GUISDAP Documentation

M. S. Lehtinen* and A. Huuskonen**

* Sodankylä Geophysical Observatory, Sodankylä, Finland
** Finnish Meteorological Institute, Helsinki, Finland

The experiment initialization file

The GUISDAP experiment specification, explained in the document `init_EISCAT`, is a sufficient description of an incoherent scatter experiment to serve as a starting point for the data analysis. It is practical, however, to calculate the various ambiguity functions in advance and store those in a file, called the experiment initialization file, for later use in the analysis. This results in great reductions in the analysis startup times. The ambiguity functions thus calculated can also be used in the documentation of the experiment. The variables are listed below.

General parameters

`GUP_iniver`		Version number of GUISDAP
`nameexpr`		Name of the experiment

Parameters defined for each virtual channel, the experiment has 12 virtual channels

`vc_Aenv`	600 by 12	Autocorrelation function of the envelope of the transmitted waveform
`vc_Ap`	126 by 12	Autocorrelation function of the receiver impulse response
`vc_penv`	600 by 12	Effective pulse form
`vc_Apenv`	600 by 12	Autocorrelation function of the effective pulse form
`vc_penvabs`	600 by 12	absolute value version of `vc_penv` for support calculations
`vc_penvo`	1 by 12	Range to the leading edge of the range ambiguity function

Parameters specified for each lag profile group, the present experiment has 120 lag profiles groups

`lpg_ND`	1 by 120	Number of crossed products
`lpg_T`	1 by 120	Common value of `lp_T(lpg)`
`lpg_bac`	1 by 120	Background lag profile group for lpg
`lpg_cal`	1 by 120	Calibration lag profile group for lpg
`lpg_bcs`	1 by 120	Type of profile, b=background, c=calibration, s=signal, o=offset, x=multipulse zero lag
`lpg_code`	1 by 120	Label to group lag profiles, e.g. in CP1 is 1 for 14μs power profiles, 2 for multipulse profiles etc.
`lpg_dt`	1 by 120	Range increment [`p_dtau`]
`lpg_h`	1 by 120	Range to the center point of the range ambiguity function of the first product
`lpg_lag`	1 by 120	Lag value, difference of `lp_t2` and `lp_t1`
`lpg_lpdata`	1 by 318	Lag profile numbers ordered by lag profile groups
`lpg_lpend`	1 by 120	Start address for lag profile numbers in `lpg_lpdata`
`lpg_lpstart`	1 by 120	Last address for lag profile numbers in `lpg_lpdata`
`lpg_nt`	1 by 120	Number of points in the profile
`lpg_ra`	1 by 120	Result memory address of the first product
`lpg_ri`	1 by 120	Result memory address increment
`lpg_w`	1 by 120	Width of the range ambiguity function
`lpg_wom`	120by 121	Reduced spectral ambiguity function

Scales and other important parameters

`p_N0`	1 by 1	Electron density scale [m^{-3}]
`p_R0`	1 by 1	Range scale in units of `p_dtau`
`p_T0`	1 by 1	Ion temperature [K]

p_m0	1 by 2	Ion masses [u]
p_D0	1 by 1	Debye term based on scale variables
p_om	121 by 1	Frequency axis in units of p_om0
p_om0	1 by 1	Frequency scale
p_XMITloc	1 by 3	transmitter location [latitude (deg N), longitude (deg E), altitude (km)]
p_RECloc	1 by 3	receiver location [latitude (deg N), longitude (deg E), altitude (km)]

Producing initialization file from general GUP variables

The initialization file is produced by the program `init_GUP`. The following explains the program execution (numbers refer to lines):

- load the GUISDAP variables from file
- `findlpg` groups lag profiles to lag profile groups. A lag profile group contains all the lag profiles which are summed to same memory locations.
- `find_vcgroups` checks which virtual channels have identical transmission and receiver impulse response. This makes many subsequent operations faster
- `find_calibrations` finds the calibration and background lag profile groups for each group.
- `ambcalc` calculates the effective pulse forms (convolution of the transmitter envelope with the receiver impulse response) and the autocorrelation functions of the transmitter envelopes and receiver impulse responses for each virtual channel (`vc_penv`, `vc_Aenv` and `vc_Ap`)
- `lpgwrcalc` makes the range ambiguity functions for the signal lag profile groups and calculates the distance to the first gate (`lpg_h`) and the width of the range ambiguity function (`lpg_w`). It also makes a Postscript file for plotting of the range ambiguity functions.
- `lpgwomcalc` calculates the spectral ambiguty functions for all signal lag profile groups (`lpg_wom`).
- `lpg_tex` makes a TEX–file containing lag profile group parameters
- save results to a file.

