SANDIA REPORT
SAND2006-2357
Unlimited Release
Printed April 2006

Viewing GFF format SAR Images with Matlab

William H. Hensley, Jr. and Armin W. Doerry

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2006-2357
Unlimited Release
Printed April 2006

Viewing GFF format SAR Images with Matlab

William H. Hensley, Jr. and Armin W. Doerry
SAR Applications Department

Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185-1330

ABSTRACT

Sandia radar systems output their images and image products in Sandia's unique GFF file format. Sandia wishes and encourages other government agencies and research institutions to use and exploit its SAR images, and consequently is making available its Matlab GFF file reading software via this report.

ACKNOWLEDGEMENTS

This work was funded by the US DOE Office of Nonproliferation & National Security, Office of Research and Development, NA-22, under the Advanced Radar System (ARS) project.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

CONTENTS

ABSTRACT 3
ACKNOWLEDGEMENTS 4
CONTENTS 5
FOREWORD 6
1 Introduction 7
2 Matlab Source Listings 7
3 Conclusions 28
Distribution 28

FOREWORD

During the initial development of its Synthetic Aperture Radar (SAR) systems, Sandia required image annotation beyond that achievable with standard image file formats available at the time. Consequently Sandia developed its own image file format known as GSAT File Format, or GFF. (The acronym GSAT denotes Ground-based SAR Applications Testbed.) The GFF file format has undegone several revisions as SAR development programs have come and gone, but remains Sandia's principal SAR image format. Sandia today has a large library of SAR images in GFF format.

1 Introduction

Sandia radar systems output their images and image products in Sandia's unique GFF file format. The GFF file format was developed to allow custom annotation of SAR images, as well as storing complex (magnitude and phase) images for post-processing and exploitation. Sandia's principal image analysis tool is the software package Matlab™, available from The MathWorks.[1] Consequently, Sandia has developed Matlab tools (programs) to read GFF files.

Sandia wishes and encourages other government agencies and research institutions to use and exploit its SAR images, and consequently is making available its Matlab GFF file reading software via this report. What follows are Matlab source listings for reading GFF files into a Matlab work space.

It should be recognized by the reader that the GFF file format is evolving, and that the following Matlab Source listings are merely a snapshot of an article in development. Consequently, this software is provided as is, with no guarantee, warranty, or other assurance of functionality or correctness for any purpose. Furthermore, users should have no expectation of support from the authors or Sandia National Laboratories. In other words, use at your own risk.

2 Matlab Source Listings

Source listings are provided for the following three files.

gffquickview.m	the top-level shell routine that calls the file reader and displays the image.
load_gff_1_8b.m	the GFF file reader.
read_gff_header_1_8b.m	called by load_gff_1_8b.m to read the header information.

What follows is the source listing for these files.

[1] http://www.mathworks.com/

gffquickview.m

```
% gffquickview      open and view Sandia GFF image file
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% This software is provided as is, with no guarantee,
%% warranty, or other assurance of functionality or
%% correctness for any purpose.  Furthermore, users should
%% have no expectation of support from the authors or
%% Sandia National Laboratories.  Use at your own risk.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Author:    A. W. Doerry, 5342
% Written:   28March2006
% Copyright: 2006 Sandia National Laboratories

% awd   20060328    initial coding
%

clear all;
close all;

pathname = '.';

[Image, Header, fname] = load_gff_1_8b('defaultPath',pathname);

if Header.BytesPerPixel>1,
    im_qp = sqrt(abs(Image));
else
    im_qp = abs(Image);
end
im_qp = 255*im_qp/max(max(im_qp));

im_qp = round(min(4*im_qp,255));

figure; image(im_qp);
colormap(gray(256));
axis('image');
```

