

# risese Group

John D. Baker

<https://github.com/bakerjd99/jackshacks/blob/main/risese.ijs>

SHA-256: 4e9d1921e7070cad9f70c1e0205ac778b2925c7fa5a246c0d28003318178ae8f

March 29, 2023

## Contents

<b>risese Overview</b>	<b>2</b>
risese Interface . . . . .	3
risese Algorithm Notes . . . . .	3
<b>risese Source Code</b>	<b>5</b>
<b>=: Index</b>	<b>41</b>

## riserset Overview

`riserset` is a collection of basic astronomical algorithms that compute the rise, transit, and set times of [IAU-named](#) stars.

`riserset` is distributed as an auxillary J addon. Auxillary addons are hosted in private GitHub repositories. `riserset` can be installed in the local J folder `~addons/jacks` with:

```
load 'pacman'
```

*NB. files from <https://github.com/bakerjd99/jackshacks>*

```
install 'github:bakerjd99/jackshacks'
```

*NB. installed files*

```
dir '~addons/jacks'
```

Once installed it can be loaded and run with:

```
load 'riserset'  
location_yellowstone 0
```

*NB. IAU stars rising/setting over Old Faithful*

```
iau_tonight 0
```

For more details about setting up, running, and “hacking” `riserset` refer to the Jupyter notebook `riserset_notebook.ipynb` or the pdf version of same `'riserset_notebook.pdf'`. Both of these files are installed with `riserset` but are also available here:

## riseset Interface

`iau_tonight` [17] *named IAU stars visible tonight*  
`loadstars` [20] *loads riseset star data*  
`riseset` [33] *rise, transit, set times of stars*

## riseset Algorithm Notes

Many `riseset` algorithms are taken from Jean Meeus's book *Astronomical Algorithms*. A PDF copy of this book is available here:

<https://ia802807.us.archive.org/20/items/astronomicalalgorithmsjeanmeeus1991/>

also here:

[https://www.amazon.com/s?i=stripbooks&rh=p\\_27%3AJean+Meeus&s=relevancerank&text=Jean+Meeus&ref=dp\\_byline\\_sr\\_book\\_1](https://www.amazon.com/s?i=stripbooks&rh=p_27%3AJean+Meeus&s=relevancerank&text=Jean+Meeus&ref=dp_byline_sr_book_1)

Nutation algorithms are from Jay Tanner's site:

<https://neoprogrammics.com/nutations/index.php>

Markdown versions of Tanner's algorithms are stored in the JOD `futs` and `utils` dictionaries — see:

1. `nututation_in_longitude_dPsi_md`
2. `nututation_in_obliquity_of_ecliptic_dEpsDeg_md`

in:

<https://github.com/bakerjd99/joddumps/blob/master/futs.ijs>

<https://github.com/bakerjd99/joddumps/blob/master/utils.ijs>

Delta T ( $\Delta T$ ) is computed using polynomial expressions by Espenak and Meeus, see:

<https://eclipse.gsfc.nasa.gov/SEhelp/deltatpoly2004.html>

A markdown version of the Delta T ( $\Delta T$ ) algorithm is in `futs` see:

```
1. nasa_polynomial_expressions_for_delta_t_md
```

You can display the markdown referenced about with the JOD expressions:

```
load 'general/jod'
od ;:'futs utils'
NB. display markdown documents
4 disp ; }.@(4&dn1)&.> 'nutation_';'nasa_'
```

Many examples showing how to use various `riseset` words are in the JOD `futs` test suite `riseset`. You can display all the test cases with the JOD expressions:

```
3 grp 'riseset'    NB. test cases in suite

3 disp 'riseset'   NB. display test suite

4 rtt 'riseset'    NB. silently run all tests in suite
```

