QUODcarb variable names

QUODcarb's mksys function creates variable names for every parameter in the system. Capitalization does matter, and the capitalization scheme is as follows:

- Default lowercase for all parameter values
  - EXCEPT temperature (T) and pressure (P)
  - EXCEPT a small selection are capitalized to avoid ambiguity
    - HF and F of the fluoride system
      - phf looked too much like free hydrogen ion and 'f' by itself has other uses, so they are HF and F
    - H2S and HS of the sulfide system
      - hs was unclear, in our opinion, so it is HS
    - fH the free hydrogen ion activity coefficient
      - we used fH as in the CO2SYSv3 code
- Default totals all uppercase
  - TC, TA, TB, TS, TF, TP, TSi, TNH4, TH2S, TCa
  - only a trailing letter is lowercase
- Default pK is lowercase p (for -log10), uppercase K, lowercase following letter(s)
  - K0, K1, K2, Kb, Kw, Ks, Kf, Kp1, Kp2, Kp3, Ksi, Knh4, Kh2s, Kar, Kca
  - OmegaCa and OmegaAr fall under this category

QUODcarb Possible Inputs

* means it is required

`obs.tp(i).var` can repeat for as many temperature-pressure dependent systems as input

Variable	Name	Units
`obs.sal`	*salinity	S_P (unitless)
`obs.usal`	salinity measurement uncertainty (± 1 sigma)	S_P (unitless)
`obs.TC`	total carbon, dissolved inorganic carbon (DIC)	µmol/kg-SW
`obs.uTC`	total carbon measurement uncertainty (± 1 sigma)	µmol/kg-SW
`obs.TA`	total alkalinity	µmol/kg-SW
`obs.uTA`	total alkalinity uncertainty (± 1 sigma)	µmol/kg-SW
`obs.tp(i).T`	*temperature	°C
`obs.tp(i).uT`	temperature uncertainty (± 1 sigma)	°C
`obs.tp(i).P`	*pressure (below surface, surface = 0 dbar)	dbar

obs.tp(i).uP	pressure uncertainty (± 1 sigma)	dbar
obs.tp(i).fco2	fugacity of CO_2 = $f(CO_2)$	µatm
obs.tp(i).ufco2	$f(CO_2)$ uncertainty (± 1 sigma)	µatm
obs.tp(i).pco2	partial pressure of CO_2 = $p(CO_2)$	µatm
obs.tp(i).upco2	$p(CO_2)$ uncertainty (± 1 sigma)	µatm
obs.tp(i).co2st	CO_2* = CO_2 *(aq)* + H_2CO_3 *(aq)*	µatm
obs.tp(i).uco2st	CO_2* uncertainty (± 1 sigma)	µatm
obs.tp(i).co3	total carbonate ion = $CO_3^{2-}{}_T$	µmol/kg-SW
obs.tp(i).uco3	$CO_3^{2-}{}_T$ uncertainty (± 1 sigma)	µmol/kg-SW
obs.tp(i).ph	-$\log_{10}$ hydrogen ion = pH	unitless
obs.tp(i).uph	pH uncertainty (± 1 sigma)	unitless
obs.TB	total borate	µmol/kg-SW
obs.uTB	total borate uncertainty (± 1 sigma)	µmol/kg-SW
obs.TS	total sulfate	µmol/kg-SW
obs.uTS	total sulfate uncertainty (± 1 sigma)	µmol/kg
obs.TF	total fluoride	µmol/kg-SW
obs.uTF	total fluoride uncertainty (± 1 sigma)	µmol/kg-SW
obs.TP	total phosphate = H_3PO_4 + $H_2PO_4^-$ + HPO_4^{2-} + PO_4^{3-}	µmol/kg-SW
obs.uTP	total phosphate uncertainty (± 1 sigma)	µmol/kg-SW
obs.TSi	total silicate	µmol/kg-SW
obs.uTSi	total silicate uncertainty (± 1 sigma)	µmol/kg-SW
obs.TNH3	total ammonia = NH_3 + NH_4^+	µmol/kg-SW
obs.uTNH3	total ammonia uncertainty (± 1 sigma)	µmol/kg-SW
obs.TH2S	total sulfide	µmol/kg-SW
obs.uTH2S	total sulfide uncertainty (± 1 sigma)	µmol/kg-SW
obs.TCa	total calcium	µmol/kg-SW
obs.uTCa	total calcium uncertainty (± 1 sigma)	µmol/kg-SW

