```
## This script creates icartt files that merge ATom icartt (.ict) files from a singl
e ATom flight.
## All the ict files should be in one folder, with the name YYYYMMDD (example:  2016
0808 for the flight of Aug 8, 2016)
## usage: source(This file)  ; # should be run using R > 3.3 and started in the fold
er with all the ict files from the flight to be merged
##
##ULOD 77777 and LLOD =-8888 must be flagged and set
##

## required functions
###                     Function to convert MEDUSA CO2 data to WMO scale
co2.scripps.noaa=function(co2.scripps,sio2gmd2=-0.00001796,sio2gmd1=0.0101942,sio2gm
d0=-1.083){
## coefficients from Britt Stephens, 20170609
co2.gmd=co2.scripps^2*sio2gmd2+co2.scripps*sio2gmd1+sio2gmd0+co2.scripps
#and an offset at 400 ppm of > 400^2*sio2gmd2+400*sio2gmd1+sio2gmd0 [1] 0.12108 (Scr
ipps O2 lab low by 0.12 ppm)
return(co2.gmd)
}
#############
############   merge 1s data to medusa using the supplied filling kernels
merge.medusa.atom=function(YYYYMMDD=20160815,path.to.flights="/documents/DATA/ATOM",
keepers=c("UTC_Start","P","T","POT","TAS","G_LAT","G_LONG","G_ALT","Static_Pressure"
, "Cabin_Pressure","CO2_AO2","O2_AO2","APO_AO2","H2O_ppmv","CO2_NOAA","CO_NOAA","CH4
_NOAA","CO2_QCLS","CH4_QCLS","CO_QCLS","N2O_QCLS","H2O_UWV", "O3_UO3", "O3_CL","HCHO
","Dist") )
## columnas of the merged file to keep
           {
medusa=NULL
#for(flt in medusa.flts){
print(paste("MEDUSA",flt) )
medusa.dat=list.files(path =paste(path.to.flights,"/",flt,sep=""),pattern="MEDUSA_DC
8") ;
if(length(medusa.dat)==0) { return(NULL)
    } else {
   medusa.data= medusa.dat[length(medusa.dat)] }
medusa.kern=list.files(path =paste(path.to.flights,"/",flt,sep=""),pattern= "MEDUSA-
Kernel_DC8" ) ;
if(length(medusa.kern)==0) { return(NULL)
   } else {
  medusa.kernel = medusa.kern[length(medusa.kern)] }
# read the medusa files for this flight
medu.data=read.1001(file=paste(path.to.flights,"/",flt,"/",medusa.data,sep=""))$dat
;
#make the names conform
colnames(medu.data)[2:ncol(medu.data)]=paste(colnames(medu.data)[2:ncol(medu.data)],
"MED",sep="_")
#
APO_MED = medu.data[,"O2_N2_MED"]+1.1 * (medu.data[,"CO2_MED"] - 350) / 0.2094
medu.data=cbind(medu.data,APO_MED)
colnames(medu.data)[ncol(medu.data)]="APO_MED"
#
medu.kernel=read.1001(file=paste(path.to.flights,"/",flt,"/",medusa.kernel,sep=""))$
dat
if(!exists("Mer.1s")){
attach(paste(path.to.flights,flt,".RData",sep="/"),pos=2)
Mor.0=get("Mer.1s",pos=2)
detach(2)
 } else {
Mor.0=Mer.1s
 }
#Mor.0 contains the merged object for each flight
#loop over the 32 flasks in each flight
## columns to keep
n.keep=match(keepers,colnames(Mor.0))  ## ; n.keep=n.keep[is.finite(n.keep)]
#Mor.k=Mor.0[,n.keep]
keep.ok=keepers[is.finite(n.keep)]
#Mor.k=as.data.frame( matrix(NA,ncol=length(keepers),nrow=nrow(Mor.0)) )
Mor.k= ( matrix(NA,ncol=length(keepers),nrow=nrow(Mor.0)) )
colnames(Mor.k)=keepers
#print(dim(Mor.k)) ; print(dim(Mor.0))
Mor.k[,keep.ok]=as.matrix ( Mor.0[,keep.ok] )
# this dance to cover case where Mor.0 may not have all the parameters in keepers
t.range=c(max(c(Mor.k[1,1],medu.kernel[1,1])),min(c(Mor.k[nrow(Mor.k),1],medu.kernel
[nrow(medu.kernel),1])) )
l.mor.ok=Mor.k[,1]>=t.range[1]&Mor.k[,1]<=t.range[2]
l.med.ok=medu.kernel[,1]>=t.range[1]&medu.kernel[,1]<=t.range[2]
dum=merge(Mor.k[l.mor.ok,],medu.kernel[l.med.ok,],by=1,all=T)
Dum=NULL
for(i in 1:nrow(medu.data)){
l.sel=dum[,1]>=medu.data[i,1]&dum[,1]<=medu.data[i,2] ## start and stop times for fl
ask i
## due to NA's an awkward procedure here
F1=apply(dum[l.sel,],2,function(x)(sum(x*dum[l.sel,"Kernel"],na.rm=T)/sum(dum[l.sel,
"Kernel"][is.finite(x)])) )
F1["Kernel"]=sum(dum[l.sel,"Kernel"])
F2=apply(dum[l.sel,],2,function(x)(sum(is.finite(x))/length(x) ) )
names(F2)=paste(names(F2),"wt",sep=".")
#f0=NULL
#for(k in 1:ncol(dum)){f0=c(f0,sum(dum[l.sel,k]*dum[l.sel,"Kernel"],na.rm=T)/sum( c(
dum[l.sel&is.finite(dum[,k]),"Kernel"])) ) }
Dum=rbind(Dum,c(medu.data[i,],F1,F2) )
}
CO2_MED_gmdscale=co2.scripps.noaa(Dum[,"CO2_MED"])
#
medusa=rbind(medusa,cbind(rep(as.numeric(flt),nrow(Dum)),Dum,CO2_MED_gmdscale) )
#}
save(medusa,file=paste(path.to.flights,"/",flt,"/medusa_",flt,".RData",sep=""))
write.table(medusa,file=paste(path.to.flights,"/",flt,"/medusa_",flt,".tbl",sep=""),
row.names=F,col.names=T,quote=F)
invisible(medusa)
}
###   function to locate and determine extent of repeated values in a data set -- us
ed here to find gaps with multiple missing data
repeated.segments=function(x,value){
#gives 2-column matrix. col 1: starting index within x of each subset of x ==value;
col 2: # repeats
if(!is.na(value)){l1=( x==value)}else{l1=is.na(x)}
l1s=c(F,l1) #pad the logical to catch runs of "value" at the start and end
l1p=c(l1,F)
t2f=which(l1s & !l1p) ; f2t = which(!l1s & l1p) # indices for T->F and F->T
rr=cbind(f2t,t2f-f2t); dimnames(rr)=list(NULL,c("i0","N"))
return(rr)
}
##################################
###########  Function to read icartt (".ict") files into R; returns a list with data
 in the first slot and then metadata objects

`read.1001` = function(file2read, l.colnames = T, field.sep=",",comm.char=";",saveli
st=T,check.file=T)
{
