A TOOL OF SEPARATING SPECTRAL WINDOWS IN MIRIAD -
Pre-processing SMA data taken from the Hybrid SWARM+ASIC correlator for further calibrations and imaging

Abstract -
As required by SMA Miriad users, we have implemented a Miriad task swarmsplt. This task is useful for Miriad users to separate SMA visibility data in various spectral chunks generated via different IFs from either ASIC or SWARM correlator components. Given two independent backends for the ASIC and SWARM data streamers and opposite sign of the channel increments for the two spectral chunks in a SWARM quadrant given a sideband, the separation is necessary in calibrations of the SMA hybrid correlator data using the existing routines in Miriad. In addition, swarmsplt provides a function to average the spectral resolution. Using the new SMA-Miriad pipeline task smalod along with swarmsplt, a pre-processing C-shell script for the new SMA data generated from the hybrid SWARM+ASIC correlator has been implemented. The products of the pre-processing data are compatible with the latest version of SMA-Miriad 1.5.1 that has been globally distributed. The pre-processing has provided a quick and useful way for Miriad users inside or outside CfA in handling the interferometer data produced from the SWARM correlator during SMA backend upgrading. RTDC can use the scripts to distribute SMA data upon request from Miriad users outside CfA.

The users' interface program in Fortran, swarmsplt.for, is given to demonstrate how to make SMA SWARM data be adapted generally to the tasks in Miriad. The new Miriad task has been implemented in Miriad SWARM3.1.0 or newer versions.

1. Why swarmsplt -
There are good reasons to separate between the SWARM and ASIC correlator data prior to calibrating and imaging in Miriad. The main concerns for handling SWARM data with general Miriad tasks are summarized in the following aspects:

1) The SWARM and ASIC data streamers are produced from independent electronic and digital backend systems; the error sources are different. In particular, one needs to pay attention to the variations of the complex gains due to instrumental issues during an observation track although the results of the preliminary analysis show that the phase drifting of the SWRAM and ASIC data appeared to track each other in an observing run of 8 hours. (see Thechnical Highlights in SMA Newsletter Number 20, July 13, 2015)

2) Spectral resolutions across the data chunks produced from SMA hybrid correlator are not uniform. In particular, the channel resolution of the SWARM chunks is about an order of magnitude finer than those of ASIC chunks. Each of SWARM chunks contains 16384 channels including those low amplitude guard-band channels near the edges of the spectra. Extra programming efforts in the specific treatment for low S/N and the implementation for large memory buffers in the Miriad code are needed in further calibrating the SWARM data in high spectral resolution.

3) In Miriad convention, the signs of spectral channel increments from various spectral windows (chunks) are assumed to be the same given a sideband. However, the signs of spectral channel increment are opposite for the two chunks in a SWARM quadrant given a sideband. The new SWARM data structure is confusing. In particular, for the calibration and continuum imaging routines, averaging is inevitable. The resultant frequency from averaging without separating the two SWARM chunks leads to an incorrect reference frequency, which could become problems in further reduction of the SWARM data, and...
could cause a failure in imaging.

4) This task can be used to pre-process the new SMA data produced from the hybrid SWARM+ASIC correlator. It is necessary for users to separate the data between ASIC and SWARM chunks in order to use the existing Miriad routines and example scripts for the reduction of the SMA SWARM data in Miriad. By separating the ASIC and SWARM chunks, and reducing a factor of two in the spectral resolution for the SWARM data, swarmsplit can furnish users a set of pre-processed ASIC and SWARM data that is compatible with Miriad software in the previous distributions.

Finally, Ch0 contains only 1 channel of pseudo continuum visibility data that is designed for the SMA scheduler to make a quick image with Miriad in the data quality assurance routine. This channel window is recommended to be eliminated prior to further data processing in Miriad in order to avoid problems related to the single-channel window.

[4] SMA Operations Logs #29874 and #29990

• 2. Help deck -

The Miriad help deck provides a description for the task swarmsplit, including the programmer who wrote the code and who is responsible to modification and further implementation. The description also gives the Keywords that are used in the task and can be supplied by users via the input deck or a Miriad command line in a shell environment.

