DCMTK - Bug #1132

DcmIODUtil::extractBinaryFrames and DcmSegmentation::concatFrames not consistently working if bits per frame is not divisible by 8

2024-07-29 17:01 - Marco Eichelberg

Status:	Closed	**Start date:**	2024-07-29
Priority:	Normal	**Due date:**	
Assignee:	Michael Onken	**% Done:**	100%
Category:	Library	**Estimated time:**	0:00 hour
Target version:	3.6.9		
Module:	dcmiod, dcmseg	**Compiler:**	
Operating System:			

Description

There seems to be a bug in DcmIODUtil::extractBinaryFrames (dcmtk/dcmiod/libsrc/iodutil.cc). This does not consistently work correctly if the number of bits per frame is not divisible by 8. Depending on how many bits overlap between the frames it can happen that already after the first frame too few bytes of pixData are considered for the next frame. Thus, a shift over the frames occurs resulting in not even reading all bytes of pixData in the end.

For example, if the pixel data contains two frames with 7 bits per frame (for the sake of the example), then frameLengthBytes (= frame->length) will be 1. Hence, exactly one byte from the input pixData will be copied in frame->pixData for each frame. This works correctly for the first frame, but the second frame would clearly have to read bits from two bytes, namely 1 bit from the first byte and 6 bits from the second byte, which it does not. In fact, as bitShift = 1 after copying the first byte into the first frame, the second frame will only read the first byte again and the second byte of pixData will never be read in this example.

A matching logic bug appears in DcmSegmentation::concatFrames (dcmtk/dcmseg/libsrc/segdoc.cc). In the example of two frames with 7 bits per frame, the second frame would only have its first bit written into the first byte and the rest of its bits would be discarded.

As the example above shows, loading or saving pixel data into frames may have to “touch” (read data from or write data into) up to bitsPerFrame / 8 + 2 bytes if bitsPerFrame is not divisible by 8 (more specifically, if bitsPerFrame % 8 is 3, 5, 6 or 7), and not always bitsPerFrame / 8 + 1. This mistake causes some frames to “lose” bits due to overwriting them during saving and due to using bits double during reading (which can also lead to more bits being set to 1 than in the input pixData). This results in write/read offset errors that accumulate over the frames. Therefore, later frames end up loading (but also saving) data completely misaligned with the logical frame boundaries in pixData, and there are bytes at the end of pixData that are never written or read.

As far as we can tell, these two bugs cancel out for the most part: binary segmentations written with DCMTK and then loaded via DCMTK usually look mostly fine. However, the bits that are mishandled at the frame boundaries can cause artefacts, particularly for segmentations that reach the corners of the frame. More importantly though, segmentations written this way are objectively wrong and cannot be meaningfully imported by other software following DICOM specifications. The imported segmentations will be nonsensical. The fact that all of this only happens for “odd” frame sizes (see above) is most likely why this bug has not received attention before.

Reported 2024-07-09 by Melanie Michels <melanie.michels@snkeos.com> and Max Langhof <max.langhof@snkeos.com>

History

#1 - 2024-12-06 13:26 - Jörg Riesmeier

- Target version changed from 3.7.1+ to 3.6.9

#2 - 2024-12-11 09:49 - Jörg Riesmeier

- Status changed from New to Closed

- % Done changed from 0 to 100

Closed with commit 6245f9cc90.