Subject: Write_Image
Posted by idlfreak on Mon, 17 Dec 2001 22:43:07 GMT
View Forum Message <> Reply to Message

hi,
Can anybody please tell me how to save an image using WRITE_IMAGE function in DICOM format.
Any help is appreciated.
Thank you,

Regards,
Akhila....

Subject: Re: Write_Image
Posted by Bhautik Joshi on Wed, 19 Dec 2001 23:36:42 GMT
View Forum Message <> Reply to Message

> Can anybody please tell me how to save an image using WRITE_IMAGE
> function in DICOM format.
> Any help is appreciated.
This is the point where I tense the muscles in my hands into clawlike gestures and laugh maniaclly, eyes glowing, brain melting... "it can't be done! no! never!!!"

ahem

I spent a good couple of weeks trying to squeeze functionality out of IDL 5.4 to be able to write DICOM files and my conclusion is that it can't be done. Currently, there is plenty of support for *reading* DICOM files but for writing them, you'll need an external package.

Rumour is that RSI will soon release an extra medical imaging package for, amongst other things, writing to DICOM files. Like I said though, at the moment, you'll have to use some sort of external package, like the Offis DICOM toolkit:

http://www.offis.de/projekte/dicom/

I haven't been able to get the damn thing to compile yet (stupid SUN .h files), but once I do I shall bore you all with my adventures of succesfully combining IDL and DICOM.

Anyway, hope this hasn't put you off too much, best of luck with it anyway :)

Cheers,
Bhautik

--
 /--(__)----- ----\
| nbj@imag.wsahs.nsw.gov.au | phone: 0404032617 |..|--\ -moo |
| ICQ #: 2464537 | http://cow.mooh.org | |--| |
|---------------------------+----------------------\OO/|| ------/
| international |
| roast. my sanity has gone |
| its lost forever |
\---------------------------/

Subject: Re: Write_Image
Posted by Bhautik Joshi on Thu, 20 Dec 2001 00:47:07 GMT
View Forum Message <> Reply to Message

> Can anybody please tell me how to save an image using WRITE_IMAGE
> function in DICOM format.
> Any help is appreciated.

Just a little follow-up to my last post:

Yes, the help files in IDL 5.4 *do* list DICOM as one of the output
file-types for write_image.

"..
Format

A scalar string containing the name of the file format to write. The
following are the supported formats:

ï¿½ BMP
ï¿½ JPEG
ï¿½ PNG
ï¿½ PPM
ï¿½ SRF
ï¿½ TIFF
ï¿½ DICOM
.."

However this bit of documentation is a bit of a furphy [1], as
illustrated below:

MOO> foo=replicate(1.0,100,100)
MOO> write_image,'foo.dcm', 'DICOM',foo

% Compiled module: WRITE_IMAGE.
% WRITE_IMAGE: Unknown image file format: DICOM

% Error occurred at: WRITE_IMAGE   142
  /usr/local/rsi/idl_5.4/lib/write_image.pro
% $MAIN$
% Execution halted at: $MAIN$

For this, RSI should be smothered in peanut oil and liberally beaten
with a wet herring.

Cheers,
Bhautik

[1] furphy n.(pl.furphies) 1 a false report or rumour. 2 an absurd
story. adj.(furphier, furphiest) absurdly false, unbelievable: thats the
furphiest bit of news I ever heard.

see: http://www.anu.edu.au/ANDC/Ozwords/November_97/6._furphy.htm

--
 /---(__)----- ----\
| nbj@imag.wsahs.nsw.gov.au | phone: 0404032617 |..|--\ -moo |
| ICQ #: 2464537 | http://cow.mooh.org | |--| |
|---------------------------+----------------------\OO/|| ------/
| international |
| roast. my sanity has gone |
| its lost forever |
\---------------------------/

Subject: Re: Write_Image
Posted by Marcus O'Brien on Fri, 21 Dec 2001 14:08:35 GMT
View Forum Message <> Reply to Message

Hi,

I've included a c prog that gives the main tags and those that are
required by the DICOM MR image module (there aren't that many that are
required), this prog converts from tiff to DICOM. Should'nt be too hard
to IDLize. Grab yourself a copy of the standard, seem to recall 3,4,5
and 6 are the bits you need to cross reference to find the essential
tags and data representations for the modality of the file you wish to
produce.