/geo/gmt/askoh/guisdap/m152/init_GUP.m

```
1  % GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
2  %
3  % Main program to find lag profile groups and calculate ambiguity functions
4  % Script and functions called:
5  % load_GUPvar : load GUISDAP experiment definition variables from a file
6  % read_specpar: defines the scale parameters and spectral grid
7  % findlpg     : lag profile groups are formed here
8  % find_vcgroups : similar virtual channels are grouped together
9  % find_calibrations : associates lpg's with calibration and background lpg
10 % ambcalc     : vc_Aenv vc_Ap vc_Apenv matrices formed
11 % lpgwrcalc   : calculation of the range ambiguity functions
12 % lpgwomcalc  : calculation of the reduced spectral ambiguity functions
13 % save_toinitfile : saves the variables to a file
14 %
15 % See also: load_GUPvar read_specpar nat_const constants lpg_tex save_toinitfile
16 %
17 % See also: findlpg find_vcgroups find_calibrations ambcalc lpgwrcalc lpgwomcalc
18
19 clg, hold off
20 glob_GUP
21
22 % Chdir to the experiment directory, store the present path as a return address
23 original_path=pwd;
24 fprintf('\n\nChdir to the experiment directory\n')
25 cdir(canon(path_expr))
26
27 nat_const
28
29 if exist('N_rcprog')~=1, N_rcprog=1; end
30 for d_rcprog=1:N_rcprog
31   Stime=clock;
32
```

```
  if N_rcprog>1, apustr=['_',int2str(d_rcprog)]; else apustr=[]; end
  load_GUPvar

  read_specpar
  constants

  findlpg
  find_vcgroups
  find_calibrations
  ambcalc,

  if all(p_XMITloc==p_RECloc) % Monostatic case
    lpgwrcalc,
    lpgwomcalc,
  else
  % Assume that the scattering volume is always completely filled
    disp([' Bistatic measurement:'])
    vc_Aenv=ones(600,length(vc_ch)); % For unit length
    fir=lp_fir; lp_fir=abs(lp_fir);
    lpgwomcalc,
    lp_fir=fir;vc_Aenv=zeros(size(vc_env));
  end

  plot(p_om,real(lpg_wom)), drawnow
  lpg_tex
  fprintf('  Time used in initialisation:%8.2f min\n',etime(clock,Stime)/60)
  save_toinitfile

  fprintf('Radar controller program %g ready\n',d_rcprog)
end
fprintf(['\nChdir back to the original directory ', original_path,'\n'])
cdir(original_path)
fprintf('\n******************************************************\n')
fprintf('*\n*\n* Execute plot_wr to see the range ambiguity functions\n')
fprintf('*\n*\n******************************************************\n')
clear original_path Stime d_rcprog
```

Routines called by the initialization

This is a function that returns the ACF of the transmission waveform on basis of the variable **vc_Aenv**. We need this, because we do not want to think about index calculations each time we need the variable for some time interval.

/geo/gmt/askoh/guisdap/m152/Aenv.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% value of ACF of transmitter envelope for virtual channel 'vch' and time lag 't'
% This if the preferred way of referencing matrix vc_Aenv, which contains the function
% for only non-negative values of 't'. Also possible references beyond the matrix
% index limits are hanled. The function also takes care of the fact that
% the value of ACF at lag 0 is stored at Matlab matrix index 1.
% Parameters
% vch : virtual channel numbers
% t: lag values, any value permitted
%
% See also: Ap Apenv
%
% function res=Aenv(vch,t);
function res=Aenv(vch,t);

global vc_Aenv

t=abs(t(:))+1;
[len,hups]=size(vc_Aenv);
ii=find(t>len);
t(ii)=len*ones(length(ii),1);
res=vc_Aenv(t,vch);
```

The following routine calculates many ambiguity functions and related functions that are useful to have in the workspace.

/geo/gmt/askoh/guisdap/m152/ambcalc.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
Nvc=length(vc_ch);
lenenv=length(vc_env(:,1));
lenp=length(vc_p(:,1));
vc_Aenv=zeros(lenenv,Nvc);
vc_penv=zeros(lenenv+lenp-1,Nvc);
vc_Apenv=zeros(lenenv+lenp-1,Nvc);
```

```
vc_penvabs=zeros(lenenv+lenp-1,Nvc);
vc_Ap=zeros(lenp,Nvc);

fprintf('#\n#\n# Ambcalc: \n')
fprintf('# calculating ACF''s of transmission envelopes\n')
fprintf('# calculating ACF''s of filter impulse responses\n')
fprintf('# calculating effective pulseforms etc\n#\n')

for group=diff_val(vc_group)
  vcs=find(vc_group==group);
  vc=vcs(1);

  fprintf('#Virtual channel group %.0f, formed by channels:', group);
  fprintf(' %.0f',vcs); fprintf('\n')

  ind=1:max(find(vc_env(:,vc)~=0));env=vc_env(ind,vc);
  ind=1:max(find(vc_p(:,vc)~=0));  imp=vc_p(ind,vc);
  Aenv=xcorr(env);
  len=ceil(length(Aenv)/2);
  vc_Aenv(1:len,vcs)=Aenv(len:2*len-1)*ones(size(vcs));
  plot((-(len-1):len-1)*p_dtau,Aenv); title(' ACF of X-mission envelope')
  drawnow

  Ap=xcorr(imp);
  len=ceil(length(Ap)/2);
  vc_Ap(1:len,vcs)=Ap(len:2*len-1)*ones(size(vcs));
%  plot((-(len-1):len-1)*p_dtau,Ap), title(' ACF of filter impulse response')
  drawnow

  penv=conv(imp,env);
  Apenv=xcorr(penv);
  len=length(penv);
  vc_penv(1:len,vcs)=penv*ones(size(vcs));
  vc_Apenv(1:len,vcs)=Apenv(len:2*len-1)*ones(size(vcs));
%  plot((0:len-1)*p_dtau,penv); title(' effective pulse form')
  drawnow

% calculate the same using absolute values of env and p; used later
% for ambiguity function support calculations
  penv=conv(abs(imp),abs(env));
  len=length(penv);
  vc_penvabs(1:len,vcs)=penv*ones(size(vcs));
  vc_penvo(vcs)=vc_envo(vcs)+1;

end
fprintf('#\n#pulseforms calculated\n#\n#\n'),
clear Nvc vc len Apenv Aenv Ap penv env imp ind lenenv lenp group
```

