation Item if of Undefined Length.
2 bytes	2 bytes	2 bytes	2 bytes	4 bytes	'Value Length' bytes

Table 2. Explicit VR other than shown in table 1



Tag		VR	Value Length	Value
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	VR (2 byte character string)	(16-bit unsigned integer)	Even number of bytes containing the Data Element Value(s) encoded according to the VR and negotiated Transfer Syntax.
2 bytes	2 bytes	2 bytes	2 bytes	'Value Length' bytes

Table 3. Implicit VR



Tag		Value Length	Value
Group Number (16-bit unsigned integer)	Element Number (16-bit unsigned integer)	32-bit unsigned integer	Even number of bytes containing the Data Elements Value encoded according to the VR specified in PS 3.6 and the negotiated Transfer Syntax. Delimited with Sequence Delimitation Item if of Undefined Length.
2 bytes	2 bytes	4 bytes	'Value Length' bytes or Undefined Length

For waveform content, it is mandatory to use explicit VR little endian transfer syntax [3]. The complete list of DICOM VR can be seen at DICOM specification PS3.5 section 6.2 [4].

3. Mandatory Tags in DICOM Waveform File

Every data element in DICOM started with 4 bytes tag code, which is separated into group number and element number. Group number and element number consist of 2 bytes unique code written in 16 bit unsigned integer. The complete list of DICOM tags can be seen at DICOM specification PS3.6 section 6 [5].

DICOM file that contains waveform must have mandatory tags as follow:

Table 4. Mandatory tags in DICOM waveform file

Tag Number	Tag Name
(0002,xxxx)	File meta group tags
(0008,xxxx)	Identifying tags
(0010,xxxx)	Patient info tags
(0018,xxxx)	Acquisition properties tags
(0020,xxxx)	Study relationship tags
(003a,xxxx)	Waveform properties tags
(0040,xxxx)	Procedure tags
(5400,xxxx)	Waveform data tags

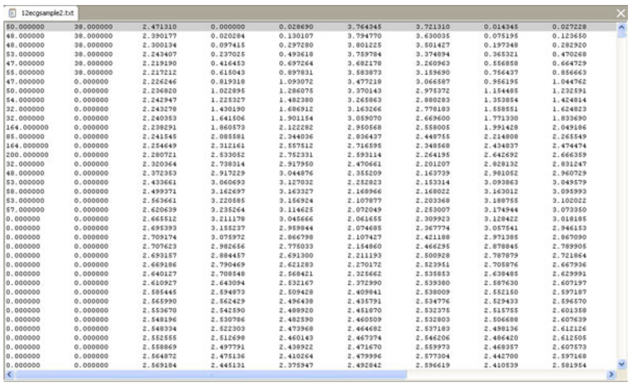
4. The DICOM Waveform Generator

We developed an application to generates waveforms in a DICOM file based on stream input signals in a text file. The result file was fits the latest DICOM specifications, which are DICOM 3.0 specification and DICOM Supplement 30 about waveform content.

We were not used any DICOM toolkit to develop the application, but we wrote the C code from the scratch using Microsoft Visual C++ 6.0. We were not used any complicated class in order to keep the code simpler written in basic C language. We got the advantages from this condition because we have plan to move the code into any embedded systems or MCU programmable systems, so we just make a little modification in the code to make it run in the new embedded environment system.

For our prototype, we've developed an application to generated 12 lead ECG signals in the text file become a DICOM waveform file. The structure of the ECG signals in the text file can be described as follow :