                print(paste("Reading", file2read))
#if(substring(file2read,nchar(file2read)-2)=="ict")licartt=T else licartt=F
if(length(grep(",",scan(file2read,nlines=1,what=character())))>0)licartt=T else lica
rtt=F
print(paste("ICARTT format ",licartt))
if(licartt)csep=field.sep else csep=""
if(licartt)print("ICARTT file format")
        n1 = scan(file2read, what = integer(), n = 2, multi.line = F,comment.char=co
mm.char,sep=csep)
        if(n1[2] == 1001) {
NLHEAD = n1[1]
        } else stop(paste("Error: read file type", n1[2], "should be 1001"))
        require(stringr)
        full.header=readLines(con=file2read,n=n1[1])
authors=full.header[2]
address=full.header[3]
d.source=full.header[4]
missn=full.header[5]
vols=as.numeric(unlist( (str_split(full.header[6],csep)) ) )
        names(vols)=c("vol.num","Nvols")
date.s=as.numeric(unlist( (str_split(full.header[7],csep)) ) )
        names(date.s)=c("YYYY","MM","DD","yyyy","mm","dd")
data.interval=as.numeric(unlist(str_split(full.header[8],csep))[1])
dum.n=unlist(str_split(full.header[9],csep) )
```

```
UTC.name=(dum.n)[1]
UTC.descr=dum.n[2:length(dum.n)]
        # Num variables
NV = as.numeric(unlist(str_split(full.header[10],csep))[1])
        # scaling values for dependent variables
VSCALE = as.numeric(unlist(str_split(full.header[11],csep))[1:NV])
        # missing values for dependent variables
VMISS = as.numeric(unlist(str_split(full.header[12],csep))[1:NV])
VNAME = rep(NA,NV)
VUNITS = rep(NA,NV)
        for(i in 1:NV){ zum = unlist(str_split(full.header[12+i],csep))[1:2]
VNAME[i]=str_trim(zum[1],"b")
VUNITS[i]=str_trim(zum[2],"b")
}
NSCOML = as.numeric(unlist(full.header[12 + NV+1])[1]) ## number of special comment
lines
if(NSCOML>=0){
spec.comm=c(full.header[(12+NV+2):(12+NV+2+NSCOML-1)]) } else {spec.comm=NULL}
#
        #number of regular comment lines
NNCOML = as.numeric(unlist(full.header[12 + NV+2+NSCOML-1+1])[1]) ## number of speci
al comment lines
norm.comm=c(full.header[(12+NV+3+NSCOML):(12+NV+2+NSCOML+NNCOML)] )
## PARSE the REGULAR COMMENT LINES if this is desired; last one is columm names, whi
ch should be the same as the variable names, ind veriable first
        #if(l.colnames) VNAME = scan(file2read, skip = NLHEAD - 1, what =
                        #character(), n = NV + 1,sep=csep,comment.char=comm.char)
dum = as.matrix(read.table(file2read, skip = NLHEAD-1,sep=csep,header=T))
colnames(dum)=c(UTC.name,VNAME)
        for(i in 1:NV) {
                dum[dum[, i + 1] == VMISS[i], i + 1] = NA
        }
        dum1 = dum * matrix(rep(c(1, VSCALE), nrow(dum)), nrow = nrow(dum), byrow =
T)
n7=c(paste("0",1:9,sep=""),10:99)
if(check.file){
f2r=unlist( str_split(file2read,"_") )
if( substring(f2r[3],1,8) != paste(date.s["YYYY"],n7[date.s["MM"]],n7[date.s["DD"]],
sep="")) stop(paste(
 f2r[3],"!=",paste(date.s["YYYY"],n7[date.s["MM"]],n7[date.s["DD"]],sep="")) )
YYYYMMDD=as.numeric(f2r[3])
REV=as.numeric(substring(f2r[4],2,3))
MER.info=paste(f2r[1],REV,YYYYMMDD,paste(date.s[4],n7[date.s[5]],n7[date.s[6]],sep="
"),sep="," )
 } else { MER.info="No file info" ; REV= NA ; YYYYMMDD=paste(date.s["YYYY"],n7[date.
s["MM"]],n7[date.s["DD"]],sep="") }
RES=list("dat"=dum1,"authors"=authors,"address"=address,"d.source"=d.source,"missn"=
missn,"vols"=vols,
         "date.s"=date.s,"data.interval"=data.interval,"NV"=NV,"VUNITS"=VUNITS,"NSCO
ML"=NSCOML,
         "spec.comm"=spec.comm, "NNCOM"=NNCOML,"norm.comm"=norm.comm,"YYYYMMDD"=YYYY
MMDD,"REV"=REV
          ,"MER.info"=MER.info)
# --------------------------------
        return(RES)
}

##    BEGIN MERGE SCRIPT
library(stringr)
#
# Directory and flight information
pdir=unlist( str_split(getwd(),"/") )
flt=as.numeric(pdir[length(pdir)] )
     YYYY=substring(as.character(flt),1,4)
     MM=substring(as.character(flt),5,6)
     DD=substring(as.character(flt),7,8)
#filenames=list.files(pattern="ict")
filenames=system("ls *.ict",intern=T)
##it is inappropriate to merge the 60s file in the basic merge procedure
if(is.finite(match("AMS-60s",substring(filenames,1,7)))){
    print("Deleting AMS-60s")
    system("rm AMS-60s*")
    filenames=list.files(pattern="ict") ## 60s deleted
```

```
  }
## files giving long time averages will be merged separately; plus the merged files,
 which never should be merged
integ.prefix=c("MER","WAS","MED","PFP","PAN")
#these are the 1Hz data sets; MMS is the first to be read so it is excluded along wi
th integ.prefix
f2br.0=filenames[!substring(filenames,1,3)%in%c("MMS",integ.prefix) & (substring(fil
enames,1,9)!="SAGA-AERO") & (substring(filenames,1,9)!="CAFS-FLUX")] ## files to be
read
#F2br.0=matrix(unlist(str_split(f2br.0,"_")) ,ncol=4,byrow=T)
F2br.0=t(apply(matrix(f2br.0,ncol=1),1,function(x)unlist(str_split(x,"_"))[1:4]))
   idd=rev( duplicated(rev(F2br.0[,1]) ) ) ##"filenames" is in ascending order; mark
s as duplicated any row with earlier revision number
   F2br=F2br.0[!idd,]
   f2br=f2br.0[!idd]
#we only merge the most recent revision

#MMS file name, if there are multiple versions take the last one
n.mms=filenames[grep("MMS",substring(filenames,1,3))]
n.mms=n.mms[length(n.mms)]
print(n.mms)