load_gff_1_8b.m

```
function  [Image, Header, FileName ]= load_gff_1_8b( FileName, PathName )
% function <[>Image<, Header<, FileName>]> = load_gff_1_8b<( FileName )>;
%    OR    <[>Image<, Header<, FileName>]> = load_gff_1_8b<( 'defaultPath', PathName )>;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% This software is provided as is, with no guarantee,
%% warranty, or other assurance of functionality or
%% correctness for any purpose.  Furthermore, users should
%% have no expectation of support from the authors or
%% Sandia National Laboratories.  Use at your own risk.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function loads an image written according to the GSAT File
% Format specification [1] into the MATLAB workspace.  It optionally
% returns the values from the header in a MATLAB structure.
%
% Inputs:
% FileName (Optional)  If the optional file name argument is present,
%          it is assumed to point to a valid GSAT file.
%          If the argument is not present, the user is prompted to
%          select a file using the file open dialog box.
% PathName (Optional)  If the command line has two arguments, and
%          the first one is the string 'defaultpath' (any case),
%          the program interprets the PathName input as the starting
%          directory path to use for the file open dialog box.
%
% Outputs:
% Image    The array containing the (non-scaled, non-calibrated) image
%          pixels from the file.
% Header   (Optional) The MATLAB structure containing all fields of
```

```
%          the file header.
% FileName (Optional) The full path- and file-name of the file read.

% References:
% 1) Mendelsohn, G. H., et al, "GSAT Image File Format Specification"
%    Revision 1-6 (Draft), 02Feb00.

% Author:    W. H. Hensley, 2344
% Written:   8Feb2000
% Copyright: 2000 Sandia National Laboratories

% Revisions:
% 27May2000 WHH Added support for input filename.
%               Improved error handling and messaging.
% 28Jun2000 DLB  Put back in Bill's revision as follows:
% 14Mar2000 WHH Updated to support GFF Draft 1-6 as of 3/14/2000.  This
%               changed the way that complex mag/phase pixels are stored,
%               so that the phase is always first, regardless of "Endian".
% 08Jul2000 WHH Added support for default path input.

% Test History:
% 12Feb2000 WHH Tested against image from RTV Rev A RadarControl tape:
%               19980726.L01L004:VRTVCHECKOUT0:PASS0000:00000.GFF

%=====================================================================
% Check for input, and get file:
%=====================================================================
if ( nargin >= 2 ),
   [Header, gffName, gffPath, fid] = read_gff_header_1_8b('defaultPath',PathName)
   FileName = [gffPath gffName];
   if fid < 1,
      if fid == -3,
         errordlg(['Requested file (' gffName ') is NOT in GSAT format!'],'load_gff_1_8b');
      elseif fid == -2,
         errordlg(['Requested file (' gffName ') does not exist'],'load_gff_1_8b');
      else
         errordlg(['Header could not be read on file ' gffName ...
```

```
                ', function aborting.'],'load_gff');