Ap and APenv are the same as Aenv, but for receiver impulse response and the effective pulse form.

/geo/gmt/askoh/guisdap/m152/Ap.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% value of ACF of receiver impulse response for virtual channels 'vch' and time lags 't'
% This if the preferred way of referencing matrix vc_Ap, which contains the function
% for only non-negative values of 't'. Also possible references beyond the matrix
% index limits are hanled. The function also takes care of the fact that
% the value of ACF at lag 0 is stored at Matlab matrix index 1.
% Parameters
% vch : virtual channel numbers
% t: lag values, any value permitted
%
% See also: Aenv Apenv
%
% function res=Ap(vch,t);
function res=Ap(vch,t);

global vc_Ap

t=abs(t(:))+1;
[len,hups]=size(vc_Ap);
ii=find(t>len);
t(ii)=len*ones(length(ii),1);
res=vc_Ap(t,vch);
```

The following gives the transmission envelope for any time interval.

/geo/gmt/askoh/guisdap/m152/env.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% value of pulseform for virtual channel 'vc' and time instant 't'
% This if the preferred way of referencing matrix vc_env, which contains the
% functions at offsetted time values.
% Parameters
```

```
% vc : virtual channel numbers
% t: time instants, any value permitted
%
% See also: penv
% function res=env(vc,t);
function res=env(vc,t);

global vc_env vc_envo

[len,hups]=size(vc_env);
t=t-vc_envo(vc);
iin=find(t>=1 & t<=len);
if length(iin)>0,
  res=zeros(size(t));
  res(iin)=vc_env(t(iin),vc);
else
  res=[];
end
```

In `find_calibrations`, we try to assign a background and calibration lag profile group to each group. We require that a calibration group is found for every group, but not the existence of a background group. The most evident examples of the latter case are the signal lag profile groups obtained from alternating codes.

/geo/gmt/askoh/guisdap/m152/find_calibrations.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Script to associate each lag profile groups with its background and
% calibration groups. Guisdap required that a calibration group is found
% It is not necessary to find a background groups exists, if the lag value
% is non-zero
%
% See also: init_GUP
%
% find_calibrations.m

% The program logic is based on treating one lpg_code value at a time
fprintf('\nLooking for calibration and background ....\n\n')
codes=sort(lpg_code);
codes(find(diff(codes)==0))=[]; % find all different values

% lpg_bac stores the background lag profile group number  (0 means no background)
% lpg_cal stores the calibration lag profile group number
lpg_bac=zeros(size(lpg_ra));
lpg_cal=zeros(size(lpg_ra));
for code=codes;

  % First we look for background and calibration measurements with the same code number
  sig_gr=find(lpg_code==code & lpg_bcs=='s');
  xxx_gr=find(lpg_code==code & lpg_bcs=='x');
  bac_gr=find(lpg_code==code & lpg_bcs=='b');
  cal_gr=find(lpg_code==code & lpg_bcs=='c');
  off_gr=find(lpg_code==code & lpg_bcs=='o');
  all_gr=[bac_gr, cal_gr, sig_gr, xxx_gr, off_gr];
  % If no calibration data for a code, look for other codes, which
  % might be on the same real channel. If any is found, use their calibrations.
  if length(cal_gr)==0,
    ch=diff_val(vc_ch(lp_vc(lpg_lp(sig_gr(1))))); % These are the real channels used
    for other=codes(find(codes~=code))
      lpg_other=find(lpg_code==other);
      ch_other=diff_val(vc_ch(lp_vc(lpg_lp(lpg_other(1))))); % These are channels used
      if length(ch)==length(ch_other) % Make sure that same channels are used
        if all(ch==ch_other) % Continue testing
          bac_gr=find(lpg_code==other & lpg_bcs=='b');
          cal_gr=find(lpg_code==other & lpg_bcs=='c');
          if length(cal_gr)>0, % Be happy, if calibration group was found
            fprintf(' Using calibrations of group %g for group %g\n',other,code)
            break,
          end
        end
      end
    end
  end

    %******* find background LPG for all LPG:s ********************
    for lpg=all_gr,
      % bac contains all possible background measurements for the given lpg
      bac=find(abs(lpg_lag(bac_gr)-lpg_lag(lpg))<=1000*eps);
      lenbac=length(bac);
      % Study first the case when at least one background measurement was found
      if lenbac>=1,
        if lenbac>1
          fprintf('\n More than one background lag profile group found for lpg %3.0f\n',lpg)
          fprintf(' Background groups are:'),fprintf(' %.0f', bac_gr(bac))
```