#
#### now start reading the files, MMS first.
#
###  read the MMS file
MMS.0=read.1001(n.mms)
Dum=MMS.0$dat
LLOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",n.mms),inter
n=T),":"))[2])
ULOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",n.mms),inter
n=T),":"))[2])
Dum[Dum == LLOD_FLAG_MMS | Dum == ULOD_FLAG_MMS ] =NA
### full range of acceptable times; time vector with 1 Hz resolution
Dum.24=Dum
if(sum(is.finite(Dum[,"G_LONG"]+Dum[,"G_LAT"]))!=0){
lna=is.finite(Dum[,"G_LONG"]+Dum[,"G_LAT"])
print(paste(sum(!lna),"NAs found in MMS"))
a7=approx(Dum[lna,1],Dum[lna,"G_LAT"],xout=Dum[!lna,1])$y
Dum[!lna,"G_LAT"]=a7
a8=approx(Dum[lna,1],Dum[lna,"G_LONG"],xout=Dum[!lna,1])$y
Dum[!lna,"G_LONG"]=a8
a9=approx(Dum[lna,1],Dum[lna,"G_ALT"],xout=Dum[!lna,1])$y
Dum[!lna,"G_ALT"]=a9
}
UTC=(Dum[,1])

## setup lines for the 10 s merge
##cut into 10 sec intervals
  Dum.mms=Dum   ##saved for later use
  t1=trunc(UTC/10)*10
  bb=unique(c(t1,t1[length(t1)]+10)) ## breaks for cut ()
  INDEX.start=bb[as.numeric(cut(UTC,right=F,breaks=bb))]  ## INDEX for each 10s inte
rval, labelled at START
  INDEX.start.mms=INDEX.start   ##save for later
MMS.1=aggregate(Dum,by=list(INDEX.start),mean,na.rm=T)
#MMS.1[,"G_LONG"]=tapply(Dum[,"G_LONG"],INDEX.start,median,na.rm=T) ## accounts for
averaging dateline issue
funny=function(x){x1=mean(x,na.rm=T) ; if(abs(diff(range(x,na.rm=T)))> 10 | median(a
bs(trunc(x)),na.rm=T) ==179) x1=180; return(x1)}
MMS.1[,"G_LONG"]=tapply(Dum[,"G_LONG"],INDEX.start,FUN=funny     ) ## accounts for a
veraging dateline issue
UTC_Stop=MMS.1[,1]+10
MMS=cbind(MMS.1[,1],UTC_Stop,MMS.1[,2:ncol(MMS.1)])
colnames(MMS)[1:3]=c("UTC_Start","UTC_Stop","UTC_Mean")
Mer=MMS
##
#### These will be for the 10s files
Special.comments="File,YYYY,MM,DD,yyyy,mm,dd,R#"
Special.comments=c(Special.comments,paste(n.mms,paste(MMS.0$date.s,collapse=","),MMS
.0$REV,sep=",") )
Params=colnames(MMS)[2:ncol(MMS)]
Units=c("s","s",MMS.0$VUNITS )
 ALLNAMES = data.frame(params=colnames(Mer),data.set=rep("MMS",ncol(Mer)) )  ### acc
```

```
ume all params with source file for metadata
####

## ###########################################################################
######
##setup lines for the  1 s merge
##make intervals every 1 s
Dum=MMS.0$dat
LLOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",n.mms),inter
n=T),":"))[2])
ULOD_FLAG_MMS=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",n.mms),inter
n=T),":"))[2])
Dum[Dum == LLOD_FLAG_MMS | Dum == ULOD_FLAG_MMS ] =NA
### full range of acceptable times; time vector with 1 Hz resolution
UTC=(Dum[,1])

  UTC.1s=min(trunc(UTC)):(max(trunc(UTC)))
UTC_Stop.1s=UTC.1s+1
  UTC_Start.1s=trunc(UTC)
MMS.1s=aggregate(Dum,by=list(UTC_Start.1s),mean,na.rm=T)
MMS.1.1s=merge(cbind(UTC.1s,UTC_Stop.1s),MMS.1s,by=1,all=T)

colnames(MMS.1.1s)[1:3]=c("UTC_Start","UTC_Stop","UTC_Actual")
Mer.1s=MMS.1.1s
##
Special.comments.1s="File,YYYY,MM,DD,yyyy,mm,dd,R#"
Special.comments.1s=c(Special.comments.1s,paste(n.mms,paste(MMS.0$date.s,collapse=",
"),MMS.0$REV,sep=",") )
Params.1s=colnames(MMS.1.1s)[2:ncol(MMS.1.1s)]
Units.1s=c("s","s",MMS.0$VUNITS )

#
for(i2 in 1:length(f2br) ){
if(F2br[i2,1]%in%c("SAGA-MC","SAGA-AERO","TOGA")) next        ## don't merge TOGA or
 SAGA to 1s, sample interval is 35s
print(paste("1Hz merge Reading:",f2br[i2]) )
Dum.0 = read.1001(f2br[i2] )
Dum=Dum.0$dat
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
l.ok=Dum[,1]> UTC.1s[1] & Dum[,1]< UTC_Stop.1s[length(UTC_Stop.1s)]
Dum=Dum[l.ok,]
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f2br[i2]),intern
=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f2br[i2]),intern
=T),":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
   t1=trunc(Dum[,1])
if(length(t1)<5) next   ## traps case with no entries
   Dum.1=aggregate(Dum,by=list(t1),mean,na.rm=T)
   colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
if(F2br[i2,1] == "DOSE")colnames(Dum.1)[2:ncol(Dum.1)]=paste(colnames(Dum.1)[2:ncol(
Dum.1)],"DOSE",sep=".")
#PATCHES for non-conforming names ############     NAMES    NAMES     NAMES ########
#####     NAMES    NAMES     NAMES
## UCATS
if(F2br[i2,1]=="UCATS-GC"){
##check if names changed, if not, append _UCATS
print("UCATS name check")
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:nco
l(Dum.1)],"UCATS",sep="_")
     } else {
if(N7[[4]][2]!="UCATS"){                         #do nothing if it is already _UCATS
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]    ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"UCATS",sep="_")
                               }
}
}

## PANTHER
if(substring(F2br[i2,1],1,4)=="GCEC" & YYYY == "2016"){
###check if names changed, if not, append _GCECD
print("Panther check 124")
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:nco
l(Dum.1)],"GCECD",sep="_")
     } else {
if(N7[[4]][2]!="GCECD"){                        #do nothing if it is already _GCECD
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]     ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"GCECD",sep="_")
                       }
}
}

## GEOS5   avoid dup variables
if(substring(F2br[i2,1],1,5)=="GEOS5" ){
### append _GEOS5 to names
print("GEOS5 file data found!")
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GEOS5",sep="_
")
}

## GMI   avoid dup variables
if(substring(F2br[i2,1],1,3)=="GMI" ){
### append _GMI to names
print("GMI file data found!")
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GMI",sep="_")

}

## AMSSD
l.AMSSD=F
if(substring(F2br[i2,1],1,5)=="AMSSD" ){
###1st 12 columns only
l.AMSSD=T
print("AMSSD check")
Dum.1=Dum.1[,1:13]
Dum.0$VUNITS=Dum.0$VUNITS[1:11]
}


#############     NAMES     NAMES     NAMES #############     NAMES     NAMES     NAME
S
Mer.new=merge(Mer.1s,Dum.1,by=1,all=T)
Mer.1s=Mer.new
######
#-#print(paste(f2br[i2],paste(Dum.0$date.s,collapse=","),Dum.0$REV,sep=","))
#-#print(Params.1s);print(colnames(Dum.1)[2:ncol(Dum.1)])
#-#print(Units.1s);print(c("s",Dum.0$VUNITS ))
Special.comments.1s=c(Special.comments.1s,paste(f2br[i2],paste(Dum.0$date.s,collapse
=","),Dum.0$REV,sep=",") )
Params.1s=c(Params.1s, colnames(Dum.1)[2:ncol(Dum.1)] )
Units.1s=c(Units.1s,"s",Dum.0$VUNITS )
#
}

##add profiles if they have been determined
if(file.exists(paste("prof",flt,"txt",sep=".")) ) {
prof.nums=read.table(paste("prof",flt,"txt",sep="."),header=T)
prof.no=rep(0,nrow(Mer.1s))
for(k in 1:nrow(Mer.1s)){ prof.no[ Mer.1s[,1]>prof.nums[k,"pr.start"]&Mer.1s[,1]<pro
f.nums[k,"pr.end"] ] = k}
prof.no[ prof.no == 0 & Mer.1s[,"G_ALT"] ] = -1  ## 0, level at alt, -1, level at bo
ttom
Mer.1s=cbind(Mer.1s,prof.no)
```

```
Units.1s = c(Units.1s,"NONE")
Params.1s=c(Params.1s,"prof.no")
Special.comments.1s=c(Special.comments.1s,"Profile Number (>0) ; high level leg = 0
; low level leg = -1")
}
#### distances will be added later

########################################################################################
##############
####                        START 10s merge                        ################
##############
########################################################################################
##############
## note the file types and parameters for 10s include TOGA and SAGA with 30s samplin
g times
###    USE midpoints for TOGA and SAGA [  NOt DONE YET ]
#
for(i2 in 1:length(f2br) ){
print(paste("Reading:",f2br[i2]) )
Dum.0 = read.1001(f2br[i2] )
Dum=Dum.0$dat
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
#                  use the time stamp for start of the interval, except for averaging
 measurements
if(! F2br[i2,1]%in%c("TOGA","DOSE","SAGA-MC","SAGA-AERO")) { j.UTC = 1 } else {j.UTC
 =3 }  ## Mid UTC (various names)
l.ok=Dum[,j.UTC ]> UTC[1] & Dum[,j.UTC ]< UTC[length(UTC)]
Dum=Dum[l.ok,]
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
## handle ULOD and LLOD
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f2br[i2]),intern
=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f2br[i2]),intern
=T),":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
if(is.null(dim(Dum)))Dum=t(as.matrix(Dum.0$dat))  ## trap error for 1 line of  null
data in the file
    if(substring(f2br[i2],1,4) == "DOSE"){
        Dum[,j.UTC ]=Dum[,"Mid_UTC"]  ## a kluge, treating the midpoint as the start
point for simple merging
        }
   t1=trunc(Dum[,j.UTC ]/10)*10
if(length(t1)<5) next
      if(! F2br[i2,1]%in%c("TOGA","DOSE","SAGA-MC","SAGA-AERO")) {           ##do n
ot do the aggregate step for these
   bb=unique(c(t1,t1[length(t1)]+10)) ## breaks for cut ()
   INDEX.start=bb[as.numeric(cut(Dum[,j.UTC ],right=F,breaks=bb))]  ## INDEX for eac
h 10s interval, labelled at START
   Dum.1=aggregate(Dum,by=list(INDEX.start),mean,na.rm=T)
   colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
     } else {
t1 = round (Dum[,j.UTC],-1)   ##note rounding here, giving the max overalp with the
interval it will round to
#Dum.1 = Dum
#Dum.1[,1]=t1
Dum.1 = cbind(t1,Dum)   ## t1 replaces "Group.1"
   colnames(Dum.1)[2]=paste("UTC_Start",F2br[i2,1],sep=".")