Task: swarmsplit
Responsible: Jun-Hui Zhao

swarmsplit special for SMA data produced from the hybrid correlators ASIC and SWARM, splitting the uv dataset into either the ch0 or the 48 spectral chunks produced from the ASIC correlator, or the broadband spectral chunks produced from the SWARM correlator.

Keyword: vis
The name of the input uv data sets. The input data must be in the original format generated with smalod in converting SMA data produced from the hybrid correlator SWARM and ASIC; i.e. the data with 51 spectral windows (ch0 + 48 ASIC chunks and 2 SWARM chunks). No default.

Keyword: out
The name of the output uv data set. No default.

Keyword: options
This gives extra processing options. Eight options given below can be used one at a time. No multiple options are allowed:

  ch0 Means taking the first channel (ch0) data for continuum.
  asic Means all the 48 spectral chunk data produced from the legacy ASIC correlator including two IF chunks.
  if1 Means the first IF in the sideband--the first 24 spectral chunk data (S1,S2,...S24) produced from the legacy ASIC correlator.
  if2 Means the second IF in the sideband--the rest 24 spectral chunk data (S25,S26,...S48) produced from the legacy ASIC correlator.
  swarm Means two SWARM spectral chunks (S49, S50).
  sw1 the first SWARM chunk (S49).
  sw2 the second SWARM chunk (S50).
  sw3 the third SWARM chunk (S51).\(^5\)
  sw4 the third SWARM chunk (S52).\(^5\)
  sw5 the third SWARM chunk (S53).\(^5\)
  sw6 the third SWARM chunk (S54).\(^5\)
  sw7 the third SWARM chunk (S55).\(^5\)
  sw8 the third SWARM chunk (S56).\(^5\)

Default is the ASIC N (48 or less) spectral chunks' data and the two (eight eventually) SWARM chunks, excluding the ch0.
Keyword: line
The normal uv linetype in the form:
line,nchan,start,width,step
The default is all channels.
To reduce down spectral resolution for swarm data,
one may set the keyword - line as:
line = channel,8192,1,2
for options = sw1 or sw2 or other SWARM chunks; or
line = channel,16384,1,2
for options = swarm.

3. Usage -
Examples are given for usage in the current version implemented for the first SWARM quadrant or two SWARM chunks:

1) for all spectral chunks excluding ch0, the pseudo continuum
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.spc
   options   =

2) for all swarm chunk data
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.swarm
   options   = swarm

3) for 1st swarm chunk (S49) data
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.swl
   options   = sw1

4) for 2nd swarm chunk (S50) data
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.sw2
   options   = sw2

5) for all swarm chunk data, averaging every two channels
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.swarm
   options   = swarm
   line      = channel,16384,1,2

6) for 1st swarm chunk (S49) data, averaging every two channels
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
   out       = 150519_rx0.usb.swl
   options   = sw1
   line      = channel,8192,1,2

7) for 2nd swarm chunk (S50) data, averaging every two channels
swarmsplit% inp
   Task: swarmsplit
   vis       = 150519_rx0.usb
8) for all ASIC chunk data, doubleband data

```bash
swarm_split
Task: swarm_split
vis  = 150519_rx0.usb
out  = 150519_rx0.usb.asic
options = asic
```

9) for the 1st IF chunk data from the ASIC correlator

```bash
swarm_split
Task: swarm_split
vis  = 150519_rx0.usb
out  = 150519_rx0.usb.if1
options = if1
```

10) for the 2nd IF chunk data from the ASIC correlator

```bash
swarm_split
Task: swarm_split
vis  = 150519_rx0.usb
out  = 150519_rx0.usb.if2
options = if2
```

11) for ch0 data

```bash
swarm_split
Task: swarm_split
vis  = 150519_rx0.usb
out  = 150519_rx0.usb.ch0
options = ch0
```