Good luck

Marc O'Brien

Akhila wrote:
>

> hi,
> Can anybody please tell me how to save an image using WRITE_IMAGE
> function in DICOM format.
> Any help is appreciated.
> Thank you,
>
> Regards,
> Akhila....

--
Tel: 0208 725 2857
Fax: 0208 725 2992

St George's Hospital Medical School
Department of Biochemistry and Immunology
Cranmer Terrace
Tooting
London SW17 0RE

```
/*
** TIFF_to_DICOM.c - converts a Tagged Image File to a DICOM
**
** Derived by Marc O'Brien from tif2ras.c, which is:
**
** Copyright (c) 1990 by Sun Microsystems, Inc.
**
** Author: Marc O'Brien
** marcus@icr.ac.uk
**
** Permission to use, copy, modify, and distribute this software and its
** documentation for any purpose and without fee is hereby granted,
** provided that the above copyright notice appear in all copies and that
** both that copyright notice and this permission notice appear in
** supporting documentation.
**
** This file is provided AS IS with no warranties of any kind.  The author
** shall have no liability with respect to the infringement of copyrights,
** trade secrets or any patents by this file or any part thereof.  In no
** event will the author be liable for any lost revenue or profits or
** other special, indirect and consequential damages.

 ************************************************************ ***************
 ************************************************************ ***************
** This program converts Cadplan radiotherapy planning system
** screendumps into secondary capture DICOM files.
** The screendumps are 1280 x 640, which represents the two
** planning windows on the system, and are stored as tiff photometric
** palette files.
** The tiff reading part of the program is taken almost entirely from
```

```
** tifftopnm.c by Jef Poskanzer.
** The DICOM file is written to conform to the secondary capture module
** although patient name is set to "SECONDARY CAPTURE", as the ctn display
** and print routines I use utilise the patient name field.
** By default the program converts the left half of the screendump to a DICOM
** image. Adding a flag after the input file allows for the wholescreen or
** just the right hand side to be converted.
** The colored contours are converted to greyscale by conversion to HSV
** color space, discarding the Hue and Saturation values, then converting
** back to RGB.
** This program does not at present represent a general tiff to DICOM
** conversion utility, as it has been written primarily to convert Cadplan
** screendumps, although with a little hacking it should be capable of
** handling most forms of tiff image.
** The program writes the dicom image to the directory where the print
** program, create_icons, looks for its images, /pcscratch1/imagespool/SC
**
** Marc O'Brien
**
 ********************************************************** **********
 ********************************************************** **********
*/

#include "./libtiff/tiffio.h"
#include "colconv.h"
#include <time.h>
#include <string.h>
#include <stdio.h>

typedef unsigned short pixel;
typedef unsigned char pixval;


#define MAXCOLORS 1024
#define MAXMAXVAL 2048
#define DEFAULT 1
#define RIGHT 2
#define BOTH 3

#define ASSIGN(p,red,grn,blu) (p) = ((pixel) (red) << 20) | ((pixel) (grn) << 10) | (pixel) (blu)
#define ASSIGN1(x,v) ASSIGN(x,0,0,v)

int main( argc, argv )
int argc;
char* argv[];

{
    int cols, rows, grayscale, format, fileexists;
```

```
    int numcolors;
    register TIFF* tif;
    int row, i;
    register int col;
    unsigned char* buf;
    register unsigned char* inP;
    int maxval;
int maxgrey, mingrey;

    int headerdump;
    register unsigned char sample, xelval;