QUODcarb Possible Outputs

- Every parameter in the system has six possible forms (except a few exceptions):
  - **1 & 2-** parameter value and parameter uncertainty in normal space
  - **3 & 4-** parameter value and parameter uncertainty in -log10 space ('p')
  - **5 & 6-** upper and lower error bounds in normal space
    - Upper/Lower error bounds are calculated in normal space, they don't exist in $-\log_{10}$ space ('p')
- `obs.tp(i).var` will repeat for as many temperature-pressure dependent systems as input, tp(1), tp(2), tp(3), etc.
- this list is also available at your command line if you use the command '`fieldnames(est)`' and '`fieldnames(est.tp)`'
  - or '`fieldnames(est(1).tp)`' if more than one datapoint in est structure

- Temperature, Salinity, and Pressure
  - `est.sal, est.usal`
  - `est.tp(i).T, est.tp(i).uT`
  - `est.tp(i).P, est.tp(i).uP`
- Total carbon, also known as DIC
  - `est.TC, est.uTC, est.uTC_u, est.uTC_l`
  - `est.pTC, est.upTC`
- Total alkalinity
  - `est.TA, est.uTA, est.uTA_u, est.uTA_l`
  - `est.pTA, est.upTA`
- Water: Kw = [h]/[oh]
  - `Kw:`
    - `est.tp(i).pKw, est.tp(i).upKw, est.tp(i).Kw, est.tp(i).uKw, est.tp(i).uKw_u, est.tp(i).uKw_l`
  - `oh:`
    - `est.tp(i).poh, est.tp(i).upoh, est.tp(i).oh, est.tp(i).uoh, est.tp(i).uoh_u, est.tp(i).uoh_l`
  - `ph:`
    - `est.tp(i).ph, est.tp(i).uph, est.tp(i).h, est.tp(1).uh, est.tp(i).uh_u, est.tp(i).uh_l`
  - `ph (free scale):`
    - `est.tp(i).ph_free, est.tp(i).uph_free, est.tp(i).h_free, est.tp(1).uh_free, est.tp(i).uh_free_u, est.tp(i).uh_free_l`
  - `ph (total scale):`

        - est.tp(i).ph_tot, est.tp(i).uph_tot, est.tp(i).h_tot, est.tp(1).uh_tot, est.tp(i).uh_tot_u, est.tp(i).uh_tot_l
    - ph (seawater scale):
        - est.tp(i).ph_sws, est.tp(i).uph_sws, est.tp(i).h_sws, est.tp(1).uh_sws, est.tp(i).uh_sws_u, est.tp(i).uh_sws_l
    - ph (nbs scale):
        - est.tp(i).ph_nbs, est.tp(i).uph_nbs, est.tp(i).h_nbs, est.tp(1).uh_nbs, est.tp(i).uh_nbs_u, est.tp(i).uh_nbs_l
- Carbonate: K0 = [co2st]/[fco2], K1 = [h][hco3]/[co2st], K2 = [h][co3]/[hco3]
    - K0:
        - est.tp(i).pK0, est.tp(i).upK0, est.tp(i).K0, est.tp(i).uK0, est.tp(i).uK0_u, est.tp(i).uK0_l
    - K1:
        - est.tp(i).pK1, est.tp(i).upK1, est.tp(i).K1, est.tp(i).uK1, est.tp(i).uK1_u, est.tp(i).uK1_l
    - K2:
        - est.tp(i).pK2, est.tp(i).upK2, est.tp(i).K2, est.tp(i).uK2, est.tp(i).uK2_u, est.tp(i).uK2_l
    - fco2:
        - est.tp(i).pfco2, est.tp(i).upfco2, est.tp(i).fco2, est.tp(i).ufco2, est.tp(i).ufco2_u, est.tp(i).ufco2_l
    - co2st:
        - est.tp(i).pco2st, est.tp(i).upco2st, est.tp(i).co2st, est.tp(i).uco2st, est.tp(i).uco2st_u, est.tp(i).uco2st_l
    - p2f: convert pco2 to fco2
        - est.tp(i).pp2f, est.tp(i).upp2f, est.tp(i).p2f, est.tp(i).up2f
    - hco3:
        - est.tp(i).phco3, est.tp(i).uphco3, est.tp(i).hco3, est.tp(i).uhco3, est.tp(i).uhco3_u, est.tp(i).uhco3_l
    - co3:
        - est.tp(i).pco3, est.tp(i).upco3, est.tp(i).co3, est.tp(i).uco3, est.tp(i).uco3_u, est.tp(i).uco3_l