4. Pre-processing and compatibility -

A C-shell script `swarmload.csh` has been implemented to pre-process the new SMA data produced from the hybrid SWARM+ASIC correlator. The pre-processing with the C-shell script involves two Miriad tasks, `smalod` and `swarm_split`. The former is used to load the new SMA data packaged in a new MIR/IDL format and write them in Miriad format. To date (August 3, 2015), the new SWARM correlator has been implemented for two chunks of spectral data; in the process of converting to Miriad format, the SMA spectral data are separated with `swarm_split` into six subsets of the SWARM and ASIC data for the two sidebands. Each sideband consists of one ASIC subset and two SWARM subsets. The spectral resolution of the two SWARM subsets can be reduced by a factor of two. These subsets are fully compatible with the latest version of the SMA Miriad software package that has been globally distributed, SMA Miriad 1.5.1 (RH 6 or CentOS 6). Then users can carry out further calibrations and imaging to follow the example scripts that have been posted in a link accessible to public, SMA MIRIAD: Data reduction scripts.

The application appears to be quite useful to the SMA users outside CfA who are not accessible to the new Miriad-SWARM software before the new system of duplication and distribution to be built up for handling SWARM data in a full spectral resolution. A guideline to process the new SWARM and ASIC data using SMA Miriad 1.5.1 (RH 6 or CentOS 6) has been posted in both the SMA data reduction page and the SMA Miriad web site, MIRIAD-SWARM Software for Users Outside CfA. With a minimum assistance from RTDC or SMA staff, SMA users who are not able to access to the CfA computer systems will be able to process the new hybrid SWARM+ASIC data in Miriad.

As more SWARM hardware components to be implemented, the pre-processing will be implemented for handling more subsets of SWARM data with the globally-distributed SMA Miriad package.

5. Source code -

The relevant Fortran code that is used for user's interface is given below along with the header file. This is a special program implemented for pre-processing the new SMA data generated from the SWARM+ASIC hybrid correlator. This new Miriad task is developed based on `smasplit`. The current version distributed in Miriad-SWARM 3.1.1 has been implemented for the first SWARM quadrant (two SWARM chunks). Eight options are support to separate the new SMA data generated from the different correlator and IF components.
Also, a function of linetype has been implemented; with averaging down the spectral resolution of the SWARM chunks, the pre-processed SWARM data are compatible with the SMA-Miriad package that was specially or originally configured and developed for reduction of the SMA ASIC correlator data with a BW of 4GHz, or doubleband.

5.1 swarmsplit for

```fortran
program swarmsplit
implicit none

c = swarmsplit - split uv dataset into either ch0 or ASIC or SWARM chunks.

c+ jhz

c = uv handling

SWARMSPiLT is a special tool for handling SMA data produced from
the hybrid correlators ASIC and SWARM, splitting the uv dataset
into either the ch0 or the ASIC (48 or smaller) spectral chunks
or the two SWARM chunks produced from the SMA hybrid correlator
(ASIC + SWARM).

vis
The name of the input uv data sets. The input data must be in the
original format generated with SMALOD in converting SMA data produced
from the hybrid correlator SWARM and ASIC; i.e. the
data with 51 spectral windows (ch0 + 48 ASIC chunks and 2 SWARM
chunks). No default.

out
The name of the output uv data set. No default.

options
This gives extra processing options. Eight options given below can be
used one at a time. No multiple options are allowed:

    ch0   Means taking the first channel (ch0) data for continuum.
    asic  Means all the 48 spectral chunk data produced from the
          legacy ASIC correlator including two IF chunks.
    if1   Means the first IF in the sideband--the first 24 spectral
          chunk data (S1,S2,...S24) produced from the legacy ASIC
          correlator.
    if2   Means the second IF in the sideband--the rest 24 spectral
          chunk data (S25,S26,...S48) produced from the legacy ASIC
          correlator.
    swarm Means two SWARM spectral chunks (S49,S50).
    sw1   the first SWARM chunk (S49).
    sw2   the second SWARM chunk (S50).
          Default is the ASIC N (48 or less) spectral chunks data and
          the two SWARM chunks, excluding the ch0.

line
The normal uv linetype in the form:
    line,nchan,start,width,step
The default is all channels.
To reduce down spectral resolution for swarm data,
one may set the keyword - line as:
    line = channel,8192,1,2
for options = sw1 or sw2; or
    line = channel,16384,1,2
for options = swarm.

--

History:
    jhz 25Nov14 borrowed setup for ASIC correlator from
    smasplit
    jhz 15Feb15 re-config uvflags index for SWARM chunks in a uv handling
    subroutine
    jhz 20Feb15 tested for ASIC data only
    jhz 15Jun15 implemented select functions for spectral chunks from SWARM
    correlator for two spectral chunks (1st quadrant)
    jhz 01Jul15 implemented for SWARM+ASIC hybrid correlator
    jhz 14Jul15 increased MAXVIS and updated inline doc
    jhz 20Jul15 implement function of linetype for swarm data
    to reduce down spectral resolution
```