Every sample of signal has been represented as a row that contains 12 columns. The order of the columns is : lead I, lead II, lead III, lead aVR, lead aVL, lead aVF, lead V1, lead V2, lead V3, lead V4, lead V5, and lead V6. Total sample signal is 2,500 signals. It means, for the signal source, we have 2,500 rows with 12 columns each row in our text file. Each lead has its value written in %3.6f float integer. For example, the value is 113.563452 or 2.536445. All values are in millivolt.



```
50.000000   38.000000   2.471310   0.000000   0.028690   3.764345   3.721310   0.014345   0.027228
48.000000   38.000000   2.390177   0.020284   0.130107   3.794770   3.630035   0.075195   0.123650
48.000000   38.000000   2.300134   0.097415   0.297280   3.801225   3.501427   0.197348   0.282920
53.000000   38.000000   2.243407   0.237025   0.493618   3.759784   3.374894   0.365321   0.470268
47.000000   38.000000   2.219190   0.416453   0.697264   3.682178   3.260963   0.556858   0.664729
55.000000   38.000000   2.217212   0.615043   0.897831   3.583873   3.159690   0.756437   0.856663
47.000000   0.000000    2.226246   0.819318   1.093072   3.477218   3.066587   0.956195   1.044762
50.000000   0.000000    2.236820   1.022895   1.286075   3.370143   2.975372   1.154485   1.232591
54.000000   0.000000    2.242947   1.225327   1.482380   3.265863   2.880283   1.353854   1.424814
32.000000   0.000000    2.243278   1.430190   1.686912   3.163266   2.778183   1.558551   1.624823
32.000000   0.000000    2.240353   1.641506   1.901154   3.059070   2.669600   1.771330   1.833690
164.000000  0.000000    2.238291   1.860573   2.122282   2.950568   2.558005   1.991428   2.049186
85.000000   0.000000    2.241545   2.085581   2.344036   2.836437   2.448755   2.214808   2.265549
164.000000  0.000000    2.254649   2.312161   2.557512   2.716595   2.340568   2.434837   2.474474
200.000000  0.000000    2.280721   2.533052   2.752331   2.593114   2.264195   2.642692   2.666359
32.000000   0.000000    2.320364   2.738314   2.917950   2.470661   2.201207   2.828132   2.831247
48.000000   0.000000    2.372353   2.917229   3.044876   2.355209   2.163739   2.981052   2.960729
53.000000   0.000000    2.433661   3.060693   3.127032   2.252823   2.153314   3.093863   3.049579
58.000000   0.000000    2.499371   3.162697   3.163327   2.168966   2.168022   3.163012   3.095993
53.000000   0.000000    2.563661   3.220585   3.156924   2.107877   2.203368   3.188755   3.102022
57.000000   0.000000    2.620639   3.235264   3.114625   2.072049   2.253007   3.174944   3.073350
0.000000    0.000000    2.665512   3.211178   3.045666   2.061655   2.309923   3.128422   3.018185
0.000000    0.000000    2.695393   3.155237   2.959844   2.074685   2.367774   3.057541   2.946153
0.000000    0.000000    2.709174   3.075972   2.866798   2.107427   2.421188   2.971385   2.867090
0.000000    0.000000    2.707623   2.982656   2.775033   2.154860   2.466295   2.878845   2.789905
0.000000    0.000000    2.693157   2.884457   2.691300   2.211193   2.500928   2.787879   2.721864
0.000000    0.000000    2.669186   2.790469   2.621283   2.270172   2.523951   2.705876   2.667936
0.000000    0.000000    2.640127   2.708548   2.568421   2.325662   2.535853   2.638485   2.629991
0.000000    0.000000    2.610927   2.643094   2.532167   2.372990   2.539380   2.587630   2.607197
0.000000    0.000000    2.585445   2.594873   2.509428   2.409841   2.538009   2.552150   2.597187
0.000000    0.000000    2.565990   2.562429   2.496438   2.435791   2.534776   2.529433   2.596570
0.000000    0.000000    2.553670   2.542590   2.488920   2.451870   2.532375   2.515755   2.601358
0.000000    0.000000    2.548196   2.530786   2.482590   2.460509   2.532803   2.506688   2.607639
0.000000    0.000000    2.548334   2.522303   2.473968   2.464682   2.537183   2.498136   2.612126
0.000000    0.000000    2.552555   2.512698   2.460143   2.467374   2.546206   2.486420   2.612505
0.000000    0.000000    2.558869   2.497791   2.438922   2.471670   2.559973   2.468357   2.607573
0.000000    0.000000    2.564872   2.475136   2.410264   2.479996   2.577304   2.442700   2.597168
0.000000    0.000000    2.569184   2.445131   2.375947   2.492842   2.596619   2.410539   2.581954
```

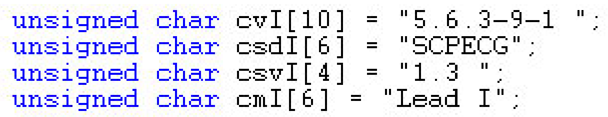
Figure 3. The raw data of the ECG signals

We've set the standard value for study attributes and patient attributes, so the result DICOM file will has same value for that kind of attributes. But the code can be easily modified to handle a dynamic input.