```
          fprintf('\n Sorry, but only one can be used at present, taking the first one\n\n')
          bac=bac(1);
        end
        lpg_bac(lpg)=bac_gr(bac);
      % If background not found, use offset for non-zero lags, if it exists
      % This background system is used in many uniprog-type experiments
      % UHF1 and UHF2, ELSA-T4, UP3A (=GEN6B)
      % UP3A has two offset profiles, use the latter which is completely
      % free of signal contributions.
      elseif lenbac==0 & lpg_lag(lpg)>0 & length(off_gr)>0;
        len=length(off_gr);
        lpg_bac(lpg)=off_gr(len);
      % ALTERNATING code (and multipulse) experiments do not need background for non-zero lags
      elseif lenbac==0 & lpg_lag(lpg)>0
         fprintf(' No background for lpg %3.0f, lag value %3.0f us\n',lpg,lpg_lag(lpg)*p_dtau)
      elseif lenbac==0 & lpg_lag(lpg)==0  % Something in error in the background measurement
         fprintf('\nERROR: No background measurement found for lpg %3.0f\n',lpg)
         fprintf(' The lag value of this lpg is zero and\n therefore data will be regarded as garbage
\n\n')
         if lpg_bcs(lpg)=='c'; % Remove from the calibration group
           ind=find(cal_gr==lpg);cal_gr(ind)=[];
         end
         lpg_bcs(lpg)='g';
      end
    end

  %******* find the LPG that gives the calibration power ***********
  cal=find(lpg_lag(cal_gr)==0);
  if length(cal)==0,
    fprintf('\nERROR: No calibration found for code %3.0f, formed by lag profile groups\n',code)
    fprintf(' %.0f', sort(all_gr))
    fprintf('\n Treating these lag profile groups as garbage\n\n')
    lpg_bcs(all_gr)='g'*ones(1,length(all_gr));
  else,
    if length(cal)>1
      fprintf('\n More than one calibration lag profile group found for code %3.0f\n',code)
      fprintf(' Calibration groups are:'),fprintf(' %.0f', cal_gr(cal))
      fprintf('\n Sorry, but only one can be used at present, taking the first one\n\n')
      cal=cal(1);
    end
    % Store the found calibration lpg to all lag profile groups with this code number
    lpg_cal(all_gr)=cal_gr(cal)*ones(1,length(all_gr));
  end

end

fprintf('\n...      Calibration and background lpg''s located\n\n')

clear all_gr sig_gr xxx_gr bac_gr cal_gr off_gr code codes cal bac lenbac len lpg
clear ch lpg_other ch_other other_codes other
```

The following script finds lag profile group parameters. We suppose here that if two lag profiles have the same `lp_ra`, they belong to the same lag profile group (that is, the parameters `lp_nt` and `lp_ri` are also equal and no other lag profiles overlap those with a single fixed `lp_ra`). This simplified idea does not necessarily work for all experiments, and this should be generalized.

/geo/gmt/askoh/guisdap/m152/findlpg.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Produce the lag profile groups.
% We assume that two lag profiles belong to the same lag profile group
% if the start addresses lp_ra agree. The program checks that various other
% parameters are equal in the lag profiles. A hidden assumption is that
% addresses attached to a lag profile group do not belong to any other group.
%
% See also: init_GUP
%
fprintf(['\n\nProducing the lag profile groups: ...\n\n'])
lpg_ra=diff_val(lp_ra); % find all different values
len=length(lpg_ra);
lpg_lag=zeros(1,len);lpg_dt=zeros(1,len);lpg_ND=zeros(1,len);lpg_T=zeros(1,len);
lpg_ri=zeros(1,len); lpg_nt=zeros(1,len);lpg_h=zeros(1,len);lpg_w=zeros(1,len);
lpg_bcs=zeros(1,len);lpg_code=zeros(1,len);
lpg_lpdata=zeros(1,len); lpg_lpind=0;
lpg_lpstart=zeros(1,len); lpg_lpend=zeros(1,len);
ad_lpg=[];
for ind=1:length(lpg_ra),
  lpg=find(lp_ra==lpg_ra(ind));
  lenlpg=length(lpg);
  lpg_lpdata(lpg_lpind+(1:lenlpg))=lpg;
  lpg_lpstart(ind)=lpg_lpind+1; lpg_lpend(ind)=lpg_lpind+lenlpg;
  lpg_lpind=lpg_lpind+lenlpg;
  lpg_lag(ind)=cheq(lp_t2(lpg)-lp_t1(lpg));
```

```
  lpg_dt(ind)=cheq(lp_dt(lpg).*lp_dec(lpg));
  lpg_ND(ind)=sum(sum(abs(lp_fir(:,lpg))));
  lpg_T(ind)=cheq(lp_T(lpg));
  lpg_ri(ind)=cheq(lp_ri(lpg));
  lpg_nt(ind)=cheq(lp_nt(lpg));
  lpg_h(ind)=mean(lp_h(lpg));
% note that this range value will be updated for signal lag profiles
% after the range ambiguity functions are calculated
  lpg_bcs(ind)=cheq(lp_bcs(lpg));
  lpg_code(ind)=cheq(lp_code(lpg));

  fprintf(['lpg=%3.0f code=%1.0f type=',setstr(lpg_bcs(ind))],ind,lpg_code(ind));
  fprintf(' lag=%3.0f dt=%3.0f', p_dtau*lpg_lag(ind), p_dtau*lpg_dt(ind));
  fprintf(' ND=%2.0f h=%5.0f',  lpg_ND(ind), p_dtau*lpg_h(ind));
  fprintf(' T=%3.0f nt=%3.0f',  lpg_T(ind),   lpg_nt(ind));
  fprintf(' ra=%3.0f ri=%2.0f',lpg_ra(ind), lpg_ri(ind));
  fprintf('\n');

  addr=lpg_addr(ind);sto=addr+1;
  if max(sto)>length(ad_lpg); ad_lpg(max(sto))=0; end
  ind=find(ad_lpg(sto)~=0);
  if length(ind)>0,
     fprintf('\n\n Conflict in the lag profile definition\n')
     fprintf(' Lag profile group %.0f\n tries to define addresses\n',ind)
     fprintf(' %5.0f',addr(ind))
     fprintf('\n which already belong to lag profile groups\n')
     fprintf(' %5.0f',ad_lpg(sto(ind)))
     fprintf('\n')
     error(' ')
  else
     ad_lpg(sto)=ind*ones(size(addr));
  end
end;
clear lpg ind len lenlpg lpg_lpind addr sto ad_lpg
```