To do: To implement more SWARM chunks.

#include 'maxim.h'
character version(*)(*)
parameter(version='SwarbSplit: version1.0.6 20-July-2015')
character uvflags*12,ltype*16,out*64
integer npol,Snpol,pol,tIn,tOut,vupd,nread,nrec,i,nbin
real intime,jyerk
logical dotaver,doflush,buffered,PolVary,first
logical ok,donenpol,doasic,doch0,doi1,doi2,dohyb
logical doswarm,dosw1,dosw2,doasicsw
double precision preamble(5),Tmin,Tmax,Tprev,interval
complex data(MAXCHAN)
logical flags(MAXCHAN)
logical ampsc,vecamp,relax,doprt
integer nspect, s2m(2,2)
integer nschan(MAXWIN)
double precision sfreq(MAXWIN)

character in*80
integer tno

Externals.

logical uvDatPrb,uvDatOpn,uvVarUpd
logical doSMAorg

common/nswindows/nspect
common/SMAIFS/s2m

Get the input parameters.

nspect = 51
dohyb = .false.
doprt = .false.
call output(version)
call keyini
uvflags = 'slr3'
call uvDatInp('vis',uvflags)
if(uvDatOpn(tIn)) then
call uvDatRd(preamble,data,flags,maxchan,nread)
dowhile(nread.gt.0)
call uvrdvi(tin,'nspect',nspect,0)
call uvgetvri(tin,'nschan',nschan,nspect)
call uvgetvrd(tin,'sfreq',sfreq,nspect)
call SpecSMA(tin,
* nspect,nschan,sfreq,MAXWIN,doprt)
call uvDatRd(preamble,data,flags,maxchan,nread)
if (nspect.gt.2) nread=0
end do
if (nspect.lt.51) dohyb = .true.
end if
call uvDatCls
doSMAorg=.false.
if(nschan(1).eq.1) doSMAorg=.true.
if(.not.doSMAorg) call bug('f',
* 'SWARMSPIT: does not work for this dataset')
call keyini
uvflags = 'dslr3'
call
* getopt(uvflags,doasic,doch0,doi1,doi2,doswarm,
* dosw1,dosw2,doasicsw)
call uvDatInp('vis',uvflags)
call keyd('interval',interval,0.d0)
call keya('out','out',')
call keyfin
c check the logic conflict in options
if(dof2.and.dof1)
  * call bug ('f','Hybrid-mode: do one IF at a time.')
c
C make a certain no averaging in the splitting
C
      interval = 0.d0
      ampsc = .false.
      vecamp = .false.
      relax = .false.
C
C Check the input parameters.
C
if(doasic.and.(.not.dohyb))
  * call bug ('w','Split the SMA ASIC data into 48 spectral chunks data.')
if(doch0)
  * call bug ('w','Split the first channel (Ch0) from the SMA ASIC data.')
if(dof1)
  * call bug ('w','Split the first IF data from the SMA ASIC data.')
if(dof2)
  * call bug ('w','Split the second IF data from the SMA ASIC data.')
if(dohyb)
  * call bug ('w','Handling the hybrid-resolution SMA ASIC data,')
if(dohyb)
  * call bug ('w','split all spectral chunks data. )
if(doswarm)
  * call bug ('w','Split all SWARM chunks.')
if(doswl)
  * call bug ('w','Split 1st SWARM chunk S49.')
if(dosw2)
  * call bug ('w','Split 2nd SWARM chunk S50. ')
if(doasicsw)
  * call bug ('w','Split all SWARM and ASIC chunks.')
if(out.eq.' ' )call bug('f','Output file must be specified')
if(interval.lt.0)call bug('f','Illegal value for interval')
C
C Various initialisation.
C
interval = interval/(24.*60.)
npol = 0
Snpol = 0
first = .true.
PolVary = .false.
doflush = .false.
buffed = .false.
donenpol = .false.
dotaver = interval.gt.0.or.uvDatPrb('polarization?',0.d0)
call BufIni
nrec = 0
C
C Open the input and the output files.
C
dowhile(uvDatOpn(tIn))
nbin = 1
if(dotaver)then
  call uvrdvri(tIn,'nbin',nbin,1)
if(nbin.gt.1)then
  call bug('w','
      * 'Time averaging or pol' 'n selection of bin-mode data')
  call bug('w','
      * This will average all bins together')
  endif
endif
call uvDatGta('ltype',ltype)
call VarInit(tIn,ltype)
call uvVarIni(tIn,vupd)
call uvVarSet(vupd,'dra')
call uvVarSet(vupd,'ddc')
call uvVarSet(vupd,'source')
call uvVarSet(vupd,'on')