5. Detailed Semantic

DICOM has strict rules for file structure byte per byte. The value length has important role in DICOM structure. We can not declare unused data in DICOM file. For example, in other application, we allowed to declare let's say 10 bytes of char, but we only use 7 bytes of them. In DICOM, it will caused error when the file has been opened.

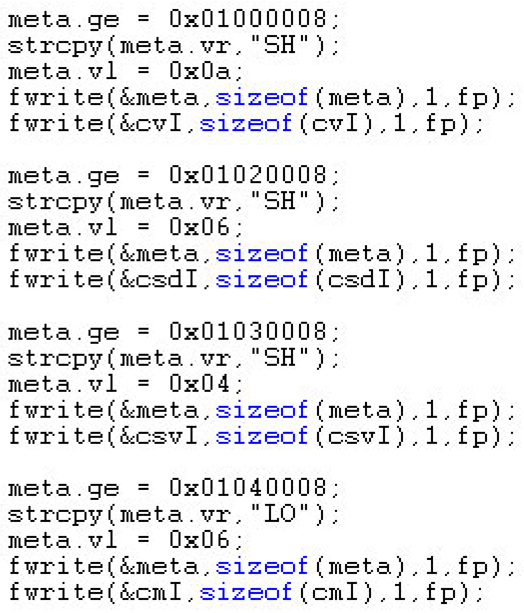
All value fields must have even number of length. If the value is char, it must be padded with space (" "). If the value is integer, it must be padded with 0x00. We can see the examples bellow :



```
unsigned char cvI[10] = "5.6.3-9-1 ";
unsigned char csdI[6] = "SCPECG";
unsigned char csvI[4] = "1.3 ";
unsigned char cmI[6] = "Lead I";
```

Figure 4. Declaration of variables

All data must be written as a 16 bit binary value. So, we had to carefully when we assigned the bit value for each data element.



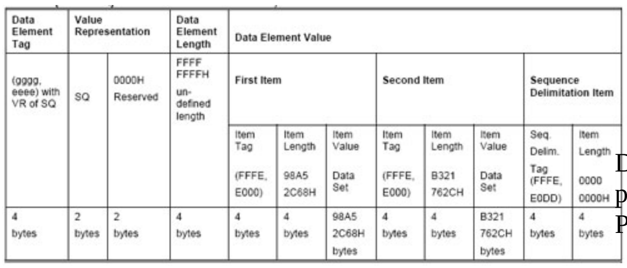
```
meta.ge = 0x01000008;
strcpy(meta.vr,"SH");
meta.vl = 0x0a;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&cvI,sizeof(cvI),1,fp);

meta.ge = 0x01020008;
strcpy(meta.vr,"SH");
meta.vl = 0x06;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&csdI,sizeof(csdI),1,fp);

meta.ge = 0x01030008;
strcpy(meta.vr,"SH");
meta.vl = 0x04;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&csvI,sizeof(csvI),1,fp);

meta.ge = 0x01040008;
strcpy(meta.vr,"LO");
meta.vl = 0x06;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&cmI,sizeof(cmI),1,fp);
```

Figure 5. Bit assignment of data elements

6. Sequence Data Element

Sequence data element is a unique data element in DICOM format. Waveform uses this data element a lot. The sequence data element has its own strict rule in order to encode the input data into stream binary bits.

