Supplemental File for "The Niwot Ridge Subalpine Forest US-NR1 AmeriFlux site – Part I: Data acquisition and site record-keeping" *by* S. P. Burns et al.

README for: USNR1_prep.zip

S. P. Burns et al.

sean@ucar.edu

Date: September 8, 2016

This is a README file for the zip archive of the `prep` software that ran on a Dell desktop PC (urquell, linux, 2.6.18-409.el5PAE #1 SMP Fri Feb 12 06:35:09 EST 2016 i686 i686 i386 GNU/Linux) for both ASTER and NIDAS versions of the NCAR EOL data-acquisition software. Several MATLAB script files used to extract the binary data using `prep` are also included. As an alternative to the MATLAB method, there is also support for reading the NIDAS data with R, which is summarized here:

https://www.eol.ucar.edu/software/r-software-analysis-time-series

The most recent NIDAS information (and source code) is available at NCAR EOL and on github at:

https://www.eol.ucar.edu/software_center/nidas

https://github.com/ncareol/nidas/wiki

The files included are:

```
-rw-r--r-- 1 sburns aster 12001280 May 16 00:10 prep_binaries_aster.tar  (see discussion paper)
-rw-r--r-- 1 sburns aster 36802560 May 16 00:09 prep_binaries_nidas.tar  (see discussion paper)
-rw-r--r-- 1 sburns aster  112640  Sep  8 09:11 matlab_mfiles_prep.tar
```

The files, "prep_binaries_aster.tar" and "prep_binaries_nidas.tar" are available from the discussion paper. The MATLAB code to run `prep` and create ASCII data files is in "matlab_mfiles_prep.tar". A short example of how `prep` is used will be provided below. The individual matlab script files are:

```
-rw-rw-r-- 1 sburns aster 23138 Feb  3  2016 data_files_prep.m
-rw-r--r-- 1 sburns aster  3998 Nov  6  2015 info_ch11_ACCEL_mark_r.m
-rw-r--r-- 1 sburns aster  8388 Aug  2 08:27 info_j0_chan200_UC.m
-rw-r--r-- 1 sburns aster 10277 Nov  4  2015 info_j1_chan201_FAST.m
-rw-r--r-- 1 sburns aster  1416 Nov  9  2012 info_j2_chan202_UCB.m
-rw-r--r-- 1 sburns aster  1779 Feb  7  2011 info_j3_chan203_PROF.m
-rw-r--r-- 1 sburns aster  3356 Dec  8  2014 info_j4_chan204_MET.m
-rw-r--r-- 1 sburns aster  1872 Feb  7  2011 info_j5_chan205_RAD.m
-rw-r--r-- 1 sburns aster  1328 Jun  2  2015 info_j7_chan207_NCTC.m
```

```
-rw-r--r-- 1 sburns aster  7989 Nov  4  2015 ops_config.m
-rw-rw-r-- 1 sburns aster  8027 Oct 26  2014 prep_extract_ascii_porter.m
-rw-rw-r-- 1 sburns aster  9893 Sep  6 10:17 prep_extract_ascii_porter_nidas.m
-rw-rw-r-- 1 sburns aster 12170 Aug 29 12:14 prep_extract_ascii_sonics.m
```