For a specified memory location, the following routine returns all lag profiles that are summed to that location. The routine also returns corresponding product sample times and the virtual channel.

/geo/gmt/askoh/guisdap/m152/findrg.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% findrg.m
%
% Function to find all lag profiles contributing to a result
% memory location m
%
% [lp,t1,t2,vc]=findrg(m)
%
  function [lp,t1,t2,vc]=findrg(m)
%

  global lp_ra lp_nt lp_ri lp_t1 lp_dt lp_t2 lp_vc

  lp=find( lp_ra<=m & m<=lp_ra+((lp_nt-1).*lp_ri) ...
             & round((m-lp_ra)./lp_ri)==(m-lp_ra)./lp_ri );
  if nargout>1
    t1=lp_t1(lp)+lp_dt(lp).*(m-lp_ra(lp))./lp_ri(lp);
    t2=t1+lp_t2(lp)-lp_t1(lp);
    vc=lp_vc(lp);
  end
```

Many of the virtual channels often have similar transmitter envelopes and similar receiver impulse responses. The immediate consequence is that the `vc_Aenv`, `vc_Ap`, `vc_penv` and `vc_Apenv` functions are also similar. Also, calculation of the range ambiguity functions and spectral ambiguity functions is faster, if the similarity of the virtual channels is taken into account. In the next routine, we group the virtual channels together so that the subsequent calculations would be faster. It would be possible to reduce the size of the initialization file by storing certain functions only once for each group. This has not yet been implemented.

/geo/gmt/askoh/guisdap/m152/find_vcgroups.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Several virtual channels often have similar transmission envelopes and receiver
% impulse responses. It will make the program execution faster, if ambiguity
% function calculations are done only once for all these channels. This function
% checks the envelopes and impulse responses and form the virtual channel groups
%
% See also: init_GUP
%
fprintf('\n\nGrouping the virtual channels in virtual channel groups.\n')
vc_group=zeros(size(vc_ch));

```

```
group=1;
vcs=find(vc_ch~=0); % These channels are in use
while length(vcs)>0,
  vc=vcs(1);
  index=find(max(abs(vc_env(:,vcs)-vc_env(:,vc)*ones(1,length(vcs))))<100*eps ...
         & max(abs(vc_p(:,vcs)-vc_p(:,vc)*ones(1,length(vcs))))<100*eps);
  vc_group(vcs(index))=group*ones(size(index));
  fprintf(' Group %.0f contains virtual channels ',group);
  fprintf(' %.0f',vcs(index)); fprintf('\n')
  group=group+1;
  vcs(index)=[];
end
```

All necessary globals definitions:

/geo/gmt/askoh/guisdap/m152/glob_GUP.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% All the global variables needed to run init_GUP
%
% See also: globals, glob_EISCAT, start_GUP, init_GUP
%
global ch_adcint ch_filter ch_fradar ch_gain p_XMITloc p_RECloc

global lp_t1 lp_t2 lp_dt lp_nt lp_vc lp_ra lp_ri
global lp_T lp_code lp_bcs lp_h  lp_nfir lp_fir lp_dec
global lpg_lag lpg_dt lpg_nt lpg_ra lpg_ri lpg_T lpg_code lpg_bac lpg_cal
global lpg_bcs lpg_h lpg_w lpg_ND lpg_wom lpg_wr lpg_lpdata lpg_lpstart lpg_lpend