The structure of sequence data element, or called as Nested Data Set described in DICOM specification PS3.5 section 7.5 [6]. It can be summarized as follow :

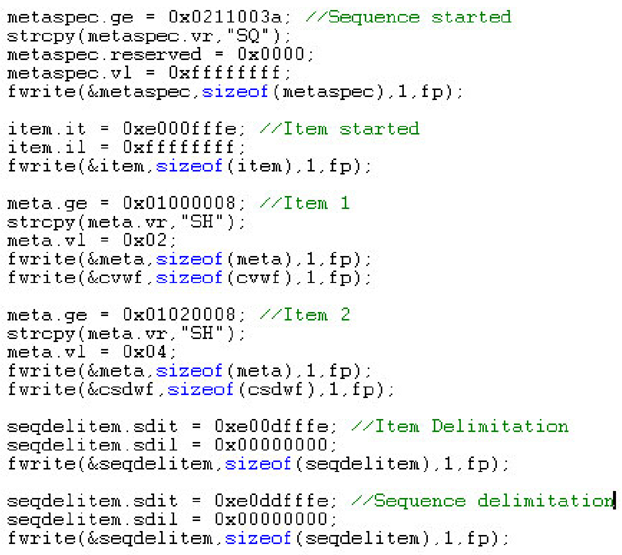


Data Element Tag	Value Representation		Data Element Length	Data Element Value							
(gggg, eeee) with VR of SQ	SQ	0000H Reserved	FFFF FFFFH un-defined length	First Item			Second Item			Sequence Delimitation Item	
				Item Tag (FFFE, E000)	Item Length 98A5 2C68H	Item Value Data Set	Item Tag (FFFE, E000)	Item Length B321 762CH	Item Value Data Set	Seq. Delim. Tag (FFFE, E0DD)	Item Length 0000 0000H
4 bytes	2 bytes	2 bytes	4 bytes	4 bytes	4 bytes	98A5 2C68H bytes	4 bytes	4 bytes	B321 762CH bytes	4 bytes	4 bytes

Figure 6. The structure of sequence data element

Since waveform must be written in explicit VR little endian transfer syntax, the structure of the sequence data element must be follow like fig 6.

The sequence data element started with 4 bytes of tag code, followed by 2 bytes of "SQ" and 2 bytes of 0x0000h bits. The value length is written in 4 bytes length integer as 0xFFFFFFFF. Stream of items started with 0xFFFE,0xE000 as item tag, followed by 4 bytes of item length and the item value. The sequence must be ended with sequence delimitation item 0xFFFE,0xE0DD and 0x00000000. All bits must be written in little endian byte order. We can see the detail implementation in real C code as follow :



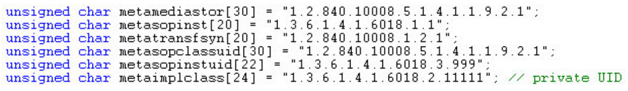
```
metaspec.ge = 0x0211003a; //Sequence started
strcpy(metaspec.vr,"SQ");
metaspec.reserved = 0x0000;
metaspec.vl = 0xffffffff;
fwrite(&metaspec,sizeof(metaspec),1,fp);

item.it = 0xe000fffe; //Item started
item.il = 0xffffffff;
fwrite(&item,sizeof(item),1,fp);

meta.ge = 0x01000008; //Item 1
strcpy(meta.vr,"SH");
meta.vl = 0x02;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&cvwf,sizeof(cvwf),1,fp);

meta.ge = 0x01020008; //Item 2
strcpy(meta.vr,"SH");
meta.vl = 0x04;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&csdwf,sizeof(csdwf),1,fp);

seqdelitem.sdit = 0xe00dfffe; //Item Delimitation
seqdelitem.sdil = 0x00000000;
fwrite(&seqdelitem,sizeof(seqdelitem),1,fp);

seqdelitem.sdit = 0xe0ddfffe; //Sequence delimitation
seqdelitem.sdil = 0x00000000;
fwrite(&seqdelitem,sizeof(seqdelitem),1,fp);
```

Figure 7. Implementation of sequence data element

7. Unique Identifiers

DICOM standard provides a mechanism to prevent duplication of medical data or images in one system. For example, a DICOM system will not has two identical DICOM files that contain ECG record of the patient named "BUDHI" that recorded on July 28, 2008 – 23:10:56 taken by using device "A". This mechanism called Unique Identifiers (UIDs).