    }
# -----------   NAMES    NAMES    NAMES    NAMES `
#### Name alterations
if(F2br[i2,1] == "DOSE")colnames(Dum.1)[2:ncol(Dum.1)]=paste(colnames(Dum.1)[2:ncol(
Dum.1)],"DOSE",sep=".")
##fix colnames for TOGA
if(F2br[i2,1]=="TOGA"){
##check if names changed, if not, append _TOGA
n7 = nchar(colnames(Dum.1) )
if(substring(colnames(Dum.1)[4],n7[4]-3) != "TOGA") {
  colnames(Dum.1)[2:ncol(Dum.1)]=paste(colnames(Dum.1)[2:ncol(Dum.1)],"TOGA",sep="_"
)}
}
## UCATS
```

```r
if(F2br[i2,1]=="UCATS-GC"){
print("UCATS name check, 1s")
##check if names changed, if not, append _UCATS
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:nco
l(Dum.1)],"UCATS",sep="_")
     } else {
if(N7[[4]][2]!="UCATS"){                          #do nothing if it is already _UCATS
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]      ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"UCATS",sep="_")
                       }
}
}

## PANTHER
if(substring(F2br[i2,1],1,4)=="GCEC" & YYYY == "2016"){
print(" GCEX fix")
##check if names changed, if not, append _GCECD
N7=str_split(colnames(Dum.1),pattern="_")
##case nothing appended
if(is.na(N7[[4]][2])) { colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:nco
l(Dum.1)],"GCECD",sep="_")
     } else {
if(N7[[4]][2]!="GCECD"){                          #do nothing if it is already _GCECD
cx5=matrix(unlist(N7),ncol=2,byrow=T)[,1]      ## first element of each list
colnames(Dum.1)[2:ncol(Dum.1)] = paste(cx5,"GCECD",sep="_")
                       }
}
}

## GEOS5   avoid dup variables
if(substring(F2br[i2,1],1,5)=="GEOS5" ){
### append _GEOS5 to names
print("GEOS5 file data found!")
N7=str_split(colnames(Dum.1),pattern="_")
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GEOS5",sep="_
")
}

## GMI   avoid dup variables
if(substring(F2br[i2,1],1,3)=="GMI" ){
### append _GMI to names
print("GMI file data found!")
N7=str_split(colnames(Dum.1),pattern="_")
colnames(Dum.1)[2:ncol(Dum.1)] = paste(colnames(Dum.1)[2:ncol(Dum.1)],"GMI",sep="_")

}

## end name alterations
l.AMSSD=F
## AMSSD
if(substring(F2br[i2,1],1,5)=="AMSSD" ){
l.AMSSD=T
###1st 12 columns only
print("AMSSD check")
Dum.1=Dum.1[,1:13]
Dum.0$VUNITS=Dum.0$VUNITS[1:11]  ## this count is a mystery.....
}


# ----------- NAMES NAMES      NAMES        NAMES
#
Mer.new=merge(Mer,Dum.1,by=1,all=T)
ALLNAMES=rbind(ALLNAMES, data.frame(params=colnames(Dum.1)[2:ncol(Dum.1)],data.set=r
ep(f2br[i2],ncol(Dum.1)-1) ) )
Mer=Mer.new
###
Special.comments=c(Special.comments,paste(f2br[i2],paste(Dum.0$date.s,collapse=","),
Dum.0$REV,sep=",") )
Params=c(Params, colnames(Dum.1)[2:ncol(Dum.1)] )
Units=c(Units,"s",Dum.0$VUNITS )
#Units=c(Units,"s",Dum.0$VUNITS[1:(ncol(Dum.1)-1)] )
```

```
#
}

#fix colnames
n.dup=which(substring(colnames(Mer),1,8)=="Stop_UTC")
Start.dup=colnames(Mer)[n.dup-1]
Stop.dup=gsub(pattern="Start",replacement="Stop",Start.dup)
colnames(Mer)[n.dup] = Stop.dup

##add profiles if they have been determined
if(file.exists(paste("prof",flt,"txt",sep=".")) ) {
prof.nums=read.table(paste("prof",flt,"txt",sep="."),header=T)
prof.no=rep(0,nrow(Mer))
for(k in 1:nrow(Mer)){ prof.no[ Mer[,1]>prof.nums[k,"pr.start"]&Mer[,1]<prof.nums[k,
"pr.end"] ] = k}
prof.no[ prof.no == 0 & Mer[,"G_ALT"] ] = -1  ## 0, level at alt, -1, level at botto
m
Mer=cbind(Mer,prof.no)
Units = c(Units,"NONE")
Params=c(Params,"prof.no")
Special.comments=c(Special.comments,"Profile Number (>0) ; high level leg = 0 ; low
level leg = -1")
ALLNAMES=rbind(ALLNAMES,c("prof.no","computed"))
}

##add distance in km since departure -- use file if it exists
if(file.exists("Dists10.txt")){Dist=scan("Dists10.txt")} else {
require(sp)
d0=0
#
for(k in 2:nrow(Dum.mms)){
d0=c(d0,spDistsN1(pts=matrix(c(Dum.mms[k,"G_LONG"],Dum.mms[k,"G_LAT"]),ncol=2,nrow=1
,byrow=T),
     pt=c(Dum.mms[k-1,"G_LONG"],Dum.mms[k-1,"G_LAT"]),longlat=T) )
}
dd=tapply(d0,INDEX.start.mms,sum)
write(d0,file="d0.all.txt")
Dist=cumsum(dd)
print("Writing Dists10.txt")
write(Dist,file="Dists10.txt")
}
Mer=cbind(Mer,Dist)
Units = c(Units,"km")
Params=c(Params,"Dist")
Special.comments=c(Special.comments,"Distance in km along the flight track")
ALLNAMES=rbind(ALLNAMES,c("Dist","computed"))

#########################################################
##add Dists to Mer.1s by interpolating in the 10s file.#
if(file.exists("Dists1.txt")){                          #
                                                        #
Dist=scan("Dists1.txt")} else {                         #
d0 = scan("d0.all.txt") # file created in 10s loop      # May not have exact 1s time
stamp of UTC_1s      #
#                                                       #
dd=tapply(d0,UTC_Start.1s,sum)                          #
if(length(dd)>=nrow(Mer.1s)) {dd=dd[1:nrow(Mer.1s)]     #
  } else {                                              #
dd=c(dd,rep(0,nrow(Mer.1s)-length(dd)) ) }              #
Dist=cumsum(dd)                                         #
print("Writing Dists1.txt")                             #
write(Dist,file="Dists1.txt")                           #
}                                                       #
                                                        #
Mer.1s=cbind(Mer.1s,Dist)                               #
                                                        #
Units.1s = c(Units.1s,"km")                             #
Params.1s=c(Params.1s,"Dist")                           #____________________________
#
Special.comments.1s=c(Special.comments.1s,"Distance in km along the flight track")
#
####    end of adding distance to 1s file               #----------------------------
#
```

```
##########################################################
# write out allnames
write.table(ALLNAMES,row.names=F,col.names=T,quote=F,file="ALLNAMES.txt")

##write out ICARTT files
### remove redundant UTC columns  1s
lUTC.1s=1:(ncol(Mer.1s) )%in%grep(pattern="UTC",x=colnames(Mer.1s))
lUTC.1s=lUTC.1s[2:ncol(Mer.1s)]
lUTC.1s[1]=F    ## this is UTC_Stop; all the others containing "UTC" will be removed

l.UTC.1s= !lUTC.1s
l.keep.1s=unlist(str_split(paste("UTC_Start",paste(Params.1s[l.UTC.1s],collapse=",")
,sep=","),pattern=","))
colnames(Mer.1s)[2]=l.keep.1s[2]

lUTC=1:(ncol(Mer) )%in%grep(pattern="UTC",x=colnames(Mer))
lUTC=lUTC[2:ncol(Mer)]
lUTC[1]=F    ## this is UTC_Stop; all the others containing "UTC" will be removed
l.UTC= !lUTC