      end

      Image = [];
      Header = [];
      return
   end
elseif ( nargin >= 1 ),
   [Header, gffName, gffPath, fid] = read_gff_header_1_8b(FileName);
   if fid < 1,
      if fid == -3,
         errordlg(['Requested file (' FileName ') is NOT in GSAT format!'],'load_gff');
      elseif fid == -2,
         errordlg(['Requested file (' FileName ') does not exist'],'load_gff');
      else
         errordlg(['Header could not be read on file ' FileName ...
                ', function aborting.'],'load_gff');
      end

      Image = [];
      Header = [];
      return
   end
else
   [Header, gffName, gffPath, fid] = read_gff_header_1_8b;
   FileName = [gffPath gffName];
   if fid < 1,
      if fid == -3,
         errordlg(['Requested file (' FileName ') is NOT in GSAT format!'],'load_gff');
      elseif fid == -2,
         errordlg(['Requested file (' FileName ') does not exist'],'load_gff');
      else
         errordlg(['Header could not be read on file ' FileName ...
                ', function aborting.'],'load_gff');
      end
      Image = [];
      Header = [];
```

```
      return
   end

end

if (Header.RowMajor)
   FirstIndexLength  = Header.RgCnt;
   SecondIndexLength = Header.AzCnt;
   Flip              = 0;
else
   FirstIndexLength  = Header.AzCnt;
   SecondIndexLength = Header.RgCnt;
   Flip              = 1;
end

switch Header.ImageType
case 0,
    fprintf('Reading sqrt(magnitude) pixels.\n');
    % Take care of RowMajor
    if Flip,
        Image = fread(fid,[FirstIndexLength,SecondIndexLength],'uchar').';
    else
        Image = fread(fid,[FirstIndexLength,SecondIndexLength],'uchar');
    end


case 1,
    fprintf('Reading complex (magnitude, phase) pixels.\n');
    if Header.BytesPerPixel == 4 %phase = 2 bytes, same with magnitude
        Image2 = fread(fid,[FirstIndexLength*2,SecondIndexLength],'ushort');
        % Convert to MATLAB complex:
        % [3/14/00 WHH All "Endian" cases now store phase first, followed
        % by magnitude.]
        Image = Image2(2:2:size(Image2,1),:) .*exp(j * Image2(1:2:size(Image2,1),:) * 2*pi/(2^16));
    else %BytesPerPixel=8 bytes=64 bits, therefore phase is 4 bytes=32 bits, mag =4 bytes=32 bits
        Image2 = fread(fid,[FirstIndexLength*2,SecondIndexLength],'ulong');
        % Convert to MATLAB complex:
```

```
        % [3/14/00 WHH All "Endian" cases now store phase first, followed
        % by magnitude.]
        Image = Image2(2:2:size(Image2,1),:) .*exp(j * Image2(1:2:size(Image2,1),:) * 2*pi/(2^32));
        % Take care of RowMajor
    end
    if Flip,
        Image = Image.';
    end

case 2,%Read in real+imag format pixels
    fprintf('Reading complex (real and imaginary) pixels.\n');
        Image2 = fread(fid,[2*FirstIndexLength SecondIndexLength],'float');
        Image = Image2(1:2:size(Image2,1),:)+j*Image2(2:2:size(Image2,1),:);

        %Take care of RowMajor
    if Flip,
        Image = Image.';
    end

otherwise,
    warndlg(['Unknown pixel type: ' num2str(Header.ImageType) '.'],'load_gff');
end %ends the switch statement

fprintf('FID in load_gff was %d.\n',fid);

fclose(fid);

return
```