For our prototype, we used 5 standard UIDs and 1 private UID. The 5 standard UIDs for 12 lead ECG waveform are : Media Storage UID, SOP Instance UID, Transfer Syntax UID, SOP Class UID, and SOP Instance UID. The private UID is Implementation Class UID (figure 8).



```
unsigned char metamediastor[30] = "1.2.840.10008.5.1.4.1.1.9.2.1";
unsigned char metasopinst[20] = "1.3.6.1.4.1.6018.1.1";
unsigned char metatransfsyn[20] = "1.2.840.10008.1.2.1";
unsigned char metasopclassuid[30] = "1.2.840.10008.5.1.4.1.1.9.2.1";
unsigned char metasopinstuid[22] = "1.3.6.1.4.1.6018.3.999";
unsigned char metaimplclass[24] = "1.3.6.1.4.1.6018.2.11111"; // private UID
```

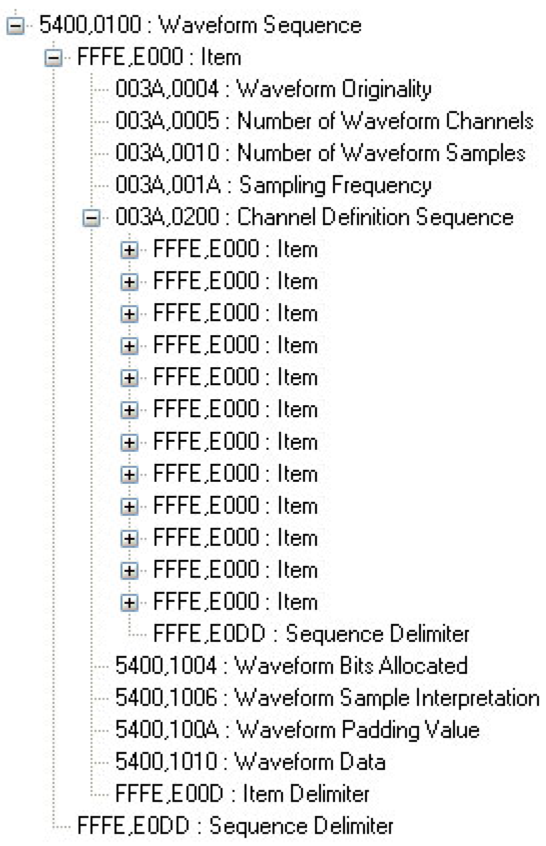
Figure 8. UID implementation in C code

Complete list of standard UID can be seen in DICOM specification PS3.6 Annex A [7]. To create a private UID, we can see the rule in DICOM specification PS3.5 Annex B [8].

8. Waveform Data Element

This element is main part of the DICOM waveform file. The data was encapsulated into a sequence data element which is not contains the signal only, but also the attributes of the waveform.

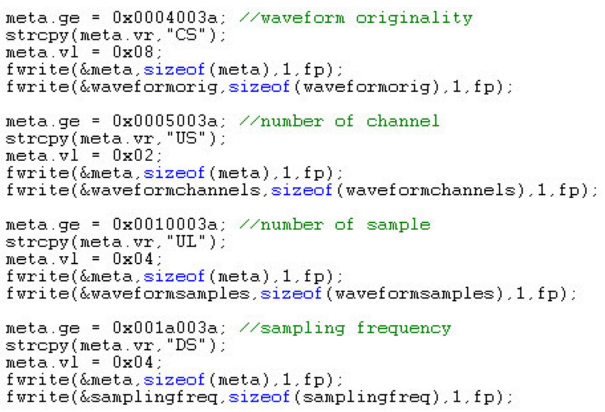
We can see the complete structure of the waveform data element bellow :



```
⊟ 5400,0100 : Waveform Sequence
    ⊟ FFFE,E000 : Item
        003A,0004 : Waveform Originality
        003A,0005 : Number of Waveform Channels
        003A,0010 : Number of Waveform Samples
        003A,001A : Sampling Frequency
        ⊟ 003A,0200 : Channel Definition Sequence
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
            ⊞ FFFE,E000 : Item
              FFFE,E0DD : Sequence Delimiter
        5400,1004 : Waveform Bits Allocated
        5400,1006 : Waveform Sample Interpretation
        5400,100A : Waveform Padding Value
        5400,1010 : Waveform Data
        FFFE,E00D : Item Delimiter
    FFFE,E0DD : Sequence Delimiter
```