    register int bitsleft;
    unsigned short bps, spp, photomet;
    unsigned short* redcolormap;
    unsigned short* greencolormap;
    unsigned short* bluecolormap;
unsigned long colormap[MAXCOLORS];
pixel* xelrow;
pixel* xP;
float h, s, v, rf, gf, bf;

unsigned int ImageBytes;

/* prototype declaration */
int WriteStringElement(unsigned short, unsigned short, char *, FILE *);
int WriteBinaryElement(unsigned short, unsigned short, unsigned short, FILE *);
int WritePixelTag(unsigned short, unsigned short, unsigned int, FILE *);
void swap(unsigned short *);

/* Time structures for constructing a unique UID */
clock_t TheClock;
char UniqueString[12];

    char* usage = "tifftodicom <tifffile>\n";
char* SOPClass = "1.2.840.10008.5.1.4.1.1.4";
char* SOPInstance = "1.2.840.10008.5.1.4.1.1.4.1";
char* outputdir = ".//";
char imgfile[64];
char SeriesInstanceUID[64];
char StudyInstanceUID[64];
char RelFrameOfReferenceUID[64];
char StudyID[17];
char ValBuffer[1025];
char windowwidth[16];
char windowcenter[16];
int DSLength;
FILE *fp;
```

```
    headerdump = 1;
    if(argc > 1)
    {
  if(argc == 3)
  {
  }
  tif = TIFFOpen( argv[1], "r" );
  printf("\nOpening %s\n", argv[1]);
    }
    else
 {
  printf("\nNo tiff file specified:\n%s\n", usage);
  exit(0);
 }

    if ( tif == NULL ) printf( "error opening TIFF file %s", argv[1]);
    else
    {

     if ( headerdump )TIFFPrintDirectory( tif, stderr, TIFFPRINT_NONE );
     if ( ! TIFFGetField( tif, TIFFTAG_BITSPERSAMPLE, &bps ) )bps = 1;
     if ( ! TIFFGetField( tif, TIFFTAG_SAMPLESPERPIXEL, &spp ) )spp = 1;
     if ( ! TIFFGetField( tif, TIFFTAG_PHOTOMETRIC, &photomet ) )printf( "error getting
photometric" );

     switch ( spp )
  {
   case 1:
   case 3:
   case 4:
   break;

   default:
   printf("can only handle 1-channel gray scale or 1- or 3-channel color" );
  }

     (void) TIFFGetField( tif, TIFFTAG_IMAGEWIDTH, &cols );
     (void) TIFFGetField( tif, TIFFTAG_IMAGELENGTH, &rows );

     if ( headerdump )
  {
   printf( "\t%dx%dx%d image\n", cols, rows, bps * spp );
   printf( "\t%d bits/sample, %d samples/pixel\n", bps, spp );
  }

     maxval = ( 1 << bps ) - 1;
     if ( maxval == 1 && spp == 1 )
     {
```

```
   if ( headerdump ) printf("monochrome" );
   grayscale = 1;
      }
      else
  {
   switch ( photomet )
       {
        case PHOTOMETRIC_MINISBLACK:
        if ( headerdump ) printf( "%d graylevels (min=black)", maxval + 1 );
        grayscale = 1;
    maxgrey = 255;
        break;

        case PHOTOMETRIC_MINISWHITE:
        if ( headerdump ) printf( "%d graylevels (min=white)", maxval + 1 );
        grayscale = 1;
        break;


    /* Format supplied by Cadplan tiff screen dumps */
        case PHOTOMETRIC_PALETTE:
        if ( headerdump ) printf( "\tcolormapped\n" );
        if ( ! TIFFGetField( tif, TIFFTAG_COLORMAP, &redcolormap, &greencolormap,
&bluecolormap ) )
     printf( "error getting colormaps" );
        numcolors = maxval + 1;
        if ( numcolors > MAXCOLORS ) printf( "too many colors" );
        maxval = numcolors;
        grayscale = 0;
        maxval = MAXMAXVAL;
    maxgrey = 0;
    mingrey = MAXMAXVAL + 1;


    /* Load colormap */
        for ( i = 0; i < numcolors; ++i )
    {
     register unsigned long r, g, b;
     r = (unsigned long) redcolormap[i] * MAXMAXVAL / 65535L;
     g = (unsigned long) greencolormap[i] * MAXMAXVAL / 65535L;
     b = (unsigned long) bluecolormap[i] * MAXMAXVAL / 65535L;

     /* Check to see if color is a grey */
     if( r == g && g == b )
     {
      ASSIGN1( colormap[i], b );
      if(b>maxgrey) maxgrey = b;
      if(b<mingrey) mingrey = b;
```

```
   }
   else
   {
    /* Color is not a grey so convert to HSV, discard color