global k_radar0 p_om p_dtau p_T0 p_N0  p_D0 p_m0 p_om0 p_R0 p_rep

global vc_ch vc_p vc_env vc_envo vc_Aenv vc_Ap vc_penv vc_Apenv vc_penvabs vc_penvo vc_group
global vc_sampling %AH 940802
```

Loading of the GUISDAP variables is done by

/geo/gmt/askoh/guisdap/m152/load_GUPvar.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% A script to load GUPvariables into the workspace
% The script assumed that variables name_expr and name_site exist in the workspace
% The script contains reference to EISCAT remote sites. However, the script works
% without modifications for other radars as long as name_site is different from K and S
%
% See also: path_expr save_toinitfile
%

temp=[path_expr name_expr, name_site];
if name_site=='K' | name_site=='S';
  temp=[path_expr name_expr 'R'];
end
if exist('d_rcprog')==1, rcp=d_rcprog; else rcp=0; end
if exist(canon([temp '_' int2str(rcp) 'GUPvar.mat'],0))==2,
  eval(canon(['load ' temp '_' int2str(rcp) 'GUPvar']))
elseif exist(canon([temp 'GUPvar.mat'],0))==2,
  eval(canon(['load ' temp 'GUPvar']))
else
  fprintf(['\n\n\n GUP variable file    ', canon([temp 'GUPvar.mat'],0),'    not found \n\n\n'])
  error(' ')
end
if GUP_iniver<1.52,
  fprintf('*\n*\n* Files produced by GUP version %.2f not usable\n', GUP_iniver)
  fprintf('* Please, reinitialize the experiment with GUP 1.52 or later\n*\n*\n')
  error('')
end
if exist('lp_firsto')
  lp_fir=cumsum(full(lp_firsto));
end
clear rcp
```

The following routine gives all lag profile numbers that belong in a lag profile group.

/geo/gmt/askoh/guisdap/m152/lpg_lp.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% lpg_lp.m
% Gives the lag profiles that belong to a lag profile group
% function lp=lpg_lp(lpg)

function lp=lpg_lp(lpg)

global lpg_lpdata lpg_lpstart lpg_lpend
lp=lpg_lpdata(lpg_lpstart(lpg):lpg_lpend(lpg));
```

The following routine creates a `.tex`-file that can be used in documentation of lag profile groups.

/geo/gmt/askoh/guisdap/m152/lpg_tex.m
```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Script to write TeX data about lpg_ parameters
% suitable to be printed with tex code in file lpg.tex
%
% See also: init_GUP
fil=canon([path_expr name_expr name_site apustr 'lpg_i.tex']);
if exist(fil)==2, delete(fil), end

for ind=1:length(lpg_ND)
  fprintf(fil,'\\+%3.0f&',ind);
  fprintf(fil,'%2.0f&',lpg_code(ind));
  fprintf(fil,['  ',setstr(lpg_bcs(ind)),'& ']);
  fprintf(fil,'%3.0f&',lpg_lag(ind)*p_dtau);
  fprintf(fil,'%6.0f&',lpg_h(ind)*p_dtau);
  fprintf(fil,'%6.0f&',lpg_w(ind)*p_dtau);
  fprintf(fil,'%3.0f&',lpg_dt(ind)*p_dtau);
  fprintf(fil,'%3.0f&',lpg_ND(ind));
  fprintf(fil,'%3.0f&',lpg_T(ind));
  fprintf(fil,'%3.0f&',lpg_nt(ind));
  fprintf(fil,'%4.0f&',lpg_ra(ind));
  fprintf(fil,'%4.0f\\cr\n',lpg_ri(ind));
end;
closefile
clear fil ind
```

Calculation of spectral ambiguity function for a lag profile group:

/geo/gmt/askoh/guisdap/m152/lpgwom.m
```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% function that calculates the reduced spectral ambiguity function
% lpg     : lag profile group number
% womsum  : reduced spectral ambiguity function
%
% See also: lpgwomcalc wl
%
% function womsum=lpgwom(lpg);
function womsum=lpgwom(lpg);

global p_om  p_dtau p_om0 lp_t1 lp_t2 lp_vc lp_nfir lp_fir vc_group
womsum=zeros(size(p_om))';
dt=-5000;wc=-1;
for lp=lpg_lp(lpg),
 % disp(lp),
  used_oldvalues=0;
  t2=lp_t2(lp);t1=lp_t1(lp);vc=lp_vc(lp);
  if (dt==t2-t1 & vc==wc)
    womsum=womsum+sum(lp_fir(:,lp))*wold; used_oldvalues=1;
  elseif (dt==t2-t1 & wc>0 & vc~=wc)
    if vc_group(vc)==vc_group(wc),
      womsum=womsum+sum(lp_fir(:,lp))*wold; used_oldvalues=1;
    end
  end
  if used_oldvalues==0,
    dt=t2-t1;
    [w,wx]=wl(vc,dt);
    ind=find(w~=0); w=w(ind); wx=wx(ind);
    ch=1;  % hyi hyi
    wnew=sum( (w*ones(size(p_om))').*exp( wx*p_om'*(p_dtau*1e-6*p_om0(ch)*sqrt(-1)) ) );
    womsum=womsum+sum(lp_fir(:,lp))*wnew;
    wc=vc;wold=wnew;
  end
end
```

Calculation of spectral ambiguity functions for all signal lag profile groups in the workspace.

/geo/gmt/askoh/guisdap/m152/lpgwomcalc.m
```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% script to calculate the spectral ambiguity functions for all signal lpg's
%
% See also: lpgwom
lpg_wom=zeros(length(lpg_bcs),length(p_om));
fprintf('\n*\n* Calculating spectral ambiguity functions for signal lpg:s\n*\n*');

for lpg=find(lpg_bcs=='s');
  fprintf(' %.0f',lpg),
  lpg_wom(lpg,:)=lpgwom(lpg);
end
fprintf('\n*\n* spectral ambiguity functions calculated\n*\n')
```