Figure 9. The waveform data element structure

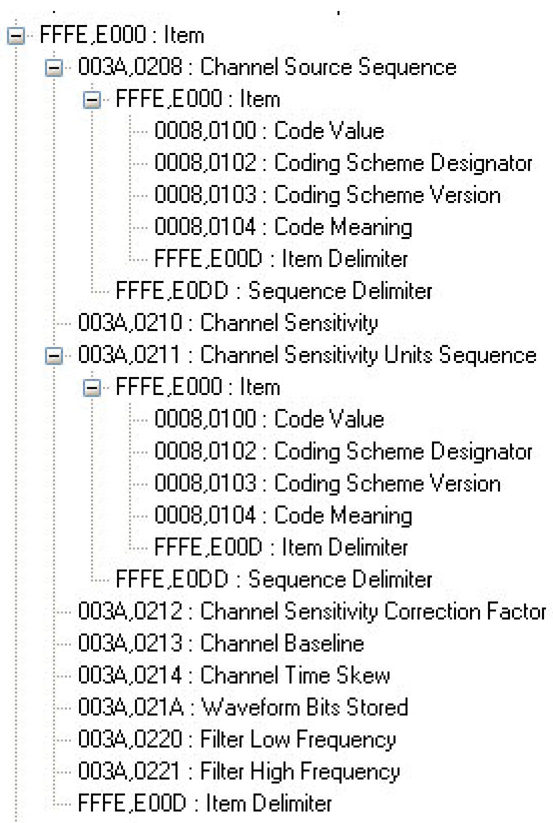
The first part of the sequence is waveform originality. For the 12 lead ECG data, the value must be “ORIGINAL” with 0x003a,0x0004 as its tag. Then followed with number of the waveform channel, which is in our case was “12”, number of waveform sample (ours was 2,500), and sampling frequency (ours was 250). The implementation can be seen bellow :



```
meta.ge = 0x0004003a; //waveform originality
strcpy(meta.vr,"CS");
meta.vl = 0x08;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&waveformorig,sizeof(waveformorig),1,fp);

meta.ge = 0x0005003a; //number of channel
strcpy(meta.vr,"US");
meta.vl = 0x02;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&waveformchannels,sizeof(waveformchannels),1,fp);

meta.ge = 0x0010003a; //number of sample
strcpy(meta.vr,"UL");
meta.vl = 0x04;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&waveformsamples,sizeof(waveformsamples),1,fp);

meta.ge = 0x001a003a; //sampling frequency
strcpy(meta.vr,"DS");
meta.vl = 0x04;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&samplingfreq,sizeof(samplingfreq),1,fp);
```

Figure 10. Writing the waveform attributes

The second part of this sequence is channel definition sequence which is contains channel attributes. We have 12 number of channel and every channel has its own attributes. We can see the attributes as follow :



```
⊟ FFFE,E000 : Item
    ⊟ 003A,0208 : Channel Source Sequence
        ⊟ FFFE,E000 : Item
            0008,0100 : Code Value
            0008,0102 : Coding Scheme Designator
            0008,0103 : Coding Scheme Version
            0008,0104 : Code Meaning
            FFFE,E00D : Item Delimiter
          FFFE,E0DD : Sequence Delimiter
      003A,0210 : Channel Sensitivity
    ⊟ 003A,0211 : Channel Sensitivity Units Sequence
        ⊟ FFFE,E000 : Item
            0008,0100 : Code Value
            0008,0102 : Coding Scheme Designator
            0008,0103 : Coding Scheme Version
            0008,0104 : Code Meaning
            FFFE,E00D : Item Delimiter
          FFFE,E0DD : Sequence Delimiter
      003A,0212 : Channel Sensitivity Correction Factor
      003A,0213 : Channel Baseline
      003A,0214 : Channel Time Skew
      003A,021A : Waveform Bits Stored
      003A,0220 : Filter Low Frequency
      003A,0221 : Filter High Frequency
      FFFE,E00D : Item Delimiter
```