- Borate: Kb = [h][boh4]/[boh3]
    - TB:
        - est.TB, est.uTB, est.uTB_u, est.uTB_l, est.pTB, est.upTB
    - Kb:
        - est.tp(i).pKb, est.tp(i).upKb, est.tp(i).Kb, est.tp(i).uKb, est.tp(i).uKb_u, est.tp(i).uKb_l
    - boh4:
        - est.tp(i).pboh4, est.tp(i).upboh4, est.tp(i).boh4, est.tp(i).uboh4, est.tp(i).uboh4_u, est.tp(i).uboh4_l
    - boh3:
        - est.tp(i).pboh3, est.tp(i).upboh3, est.tp(i).boh3, est.tp(i).uboh3, est.tp(i).uboh3_u, est.tp(i).uboh3_l
- Sulfate: Ks = [fH][so4]/[hso4]
    - TS:
        - est.TS, est.uTS, est.uTS_u, est.uTS_l, est.pTS, est.upTS
    - Ks:
        - est.tp(i).pKs, est.tp(i).upKs, est.tp(i).Ks, est.tp(i).uKs, est.tp(i).uKs_u, est.tp(i).uKs_l
    - fH:
        - est.tp(i).pfH, est.tp(i).upfH, est.tp(i).fH, est.tp(i).ufH, est.tp(i).ufH_u, est.tp(i).ufH_l
    - so4:
        - est.tp(i).pso4, est.tp(i).upso4, est.tp(i).so4, est.tp(i).uso4, est.tp(i).uso4_u, est.tp(i).uso4_l
    - hso4:
        - est.tp(i).phso4, est.tp(i).uphso4, est.tp(i).hso4, est.tp(i).uhso4, est.tp(i).uhso4_u, est.tp(i).uhso4_l
- Fluoride: Kf = [h][F]/[HF]
    - TF:
        - est.TF, est.uTF, est.uTF_u, est.uTF_l, est.pTF, est.upTF
    - Kf:
        - est.tp(i).pKf, est.tp(i).upKf, est.tp(i).Kf, est.tp(i).uKf, est.tp(i).uKf_u, est.tp(i).uKf_l

    - F:
        - est.tp(i).pF, est.tp(i).upF, est.tp(i).F, est.tp(i).uF, est.tp(i).uF_u, est.tp(i).uF_l
    - HF:
        - est.tp(i).pHF, est.tp(i).upHF, est.tp(i).HF, est.tp(i).uHF, est.tp(i).uHF_u, est.tp(i).uHF_l
- Phosphate: Kp1 = [h][h2po4]/[h3po4], Kp2 = [h][hpo4]/[h2po4], Kp3 = [h][po4]/[hpo4]
    - TP:
        - est.TP, est.uTP, est.uTP_u, est.uTP_l, est.pTP, est.upTP
    - Kp1:
        - est.tp(i).pKp1, est.tp(i).upKp1, est.tp(i).Kp1, est.tp(i).uKp1, est.tp(i).uKp1_u, est.tp(i).uKp1_l
    - Kp2:
        - est.tp(i).pKp2, est.tp(i).upKp2, est.tp(i).Kp2, est.tp(i).uKp2, est.tp(i).uKp2_u, est.tp(i).uKp2_l
    - Kp3:
        - est.tp(i).pKp3, est.tp(i).upKp3, est.tp(i).Kp3, est.tp(i).uKp3, est.tp(i).uKp3_u, est.tp(i).uKp3_l
    - h3po4:
        - est.tp(i).ph3po4, est.tp(i).uph3po4, est.tp(i).h3po4, est.tp(i).uh3po4, est.tp(i).uh3po4_u, est.tp(i).uh3po4_l
    - h2po4:
        - est.tp(i).ph2po4, est.tp(i).uph2po4, est.tp(i).h2po4, est.tp(i).uh2po4, est.tp(i).uh2po4_u, est.tp(i).uh2po4_l
    - hpo4:
        - est.tp(i).phpo4, est.tp(i).uphpo4, est.tp(i).hpo4, est.tp(i).uhpo4, est.tp(i).uhpo4_u, est.tp(i).uhpo4_l
    - po4:
        - est.tp(i).ppo4, est.tp(i).uppo4, est.tp(i).po4, est.tp(i).upo4, est.tp(i).upo4_u, est.tp(i).upo4_l
- Silicate: Ksi = [h][siooh3]/[sioh4]