To run `prep` correctly, first the following modification (or one like it) should be made to the cshrc or bash file:

```
################  ASTER stuff ############################
 if (`uname -n` == specialk)  setenv ASTER /usr/local/aster
 # if (`uname -n` == urquell)  setenv ASTER /usr/local/aster
 if (`uname -n` == urquell.colorado.edu)  then
    setenv ISFF /usr/local/PORTER/cuff/aster
    setenv ASTER /usr/local/PORTER/cuff/aster
endif

 # if (`uname -n` == porter.colorado.edu)  setenv ASTER /usr/local/cuff/aster
 if (`uname -n` == porter.colorado.edu)  setenv ASTER /usr/local/PORTER/cuff/aster

 # ASTER group .cshrc file
 #if ( -f $ASTER/scripts/.cshrc ) source $ASTER/scripts/.cshrc

 # comment these out if you want to use new aster:
 if ( -f $ASTER/scripts/aster.cshrc ) source $ASTER/scripts/aster.cshrc
 if (-f $ASTER/scripts/aster.login) source $ASTER/scripts/aster.login

 # this forces the new aster to be on the path (for using new prep in matlab):
 set path=(/opt/nidas/bin $path)

 # setenv RAWDATADIR /data/raw_data_2011
 # setenv RAWDATADIR /data/raw_data_2010
 # setenv RAWDATADIR /data/raw_data_2009
 # setenv RAWDATADIR /data/raw_data_2008
 # setenv RAWDATADIR /data/raw_data_2007
 # setenv RAWDATADIR /data/raw_data_2006
 # setenv RAWDATADIR /data/raw_data_2005
 # setenv RAWDATADIR /data/raw_data_2004
 # setenv RAWDATADIR /data/raw_data_2003
 # setenv RAWDATADIR /data/raw_data_2002
 # setenv RAWDATADIR /data/raw_data_2001
 # setenv RAWDATADIR /data/raw_data_2000
 # setenv RAWDATADIR /data/raw_data_1999
 # setenv RAWDATADIR /data/raw_data_1998
 # setenv RAWDATADIR /data/projects/NIWOT/raw_data

alias newaster 'set path=(/opt/nidas/bin $path)'
alias oldaster 'set path=(/usr/local/aster/bin/Linuxi686/ $path)'

 if ( $?cdpath ) then
  #set cdpath = ($cdpath $ASTER $ASTER/apps)
  set cdpath = ($cdpath $ASTER $ASTER/isff/src)
 else
  set cdpath = ($ASTER $ASTER/apps)
 endif

################  End of ASTER stuff ############################
```

For both NIDAS and ASTER, `prep` is used to create ASCII data files which are subsequently loaded into MATLAB using the mfiles provided above. The specific flags used with `prep` in NIDAS and ASTER differ, as shown in the examples below. Here we use "more" to show the data rather than piping it into a ASCII data file.

With NIDAS, an example with `prep` is:

```
/opt/nidas/bin/prep -D u.csi.24m,v.csi.24m,w.csi.24m,tc.csi.24m,diag,co2.ec,t.ec -H -d niwot \
                    -B "2016 04 01 07:00" -E "2016 04 02 07:10" -A  | more
2016-05-16,00:49:36|NOTICE|parsing: /usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/configs.xml
2016-05-16,00:49:36|INFO|parsed:/usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/configs.xml
2016-05-16,00:49:36|NOTICE|parsing: /usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/niwot.xml
2016-05-16,00:49:36|INFO|SamplePipelineProcSorter: start: Operation not permitted. Trying again with non-real-time p
2016-05-16,00:49:36|INFO|SamplePipelineRawSorter: start: Operation not permitted. Trying again with non-real-time pr
2016-05-16,00:49:36|INFO|SamplePipelineRawSorter(id=3054439312,Non-RT,prior=0,joinable,cancel=deferred) run...
2016-05-16,00:49:36|INFO|SamplePipelineRawSorter: sorterLength=1.000 sec, heapMax=1000000, heapBlock=1
2016-05-16,00:49:36|INFO|SamplePipelineProcSorter(id=3075419024,Non-RT,prior=0,joinable,cancel=deferred) run...
2016-05-16,00:49:36|INFO|SamplePipelineProcSorter: sorterLength=1.000 sec, heapMax=1000000, heapBlock=1
2016-05-16,00:49:36|INFO|opening: /data//projects/NIWOT/raw_data/niwot_20160401_040000.dat
2016 04 01 07:00:00.0828     -1.922     -4.085     -0.415     -10.37         54     1393.3      154.4
2016 04 01 07:00:00.1828     -1.818     -5.082     -1.064     -10.15         55     1393.7      154.4
2016 04 01 07:00:00.2828     -1.878     -4.906     -0.345     -10.23         56       1393      154.4
2016 04 01 07:00:00.3828     -2.112     -4.665     -0.658     -10.12         57       1394      154.4
2016 04 01 07:00:00.4828     -2.268     -4.543     -0.727     -10.14         58     1394.2      154.4
2016 04 01 07:00:00.5828     -3.125     -4.475     -0.558     -10.37         59       1394      154.4
2016 04 01 07:00:00.6828     -2.439     -4.685     -0.515     -10.21         60       1394      154.4
2016 04 01 07:00:00.7828      -2.57     -4.193     -0.643     -10.16         61     1393.2      154.4
2016 04 01 07:00:00.8828     -2.109     -4.248      -0.16     -10.23         62     1393.3      154.4
2016 04 01 07:00:00.9828     -2.242     -4.257     -0.417     -10.42         63     1393.8      154.4
etc, etc
```