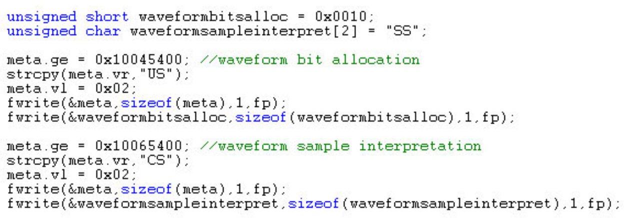
Figure 11. The channel attributes

Each channel started with annotation parts and followed with detailed attributes such as channel sensitivity correction factor, channel baseline, channel time skew, waveform bit stored, and frequency filters.

The number of channel represents the leads of the ECG data, which are lead I, lead II, lead III, lead aVR, lead aVL, lead aVF, lead V1, lead V2, lead V3, lead V4, lead V5, and lead V6.

The third part of the waveform data element is the interpretation part. It contains two data element : waveform bits allocated and waveform sample interpretation. This is the critical part, because the value will guide the DICOM viewer to interpret the entire sequence of the waveform data.

For 12 lead ECG, the value length of the waveform bits allocated must be 2 bytes and the value field must be “16”, written in binary little endian format. The value length for waveform sample interpretation must be 2 bytes with value field “SS”. The implementation of this part can be seen as follow :



```
unsigned short waveformbitsalloc = 0x0010;
unsigned char waveformsampleinterpret[2] = "SS";

meta.ge = 0x10045400; //waveform bit allocation
strcpy(meta.vr,"US");
meta.vl = 0x02;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&waveformbitsalloc,sizeof(waveformbitsalloc),1,fp);

meta.ge = 0x10065400; //waveform sample interpretation
strcpy(meta.vr,"CS");
meta.vl = 0x02;
fwrite(&meta,sizeof(meta),1,fp);
fwrite(&waveformsampleinterpret,sizeof(waveformsampleinterpret),1,fp);
```

Figure 12. Implementation of the interpretation part

The rest parts of the waveform data element are waveform padding value and waveform data.

The waveform padding value is a sample that represents a point in time but not of value. Equipment which produces digitized waveform curves may encode a specific value when the source is disconnected or otherwise invalid. This value is encoded like the Waveform Data attribute with one sample only [9].

We know that the data range of signed short integer is -32768～32768. In many cases the value of Waveform Padding Value is 32768 which is just not within the range of signed short integer (-32768～32768). In our case, we used 32768 or 0x8000 as our waveform padding value. Here is the implementation code :

```
unsigned short wfpaddingval = 0x8000;

metaspec.ge = 0x100a5400; //writting padding value
strcpy(metaspec.vr,"OW");
metaspec.reserved = 0x0000;
metaspec.vl = 0x00000002;
fwrite(&metaspec,sizeof(metaspec),1,fp);

fwrite(&wfpaddingval,sizeof(wfpaddingval),1,fp);
```

Figure 12. Implementation of waveform padding value

The waveform data itself can be described as follow :

Each waveform sample is written as a 16 bit binary value. The values are concatenated together with no delimiters (neither ASCII, not the Item delimiters that are used with undefined length Pixel Data). All waveform data requires defined length.

Since the values in our text file are floating point values and in millivolt, we have to multiply the values by 1000 before encoding them as 16 bit words. In this case, we were not depending on the native C data types.

A call like fwrite() is unsafe for two reasons:

the number of bytes written is dependent on the compiler and the host and target.

the byte order written is dependent on the target machine.

For each value written we must extract the bytes that we want into an array of char in the correct order and write it by hand. So in order to handle this issue, we wrote the code as follow :

```
float I;
long wfdataI;

fscanf(fpecg,"%f", &I);

wfdataI = (long)(I*1000);
bufferI[0]=(char)(wfdataI&0xff);
bufferI[1]=(char)((wfdataI>>8)&0xff);
fwrite(bufferI,2,1,fp);
```

Figure 13. Float to 16 bit words encoding

The “fpecg” is pointer to open the text file, and “fp” is the pointer to write the .dcm file as the result.