```
clear lpg
```

The following gives the amplitude range ambiguity for any time interval.

/geo/gmt/askoh/guisdap/m152/penv.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% value of effective pulseform for virtual channel 'vc' and time instant 't'
% This if the preferred way of referencing matrix vc_penv, which contains the
% functions at offsetted time values.
% Parameters
% vc : virtual channel numbers
% t: time instants, any value permitted
%
% See also: env
% function res=penv(vc,t);
function res=penv(vc,t);

global vc_penv vc_penvo

[len,hups]=size(vc_penv);
t=t-vc_penvo(vc);
iin=find(t>=1 & t<=len);
if length(iin)>0,
  res=zeros(size(t));
  res(iin)=vc_penv(t(iin),vc);
else
  res=[];
end
```

The following routine saves initialization results in the `init.m`-file.

/geo/gmt/askoh/guisdap/m152/save_toinitfile.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% Script to save ambiguity functions and other variables to a file
% These variables are needei in the  data analysis
%
% See also: load_initfile save_GUPvar load_GUPvar path_expr

GUP_iniver=GUP_ver;
nameexpr=[name_expr name_site];

% We produce groupwise variables to save disk space
[a,ind]=wind(diff_val(vc_group),vc_group);
vcg_Aenv=vc_Aenv(:,ind);
vcg_Ap=vc_Ap(:,ind);
vcg_Apenv=vc_Apenv(:,ind);
vcg_penv=vc_penv(:,ind);
vcg_penvabs=vc_penvabs(:,ind);

str='GUP_iniver ch_fradar ch_gain lp_vc lpg_ND lpg_T lpg_bcs lpg_code lpg_lpstart lpg_lpend';
str=[str ' lpg_lpdata lpg_dt lpg_h lpg_lag lpg_nt lpg_ra lpg_ri lpg_w lpg_wom lpg_bac lpg_cal'];
str=[str ' nameexpr p_XMITloc p_RECloc p_D0 p_N0 p_R0 p_T0 p_dtau p_m0 p_om p_om0'];
%str=[str ' vc_penv vc_penvabs vc_penvo']; % these saved once for each vc_group
%str=[str ' vc_ch vc_Aenv vc_Ap vc_Apenv vc_group']; % these saved once for each vc_group
str=[str ' vcg_penv vcg_penvabs vc_penvo'];
str=[str ' vc_ch vcg_Aenv vcg_Ap vcg_Apenv vc_group'];
if ~exist('apustr'), apustr=''; end
eval([canon(['save ' path_expr name_expr name_site apustr 'init.mat ']), str]);

clear GUP_iniver nameexpr a ind vcg_Aenv vcg_Ap vcg_Apenv vcg_penv vcg_penvabs
```

The following gives the lag ambiguity function and its support for a given lag and virtual channel.

/geo/gmt/askoh/guisdap/m152/wl.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% short form of the reduced lag ambiguity function
% vc:   virtual channel number
% lag: lag value
% w : lag ambiguity function
% l : lag ambiguity function support
%     plot(l*p_dtau,w) shows the function with correct lag values in us
%
% See also: lpgwom wr Ap Aenv
%
%  function [w,l]=wl(vc,lag)
  function [w,l]=wl(vc,lag)

global vc_Ap

len=length(vc_Ap(:,vc));
```

```
l=((lag-len+1):(lag+len-1))';
w=Ap(vc,l-lag).*Aenv(vc,l);
```

Range ambiguity function and its support for a given virtual channel and sample time pair:

/geo/gmt/askoh/guisdap/m152/wr.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% function to calculate range ambiguity function
% Parameters:
% vch:    virtual channel number
% t1,t2 : sample times of first and second factor in the product
% dumdum: with four arguments calculates support for two-dimensional ambiguity
%         functions. There are cases where reduced ambiguity function is null
%         but the two-dimensional is not.
%
% See also: wrlpg
%
% function [wwr,r]=wr(vch,t1,t2,dumdum);
function [wwr,r]=wr(vch,t1,t2,dumdum);