read_gff_header_1_8b.m

```
function [Header, gffName, gffPath, fid_out] = read_gff_header_1_8b( FileName, PathName );
% function <[>Header<, gffName, gffPath <,fid_out>]> = read_gff_header_1_8b( <FileName> );
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% This software is provided as is, with no guarantee,
%% warranty, or other assurance of functionality or
%% correctness for any purpose.  Furthermore, users should
%% have no expectation of support from the authors or
%% Sandia National Laboratories.  Use at your own risk.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function reads the header of an image file written according
% to the GSAT File Format specification [1].  It returns the values
% from the header in a MATLAB structure, and (optionally) the file
% handle to the valid GFF file.
%
% It leaves the file open (with the proper "endian" setting), and
% positioned just past the end of the header (at the beginning of
% pixel data).
%
% If the file is NOT in GFF or is invalid in any other way, it is
% closed and fid_out will be < 0.  If the requested file does not
% exist, fid_out will be -2.
%
% Inputs:
% FileName (Optional)  If the optional file name argument is present,
%         it is assumed to be the name of a valid, existing GFF file.
%
%         If the argument is not present, the user is prompted to
%         select a file using the file open dialog box.
```

```
%
% Outputs:
% Header  The MATLAB structure containing all fields of the file header.
% gffName (Optional) The name of the opened file.  The input name is
%         returned if "FileName" is passed in.
% gffPath (Optional) The path to the opened file.  Null string returned
%         if "FileName" is passed in.
% fid_out (Optional) This function will sometimes close the file and
%         re-open it.  It also can be used to find a new file without
%         having a handle passed in.  In either case, the calling
%         routine can get the active file handle from this parameter.

% References:
% 1) Mendelsohn, G. H., et al, "GSAT Image File Format Specification"
%    Revision 1-6 (Draft), 02Feb00.

% Author:    W. H. Hensley, 2344
% Written:   4Feb2000
% Copyright: 2000 Sandia National Laboratories

% Revisions:
% 08Feb2000 WHH Changed scaling back to 2^-16 on APCAlt.
% 09Feb2000 WHH Changed scaling on RgChirpRate to 2^+12 from 2^-16!
%               Scaling on RgChirpRate now dependent on rev number.
% 27May2000 WHH Changed input argument to FileName and implemented.
% 08Jul2000 DLB/WHH Added multiple-extension support for Unix
%               Added default path support.
% 18Jul2000 WHH Now closes file if the 4th input parameter (fid_out)
%               is NOT supplied.
% 21Aug2000 WHH/DLB Fixed the order of AzResolution and RgResolution.
%               Prior to this, function did NOT meet gff 1/6 spec!

%=====================================================================
% Get a valid file, assure it's at beginning of file, and open with
% the correct "endian" interpretation:
%=====================================================================
% Only use the error dialogs if this was the top-level called function!
```

```
UseWarnMsgs = length(dbstack) < 2;

if (nargin >= 2)

   if isunix,
      if (PathName(length(PathName)) ~= '/') & (PathName(length(PathName)) ~= '\'),
         PathName = [PathName '/'];
      end
      FilterSpec = [[PathName '*.GFF|'],[PathName '*.gff']];
   else
      if (PathName(length(PathName)) ~= '/') & (PathName(length(PathName)) ~= '\'),
         PathName = [PathName '\'];
      end
      FilterSpec = [PathName '*.gff;*.GFF'];
   end

   [gffName, gffPath] = uigetfile(FilterSpec, 'Select GFF file to open');