    - TSi:
        - est.TSi, est.uTSi, est.uTSi_u, est.uTSi_l, est.pTSi, est.upTSi
    - Ksi:
        - est.tp(i).pKsi, est.tp(i).upKsi, est.tp(i).Ksi, est.tp(i).uKsi, est.tp(i).uKsi_u, est.tp(i).uKsi_l
    - siooh3:
        - est.tp(i).psiooh3, est.tp(i).upsiooh3, est.tp(i).siooh3, est.tp(i).usiooh3, est.tp(i).usiooh3_u, est.tp(i).usiooh3_l
    - sioh4:
        - est.tp(i).psioh4, est.tp(i).upsioh4, est.tp(i).sioh4, est.tp(i).usioh4, est.tp(i).usioh4_u, est.tp(i).usioh4_l
- Ammonia: Knh4 = [h][nh3]/[nh4]
    - TNH4:
        - est.TNH4, est.uTNH4, est.uTNH4_u, est.uTNH4_l, est.pTNH4, est.upTNH4
    - Knh4:
        - est.tp(i).pKnh4, est.tp(i).upKnh4, est.tp(i).Knh4, est.tp(i).uKnh4, est.tp(i).uKnh4_u, est.tp(i).uKnh4_l
    - nh3:
        - est.tp(i).pnh3, est.tp(i).upnh3, est.tp(i).nh3, est.tp(i).unh3, est.tp(i).unh3_u, est.tp(i).unh3_l
    - nh4:
        - est.tp(i).pnh4, est.tp(i).upnh4, est.tp(i).nh4, est.tp(i).unh4, est.tp(i).unh4_u, est.tp(i).unh4_l
- Sulfide: Kh2s = [h][HS]/[H2S]
    - TH2S:
        - est.TH2S, est.uTH2S, est.uTH2S_u, est.uTH2S_l, est.pTH2S, est.upTH2S
    - Kh2s:
        - est.tp(i).pKh2s, est.tp(i).upKh2s, est.tp(i).Kh2s, est.tp(i).uKh2s, est.tp(i).uKh2s_u, est.tp(i).uKh2s_l
    - HS:

        - est.tp(i).pHS, est.tp(i).upHS, est.tp(i).HS, est.tp(i).uHS, est.tp(i).uHS_u, est.tp(i).uHS_l
    - H2S:
        - est.tp(i).pH2S, est.tp(i).upH2S, est.tp(i).H2S, est.tp(i).uH2S, est.tp(i).uH2S_u, est.tp(i).uH2S_l
- Aragonite: Kar = [co3][ca]/OmegaAr
    - TCa:
        - est.TCa, est.uTCa, est.uTCa_u, est.uTCa_l, est.pTCa, est.upTCa
    - Kar:
        - est.tp(i).pKar, est.tp(i).upKar, est.tp(i).Kar, est.tp(i).uKar, est.tp(i).uKar_u, est.tp(i).uKar_l
    - ca:
        - est.tp(i).pca, est.tp(i).upca, est.tp(i).ca, est.tp(i).uca, est.tp(i).uca_u, est.tp(i).uca_l
    - OmegaAr:
        - est.tp(i).pOmegaAr, est.tp(i).upOmegaAr, est.tp(i).OmegaAr, est.tp(i).uOmegaAr, est.tp(i).uOmegaAr_u, est.tp(i).uOmegaAr_l
- Calcite: Kca = [co3][ca]/OmegaCa
    - Kca:
        - est.tp(i).pKca, est.tp(i).upKca, est.tp(i).Kca, est.tp(i).uKca, est.tp(i).uKca_u, est.tp(i).uKca_l
    - OmegaCa:
        - est.tp(i).pOmegaCa, est.tp(i).upOmegaCa, est.tp(i).OmegaCa, est.tp(i).uOmegaCa, est.tp(i).uOmegaCa_u, est.tp(i).uOmegaCa_l
- Organic Alkalinity: Kalpha = [h][alpha]/[halpha]
    - If opt.pKalpha = 1
    - TAlpha:
        - est.TAlpha, est.uTAlpha, est.uTAlpha_u, est.uTAlpha_l, est.pTAlpha, est.upTAlpha
    - Kalpha:
        - est.tp(i).pKalpha, est.tp(i).upKalpha, est.tp(i).Kalpha, est.tp(i).uKalpha, est.tp(i).uKalpha_u, est.tp(i).uKalpha_l
    - alpha:

        - est.tp(i).palpha, est.tp(i).upalpha, est.tp(i).alpha, est.tp(i).ualpha, est.tp(i).ualpha_u, est.tp(i).ualpha_l
    - halpha:
        - est.tp(i).phalpha, est.tp(i).uphalpha, est.tp(i).halpha, est.tp(i).uhalpha, est.tp(i).uhalpha_u, est.tp(i).uhalpha_l
- Organic Alkalinity: Kbeta = [h][beta]/[hbeta]
    - If opt.pKbeta = 1
    - TBeta:
        - est.TBeta, est.uTBeta, est.uTBeta_u, est.uTBeta_l, est.pTBeta, est.upTBeta
    - Kbeta:
        - est.tp(i).pKbeta, est.tp(i).upKbeta, est.tp(i).Kbeta, est.tp(i).uKbeta, est.tp(i).uKbeta_u, est.tp(i).uKbeta_l
    - beta:
        - est.tp(i).pbeta, est.tp(i).upbeta, est.tp(i).beta, est.tp(i).ubeta, est.tp(i).ubeta_u, est.tp(i).ubeta_l
    - hbeta:
        - est.tp(i).phbeta, est.tp(i).uphbeta, est.tp(i).hbeta, est.tp(i).uhbeta, est.tp(i).uhbeta_u, est.tp(i).uhbeta_l
- est.f
    - residual ‘f’ value for internal consistency analysis