global vc_penvabs vc_penv vc_penvo

rt=wnz(vc_penvabs,vch);
rt=[min(rt)-1;rt;max(rt)+1];
len=length(rt); lenn=len-(t2-t1);
st=rt(1)-1;  % Origin for the part of penvabs used
wwr=vc_penv(st+(1:lenn),vch).*vc_penv(st+(1+t2-t1:len),vch);
wwr=flipud(wwr);
r=t1-vc_penvo(vch)-st+(-lenn+1:0);
if nargin>3, % calculate support for two-dim. ambiguities
  wwra=vc_penvabs(st+(1:lenn),vch).*vc_penvabs(st+(1+t2-t1:len),vch);
  wwra=flipud(wwra);
  suppind=find(wwra~=0);
  wwr=wwr(suppind); r=r(suppind);
end;
```

Range ambiguity function for a given lag profile group. No support is returned, because values at all heights starting from `1*p_dtau` are given.

/geo/gmt/askoh/guisdap/m152/wrlpg.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% function to calculate the range ambiguity function for a lag profile group
%
% See also: lpgwrcalc wr
%
%  function wsum=wrlpg(lpg)
  function wsum=wrlpg(lpg)

global lp_t1 lp_t2 lp_dt lp_vc lp_nfir lp_fir

wsum=0;
for lp=lpg_lp(lpg)
  [w,r]=wr(lp_vc(lp),lp_t1(lp),lp_t2(lp));
  if length(r)>0,
    maxr=r(length(r))+(lp_nfir(lp)-1)*lp_dt(lp);
    if length(wsum)<maxr,wsum(maxr,1)=0;end;
      for ind=1:lp_nfir(lp)
        R=r+(ind-1)*lp_dt(lp);
        wsum(fix(R))=wsum(fix(R))+lp_fir(ind,lp)*w;
      end
  else
    fprintf('For lag profile %.0f the range ambiguity function is empty\n',lp)
  end
end
```

The following routine calculates an estimate for range to the first gates in the lag profile group (`lpg_h`) and an estimate of the width of the ambiguity function (`lpg_w`).

/geo/gmt/askoh/guisdap/m152/lpgwrcalc.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% lpgwrcalc.m
%
% calculates the range ambiguity functions for all signal lag profile groups
% and also the range to the first gate and the width of the ambiguity functions
%
% Variables produced:
% lpg_h: Range to the center of range ambiguity function of the first gate
% lpg_w: Width of the range ambiguity function (twice the second moment)
% lpg_wr: The range ambiguity function
```

```
%
% See also: init_GUP wrlpg
fprintf('\n Calculating the range ambiguity functions for signal lag profile groups:\n\n')
lpg_wr=zeros(1000,length(lpg_ra));
for i=find(lpg_bcs=='s' | lpg_bcs=='x')
  w=wrlpg(i)/10; r=col(1:length(w));
  lpg_wr(length(w),i)=0;lpg_wr(r,i)=w;
  indw=find(w>0.065*max(w));   % main body of ambiguity function
  if length(indw)>0;
    pp=sum(indw.*w(indw))/sum(w(indw));
    lpg_h(i)=pp;
    lpg_w(i)=2*sqrt(sum(w(indw).*(indw-pp).^2)/sum(w(indw)));
  else
    lpg_h(i)=0; lpg_w(i)=0;
  end
  fprintf('Lag profile group %3.0f first range %5.0f us',i,lpg_h(i)*p_dtau)
  fprintf(' (%5.1f km)',lpg_h(i)*p_dtau*.150)
  fprintf(' width %5.0f us \n',lpg_w(i)*p_dtau)
  plot(r*p_dtau,w);
  title(['range ambiguity function for lpg=' num2str(i)]);grid;%prtsc
  drawnow
end
fprintf('\n\nRange ambiguity functions calculated\n\n')
clear ind file fid i w z0 z1 k k0 dt nt kold indw left right j pp jold r
```

The following routine defines temperature, density, frequency and other scales. If the hard coded values are not suitable for some experiment, they can be redifined by a Matlab file called *EXPRNAME*`_specpar.m`

/geo/gmt/askoh/guisdap/m152/read_specpar.m

```
% GUISDAP v1.50   94-03-10 Copyright Markku Lehtinen, Asko Huuskonen
%
% this script loads the _specpar file for the experiment, if it is available
% If the file is not found, the scale parameters will be those given in this routine
%
% See also: init_GUP

p_T0=300;
p_N0=1e11;
p_m0=[30.5 16];

% p_om=(-6:.1:6)'; % This range is not wide enough
p_om=2*sinh(-3:0.05:3.001)'; % Positive values shown below
%  0.00  0.10  0.20  0.30  0.40  0.51  0.61  0.71  0.82  0.93  1.04  1.16  1.27
%  1.39  1.52  1.64  1.78  1.91  2.05  2.20  2.35  2.51  2.67  2.84  3.02  3.20
%  3.40  3.60  3.81  4.03  4.26  4.50  4.75  5.01  5.29  5.58  5.88  6.20  6.54
%  6.89  7.25  7.64  8.04  8.47  8.91  9.38  9.87 10.39 10.93 11.50 12.10 12.73
% 13.39 14.08 14.81 15.58 16.38 17.23 18.12 19.05 20.04
p_R0=1000;

if exist('name_expr')==1,

  file=canon([name_expr name_site '_specpar'],0);

  if exist(file)==2,
     eval(file),
  else
    fprintf(['\n    ',file,' file is not available (need not be!)\n'])
    fprintf('    Hard coded values for scale parameters will be used\n')
    fprintf('\n')
  end
end
fprintf('Temperature scale is %.0f K\n', p_T0)
fprintf('Electron density scale is %.1e m^-3\n', p_N0)
fprintf('Ion masses are %.1f u and  %.1f u\n', p_m0(1), p_m0(2))
fprintf('Reference range is %.0f us\n', p_R0)
fprintf('Frequency values in scaled units range from %.1f to %.1f\n',min(p_om),max(p_om))

clear file
```