And here is the effect of using the resampling with the “-r 10” option in prep (as discussed in the manuscript at the end of Section 4.2):

```
 /opt/nidas/bin/prep -D u.csi.24m,v.csi.24m,w.csi.24m,tc.csi.24m,diag,co2.ec,t.ec -H -d niwot \
                    -r 10 -B "2016 04 01 07:00" -E "2016 04 02 07:10" -A | more
2016-09-08,09:22:35|NOTICE|parsing: /usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/configs.xml
2016-09-08,09:22:35|INFO|parsed:/usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/configs.xml
2016-09-08,09:22:35|NOTICE|parsing: /usr/local/PORTER/cuff/aster/projects/NIWOT/ISFF/config/niwot.xml
2016-09-08,09:22:35|INFO|SamplePipelineProcSorter: start: Operation not permitted. Trying again with non-real-time p
2016-09-08,09:22:35|INFO|SamplePipelineRawSorter: start: Operation not permitted. Trying again with non-real-time pr
2016-09-08,09:22:35|INFO|SamplePipelineRawSorter(id=3055369104,Non-RT,prior=0,joinable,cancel=deferred) run...
2016-09-08,09:22:35|INFO|SamplePipelineRawSorter: sorterLength=1.000 sec, heapMax=1000000, heapBlock=1
2016-09-08,09:22:35|INFO|SamplePipelineProcSorter(id=3076348816,Non-RT,prior=0,joinable,cancel=deferred) run...
2016-09-08,09:22:35|INFO|SamplePipelineProcSorter: sorterLength=1.000 sec, heapMax=1000000, heapBlock=1
2016-09-08,09:22:35|INFO|opening: /data//projects/NIWOT/raw_data/niwot_20160401_040000.dat
2016 04 01 07:00:00.0500     -1.922     -4.085     -0.415     -10.37         54     1393.3      154.4
2016 04 01 07:00:00.1500     -1.818     -5.082     -1.064     -10.15         55     1393.7      154.4
2016 04 01 07:00:00.2500     -1.878     -4.906     -0.345     -10.23         56       1393      154.4
2016 04 01 07:00:00.3500     -2.112     -4.665     -0.658     -10.12         57       1394      154.4
2016 04 01 07:00:00.4500     -2.268     -4.543     -0.727     -10.14         58     1394.2      154.4
2016 04 01 07:00:00.5500     -3.125     -4.475     -0.558     -10.37         59       1394      154.4
2016 04 01 07:00:00.6500     -2.439     -4.685     -0.515     -10.21         60       1394      154.4
2016 04 01 07:00:00.7500      -2.57     -4.193     -0.643     -10.16         61     1393.2      154.4
2016 04 01 07:00:00.8500     -2.109     -4.248      -0.16     -10.23         62     1393.3      154.4
2016 04 01 07:00:00.9500     -2.242     -4.257     -0.417     -10.42         63     1393.8      154.4
etc, etc
```

Note that only the time stamp is changed, not the numerical data values.