Special processing the first time around.
if(first)then
   call uvopen(tOut,out,'new')
call uvset(tOut,'preamble','uvw/time/baseline',0,0,0,0,0,0)
call hdcopy(tIn,tOut,'history')
call hisopen(tOut,'append')
call hiswritet(tOut,'SWARMSPLT: Miriad '/version)
call hisinput(tOut,'SWARMSPLT')
call hisclose(tOut)
first = .false.
endif
   call VarOnit(tIn,tOut,ltype)
endif

Loop over the data.
call uvDatRd(preamble,data,flags,maxchan,nread)
Tprev = preamble(4)
Tmin = Tprev
Tmax = Tmin
dowhile(nread.gt.0)

Count the number of records read.
nrec = nrec + 1

Do we want to keep this record.
ok = relax.or.donenpol
   if(.not.ok)then
      do i=1,nread
         ok = ok.or.flags(i)
      enddo
   endif

determine if we need to flush out the averaged data.
doflush = ok.and.dotaver
   if(doflush)then
      doflush = uvVarUpd(vupd)
      oflush = (doflush.or.preamble(4)-Tmin.gt.interval.or.
         Tmax-preamble(4).gt.interval)
      .and.buffered
   endif

Flush out the accumulated data -- the case of time averaging.
if(doflush)then
   call bufflush(tOut,ampsc,vecamp,npol)
   PolVary = PolVary.or.npol.eq.0.or.
      (Snpol.ne.npol.and.Snpol.gt.0)
   Snpol = npol
   Tmin = preamble(4)
   Tmax = Tmin
   buffered = .false.
endif

Flush out the accumulated data -- the case of no time averaging.
else if(.not.dotaver)then
   if(npol.le.0)call uvDatGti('npol',npol)
   if(ok)then
if (.not. donepol) then  
call uvputvri(tOut,'npol',npol,1)  
PolVary = PolVary.or.  
   (Snpol.ne.npol.and.Snpol.gt.0)  
Snpol = npol  
endif  
call uvDatGti('pol',pol)  
call uvputvri(tOut,'pol',pol,1)  
call VarCopy(tIn,tOut)  
call uvDatGtr('jyperk',jyperk)  
call uvputvrr(tOut,'jyperk',jyperk,1)  
call uwrite(tOut,preamble,data,flags,nread)  
donepol = npol.gt.1  
endif  
npol = npol - 1  
endif  
c  
  Accumulate more data, if we are time averaging.  
  