   if (gffName == 0)
      fid_out = -1;
      Header = [];
      gffName = [];
      gffPath = [];
      return
   else
      fid = fopen([gffPath gffName],'r','ieee-be');
   end
elseif (nargin >=1)
   if exist( FileName, 'file' ),
      fid = fopen( FileName );
      if fid < 1,

         if UseWarnMsgs,
            errordlg(['GSAT File ' gffPath gffName ' cannot be opened.'],'read_gff_header');
         end

         fid_out = fid;
         Header = [];
```

```
            gffName = [];
            gffPath = [];
            return
         end
      else

         if UseWarnMsgs,
            errordlg(['GSAT File ' FileName ' does not exist.'],'read_gff_header');
         end

         fid_out = -2;
         Header = [];
         gffName = [];
         gffPath = [];
         return
      end
      gffName = FileName;
      gffPath = [];
   else % nargin = 0
      if isunix,
         FilterSpec = ['*.GFF|','*.gff'];
      else
         FilterSpec = '*.gff;*.GFF';
      end

      [gffName, gffPath] = uigetfile(FilterSpec, 'Select GFF file to open');
      if (gffName == 0)
         fid_out = -1;
         Header = [];
         gffName = [];
         gffPath = [];
         return
      else
         fid = fopen([gffPath gffName],'r','ieee-be');
      end
   end
```

```
% Check that it's a GSAT file:
StructID = char(fread(fid,[1,7],'char'));
if ~strcmp(StructID,'GSATIMG'),

   if UseWarnMsgs,
      errordlg('Selected file is not in GFF.','read_gff_header');
   end

   fclose(fid);
   fid_out = -3;
   Header = [];
   gffName = [];
   gffPath = [];
   return;
end

% Check for "endian"
fseek(fid,54,'bof');
Header.Endian = fread(fid,1,'ushort');  % 0 = little-endian

% Reopen the file with the proper endian interpretation:
if ( ~Header.Endian )
   fclose(fid);
   fid = fopen([gffPath gffName],'r','ieee-le');
   fprintf(['Opened ' strrep([gffPath gffName],'\','\\') ' as ieee-le\n']);
else
   fclose(fid);
   fid = fopen([gffPath gffName],'r','ieee-be');
   fprintf(['Opened ' strrep([gffPath gffName],'\','\\') ' as ieee-be\n']);
end


%=====================================================================
% The file is now open correctly.  Read the parameters:
%=====================================================================
```

```
% Check for "endian"
fseek(fid,8,'bof');
Header.Version_Minor     = fread(fid,1,'ushort');
Header.Version_Major     = fread(fid,1,'ushort');
Header.Length            = fread(fid,1,'ulong');
Header.Creator_Length    = fread(fid,1,'ushort');
Header.Creator           = char(fread(fid,[1,Header.Creator_Length],'char'));
                           fread(fid,24-Header.Creator_Length,'char');

if (ftell(fid) ~= 42)
   warndlg('file marker not at 42 to start date & time.');
end

Header.DateTime          = fread(fid,6,'ushort'); % yr, mo, da, hr, min, sec.

Header.Endian            = fread(fid,1,'ushort');  % 0 = little-endian
if (( Header.Version_Major == 1 ) & ( Header.Version_Minor > 7 ) | ...
      (Header.Version_Major > 1)),

      Header.BytesPerPixel     = fread(fid,1,'float32');
else
      Header.BytesPerPixel     = fread(fid,1,'uint32');
end



Header.FrameCnt            = fread(fid,1,'ulong');
Header.ImageType           = fread(fid,1,'ulong');
Header.RowMajor            = fread(fid,1,'ulong');
Header.RgCnt                 = fread(fid,1,'ulong');
Header.AzCnt                 = fread(fid,1,'ulong');
Header.ScaleExponent     = fread(fid,1,'long');
Header.ScaleMantissa     = fread(fid,1,'long');
Header.OffsetExponent    = fread(fid,1,'long');
Header.OffsetMantissa    = fread(fid,1,'long');
```

```
                          fread(fid,32,'uchar'); % Throw away "Res2" required filler

if (ftell(fid) ~= 128)