l.keep=unlist(str_split(paste("UTC_Start",paste(Params[l.UTC],collapse=","),sep=",")
,pattern=","))
colnames(Mer)[2]=l.keep[2] ;
##
## flat tables --
write.table(Mer.1s,file="Mer.1s.tbl",row.names=F,col.names=T,quote=F)
write.table(Mer,file="Mer.tbl",row.names=F,col.names=T,quote=F)
system("gzip -f Mer.tbl Mer.1s.tbl")
## end flat tables

## construct the 10s ICARTT file
Normal.Comments=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
## deal with revision number
if(file.exists("rev_curr.txt") ) {            ## case REV is specified
REV=scan("rev_curr.txt")
N.C=Normal.Comments[1:(length(Normal.Comments) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments=c(N.C,N.C.addon)
}
##
Normal.Comments=c(Normal.Comments,paste("UTC_Start",paste(Params[l.UTC],collapse=","
),sep=","))
N.Normal.Comments=length(Normal.Comments)  ##add line of parameters
NV=length(Params[(l.UTC)])
Scale.Factors=paste(rep(1,NV),collapse=",")
Missing.Values=paste(rep(-99999,NV),collapse=",")
Var.names.units=paste(Params[l.UTC],Units[l.UTC],sep=", ")
NLINES=14+NV+length(Special.comments)+(N.Normal.Comments)
Params.units=Params[l.UTC] ; names(Params.units) = Units[l.UTC]  ## note: missing UT
C_Start


##write the 10 s file
write(paste(NLINES,"1001",sep=", "),file="Mer10.icartt")
write(c("Wofsy,S", "Harvard_University", "10 second merge of ATom DC-8 data", "ATom"
, "1,1"),
ncol=1,file="Mer10.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="Mer10
.icartt",append=T)
write("10",file="Mer10.icartt",append=T)
write("UTC_Start, s",file="Mer10.icartt",append=T)
write(NV ,file="Mer10.icartt",append=T)
write(Scale.Factors ,file="Mer10.icartt",append=T)
write(Missing.Values ,file="Mer10.icartt",append=T)
write(Var.names.units ,file="Mer10.icartt",ncol=1,append=T)
write(length(Special.comments),file="Mer10.icartt",append=T)
write(Special.comments,file="Mer10.icartt",ncol=1,append=T)
write(N.Normal.Comments,file="Mer10.icartt",append=T)
write(Normal.Comments,file="Mer10.icartt",ncol=1,append=T)
write.table(Mer[,l.keep],file="Mer10.icartt", append=T,row.names=F,col.names=F,sep="
,",na="-99999",quote=F)
if(file.exists("rev_curr.txt") ) {            ## case REV is specified
```

```
system(paste("cp Mer10.icartt MER10_DC8_",flt,"_R",REV+1,".ict",sep=""))
                                    }
write.table(ALLNAMES[l.keep,],row.names=F,col.names=T,quote=F,file="ALLNAMES_keep.tx
t")
###########   start write 1s merge file
Normal.Comments.1s=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
if(file.exists("rev_curr.txt") ) {           ## case REV is specified
REV=scan("rev_curr.txt")
N.C=Normal.Comments.1s[1:(length(Normal.Comments.1s) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.1s=c(N.C,N.C.addon)
}
Normal.Comments.1s=c(Normal.Comments.1s,paste("UTC_Start",paste(Params.1s[l.UTC.1s],
collapse=","),sep=","))
#
N.Normal.Comments.1s=length(Normal.Comments.1s)  ##add line of parameters
NV.1s=length(Params.1s[(l.UTC.1s)])
Scale.Factors.1s=paste(rep(1,NV.1s),collapse=",")
Missing.Values.1s=paste(rep(-99999,NV.1s),collapse=",")
Var.names.units.1s=paste(Params.1s[l.UTC.1s],Units.1s[l.UTC.1s],sep=", ")
NLINES.1s=14+NV.1s+length(Special.comments.1s)+(N.Normal.Comments.1s)
Params.units.1s=Params.1s[l.UTC.1s] ; names(Params.units.1s) = Units.1s[l.UTC.1s]

write(paste(NLINES.1s,"1001",sep=", "),file="Mer01.icartt")
write(c("Wofsy,S", "Harvard_University", "1 second merge of ATom DC-8 data", "ATom",
 "1,1"),
ncol=1,file="Mer01.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="Mer01
.icartt",append=T)
write("1",file="Mer01.icartt",append=T)
write("UTC_Start, s",file="Mer01.icartt",append=T)
write(NV.1s ,file="Mer01.icartt",append=T)
write(Scale.Factors.1s ,file="Mer01.icartt",append=T)
write(Missing.Values.1s ,file="Mer01.icartt",append=T)
write(Var.names.units.1s ,file="Mer01.icartt",ncol=1,append=T)
write(length(Special.comments.1s),file="Mer01.icartt",append=T)
write(Special.comments.1s,file="Mer01.icartt",ncol=1,append=T)
write(N.Normal.Comments.1s,file="Mer01.icartt",append=T)
write(Normal.Comments.1s,file="Mer01.icartt",ncol=1,append=T)
write.table(Mer.1s[,l.keep.1s],file="Mer01.icartt", append=T,row.names=F,col.names=F
,sep=",",na="-99999",quote=F)
if(file.exists("rev_curr.txt") ) {           ## case REV is specified
system(paste("cp Mer01.icartt MER-1HZ_DC8_",flt,"_R",REV+1,".ict",sep=""))}

###########  end write 1s merge file

##Start: merge for WAS ###############################################################
############################
##merge for WAS
##merge for WAS
file.WAS=system(paste("ls -1 WAS_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.WAS)>0) {    ##proceed with WAS
f.WAS=file.WAS[length(file.WAS)] ## take the last revision
Dum.0 = read.1001(f.WAS )
Dum=Dum.0$dat
#                 use the time stamp for start of the interval, except for averaging
 measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames which have "_ppt" at the end.  THIS SECTION TO CHANGE IF WAS FILE
S CHANGE
nom.cols=matrix(unlist(str_split(colnames(Dum),pattern="_")),ncol=2,byrow=T)
colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum),1],"WAS",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_WAS","UTC_Stop_WAS","UTC_Mid_WAS")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.WAS),intern=T)
,":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.WAS),intern=T)
,":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#
```

```
was.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
   was.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,"UTC_Start_WAS"] & Mer.1s[,"UTC_Start"]<=Dum
[i,"UTC_Stop_WAS"]]=i
                          }
#was.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:(
2*nrow(Dum) - 1))) )
#was.cuts[was.cuts%%2==0] = NA
AA=as.matrix ( aggregate(Mer.1s,by=list(was.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[was.cuts!=0,"prof.no"],list(was.cuts[was.cuts!=0]),max,na.rm=T)
)
AA[,"prof.no"]=prf
lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.WAS = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.WAS=c(Special.comments.1s,paste(f.WAS ,paste(Dum.0$date.s,collapse=
","),Dum.0$REV,sep=",") )
# Params.WAS=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.WAS=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.WAS=colnames(Mer.WAS)[2:ncol(Mer.WAS)]
Units.WAS=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ]
 )