       information, then convert back to RGB */

    rf = (float)r / (MAXMAXVAL + 1);
    gf = (float)g / (MAXMAXVAL + 1);
    bf = (float)b / (MAXMAXVAL + 1);
    RGB_to_HSV(rf,gf,bf,&h,&s,&v);
    h = UNDEFINED; s = 0;

    if(HSV_to_RGB(h,s,v,&rf,&gf,&bf))
    {
     r = (unsigned long)(rf * (MAXMAXVAL + 1));
     g = (unsigned long)(gf * (MAXMAXVAL + 1));
     b = (unsigned long)(bf * (MAXMAXVAL + 1));
     ASSIGN1( colormap[i], r );
     if(r>maxgrey) maxgrey = r;
     if(r<mingrey) mingrey = r;
    }
    else
    {
     printf("\tHSV_to_RGB failed for r=%u, g=%u, b=%u, h=%f, s=%f, v=%f\n",
      r,g,b,h,s,v);
     ASSIGN1( colormap[i], 0);
    }

   }

  } /* End of for loop */


 } /* End of switch statement */
} /* End of else not greyscale */

printf("\tmaxgrey = %i\n\tmingrey = %i\n", maxgrey, mingrey);

/* Calculate window width and window center */
sprintf(windowwidth, "%i",(maxgrey - mingrey));
DSLength = strlen(windowwidth);
if((DSLength % 2) != 0)
{
 windowwidth[DSLength] = ' ';
 windowwidth[DSLength+1] = NULL;
}

sprintf(windowcenter, "%i", ((maxgrey - mingrey)/(unsigned int)2)+1);
```

```
DSLength = strlen(windowcenter);
if((DSLength % 2) != 0)
{
 windowcenter[DSLength] = ' ';
 windowcenter[DSLength+1] = NULL;
}

/* Read and write scanlines here */

    buf = (unsigned char*) malloc(TIFFScanlineSize(tif));
    if ( buf == NULL )
printf( "can't allocate memory for scanline buffer" );

/* Write DICOM header here */

/* Setup unique UIDs */
TheClock = clock();
time((time_t *)&TheClock);
sprintf(UniqueString, ".%i", TheClock);

strcpy(imgfile, outputdir);
strcat(imgfile, "MR");
strcat(imgfile, UniqueString);
strcpy(StudyInstanceUID, SOPInstance);
strcat(StudyInstanceUID, UniqueString);
strcpy(SeriesInstanceUID, StudyInstanceUID);
strcpy(RelFrameOfReferenceUID, StudyInstanceUID);
strcat(SeriesInstanceUID, ".1");
strcat(RelFrameOfReferenceUID, ".2");
strcat(StudyID, "SIGNA ");

if((fp = fopen(imgfile, "w")) != NULL)
{

 /* Write tags and data elements */

 /* Write group 0008H tags here */

 strcpy((char *)ValBuffer,"ORIGINAL\\PRIMARY\\OTHER");
 /* Type of MR image */
 if(WriteStringElement(0x0008, 0x0008, (char *)ValBuffer, fp))
      printf("\tWrote 0008H, 0008H data element\n");
 else printf("\tFailed to write 0008H, 0008H data element\n");

 /* Instance date */
 if(WriteStringElement(0x0008, 0x0012, (char *)"20010820", fp))
      printf("\tWrote 0008H, 0012H data element\n");
 else printf("\tFailed to write 0008H, 0012H data element\n");
```

```
/* Instance time */
if(WriteStringElement(0x0008, 0x0013, (char *)"20010820", fp))
    printf("\tWrote 0008H, 0013H data element\n");
else printf("\tFailed to write 0013H, 0008H data element\n");


/* SOP Class */
if(WriteStringElement(0x0008, 0x0016, SOPClass, fp))
    printf("\tWrote 0008H, 0016H data element\n");
else printf("\tFailed to write 0008H, 0016H data element\n");

/* SOP instance */
if(WriteStringElement(0x0008, 0x0018, SOPInstance, fp))
    printf("\tWrote 0008H, 0018H data element\n");
else printf("\tFailed to write 0008H, 0018H data element\n");

/* Study date */
if(WriteStringElement(0x0008, 0x0020, (char *)"", fp))
    printf("\tWrote 0008H, 0020H data element\n");
else printf("\tFailed to write 0008H, 0020H data element\n");

/* Image date */
if(WriteStringElement(0x0008, 0x0023, (char *)"", fp))
    printf("\tWrote 0008H, 0023H data element\n");
else printf("\tFailed to write 0008H, 0023H data element\n");

/* Study time */
if(WriteStringElement(0x0008, 0x0030, (char *)"", fp))
    printf("\tWrote 0008H, 0030H data element\n");
else printf("\tFailed to write 0008H, 0030H data element\n");

/* Image time */