   warndlg('file marker not at 128 to start comment.');
end

Header.Comment_Length    = fread(fid,1,'ushort');
Header.Comment           = char(fread(fid,[1,Header.Comment_Length],'char'));
                          fread(fid,166-Header.Comment_Length,'char');

Header.ImagePlane        = fread(fid,1,'ulong');
Header.RgPixelSz         = fread(fid,1,'ulong') * 2^(-16);
Header.AzPixelSz         = fread(fid,1,'ulong') * 2^(-16);
Header.AzOverlap         = fread(fid,1,'long') * 2^(-16);

Header.SRPLat            = fread(fid,1,'long') * 2^(-23);
Header.SRPLong           = fread(fid,1,'long') * 2^(-23);
Header.SRPAlt            = fread(fid,1,'long') * 2^(-16);

Header.RFOA             = fread(fid,1,'long') * 2^(-23);

Header.XtoSRP            = fread(fid,1,'long') * 2^(-16);

                          fread(fid,32,'uchar'); % Throw away "Res3" required filler

if (ftell(fid) ~= 364)
   warndlg('file marker not at 364 to start phase history ID.');
end

Header.PhaseName_Length  = fread(fid,1,'ushort');
Header.PhaseName         = char(fread(fid,[1,Header.PhaseName_Length],'char'));
                          fread(fid,128-Header.PhaseName_Length,'char');
Header.ImageName_Length  = fread(fid,1,'ushort');
Header.ImageName         = char(fread(fid,[1,Header.ImageName_Length],'char'));
                          fread(fid,128-Header.ImageName_Length,'char');
```

```
Header.LookCnt           = fread(fid,1,'ulong');
Header.ParamRefAp        = fread(fid,1,'ulong');
Header.ParamRefPos       = fread(fid,1,'ulong');

Header.GrzAngle          = fread(fid,1,'ulong') * 2^(-23);
Header.Squint            = fread(fid,1,'long') * 2^(-23);
Header.GTA               = fread(fid,1,'long') * 2^(-23);
Header.RgBeamCtr         = fread(fid,1,'ulong') * 2^(-8);
Header.FlightTime        = fread(fid,1,'ulong') * 10^(-3);

if (( Header.Version_Major == 1 ) & ( Header.Version_Minor > 5 ) | ...
      (Header.Version_Major > 1)),
   Header.RgChirpRate       = fread(fid,1,'float');
else
   Header.RgChirpRate       = fread(fid,1,'long') * 2^(+12);
end

Header.XtoStart          = fread(fid,1,'long') * 2^(-16);

Header.MoCompMode        = fread(fid,1,'ulong');

Header.V_X               = fread(fid,1,'ulong') * 2^(-16);

Header.APCLat            = fread(fid,1,'long') * 2^(-23);
Header.APCLong           = fread(fid,1,'long') * 2^(-23);
Header.APCAlt            = fread(fid,1,'ulong') * 2^(-16);

Header.Calparm           = fread(fid,1,'ulong') * 2^(-24);

Header.LogicalBlkAdrr    = fread(fid,1,'ulong');

if (ftell(fid) ~= 692)
   warndlg('file marker not at 692 to start Azimuth Resolution.');
end

Header.AzResolution      = fread(fid,1,'ulong') * 2^(-16);
```

```
Header.RgResolution      = fread(fid,1,'ulong') * 2^(-16);

% Even though draft 1-6 says this is unsigned, it is actually SIGNED.
Header.DesSigmaN         = fread(fid,1,'long') * 2^(-23);

Header.DesGrazAngle      = fread(fid,1,'ulong') * 2^(-23);
Header.DesSquint         = fread(fid,1,'long') * 2^(-23);
Header.DesRng            = fread(fid,1,'ulong') * 2^(-8);

Header.SceneTrackAngle   = fread(fid,1,'long') * 2^(-23);

if (ftell(fid) ~= 720)
   warndlg('file marker not at 720 to start User Specified Data.');
end

Header.UserParam         = fread(fid,48,'uchar');  % Skip User Specified Data

if (ftell(fid) ~= 768)
   warndlg('file marker not at 768 to start CCD Parameters.');
end

Header.CoarseSNR         = fread(fid,1,'long');  % CCD Parameters
Header.CoarseAzSubSamp   = fread(fid,1,'long');
Header.CoarseRngSubSamp  = fread(fid,1,'long');
Header.MaxAzShift        = fread(fid,1,'long');
Header.MaxRngShift       = fread(fid,1,'long');
Header.CoarseDltAz       = fread(fid,1,'long');
Header.CoarseDltRng      = fread(fid,1,'long');