## do the averaging for each flask


###########   start write WAS merge file
Normal.Comments.WAS=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
if(file.exists("rev_curr_WAS.txt") ) {            ## case REV is specified
REV=scan("rev_curr_WAS.txt")
N.C=Normal.Comments.WAS[1:(length(Normal.Comments.WAS) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.WAS=c(N.C,N.C.addon)
}
Normal.Comments.WAS=c(Normal.Comments.WAS,paste("UTC_Start",paste(Params.WAS,collaps
e=","),sep=","))
N.Normal.Comments.WAS=length(Normal.Comments.WAS)  ##add line of parameters
NV.WAS=length(Params.WAS )
Scale.Factors.WAS=paste(rep(1,NV.WAS),collapse=",")
Missing.Values.WAS=paste(rep(-99999,NV.WAS),collapse=",")
Var.names.units.WAS=paste(Params.WAS,Units.WAS,sep=",  ")
NLINES.WAS=14+NV.WAS+length(Special.comments.WAS)+(N.Normal.Comments.WAS)

write(paste(NLINES.WAS,"1001",sep=", "),file="MER-WAS.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with WAS flask
samples", "ATom", "1,1"),
ncol=1,file="MER-WAS.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-W
AS.icartt",append=T)
write("0",file="MER-WAS.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-WAS.icartt",append=T)
write(NV.WAS ,file="MER-WAS.icartt",append=T)
write(Scale.Factors.WAS ,file="MER-WAS.icartt",append=T)
write(Missing.Values.WAS ,file="MER-WAS.icartt",append=T)
write(Var.names.units.WAS ,file="MER-WAS.icartt",ncol=1,append=T)
write(length(Special.comments.WAS),file="MER-WAS.icartt",append=T)
write(Special.comments.WAS,file="MER-WAS.icartt",ncol=1,append=T)
write(N.Normal.Comments.WAS,file="MER-WAS.icartt",append=T)
write(Normal.Comments.WAS,file="MER-WAS.icartt",ncol=1,append=T)
write.table(Mer.WAS,file="MER-WAS.icartt", append=T,row.names=F,col.names=F,sep=",",
na="-99999",quote=F)
#
if(file.exists("rev_curr_WAS.txt") ) {            ## case REV is specified
system(paste("cp MER-WAS.icartt MER-WAS_DC8_",flt,"_R",REV+1,".ict",sep=""))}

}
```

```
##End: merge for WAS ################################################################
##########################
#
##Start: merge for PFP ##############################################################
############################

file.PFP=system(paste("ls -1 PFP_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.PFP)>0) {    ##proceed with PFP
f.PFP=file.PFP[length(file.PFP)] ## take the last revision
Dum.0 = read.1001(f.PFP )
Dum=Dum.0$dat
#                  use the time stamp for start of the interval, except for averaging
 measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## PFP's do not have a mid time
UTC_Mid=(Dum[,1] + Dum[,2])/2
Dum=cbind(Dum[,1:2],UTC_Mid,Dum[,3:ncol(Dum)] )
nom.cols=sapply(str_split(colnames(Dum),pattern="_"),function(x)return(x[1]))
colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum)],"PFP",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_PFP","UTC_Stop_PFP","UTC_Mid_PFP")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.PFP),intern=T)
,":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.PFP),intern=T)
,":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#
#pfp.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:(
2*nrow(Dum) - 1))) )
#pfp.cuts[pfp.cuts%%2==0] = NA
pfp.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
   pfp.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2] ] =i
                          }
AA=as.matrix ( aggregate(Mer.1s,by=list(pfp.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[pfp.cuts!=0,"prof.no"],list(pfp.cuts[pfp.cuts!=0]),max,na.rm=T)
)
AA[,"prof.no"]=prf

lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.PFP = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.PFP=c(Special.comments.1s,paste(f.PFP ,paste(Dum.0$date.s,collapse=
",") ,Dum.0$REV,sep=",") )
# Params.PFP=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.PFP=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.PFP=colnames(Mer.PFP)[2:ncol(Mer.PFP)]
Units.PFP=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1) ]
 )
## do the averaging for each flask


###########   start write PFP merge file
Normal.Comments.PFP=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
if(file.exists("rev_curr_PFP.txt") ) {           ## case REV is specified
REV=scan("rev_curr_PFP.txt")
N.C=Normal.Comments.PFP[1:(length(Normal.Comments.PFP) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.PFP=c(N.C,N.C.addon)
}
Normal.Comments.PFP=c(Normal.Comments.PFP,paste("UTC_Start",paste(Params.PFP,collaps
e=","),sep=","))
N.Normal.Comments.PFP=length(Normal.Comments.PFP)  ##add line of parameters
```

```
NV.PFP=length(Params.PFP )
Scale.Factors.PFP=paste(rep(1,NV.PFP),collapse=",")
Missing.Values.PFP=paste(rep(-99999,NV.PFP),collapse=",")
Var.names.units.PFP=paste(Params.PFP,Units.PFP,sep=", ")
NLINES.PFP=14+NV.PFP+length(Special.comments.PFP)+(N.Normal.Comments.PFP)

write(paste(NLINES.PFP,"1001",sep=", "),file="MER-PFP.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with PFP flask
samples", "ATom", "1,1"),
ncol=1,file="MER-PFP.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-P
FP.icartt",append=T)
write("0",file="MER-PFP.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-PFP.icartt",append=T)
write(NV.PFP ,file="MER-PFP.icartt",append=T)
write(Scale.Factors.PFP ,file="MER-PFP.icartt",append=T)
write(Missing.Values.PFP ,file="MER-PFP.icartt",append=T)
write(Var.names.units.PFP ,file="MER-PFP.icartt",ncol=1,append=T)
write(length(Special.comments.PFP),file="MER-PFP.icartt",append=T)
write(Special.comments.PFP,file="MER-PFP.icartt",ncol=1,append=T)
write(N.Normal.Comments.PFP,file="MER-PFP.icartt",append=T)
write(Normal.Comments.PFP,file="MER-PFP.icartt",ncol=1,append=T)
write.table(Mer.PFP,file="MER-PFP.icartt", append=T,row.names=F,col.names=F,sep=",",
na="-99999",quote=F)
#
if(file.exists("rev_curr_PFP.txt") ) {           ## case REV is specified
system(paste("cp MER-PFP.icartt MER-PFP_DC8_",flt,"_R",REV+1,".ict",sep=""))}
}
##End: merge for PFP ###################################################################
##########################
# ************************
### Start Merge for SAGA-AERO
file.SAGA.AERO=system(paste("ls -1 SAGA-AERO_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.SAGA.AERO)>0) {     ##proceed with SAGA-AERO
f.SAGA.AERO=file.SAGA.AERO[length(file.SAGA.AERO)] ## take the last revision
Dum.0 = read.1001(f.SAGA.AERO )
Dum=Dum.0$dat