if(WriteStringElement(0x0008, 0x0033, (char *)"", fp))
    printf("\tWrote 0008H, 0033H data element\n");
else printf("\tFailed to write 0008H, 0033H data element\n");

/* Study accession number */
if(WriteStringElement(0x0008, 0x0050, (char *)"", fp))
    printf("\tWrote 0008H, 0050H data element\n");
else printf("\tFailed to write 0008H, 0050H data element\n");

/* Modality */
if(WriteStringElement(0x0008, 0x0060, (char *)"MR", fp))
    printf("\tWrote 0008H, 0060H data element\n");
else printf("\tFailed to write 0008H, 0060H data element\n");

/* Manufacturer */
```

```
if(WriteStringElement(0x0008, 0x0070, (char *)"GE", fp))
    printf("\tWrote 0008H, 0070H data element\n");
else printf("\tFailed to write 0008H, 0070H data element\n");

/* Study physicians name */
if(WriteStringElement(0x0008, 0x0090, (char *)"Dominick Mcintyre", fp))
    printf("\tWrote 0008H, 0090H data element\n");
else printf("\tFailed to write 0008H, 0090H data element\n");


/* Write group 0010H tags here */
/* Patient name */
if(WriteStringElement(0x0010, 0x0010, (char *)"Use^^Miss", fp))
    printf("\tWrote 0010H, 0010H data element\n");
else printf("\tFailed to write 0010H, 0010H data element\n");

/* Patient ID */
if(WriteStringElement(0x0010, 0x0020, (char *)"testID", fp))
    printf("\tWrote 0010H, 0020H data element\n");
else printf("\tFailed to write 0010H, 0020H data element\n");

/* Patient birth date */
if(WriteStringElement(0x0010, 0x0030, (char *)"20010820", fp))
    printf("\tWrote 0010H, 0030H data element\n");
else printf("\tFailed to write 0010H, 0030H data element\n");

/* Patient sex */
if(WriteStringElement(0x0010, 0x0040, (char *)"F", fp))
    printf("\tWrote 0010H, 0040H data element\n");
else printf("\tFailed to write 0010H, 0040H data element\n");

/* Patient weight */
if(WriteStringElement(0x0010, 0x1030, (char *)"F", fp))
    printf("\tWrote 0010H, 1030H data element\n");
else printf("\tFailed to write 0010H, 1030H data element\n");


/* Write group 0018H tags here */
/* Contrast */
if(WriteStringElement(0x0018, 0x0010, (char *)"", fp))
    printf("\tWrote 0018H, 0010H data element\n");
else printf("\tFailed to write 0018H, 0010H data element\n");

/* Scanning sequence */
if(WriteStringElement(0x0018, 0x0020, (char *)"RM", fp))
    printf("\tWrote 0018H, 0020H data element\n");
else printf("\tFailed to write 0018H, 0020H data element\n");
```

```
/* Sequence variant */
if(WriteStringElement(0x0018, 0x0021, (char *)"NONE", fp))
    printf("\tWrote 0018H, 0021H data element\n");
else printf("\tFailed to write 0018H, 0021H data element\n");

/* Scan options */
if(WriteStringElement(0x0018, 0x0022, (char *)"", fp))
    printf("\tWrote 0018H, 0022H data element\n");
else printf("\tFailed to write 0018H, 0022H data element\n");

/* Aquisition type */
if(WriteStringElement(0x0018, 0x0023, (char *)"2D", fp))
    printf("\tWrote 0018H, 0023H data element\n");
else printf("\tFailed to write 0018H, 0023H data element\n");

/* Slice thickness */
if(WriteStringElement(0x0018, 0x0050, (char *)"5.000000", fp))
    printf("\tWrote 0018H, 0050H data element\n");
else printf("\tFailed to write 0018H, 0050H data element\n");

/* Repetition time */
if(WriteStringElement(0x0018, 0x0080, (char *)"", fp))
    printf("\tWrote 0018H, 0080H data element\n");
else printf("\tFailed to write 0018H, 0080H data element\n");

/* Echo time */
if(WriteStringElement(0x0018, 0x0081, (char *)"", fp))
    printf("\tWrote 0018H, 0081H data element\n");
else printf("\tFailed to write 0018H, 0081H data element\n");

/* Inversion time */
if(WriteStringElement(0x0018, 0x0082, (char *)"", fp))
    printf("\tWrote 0018H, 0082H data element\n");