To read all records in the text file, we just modified the fscanf function and looped it until the end of the file.

The value length (in bytes) of the waveform data attribute is the number of samples multiplied by number of leads multiplied by the number of bytes per sample (which is 2 for OW). So in our case, the length of the waveform data attribute is 12 * 2500 * 2 = 60,000 or 0x0000EA60.

9. The Result

We use a web-based DICOM viewer which is support waveform data at http://www.excel-medical.com/waveforms/Viewer/DicomWaveformViewer.htm. This web-based DICOM viewer has been developed by using activeX technology from Microsoft.

Here is the view of our result file that produced by our DICOM Waveform Generator :

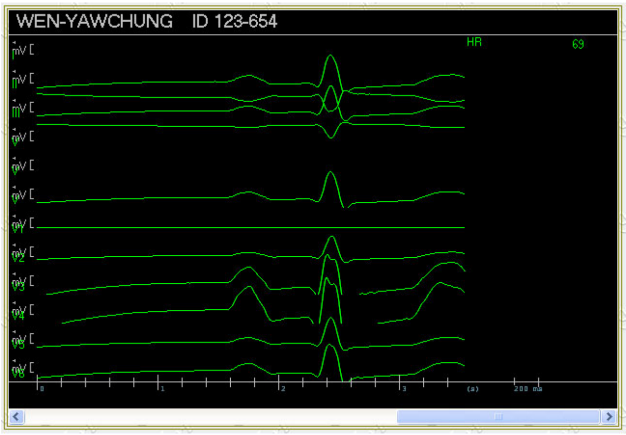


Figure 13. The DICOM result file

10. Discussion

We have developed DICOM Waveform Generator from the scratch based on DICOM 3.0 specification and DICOM Supplement 30 specification by using basic C language. The code was easily transferred and modified into embedded system or MCU based system.

Our DICOM waveform generator can be extended to cover more complicated waveform data in DICOM file since the need of medical treatments which are involves waveform data is rapidly increased.

References

[1] “Electrocardiogram”, [Last access 06/30/2008], available URL : http://en.wikipedia.org/wiki/Ecg
[2] Wang Ling-ling, Rao Ni-ni, Pu Li-xin, Wang Gang, “Developing a DICOM Middleware to Implement ECG Conversion and Viewing”, Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-4, 2005.
[3] DICOM Supplement 30: “Waveform Interchange”, Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2000.
[4] DICOM 3.0 PS3.5 Section 6.2: “Value Representation”, Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2004.
[5] DICOM 3.0 PS3.6 Section 6: “Registry of DICOM Data Elements”, Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2004.
[6] DICOM 3.0 PS3.5 Section 7.5: “Nesting of Data Sets”, Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2004.

[7] DICOM 3.0 PS3.6 Annex A: "Registry of DICOM Unique Identifiers", Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2004.
[8] DICOM 3.0 PS3.5 Annex B: "Creating a Privately Defined Unique Identifier", Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2004.
[9] DICOM Supplement 30 Section C.10.5.1.6: "Waveform Padding Value", Nat. Elect. Manufacturers Assoc.: ACR-NEMA, Digital Imaging and Communications in Medicine, NEMA, Washington D.C., 2000.
[10] V Sakkalis, F Chiarugi, S Kostomanolakis, CE Chronaki, M Tsiknakis, SC Orphanoudakis, "A Gateway Between the SCP-ECG and the DICOM Supplement 30 Waveform Standard", Computers in Cardiology, 2003.
[11] Shuqun Xie, Donglan Yu, Xianli Wei, Kuijian Wang, "The semantic extension and storage of EECP Hemodynamic Waveforms based on DICOM Standard", Medical Biology and Engineering Computation (2008) 46:391–397.
[12] T Becker, ET van der Velde, M BalJon, G de Stem, MGJM Gerritsen, S MeiJ, "The Development of an Application Profile for DICOM Waveforms in the Cathlab", IEEE Computers in Cardiology 1999;26:97-99.