#                 use the time stamp for start of the interval, except for averaging
 measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## SAGA-AERO's do have a mid time
#UTC_Mid=(Dum[,1] + Dum[,2])/2
#Dum=cbind(Dum[,1:2],UTC_Mid,Dum[,3:ncol(Dum)] )
nom.cols=sapply(str_split(colnames(Dum),pattern="_"),function(x)return(x[1]))
colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum)],"SAGA.AERO",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_SAGA.AERO","UTC_Stop_SAGA.AERO","UTC_Mid_SAGA.AERO")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.SAGA.AERO),int
ern=T),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.SAGA.AERO),int
ern=T),":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#
bkk=c(t(Dum[,1:2]))
 brk=bkk
 brk[duplicated(bkk)]=bkk[duplicated(bkk)]+1

#sga.cuts=as.numeric( cut(Mer.1s[,1],breaks=brk ,labels=as.character(1:(length(brk)
- 1))))
#sga.cuts[sga.cuts%%2==0] = NA
sga.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
   sga.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[i,2] ] =i
                          }
AA=as.matrix ( aggregate(Mer.1s,by=list(sga.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[sga.cuts!=0,"prof.no"],list(sga.cuts[sga.cuts!=0]),max,na.rm=T)
)
AA[,"prof.no"]=prf

lU=grep(pattern="UTC",colnames(AA))
```

```
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.SAGA.AERO = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.SAGA.AERO=c(Special.comments.1s,paste(f.SAGA.AERO ,paste(Dum.0$date
.s,collapse=","),Dum.0$REV,sep=",") )
# Params.SAGA.AERO=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.SAGA.AERO=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.SAGA.AERO=colnames(Mer.SAGA.AERO)[2:ncol(Mer.SAGA.AERO)]
Units.SAGA.AERO=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum
)-1) ] )
## do the averaging for each flask


###########   start write SAGA-AERO merge file
Normal.Comments.SAGA.AERO=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt
")
if(file.exists("rev_curr_SAGA.AERO.txt") ) {           ## case REV is specified
REV=scan("rev_curr_SAGA.AERO.txt")
N.C=Normal.Comments.SAGA.AERO[1:(length(Normal.Comments.SAGA.AERO) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.SAGA.AERO=c(N.C,N.C.addon)
}
Normal.Comments.SAGA.AERO=c(Normal.Comments.SAGA.AERO,paste("UTC_Start",paste(Params
.SAGA.AERO,collapse=","),sep=","))
N.Normal.Comments.SAGA.AERO=length(Normal.Comments.SAGA.AERO)  ##add line of paramet
ers
NV.SAGA.AERO=length(Params.SAGA.AERO )
Scale.Factors.SAGA.AERO=paste(rep(1,NV.SAGA.AERO),collapse=",")
Missing.Values.SAGA.AERO=paste(rep(-99999,NV.SAGA.AERO),collapse=",")
Var.names.units.SAGA.AERO=paste(Params.SAGA.AERO,Units.SAGA.AERO,sep=", ")
NLINES.SAGA.AERO=14+NV.SAGA.AERO+length(Special.comments.SAGA.AERO)+(N.Normal.Commen
ts.SAGA.AERO)

write(paste(NLINES.SAGA.AERO,"1001",sep=", "),file="MER-SAGA.AERO.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with SAGA.AERO
flask samples", "ATom", "1,1"),
ncol=1,file="MER-SAGA.AERO.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-S
AGA.AERO.icartt",append=T)
write("0",file="MER-SAGA.AERO.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-SAGA.AERO.icartt",append=T)
write(NV.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Scale.Factors.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Missing.Values.SAGA.AERO ,file="MER-SAGA.AERO.icartt",append=T)
write(Var.names.units.SAGA.AERO ,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write(length(Special.comments.SAGA.AERO),file="MER-SAGA.AERO.icartt",append=T)
write(Special.comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write(N.Normal.Comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",append=T)
write(Normal.Comments.SAGA.AERO,file="MER-SAGA.AERO.icartt",ncol=1,append=T)
write.table(Mer.SAGA.AERO,file="MER-SAGA.AERO.icartt", append=T,row.names=F,col.name
s=F,sep=",",na="-99999",quote=F)
#
if(file.exists("rev_curr_SAGA.AERO.txt") ) {           ## case REV is specified
system(paste("cp MER-SAGA.AERO.icartt MER-SAGA-AERO_DC8_",flt,"_R",REV+1,".ict",sep=
""))}
}
##End: merge for SAGA-AERO ##########################################################
##################################
### end Merge for SAGA-AERO

##Start: merge for MED ##############################################################
##############################

file.MED=system(paste("ls -1 MEDUSA_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.MED)>0) {    ##proceed with MED
f.MED=file.MED[length(file.MED)] ## take the last revision
Dum.x = merge.medusa.atom(flt) ## this function does the full merge with filling ker
```

```
nel
Dum=Dum.x[,2:ncol(Dum.x)]
Dum.0 = read.1001(f.MED ) ## we read the file to get ICARTT info
#                  use the time stamp for start of the interval, except for averaging
 measurements
#########################  SKIP ALL the MERGING here !
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames
## MED's do have a mid time
colnames(Dum)[1:4]=c("UTC_Start","UTC_Stop_MED","UTC_Mid_MED","UTC_Mean_MED")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.MED),intern=T)
,":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.MED),intern=T)
,":"))[2])
#Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#

prof.no=rep(0,nrow(Dum))
for(i in 1:nrow(Dum)){
   prof.no[i]=max(Mer.1s[Mer.1s[,"UTC_Start"]>=Dum[i,1] & Mer.1s[,"UTC_Start"]<=Dum[
i,2] ,"prof.no"])
                                }
AA=cbind(Dum,prof.no)

#Mer.MED = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
Mer.MED = AA
#
##                                                         v       v
          v
Special.comments.MED=c(Special.comments.1s,paste(f.MED ,paste(Dum.0$date.s,collapse=
","),Dum.0$REV,sep=",") )
Params.MED=colnames(Mer.MED)[2:ncol(Mer.MED)]
#Units.MED=c("s","s","s",Dum.0$VUNITS[3:(ncol(Dum)-1) ] )
Units.MED=c("s","s","s",Dum.0$VUNITS[4:(length(Dum.0$VUNITS)) ],"ppm","s","hPa","K",
"K","m/s","deg","deg","m","hPa","hPa","ppm","permil","ppm","ppm","ppm","ppb",
            "ppm","ppm","ppb","ppb","ppb","ppm","ppb","ppb","ppt","km","None","None"
,rep("None",26),"ppm","None")
names(Units.MED)=Params.MED
## do the averaging for each flask


###########   start write MED merge file
Normal.Comments.MED=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
if(file.exists("rev_curr_MED.txt") ) {           ## case REV is specified