else printf("\tFailed to write 0018H, 0082H data element\n");


/* Echo train length */
if(WriteStringElement(0x0018, 0x0091, (char *)"", fp))
    printf("\tWrote 0018H, 0091H data element\n");
else printf("\tFailed to write 0018H, 0091H data element\n");

/* Trigger time */
if(WriteStringElement(0x0018, 0x1060, (char *)"", fp))
    printf("\tWrote 0018H, 1060H data element\n");
else printf("\tFailed to write 0018H, 1060H data element\n");

/* Patient position */
if(WriteStringElement(0x0018, 0x5100, (char *)"HFS", fp))
```

```
    printf("\tWrote 0018H, 5100H data element\n");
  else printf("\tFailed to write 0018H, 5100H data element\n");


  /* Write group 0020H tags here */
  /* Study Instance */
  if(WriteStringElement(0x0020, 0x000D, StudyInstanceUID, fp))
    printf("\tWrote 0020H, 000DH data element\n");
  else printf("\tFailed to write 0020H, 000DH data element\n");

  /* Series instance UID */
  if(WriteStringElement(0x0020, 0x000E, SeriesInstanceUID, fp))
    printf("\tWrote 0020H, 000EH data element\n");
  else printf("\tFailed to write 0020H, 000EH data element\n");

  /* Study ID */
  if(WriteStringElement(0x0020, 0x0010, StudyID, fp))
    printf("\tWrote 0020H, 0010H data element\n");
  else printf("\tFailed to write 0020H, 0010H data element\n");

  /* Series number */
  if(WriteStringElement(0x0020, 0x0011, (char *)"", fp))
    printf("\tWrote 0020H, 0011H data element\n");
  else printf("\tFailed to write 0020H, 0011H data element\n");

  /* Image Number */
  if(WriteStringElement(0x0020, 0x0013, (char *)"", fp))
    printf("\tWrote 0020H, 0013H data element\n");
  else printf("\tFailed to write 0020H, 0013H data element\n");

  /* Patient orientation */
  if(WriteStringElement(0x0020, 0x0020, (char *)"", fp))
    printf("\tWrote 0020H, 0020H data element\n");
  else printf("\tFailed to write 0020H, 0020H data element\n");

  /* Image position (patient) */
  if(WriteStringElement(0x0020, 0x0032, (char *)"-178.149664\\-1.807587\\150.259369", fp))
    printf("\tWrote 0020H, 0032H data element\n");
  else printf("\tFailed to write 0020H, 0032H data element\n");

  /* Image orientation (patient) */
  if(WriteStringElement(0x0020, 0x0037, (char
*)"1.000000\\0.000000\\0.000000\\0.000000\\0.000000\\-1.000000 ", fp))
    printf("\tWrote 0020H, 0037H data element\n");
  else printf("\tFailed to write 0020H, 0037H data element\n");

  /* Frame of reference */
  if(WriteStringElement(0x0020, 0x0052, RelFrameOfReferenceUID, fp))
```

```
    printf("\tWrote 0020H, 0052H data element\n");
else printf("\tFailed to write 0020H, 0052H data element\n");

/* Laterality */
if(WriteStringElement(0x0020, 0x0060, (char *)"", fp))
    printf("\tWrote 0020H, 0060H data element\n");
else printf("\tFailed to write 0020H, 0060H data element\n");

/* Position reference indicator */
if(WriteStringElement(0x0020, 0x1040, (char *)"", fp))
    printf("\tWrote 0020H, 1040H data element\n");
else printf("\tFailed to write 0020H, 1040H data element\n");



/* Write group 0028H tags here */
/* Samples per pixel */
if(WriteBinaryElement(0x0028, 0x0002, (unsigned short)spp, fp))
    printf("\tWrote 0028H, 0002H data element\n");
else printf("\tFailed to write 0028H, 0002H data element\n");

/* Photometric interpretation */
if(WriteStringElement(0x0028, 0x0004, (char *)"MONOCHROME2", fp))
    printf("\tWrote 0028H, 0004H data element\n");
else printf("\tFailed to write 0028H, 0004H data element\n");

/* Rows in image */
if(WriteBinaryElement(0x0028, 0x0010, (unsigned short)rows, fp))
    printf("\tWrote 0028H, 0010H data element\n");
else printf("\tFailed to write 0028H, 0010H data element\n");

/* Columns in image */
if(WriteBinaryElement(0x0028, 0x0011, (unsigned short)cols, fp))