  if (dotaver.and.ok) then  
call uvrdvrr(tIn,'inttime',inttime,10.)  
call bufacc(preamble,inttime,data,flags,nread)  
buffered = .true.  
call VarCopy(tIn,tOut)  
if (nbin.gt.1) call uvputvri(tOut,'nbin',1,1)  
endif  
c  
  Keep on going. Read in another record.  
  
  if (ok) then  
     Tprev = preamble(4)  
     Tmin = min(Tmin,Tprev)  
     Tmax = max(Tmax,Tprev)  
  endif  
call uvDatRd(preamble,data,flags,maxchan,nread)  
enddo  
c  
  Flush out anything remaining.  
  
  if (buffered) then  
call bufflush(tOut,ampsc,vecamp,npol)  
PolVary = PolVary.or.npol.le.0.or.  
   (Snpol.ne.npol.and.Snpol.gt.0)  
Snpol = npol  
buffered = .false.  
endif  
call uvDatCls  
enddo  
c  
  Write things to the header which describe the data as a whole.  
  
  if (first) call bug('f','Error opening input')  
  if (nrec.eq.0) call bug('f','No data found')  
  if (.not.PolVary) call wrhd(i(tOut,'npol',Snpol)  
c  
  Update the history and close up files.  
  
  call uvclose(tOut)  
  if (doasic.or.doasicsw.or.doswarm.or.  
     * doif1.or.doif2.or.dosw1.or.dosw2)  
     * call output('The program ends successfully.')  
  end  
c  
 **************************************************************************  
c  subroutine getopt(uvflags,doasic,doch0,doif1,doif2,  
     * doswarm,dosw1,dosw2,doasicsw)  
implicit none  
logical doasic,doch0,doif1,doif2
logical doswarm,dosl1,dosl2,doasicsw
character uvflags(*)

C Determine the flags to pass to the uvdat routines.

C Output:

C uvflags Flags to pass to the uvdat routines.

C doch0 Handling the SMA DB data for the ch0, the 1st window.

C doif1 Handling the SMA DB data for the first 24 windows
C (2,3,...,25).

C doif2 Handling the SMA DB data for the next 24 windows
C (26,27,...,49).

C doswarm Handling the SMA SWARM data for 50 (S50) and 51 (S49)

C doswl Handling the SMA SWARM data for 51 (S49), the 1st swarm chunk

C dosw2 Handling the SMA SWARM data for 50 (S50), the 2nd swarm chunk

C doasicsw Handling all the spectral windows from both ASIC and SWARM
C but excluding the single-channel window Ch0

integer nopts
parameter(nopts=10)
character opts(nopts)*8
integer l, nopt
logical present(nopts),docal,dopol,dopass

data opts/'nocal','nopol','nopass','ch0','asic','if1','if2','swarm','
* 'swl','sw2'
C
c call options('options',opts,present,nopts)

nopt=0

doch0    = present(4)
doasic   = present(5)
doif1    = present(6)
doif2    = present(7)
doswarm  = present(8)
dosl1    = present(9)
dosl2    = present(10)
if(doch0) nopt=nopt+1
if(doasic) nopt=nopt+1
if(doif1) nopt=nopt+1
if(doif2) nopt=nopt+1
if(doswarm) nopt=nopt+1
if(dosl1) nopt=nopt+1
if(dosl2) nopt=nopt+1
if(nopt.gt.1) call bug('f', 'One options at a time.')</n
nopt=0
if(.not.doch0) nopt=nopt+1
if(.not.doasic) nopt=nopt+1
if(.not.doif1) nopt=nopt+1
if(.not.doif2) nopt=nopt+1
if(.not.doswarm) nopt=nopt+1
if(.not.dosl1) nopt=nopt+1
if(.not.dosl2) nopt=nopt+1
if(nopt.eq.7) doasicsw = .true.
docal = .false.
dopol = .false.
dopass = .false.