With ASTER, `prep` is used in the following way:

```
/usr/local/aster/bin/Linuxi686/prep -D u.csi.24m,v.csi.24m,w.csi.24m,tc.csi.24m,diag,co2.ec,t.ec \
-f -B "2014 04 01 07:00" -E "2014 04 02 07:10" -A "%3.3f  %3.3f %3.3f %3.3f %8g %4.3f %4.2f" -t | more

Set OPS period: OPS=ops68
Opened: /data/raw_data_2014/all/nwt140401.000000
07:00:00.069 1.327  0.299 0.002 -7.110       62 1370.600 614.10
07:00:00.169 1.498  0.357 0.080 -7.090       63 1370.100 614.10
07:00:00.267 1.510  0.261 0.175 -6.981        0 1369.300 614.10
07:00:00.369 1.541  0.185 0.086 -6.975        1 1369.300 614.10
07:00:00.469 1.346  0.237 0.123 -7.050        2 1369.500 614.10
07:00:00.569 1.381  0.041 0.170 -7.040        3 1369.300 614.10
07:00:00.669 1.258  0.282 0.233 -7.130        4 1370.500 614.10
07:00:00.769 1.382  0.247 0.074 -7.070        5 1370.000 614.10
07:00:00.869 1.287  0.141 0.272 -7.270        6 1370.500 614.10
etc, etc
```

The end result of running the MATLAB `prep` mfiles is to create a series of daily gzip'ed ASCII data files which can subsequently be loaded into MATLAB. For example, these are like:

```
-rw-rw-r-- 1 sburns aster 35475088 May  2 10:37 /data/data_ascii/160401_quack_200.dat.gz
-rw-rw-r-- 1 sburns aster 46443927 May  2 10:38 /data/data_ascii/160401_quack_201.dat.gz
-rw-rw-r-- 1 sburns aster  1808408 May  2 10:38 /data/data_ascii/160401_quack_202.dat.gz
-rw-rw-r-- 1 sburns aster  1094190 May  2 10:38 /data/data_ascii/160401_quack_203.dat.gz
-rw-rw-r-- 1 sburns aster  3492846 May  2 10:39 /data/data_ascii/160401_quack_204.dat.gz
-rw-rw-r-- 1 sburns aster  1400949 May  2 10:39 /data/data_ascii/160401_quack_205.dat.gz
-rw-rw-r-- 1 sburns aster  2581480 May  2 10:39 /data/data_ascii/160401_quack_207.dat.gz
-rw-rw-r-- 1 sburns aster 14537249 May  2 10:40 /data/data_ascii/160401_quack_209.dat.gz
```