REV=scan("rev_curr_MED.txt")
N.C=Normal.Comments.MED[1:(length(Normal.Comments.MED) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.MED=c(N.C,N.C.addon)
}
Normal.Comments.MED=c(Normal.Comments.MED,paste("UTC_Start",paste(Params.MED,collaps
e=","),sep=","))
N.Normal.Comments.MED=length(Normal.Comments.MED)  ##add line of parameters
NV.MED=length(Params.MED )
Scale.Factors.MED=paste(rep(1,NV.MED),collapse=",")
Missing.Values.MED=paste(rep(-99999,NV.MED),collapse=",")
Var.names.units.MED=paste(Params.MED,Units.MED,sep=", ")
NLINES.MED=14+NV.MED+length(Special.comments.MED)+(N.Normal.Comments.MED)

write(paste(NLINES.MED,"1001",sep=", "),file="MER-MED.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with MED flask
samples", "ATom", "1,1"),
ncol=1,file="MER-MED.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-M
ED.icartt",append=T)
write("0",file="MER-MED.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-MED.icartt",append=T)
write(NV.MED ,file="MER-MED.icartt",append=T)
write(Scale.Factors.MED ,file="MER-MED.icartt",append=T)
write(Missing.Values.MED ,file="MER-MED.icartt",append=T)
write(Var.names.units.MED ,file="MER-MED.icartt",ncol=1,append=T)
write(length(Special.comments.MED),file="MER-MED.icartt",append=T)
write(Special.comments.MED,file="MER-MED.icartt",ncol=1,append=T)
write(N.Normal.Comments.MED,file="MER-MED.icartt",append=T)
write(Normal.Comments.MED,file="MER-MED.icartt",ncol=1,append=T)
write.table(Mer.MED,file="MER-MED.icartt", append=T,row.names=F,col.names=F,sep=",",
na="-99999",quote=F)
#
if(file.exists("rev_curr_MED.txt") ) {           ## case REV is specified
system(paste("cp MER-MED.icartt MER-MED_DC8_",flt,"_R",REV+1,".ict",sep=""))}
}
##End: merge for MED #############################################################
##########################
##
##Start: merge for TOGA ##########################################################
#############################
##merge for TOGA
##merge for TOGA
file.TOGA=system(paste("ls -1 TOGA_DC8_",flt,"_R?.ict",sep=""),intern=T)
if(length(file.TOGA)>0) {    ##proceed with TOGA
f.TOGA=file.TOGA[length(file.TOGA)] ## take the last revision
Dum.0 = read.1001(f.TOGA )
Dum=Dum.0$dat
#                  use the time stamp for start of the interval, except for averaging
 measurements
l.ok=Dum[, 1 ]> Mer.1s[1,1] & Dum[,1 ]< Mer.1s[nrow(Mer.1s),1]
Dum=Dum[l.ok,]
##fix the colnames which have "_ppt" at the end.  THIS SECTION TO CHANGE IF TOGA FIL
ES CHANGE
#nom.cols=matrix(unlist(str_split(colnames(Dum),pattern="_")),ncol=2,byrow=T)
#colnames(Dum)[4:ncol(Dum)]=paste(nom.cols[4:ncol(Dum),1],"TOGA",sep="_")
colnames(Dum)[1:3]=c("UTC_Start_TOGA","UTC_Stop_TOGA","UTC_Mid_TOGA")
#
LLOD_FLAG=as.numeric(unlist(str_split(system(paste("grep LLOD_FLAG",f.TOGA),intern=T
),":"))[2])
ULOD_FLAG=as.numeric(unlist(str_split(system(paste("grep ULOD_FLAG",f.TOGA),intern=T
),":"))[2])
Dum[Dum == LLOD_FLAG | Dum == ULOD_FLAG] =NA
#
toga.cuts=rep(0,nrow(Mer.1s))
for(i in 1:nrow(Dum)){
   toga.cuts[Mer.1s[,"UTC_Start"]>=Dum[i,"UTC_Start_TOGA"] & Mer.1s[,"UTC_Start"]<=D
um[i,"UTC_Stop_TOGA"]]=i
                              }
#toga.cuts=as.numeric( cut(Mer.1s[,1],breaks=c(t(Dum[,1:2])) ,labels=as.character(1:
(2*nrow(Dum) - 1))) )
#toga.cuts[toga.cuts%%2==0] = NA
AA=as.matrix ( aggregate(Mer.1s,by=list(toga.cuts),mean,na.rm=T) )
AA=AA[AA[,1]!=0,]
prf=( tapply(Mer.1s[toga.cuts!=0,"prof.no"],list(toga.cuts[toga.cuts!=0]),max,na.rm=
T) )
AA[,"prof.no"]=prf
lU=grep(pattern="UTC",colnames(AA))
lU=lU[2:length(lU)]
lkk=rep(T,ncol(AA))
lkk[lU]=F
AB=AA[,lkk]
colnames(AB)[2]="UTC_Mean_1s"
Mer.TOGA = cbind(Dum[,1:3],AB[,2:ncol(AB)],Dum[,4:ncol(Dum)])
#
Special.comments.TOGA=c(Special.comments.1s,paste(f.TOGA ,paste(Dum.0$date.s,collaps
e=","),Dum.0$REV,sep=",") )
# Params.TOGA=c(Params.1s, colnames(Dum)[4:ncol(Dum)] )
# Units.TOGA=c(Units.1s,Dum.0$VUNITS[4:ncol(Dum)] )
Params.TOGA=colnames(Mer.TOGA)[2:ncol(Mer.TOGA)]
Units.TOGA=c("s","s","s",Units.1s[lkk[3:length(lkk)] ],Dum.0$VUNITS[3:(ncol(Dum)-1)
] )
## do the averaging for each flask


###########   start write TOGA merge file
Normal.Comments.TOGA=readLines("/documents/DATA/ATOM/RPROGS/Normal_Comments.txt")
if(file.exists("rev_curr_TOGA.txt") ) {           ## case REV is specified
REV=scan("rev_curr_TOGA.txt")
N.C=Normal.Comments.TOGA[1:(length(Normal.Comments.TOGA) -1 ) ]
N.C.addon=paste("REVISION: R",REV+1,sep="")
if(REV>=0) { for(i in 1:(REV+1)) { N.C.addon= c( N.C.addon , paste("R",i,": updated"
, sep="")) } }  ## creating next rev
Normal.Comments.TOGA=c(N.C,N.C.addon)
}
Normal.Comments.TOGA=c(Normal.Comments.TOGA,paste("UTC_Start",paste(Params.TOGA,coll
apse=","),sep=","))
N.Normal.Comments.TOGA=length(Normal.Comments.TOGA)  ##add line of parameters
NV.TOGA=length(Params.TOGA )
Scale.Factors.TOGA=paste(rep(1,NV.TOGA),collapse=",")
Missing.Values.TOGA=paste(rep(-99999,NV.TOGA),collapse=",")
Var.names.units.TOGA=paste(Params.TOGA,Units.TOGA,sep=", ")
NLINES.TOGA=14+NV.TOGA+length(Special.comments.TOGA)+(N.Normal.Comments.TOGA)

write(paste(NLINES.TOGA,"1001",sep=", "),file="MER-TOGA.icartt")
write(c("Wofsy,S", "Harvard_University", "merge of 1s ATom DC-8 data with TOGA flask
 samples", "ATom", "1,1"),
ncol=1,file="MER-TOGA.icartt",append=T)
write(paste(YYYY, MM,DD, format(Sys.time(),tz="GMT","%Y,%m,%d"),sep=","),file="MER-T
OGA.icartt",append=T)
write("0",file="MER-TOGA.icartt",append=T) ## interval =0 , not regular apcing
write("UTC_Start, s",file="MER-TOGA.icartt",append=T)
write(NV.TOGA ,file="MER-TOGA.icartt",append=T)
write(Scale.Factors.TOGA ,file="MER-TOGA.icartt",append=T)
write(Missing.Values.TOGA ,file="MER-TOGA.icartt",append=T)
write(Var.names.units.TOGA ,file="MER-TOGA.icartt",ncol=1,append=T)
write(length(Special.comments.TOGA),file="MER-TOGA.icartt",append=T)
write(Special.comments.TOGA,file="MER-TOGA.icartt",ncol=1,append=T)
write(N.Normal.Comments.TOGA,file="MER-TOGA.icartt",append=T)
write(Normal.Comments.TOGA,file="MER-TOGA.icartt",ncol=1,append=T)
write.table(Mer.TOGA,file="MER-TOGA.icartt", append=T,row.names=F,col.names=F,sep=",
",na="-99999",quote=F)
#
if(file.exists("rev_curr_TOGA.txt") ) {           ## case REV is specified
system(paste("cp MER-TOGA.icartt MER-TOGA_DC8_",flt,"_R",REV+1,".ict",sep=""))}

}
##End: merge for TOGA ##################################################################
###########################
```