C Set up calibration flags

C

uvflags = 'dsir3'

l = 5
if(docal)then
    l = l + 1
    uvflags(l:1) = 'c'
endif
if(dopass)then
    l = l + 1
    uvflags(l:1) = 'f'
endif
if(dopol)then
  l = l + 1
  uvflags(l:1) = 'e'
endif
if(doasic) then
  l = l + 1
  uvflags(l:1) = 'z'
endif
if(docho0) then
  l = l + 1
  uvflags(l:1) = '0'
endif
if(doif1) then
  l = l + 1
  uvflags(l:1) = 'y'
endif
if(doif2) then
  l = l + 1
  uvflags(l:1) = '2'
endif
if(doswarm) then
  l = l + 1
  uvflags(l:1) = '4'
endif
if(dosw1) then
  l = l + 1
  uvflags(l:1) = '5'
endif
if(dosw2) then
  l = l + 1
  uvflags(l:1) = '6'
endif
if(doasicsw) then
  l = l + 1
  uvflags(l:1) = '7'
endif
end

************

subroutine bufini
implicit none

Initialise the routines which do the buffering and averaging of
the visibility data.
All the buffering/averaging is performed in arrays stored in a
common block.

--------------------
include 'swarmsplt.h'
free = 1
mbase = 0
end

************

subroutine bufflush(tOut,ampsc,vecamp,npol)
implicit none
integer tOut,npol
logical ampsc,vecamp

This writes out the averaged data. The accumulated data is in common.
This starts by dividing the accumulated data by "N", and then writes
it out.
Inputs:
tOut The handle of the output file.
include 'swarmsplit.h'
complex data(MAXCHAN)
real amp,intime
double precision preamble(5),time(MAXBASE)
logical flags(MAXCHAN)
integer i,j,jd,k,ngood,nbp,p,idx1(MAXBASE),idx2(MAXBASE)
logical PolVary,doamp

c Externals.

c logical PolsPara
PolVary=.false.

c Determine the number of good baselines, and sort them so we have an
c index of increasing time.

c ngood = 0
do j=1,mbase
  if(cnt(j).gt.0)then
    ngood = ngood + 1
    time(ngood) = preamble(4,j) / cnt(j)
    idx2(ngood) = j
  endif
endo
if(ngood.le.0)return
call sortidxd(ngood,time,idx1)

c Now loop through the good baselines, writing them out.

c nbp = 0
npol = 0
do jd=1,ngood
  j = idx2(idx1(jd))
  if(npols(j).ne.npol)then
    call uvputvri(tOut,'npol',npols(j),1)
    PolVary = npol.gt.0
    npol = npols(j)
  endif
  preamble(1) = preamble(1,j) / cnt(j)
  preamble(2) = preamble(2,j) / cnt(j)
  preamble(3) = preamble(3,j) / cnt(j)
  preamble(4) = preamble(4,j) / cnt(j)
  preamble(5) = preamble(5,j) / cnt(j)
  intime = preamble(6,j) / npols(j)
  call uvputvrr(tOut,'inttime',intime,1)
endo

c Average the data in each polarisation. If there is only one scan in the
c average, not bother to average it.

dois1,npol
  p = pnt(i,j) - 1
  call uvputvri(tOut,'pol',pol(i,j),1)
doamp = ampsc.or.(vecamp.and.PolsPara(pol(i,j)))
  nbp = nbp + 1
endo

c Loop over the channels. If we are doing amp-scalar averaging, and
c the average visibility is zero, flag the data. Otherwise just
c depend on whether we have good data or not.