     printf("\tWrote 0028H, 0011H data element\n");
else printf("\tFailed to write 0011H, 0002H data element\n");

/* Pixel spacing */
if(WriteStringElement(0x0028, 0x0030, (char *)"1.464844\\1.464844", fp))
    printf("\tWrote 0028H, 0030H data element\n");
else printf("\tFailed to write 0028H, 0030H data element\n");

/* Bits allocatted per sample */
if(WriteBinaryElement(0x0028, 0x0100, (unsigned short)16, fp))
    printf("\tWrote 0028H, 0100H data element\n");
else printf("\tFailed to write 0028H, 0100H data element\n");

/* Bits stored per sample */
if(WriteBinaryElement(0x0028, 0x0101, (unsigned short)16, fp))
```

```
    printf("\tWrote 0028H, 0101H data element\n");
else printf("\tFailed to write 0028H, 0101H data element\n");

/* High bit */
if(WriteBinaryElement(0x0028, 0x0102, (unsigned short)15, fp))
    printf("\tWrote 0028H, 0102H data element\n");
else printf("\tFailed to write 0028H, 0102H data element\n");

/* Pixel representation */
if(WriteBinaryElement(0x0028, 0x0103, 0x0000, fp))
    printf("\tWrote 0028H, 0103H data element\n");
else printf("\tFailed to write 0028H, 0103H data element\n");


/* Write pixel data */

/* if(ImageCropFlag == DEFAULT || ImageCropFlag == RIGHT)
{
 ImageBytes = rows * (cols/2) * 2;
}
else if(ImageCropFlag == BOTH) */ ImageBytes = rows * cols * 2;

if(WritePixelTag(0x7FE0, 0x0010, ImageBytes, fp))
    printf("\tWrote 7FE0H, 0010H data element\n");
else printf("\tFailed to write 7FE0H, 0010H data element\n");



/* Read scanlines and write DICOM image data here */
xelrow = (pixel *)malloc(cols * sizeof(pixel));

/* An inline macro to read the bits from the scan line */
#define NEXTSAMPLE \
   { \
    if ( bitsleft == 0 ) \
 {  \
  ++inP; \
  bitsleft = 8; \
 } \
    bitsleft -= bps; \
    sample = ( *inP >> bitsleft ) & maxval; \
   }

   for ( row = 0; row < rows; ++row )
{
 if ( TIFFReadScanline( tif, buf, row, 0 ) < 0 ) printf( "bad data read on line %d", row );
 inP = buf;
 bitsleft = 8;
```

```
xP = xelrow;

switch ( photomet )
   {
    case PHOTOMETRIC_MINISBLACK:
    for ( col = 0; col < cols; ++col, ++xP )
{
 NEXTSAMPLE
 ASSIGN1( *xP, sample );
}
    break;

    case PHOTOMETRIC_MINISWHITE:
    for ( col = 0; col < cols; ++col, ++xP )
{
 NEXTSAMPLE
 sample = maxval - sample;
 ASSIGN1( *xP, sample );
}
    break;

    case PHOTOMETRIC_PALETTE:
    for ( col = 0; col < cols; ++col, ++xP )
{
 NEXTSAMPLE
 *xP = (unsigned short)colormap[sample];
}
    break;

    case PHOTOMETRIC_RGB:
    for ( col = 0; col < cols; ++col, ++xP )
{
 register unsigned char r, g, b;

 NEXTSAMPLE
 r = sample;
 NEXTSAMPLE
 g = sample;
 NEXTSAMPLE
 b = sample;
 if ( spp == 4 )
    NEXTSAMPLE  /* skip alpha channel */
 ASSIGN( *xP, r, g, b );
}
    break;

    default:
    printf( "unknown photometric: %d", photomet );
```

```
        }

    /* Byte swap the image data */
    for(i=0;i<(rows*sizeof(unsigned short));i++)
    {
     swap(&(xelrow[i]));
    }
    /* Write row */
    if(!fwrite((char *)xelrow, sizeof(pixel), cols, fp)) printf("\tFailed to write row\n");



   } /* End of for rows loop */

   fclose(fp);
   free(xelrow);

  } /* End of if file opened ok */
  else printf("\tUnable to open output file %s\n", imgfile);
  free(buf);

 } /* End of else able to open tiff file */


     exit((int) 0 );
} /* End of main */



int WriteStringElement(unsigned short Group, unsigned short Element, char *Buffer, FILE
*fpointer )
{
 void swap(unsigned short *);
 int result = 0;
 int i;
 unsigned char temp;

 /* Storage for DICOM data element */
 unsigned short DataTag[2];  /* fixed */
 unsigned short ValueLength;
 unsigned short Val;
 unsigned short Padding = 0x000;
