**About mfiles for HydroBase3:**

The basic read function is **[profiles, nsta] = hb_read_file(dir, root, extent);**

INPUT: 1 or 3 strings that are concatenated to specify a filename.
OUTPUT: **profiles** is a struct array with the fields listed below
**nsta** : length(profiles)

Returns a structure with the following form:

```
%  profile.country
%        .ship,
%        .cruise
%        .station
%        .year
%        .month
%        .day
%        .orig
%        .instr
%        .lat
%        .lon
%        .bdpth
%        .nobs
%        .nprops
%        .ms10
%        .ms1
%        .prop_id  ( names of variable fields in profile.data)
%        .qual
%        .data  (structure containing nprops separate fields )
%            .pr
%            .te  (etc....)
```

profiles(1).station stores the station #
profiles(1).nobs stores number of observation levels in the profile
profiles(1).nprops stores number of properties
profiles(1).prop_id lists the names of available properties

property values are stored as column vectors in the field data which is a struct array:

```
profiles(1).data.pr    % pressure
profiles(1).data.t90    % in situ temperature
profiles(1).data.sa    % salinity
profiles(1).data.o2   % oxygen   ( in micromoles/kg)
```

**********

The basic write function is :

```
function err = hb_write_file(profiles, fname, mode)
% Creates file for writing in specified mode, writes profiles
% and closes the file.
%  profiles is struct array of profiles to be written
%  fname is full pathname of file
%  mode is 0 (zero) or 'w' for Overwrite
%         1 NoClobber
%         2 or 'a' for Append
% Returns err > 0 for an error.
```

which performs the following:

```
fid = hb_create_file(fname, 0);
[m,nsta] = size(profiles);
for ii=1:nsta
```

```
   err = hb_write_profile(fid,profiles(ii));
end
 fclose(fid);
```

## Useful ways of searching through the data…..

**[P,nsta]=hb_read_file('2008may.hb2_ctd');**

**staid = [P(:).station];** % returns array of all station #s
**lats = [P(:).lat];** % returns array of all station latitudes

**indx = find(staid==9009);** % search for a particular station
OR
**indx = find([P(:).station]==9014)**

% list properties available for a particular profile:
**P(indx).prop_id**
Returns names (strings) of each property:
pr
de
t90
sa
o2

These strings form the fields of the structure **data** which is itself a field of **P**()
P(indx).data.pr -- array of pressure for station # 9014
P(indx).data.de -- array of depths
propname = P(indx).prop_id(3) -- name of 3rd property in station #9014
P(indx).data.(propname) -- **(propname)** denotes a dynamic variable : propname is a string containing name of variable.

```
%Load properties from a single station
indx= find(  [P(:).station]  ==  9009);
pr = P(indx).data.pr;
t90 = P(indx).data.t90;
sa = P(indx).data.sa);

%  Compute potential temperature
th= sw_ptmp(sa, t90, pr, 0.0);

%  Load each property from all profiles into a matrix
pr=P(1).data.pr;
t90=P(1).data.t90;
sa=P(1).data.sa;
o2=P(1).data.o2;
for ii=2:nsta
   pr=merge(pr,P(ii).data.pr);
   t90=merge(te,P(ii).data.t90);
   sa=merge(sa,P(ii).data.sa);
    if  ~ isempty(strmatch('o2 ' ,P(ii).prop_id)     % checks if the profile has o2 data
      o2=merge(o2,P(ii).data.o2);
    else
      o2=merge(o2,[NaN]);
   end
end
```