The man page for the NIDAS version of `prep`:

```
/opt/nidas/bin/prep -version
Version: 7151

/opt/nidas/bin/prep
Usage: /opt/nidas/bin/prep [-A] [-C] [-r rate] [-d dsmname] -D var[,var,...] [-B time] [-E time]
        [-h] [-s sorterLength] [-S dataSet_name ] [-x xml_file] [input ...]
    -A :ascii output (default)
    -C :binary column output, double seconds since Jan 1, 1970, followed by floats for each var
    -d dsmname: Look for a <fileset> belonging to the given dsm to determine input file names
    -D var[,var,...]: One or more variable names to output at the current rate
    -B "yyyy mm dd HH:MM:SS": begin time (optional)
    -E "yyyy mm dd HH:MM:SS": end time (optional)
    -h : this help
    -H : don't print out initial two line ASCII header of variable names and units
    -l log_level: 7=debug,6=info,5=notice,4=warn,3=err, default=6
    -n server:dir:file:interval:length:cdlfile:missing:timeout:batchperiod
        server: host name of system running nc_server RPC process
        dir: directory on server to write files
        file: format of NetCDF file names. For example: xxx_%Y%m%d.nc
        interval: deltaT in seconds between time values in file. Default: 1
        length: length of file, in seconds. 0 for no limit to the file size. Default: 1
        cdlfile: name of NetCDF CDL file on server that is used for initialization of new files
        missing: missing data value in file. Default: 1e+37
        timeout: time in seconds that nc_server is expected to respond. Default: 60
        batchperiod: check for response back from server after this number of seconds.
            Default: 1
    -p precision: number of digits in ASCII output values, default is 5
    -r rate: optional resample rate, in Hz for successive variables.
       Output timetags will be in middle of periods.
       When writing to NetCDF files, it can be useful for prep to generate
       output at several rates:  -r 1 -D v1,v2 -r 20 -D v3,v4
       If there are more than one -D option, specify the -r rate BEFORE the -D var
    -R rate: optional resample rate, in Hz. Output timetags will be at integral deltaTs.
       As with the -r option, prep can output at more than one rate
    -s sorterLength: input data sorter length in seconds (optional)
    -S dataSet_name from $ISFF/projects/$PROJECT/ISFF/config/datasets.xml
    -v : show version
    -w : windows/dos output (records terminated by CRNL instead of just NL)
    -x xml_file: if not specified, the xml file name is determined by either reading
       the data file header or from $ISFF/projects/$PROJECT/ISFF/config/configs.xml
    input: data input (optional). One of the following:
        sock:host[:port]          Default port is 30000
        unix:sockpath             unix socket name
        file [file ...]           one or more archive file names

If no inputs are specified, then the -B time option must be given, and
prep will read $ISFF/projects/$PROJECT/ISFF/config/configs.xml, to
find an xml configuration for the begin time, read it to find a
<fileset> archive for the given variables, and then open data files
matching the <fileset> path descriptor and time period.

prep does simple resampling, using the nearest sample to the times of the first
variable requested, or if the -r or -R rate options are specified, to evenly spaced times
at the given rate.

Examples:
prep -D u.10m,v.10m,w.10m -B "2006 jun 10 00:00" -E "2006 jul 3 00:00"
prep -D u.10m,v.10m,w.10m sock:dsmhost
prep -D u.10m,v.10m,w.10m -r 60 unix:/tmp/data_socket

Notes on choosing rates with -r or -R:
    For rates less than 1 Hz it is best to choose a value such that 10^6/rate is an integer.
    If you really want rate=1/3 Hz, specify rate to 7 significant figures,
    0.3333333, and you will avoid round off errors in the time tag.
    Output rates > 1 should be integers, or of a value with enough significant
    figures such that 10^6/rate is an integer.
```

The man page for the ASTER version of `prep`:

```
/usr/local/aster/bin/Linuxi686/prep
Either -D name,name,... or -a adam:chan,chan options must be specified

Usage: /usr/local/aster/bin/Linuxi686/prep [ -D dataID,dataID,... ] [-a adam:chan,chan,...]
        [-B "begin time"] [-E "end time"] [-j yday]
        [ -f [archiveFile ...] ]
        [-A [printfFormat]]
        [-i interval] [-r [rate]]
        [-b] [-C] [-h] [-p] [-t]
        [-c prepConfigFile]

dataID          ASTER data Id name, like  u.prop.10m
adam            Adam name, followed by channel numbers separated by commas
begin time      Date and/or time in one of the following formats
end time
        format                  examples (quotes required if date is specified)
        year mon day hrmnsc     "94 2 15 120000", "1994 feb 15 12:00:00.5"
        year yday hr:mn:sc      "94 046 12:00:00" (colons required in time)
        hrmnsc                  120000, 12:00:00 (must specify -j option)

-j yday         Day of year (1-366).  Required if no date in begin time
-f              Read from disk archive files. If no -f option is specified
                attach to ingestor and read data in real time (in which
                case -B and -j options are not required).
archiveFile     One or more archive files. A dash ("-") means stdin.
                If no files are listed, $RAWDATADIR will be searched for
                appropriate files
-A              ASCII column output, time in msec, print data using optional printf format
-b              Binary stream output
-C              Binary column output
-h              ASCII output, time in fractional hours
-i interval
-r rate         If -i or -r are not specified, samples are synchronized
                using nearest samples to times of first dataId
                Use -i to specify the average interval in seconds, or
                -r to specify the output average rate in Hz.
-t              ASCII output, time in HHMMSS, use format of preceding -A if specified
prepConfigFile  Defaults to $ASTER/projects/$PROJECT/$OPS/prep.config
```