Reading FITS files from the LSS receiver

file = name of FITS file
The file has 2 levels of header plus 1 level of data.

Reading the level 0 header (PRIMARY) :

header0 = headfits(file, exten=0)

The routine **sxpar** allows then to access each parameter, e.g. :

number of elementary accumulations per spectrum =	**sxpar(header,'ACC')**	[long]
spectrum duration (msec) =	**sxpar(header,'DT')**	[float]
start time =	**sxpar(header,'TIME-OBS')**	[string]
end time =	**sxpar(header,'TIME-END')**	[string]
receiver name =	**strcompress(strupcase(sxpar(header,'INSTRUME')),/remove_all)**	

To list all parameters, type : **print, header0**

Reading the level 1 header and data (SETUP) :

header1 = headfits(file, exten=1)

To list all parameters, type : **print, header1**

a = mrdfits(file, 1)

Content of structure a : **help,/struct,a**

number of frequencies =	a.nf	[int]
list of frequencies =	a.frq	[float array of dimension (a.nf)]
number of input channels =	a.nbchan	[int]
list of correlations computed =	a.chan	[int array of dimension (a.nbchan, 2)]

Data of channel k consists of the correlation of antennas a.chan[k,0] and a.chan[k,1]

a.chan		MR 1		MR 2		MR 3	
		NW	NE	NW	NE	NW	NE
MR 1	north-west	[0,0]	[1,0]	[2,0]	[3,0]	[4,0]	[5,0]
	north-east	[0,1]	[1,1]	[2,1]	[3,1]	[4,1]	[5,1]
MR 2	north-west	[0,2]	[1,2]	[2,2]	[3,2]	[4,2]	[5,2]
	north-east	[0,3]	[1,3]	[2,3]	[3,3]	[4,3]	[5,3]
MR 3	north-west	[0,4]	[1,4]	[2,4]	[3,4]	[4,4]	[5,4]
	north-east	[0,5]	[1,5]	[2,5]	[3,5]	[4,5]	[5,5]

Real part of a cross-correlation

Imaginary part of a cross-correlation

Auto-correlation

Reading the level 2 header and data (DATA CUBE) :

header2 = headfits(file, exten=2)

To list all parameters, type : **print, header2**

Total number of spectra in file is given by : **ns=sxpar(header2,'NAXIS2')**

Reading data cube from spectrum ns_begin to spectrum ns_end (included) :

cube=mrdfits(file, 2, range=[ns_begin,ns_end])

Size of structure cube : **help,cube**
Content of structure cube[i] : **help,/struct,cube**

julian date (0.1 msec accuracy)	cube[i].jd	[double]
millisecond (within current sec)	cube[i].msec	[double]
data	cube[i].data	[float array of dimension (a.nf, a.nbchan)]

To decode time, use the routine **CALDAT**:
CALDAT,cube[i].jd, month,day,year,hour,minute,second

cube[i].msec is included in the decimal part of second