do k=1,nchan(i,j)
  if(doamp.and.
    * abs(real(buf(k+p)))+abs(aimag(buf(k+p))).eq.0)
* count(k+p) = 0
flags(k) = count(k+p).gt.0
if(.not.flags(k))then
  data(k) = 0
else if(doamp)then
  amp = abs(buf(k+p))
  data(k) = (bufr(k+p) / count(k+p)) * (buf(k+p) / amp)
else
  data(k) = buf(k+p) / count(k+p)
endif
enddo
call uvwrite(tOut,preamble,data,flags,nchan(i,j))
enddo
c
Reset the counters.
c
free = 1
mbase = 0
c
If the number of polarisations varied, zero npol.
c
if(PolVary) npol = 0
end
c
******************************************************************************
c
subroutine bufacc(preamble,inttime,data,flags,nread)
c
implicit none
integer nread
double precision preamble(5)
real inttime
complex data(nread)
logical flags(nread)
c
This accumulates the visibility data. The accumulated data is left
c in common.
c
Input/Output:
c preamble Preamble. Destroyed on output.
c data The correlation data to be averaged. Destroyed on output.
c flags The data flags.
c nread The number of channels.
c------------------------------------------------------------------------------
include 'swarmsplt.h'
integer i,i1,i2,p,bl,pol
c
Determine the baseline number, and conjugate the data if necessary.
c
call BasAnt(preamble(5),i1,i2)
bl = ((i2-1)*i2)/2 + i1
if(bl.gt.MAXBASE)
  * call bug('f','Too many baselines for me to handle, in BUFACC')
c
Zero up to, and including, this baseline.
c
do i=mbase+1,bl
  cnt(i) = 0
enddo
mbase = max(mbase,bl)
c
Add in this visibility.
c
if(cnt(bl).eq.0)then
  cnt(bl) = inttime
  npols(bl) = 0
  preamble(1,bl) = inttime * preamble(1)
preamble(2,bl) = inttime * preamble(2)
preamble(3,bl) = inttime * preamble(3)
preamble(4,bl) = inttime * preamble(4)
preamble(5,bl) = inttime * preamble(5)
preamble(6,bl) = inttime
else
cnt(bl) = cnt(bl) + inttime
preamble(1,bl) = preamble(1,bl) + inttime * preamble(1)
preamble(2,bl) = preamble(2,bl) + inttime * preamble(2)
preamble(3,bl) = preamble(3,bl) + inttime * preamble(3)
preamble(4,bl) = preamble(4,bl) + inttime * preamble(4)
preamble(5,bl) = preamble(5,bl) + inttime * preamble(5)
preamble(6,bl) = preamble(6,bl) + inttime
endif

c Determine the polarisation.
c
call uvDatGti('pol',pol)
p = 0
do i=1,npols(bl)
  if(pols(i,bl).eq.pol) p = i
endo
c
A new baseline. Set up the description of it.
c
if(p.eq.0) then
  npols(bl) = npols(bl) + 1
  p = npols(bl)
  if(p.gt.MAXPOL) call bug('f',
    * 'Too many polarizations, in BufAcc')
  pols(p,bl) = pol
  nchan(p,bl) = nread
  pnt(p,bl) = free
  free = free + nread
  if(free.gt.MAXAVER) call bug('f',
    * 'Too much data to accumulate, in BufAcc')
c
Copy across the new data.
c
p = pnt(p,bl) - 1
do i=1,nread
  if(flags(i)) then
    buf(i+p) = inttime * data(i)
    bufr(i+p) = inttime * abs(data(i))
    count(i+p) = inttime
  else
    buf(i+p) = (0.0,0.0)
    bufr(i+p) = 0.0
    count(i+p) = 0
  endif
endo
c
Else accumulate new data for old baseline.
c
else
  nread = min(nread,nchan(p,bl))
nchan(p,bl) = nread
  p = pnt(p,bl) - 1
  do i=1,nread
    if(flags(i)) then
      buf(i+p) = buf(i+p) + inttime * data(i)
      bufr(i+p) = bufr(i+p) + inttime * abs(data(i))
      count(i+p) = count(i+p) + inttime
    endif
  enddo
enddo
c
c end
5.2 swarmsplt.h

#include 'maxdim.h'
integer MAXAVER,MAXPOL
parameter(MAXAVER=20000000,MAXPOL=4)
complex buf(MAXAVER)
real       bufr(MAXAVER)
double precision count(MAXAVER),cnt(MAXBASE)
integer     pnt(MAXPOL,MAXBASE),nchan(MAXPOL,MAXBAS),free,mbase
integer npols(MAXBASE),pols(MAXPOL,MAXBAS)
double precision preamble(6,MAXBAS)
column/uvavcom/preamble,count,cnt,buf,bufr,pnt,nchan,npols,*
pols,free,mbase