Header.TotProcs          = fread(fid,1,'long');
Header.TptBoxCMode       = fread(fid,1,'long');
Header.SNRThresh         = fread(fid,1,'long');
Header.RngSize           = fread(fid,1,'long');
Header.MapBoxSize        = fread(fid,1,'long');
Header.BoxSize           = fread(fid,1,'long');
Header.BoxSpc            = fread(fid,1,'long');
Header.TotTPts           = fread(fid,1,'long');
```

```
Header.GoodTPts          = fread(fid,1,'long');
Header.RndSeed           = fread(fid,1,'long');
Header.RngShift          = fread(fid,1,'long');
Header.AzShift           = fread(fid,1,'long');
Header.SumXRamp          = fread(fid,1,'long');
Header.SumYRamp          = fread(fid,1,'long');


if (ftell(fid) ~= 852)
   warndlg('file marker not at 852 to start General Fields for RTV.');
end

Header.Cy9kTapeBlock     = fread(fid,1,'ulong');

Header.NominalCenterFreq = fread(fid,1,'float');

Header.ImageFlags        = fread(fid,1,'ulong');

Header.LineNumber        = fread(fid,1,'ulong');
Header.PatchNumber       = fread(fid,1,'ulong');

Header.Lambda0           = fread(fid,1,'float');
Header.SRngPixSpace      = fread(fid,1,'float');
Header.DoppPixSpace      = fread(fid,1,'float');
Header.DoppOffset        = fread(fid,1,'float');
Header.DoppRngScale      = fread(fid,1,'float');

Header.MuxTimeDelay      = fread(fid,1,'float');

Header.APCXECEF          = fread(fid,1,'double');
Header.APCYECEF          = fread(fid,1,'double');
Header.APCZECEF          = fread(fid,1,'double');

Header.VxECEF            = fread(fid,1,'float');
Header.VyECEF            = fread(fid,1,'float');
Header.VzECEF            = fread(fid,1,'float');
```

```
if (ftell(fid) ~= 932)
   warndlg('file marker not at 932 to start Phase Calibration for IFSAR.');
end

Header.PhaseCal          = fread(fid,1,'float');

Header.SRPxECEF          = fread(fid,1,'double');
Header.SRPyECEF          = fread(fid,1,'double');
Header.SRPzECEF          = fread(fid,1,'double');

Header.Res5              = fread(fid,64,'uchar');  % filler Res5


%START REV 1.8 CHANGES

if (( Header.Version_Major == 1 ) & ( Header.Version_Minor > 7 ) | ...
      (Header.Version_Major > 1)),


Header.HeaderLen1        = fread(fid,1,'ulong');

Header.ImgDate.Year        = fread(fid,1,'ushort');
Header.ImgDate.Month       = fread(fid,1,'ushort');
Header.ImgDate.Day         = fread(fid,1,'ushort');
Header.ImgDate.Hour        = fread(fid,1,'ushort');
Header.ImgDate.Minute      = fread(fid,1,'ushort');
Header.ImgDate.Second      = fread(fid,1,'ushort');

Header.CompFileName      = fread(fid,128,'uchar');
Header.RefFileName       = fread(fid,128,'uchar');
Header.IEPlatform        = fread(fid,24,'uchar');
Header.IEProcID          = fread(fid,12,'uchar');
Header.IERadarModel      = fread(fid,12,'uchar');
Header.IERadarID         = fread(fid,1,'ulong');
```

```
Header.IESWID            = fread(fid,24,'uchar');
Header.IFPlatform        = fread(fid,24,'uchar');
Header.IFProcID          = fread(fid,12,'uchar');
Header.IFRadarModel      = fread(fid,12,'uchar');
Header.IFRadarID         = fread(fid,1,'ulong');
Header.IFSWID            = fread(fid,24,'uchar');
Header.IFAlgo            = fread(fid,8,'uchar');
Header.PHPlatform        = fread(fid,24,'uchar');
Header.PHProcID          = fread(fid,12,'uchar');
Header.PHRadarModel      = fread(fid,12,'uchar');
Header.PHRadarID         = fread(fid,1,'ulong');
Header.PHSWID            = fread(fid,24,'uchar');

Header.PHDataRcd         = fread(fid,1,'ulong');

Header.ProcProduct       = fread(fid,1,'ulong');
Header.MissionText       = fread(fid,8,'uchar');

Header.PHSource          = fread(fid,1,'ulong');

Header.GPSWeek           = fread(fid,1,'ulong');
Header.DataCollectReqH   = fread(fid,14,'uchar');