 unsigned char ValueBuffer[1025]; /* Used for all string data types */


 DataTag[0] = Group;
 swap(&(DataTag[0]));
 DataTag[1] = Element;
```

```
 swap(&(DataTag[1]));


 strcpy((char *)ValueBuffer, Buffer);
 ValueLength = (unsigned short)strlen((const char *)ValueBuffer);

 /* Check if value field needs padding to an even length */
 if((ValueLength % 2) != 0)
 {
  ValueBuffer[ValueLength] = NULL;
  ValueLength += 1;
 }
 Val = ValueLength;
 swap(&Val);


 if(fwrite((char *)&DataTag, sizeof(int), 1, fpointer) && fwrite((char *)&Val, sizeof(unsigned short),
1, fpointer) &&
 fwrite((char *)&Padding, sizeof(unsigned char), 2, fpointer))
 {
  /* Check for type 2 field, tags must be present but length 0 and no value */
  if(ValueLength != 0)
  {
   if(fwrite((char *)ValueBuffer, sizeof(unsigned char), (int)ValueLength, fpointer)) result = 1;
  }
  else
  {
   result = 1;
  }
 }
 return result;

} /* End of WriteStringElement */



int WriteBinaryElement(unsigned short Group, unsigned short Element, unsigned short Value,
FILE *fpointer )
{
 void swap(unsigned short *);
 int result = 0;

 /* Storage for DICOM data element */
 unsigned short DataTag[2];  /* fixed */
 unsigned short ValueLength;
 unsigned short Padding = 0x0000;

 DataTag[0] = Group;
```

```
 swap(&(DataTag[0]));
 DataTag[1] = Element;
 swap(&(DataTag[1]));

 ValueLength = 2;
 swap(&ValueLength);
 swap(&Value);

 if(fwrite((char *)&DataTag, sizeof(int), 1, fpointer) && fwrite((char *)&ValueLength,
sizeof(unsigned short), 1, fpointer) &&
 fwrite((char *)&Padding, sizeof(unsigned char), 2, fpointer) && fwrite((char *)&Value,
sizeof(unsigned short), 1, fpointer))
 {
  result = 1;
 }
 return result;

} /* End of WriteBinaryElement */



int WritePixelTag(unsigned short Group, unsigned short Element, unsigned int ValueLength, FILE
*fpointer )
{
 void swap(unsigned short *);
 int result = 0;
 int i;

 /* Storage for DICOM data element */
 unsigned short DataTag[2];  /* fixed */
 unsigned char *Val;
 unsigned char temp;


 DataTag[0] = Group;
 swap(&(DataTag[0]));
 DataTag[1] = Element;
 swap(&(DataTag[1]));


 /* Swap to little endian byte order */
 Val = (unsigned char *)&ValueLength;
 temp = Val[3];
 Val[3] = Val[0];
 Val[0] = temp;
 temp = Val[2];
 Val[2] = Val[1];
 Val[1] = temp;
```

```
 if(fwrite((char *)&DataTag, sizeof(int), 1, fpointer) && fwrite((char *)Val, sizeof(int), 1, fpointer))
 {
  result = 1;
 }

 return result;

} /* End of WritePixelTag */



/* Function to byte swap a word (16 bit) */
void swap(unsigned short *word)
{
 unsigned char *swapbuffer;
 unsigned char temp;

 swapbuffer = (unsigned char *)word;
 temp = swapbuffer[0];
 swapbuffer[0] = swapbuffer[1];
 swapbuffer[1] = temp;
} /* End of swap */
```

File Attachments

1) tifftodicom.c, downloaded 118 times

Subject: Re: Write_Image
Posted by idlfreak on Sun, 30 Dec 2001 19:00:36 GMT
View Forum Message <> Reply to Message

thanx for the help...but i'm a new user to IDL and i have no idea how
to compile and run the C code in IDL. I've got no idea about the
conversion procedure involved. Can anybody please guide me through
that?

Regards,
Akhila...

Subject: Re: Write_Image
Posted by David Fanning on Sun, 30 Dec 2001 19:34:35 GMT
View Forum Message <> Reply to Message

Akhila (idlfreak@yahoo.com) writes:

> thanx for the help...but i'm a new user to IDL and i have no idea how
> to compile and run the C code in IDL. I've got no idea about the
> conversion procedure involved. Can anybody please guide me through
> that?

You might have to do what all the rest of us do:
buy Ronn Kling's book:

http://www.kilvarock.com/books/booksforsale.htm

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155