Header.Res6              = fread(fid,2,'uchar');
Header.GridName          = fread(fid,24,'uchar');

Header.PixValLinearity   = fread(fid,1,'ulong');

Header.ComplexOrReal     = fread(fid,1,'ulong');

Header.BitsPerMagnitude  = fread(fid,1,'ushort');
Header.BitsPerPhase      = fread(fid,1,'ushort');

Header.ComplexOrderType  = fread(fid,1,'ulong');

Header.PixDataType       = fread(fid,1,'ulong');
```

```
Header.ImageLength       = fread(fid,1,'ulong');

Header.ImageCmpScheme    = fread(fid,1,'ulong');

Header.APBO              = fread(fid,1,'float');
Header.AsaPitch          = fread(fid,1,'float');
Header.AsaSquint         = fread(fid,1,'float');
Header.DsaPitch          = fread(fid,1,'float');
Header.IRA               = fread(fid,1,'float');
Header.RxPolarization    = fread(fid,2,'float');
Header.TxPolarization    = fread(fid,2,'float');
Header.VxAvg             = fread(fid,1,'float');
Header.VyAvg             = fread(fid,1,'float');
Header.VzAvg             = fread(fid,1,'float');
Header.APCxAvg           = fread(fid,1,'float');
Header.APCyAvg           = fread(fid,1,'float');
Header.APCzAvg           = fread(fid,1,'float');
Header.AveragingTime     = fread(fid,1,'float');
Header.Dgta              = fread(fid,1,'float');
Header.VelocY            = fread(fid,1,'ulong')*2^(-16);
Header.VelocZ            = fread(fid,1,'ulong')*2^(-16);
Header.Ba                = fread(fid,1,'float');
Header.Be                = fread(fid,1,'float');

Header.AzGeomCorr        = fread(fid,1,'ulong');

Header.RngGeomCorr       = fread(fid,1,'ulong');

Header.AzWinFacBW        = fread(fid,1,'float');
Header.RngWinFacBW       = fread(fid,1,'float');
Header.AzWinID           = fread(fid,48,'uchar');
Header.RngWinID          = fread(fid,48,'uchar');
Header.KeepOutViolPrcnt  = fread(fid,1,'float');
Header.AzCoeff           = fread(fid,6,'float');
Header.PosUncertDown     = fread(fid,1,'float');
Header.PosUncertE        = fread(fid,1,'float');
Header.PosUncertN        = fread(fid,1,'float');
```

```
Header.NavAidingType     = fread(fid,1,'ulong');

Header.TwoDNLPhaseCoeffs = fread(fid,10,'float');
Header.ClutterSNRThresh  = fread(fid,1,'float');
Header.ElevationCoeff    = fread(fid,9,'float');
Header.MonopulseCoeff    = fread(fid,12,'float');
Header.TwistPntErrPrcnt  = fread(fid,1,'float');
Header.TiltPntErrPrcnt   = fread(fid,1,'float');
Header.AzPntErrPrcnt     = fread(fid,1,'float');
Header.SigmaN            = fread(fid,1,'ulong')*2^(-23);
Header.TakeNum           = fread(fid,1,'ulong');
Header.IFSARFlags        = fread(fid,5,'ulong');
Header.MuThreshold       = fread(fid,1,'float');

Header.GffAppType        = fread(fid,1,'ulong');

Header.Res7              = fread(fid,8,'uchar');
end

if (ftell(fid) ~= Header.Length)
    warndlg('File marker not at header length - moving to end.');
   % Position file at end of header:
    fseek(fid,Header.Length,'bof');
end

if (nargout > 3)
   fid_out = fid;
else
   fclose(fid);
end

return
```

3 Conclusions

The Matlab source code listings herein will read and display the Sandia developed GFF image file format.

Distribution

1	MS 1330	W. H. Hensley	5342
1	MS 1330	A. W. Doerry	5342
1	MS 1330	S. S. Kawka	5342
1	MS 1330	D. Harmony	5342
1	MS 1330	M. S. Murray	5342
1	MS 1330	B. G. Rush	5342
1	MS 1330	G. J. Sander	5342
1	MS 1330	B. L. Remund	5340
1	MS 1330	B. L. Burns	5340
1	MS 1330	K. W. Sorensen	5345
1	MS 1330	D. F. Dubbert	5345
1	MS 1330	G. R. Sloan	5345
1	MS 1330	S. M. Becker	5348
1	MS 1330	M. W. Holzrichter	5348
1	MS 1330	D. M. Small	5348
1	MS 1330	A. D. Sweet	5348
1	MS 1330	D. C. Sprauer	5348
1	MS 0519	L. M. Wells	5354
1	MS 1330	D. L. Bickel	5354
2	MS 9960	Central Technical Files	8945-1
2	MS 0899	Technical Library	4536