## riset Source Code

*NB.\*riset s-- compute rise, transit and set times of IAU named stars.*

*NB.*

*NB. verbatim: interface word(s):*

*NB. -----*

*NB. iau\_tonight - named IAU stars visible tonight*

*NB. loadstars - loads riset star data*

*NB. riset - rise, transit, set times of stars*

*NB.*

*NB. created: 2023mar09*

*NB. changes: -----*

*NB. 23mar28 (iau\_tonight) added*

`coclass 'riset'`

*NB.\*end-header*

*NB. Byte Order Mark: UTF-8 EF, BB, BF (hex) or 239 187 191 (dec)*

`BOM=: 239 187 191{a.`

*NB. carriage return character*

`CR=: 13{a.`

*NB. seconds per day*

`DAYSECS=: 86400`

*NB. interface words (IFACEWORDSriset) group*

IFACEWORDSriset=: <;.\_1 ' iau\_tonight loadstars riset'

*NB. horizon limit in degrees*

LIMITHORZ=: 20

*NB. limiting magnitude*

LIMITMAG=: 3.

*NB. approximate epoch J2000 obliquity of the ecliptic degrees, minutes, seconds*

OBLIQUITYDMS2000=: 23 26 21.448000000000004

*NB. observer latitude longitude, west longitudes negative*

OBSLOCATION=: \_116.375956000000002 43.6467749999999981

*NB. root words (ROOTWORDSriset) group*

ROOTWORDSriset=: <;.\_1 ' IFACEWORDSriset ROOTWORDSriset VMDriset iau\_tonight yyyymmfd'

*NB. standard altitude stars - compensates for horizon atmospheric refraction*

STDALTITUDE=: 0.566699999999999982

*NB. UTC time zone offset in hours*

UTCOFFSET=: 6

*NB. version, make count and date*

VMDriset=: '0.8.0';10;'29 Mar 2023 10:27:21'

*NB. all zero, first, second, ... nth differences of nl: alldifs ?.10#100*  
 alldifs=: ([: >: [: i. [: - #) {.&.> [: <"1 (}. - }:)^(i.@#@[)

apparRADEC=: 4 : 0

*NB.\*apparRADEC v-- apparent RA and DEC for epoch (x) from J2000.0*  
*NB. RA and DEC.*

*NB.*

*NB. This verb adjusts J2000 RA and DEC coordinates to another*  
*NB. epoch. The method is based on Meeus (20.3) pg 126. This*  
*NB. calculation ignores stellar proper motions and assumes that*  
*NB. (y) RA DEC values are J2000.0. The resulting positions are*  
*NB. accurate enough for basic rise, transit, and set*  
*NB. calculations.*

*NB.*

*NB. dyad: ft =. flymd apparRADEC ftRADEC*

*NB.*

*NB. 2028 11 13.19 apparRADEC 41.054063 ,. 49.227750*

*NB.*

*NB. ({."1 ciau)=: {:"1 ciau*

*NB. 2023 4 22 apparRADEC RA\_J2000 ,: Dec\_J2000*

*'zet z th'=.* zetzthT0 *x NB. final epoch t*

*'ra dec'=.* y *NB. J2000 ra,dec*

*NB. meeus (20.4) pg. 126*

*A=. (cosd dec)\*sind ra + zet*

*B=. ((cosd th)\*(cosd dec)\*cosd ra + zet) - (sind th)\*sind dec*

```
C=. ((sind th)*(cosd dec)*cosd ra + zet) + (cosd th)*sind dec
```

*NB. NIMP star close celestial poles*

*NB. new dec,ra*

```
ran=. z + atan2 A ,: B [ decn=. dfr arcsin C  
ran ,: decn  
)
```

*NB. seconds correction apparent sidereal time - meeus pg. 84 - ( $\Delta\psi * \cos(\epsilon)$ )/15*  
apparsecs=: 15 %~ (3600 \* nutation\_longitude\_dPsi) \* [: cosd meanobliquityjd0

*NB. apparent Greenwich sidereal - hms: apparsidjd0 julfrcal |: 2023 1 3,:1991 2 8.5*  
apparsidjd0=: ([: dmsfrdd 15 %~ [: nth0 meansidjd0) + 0 0 ,"1 [: ,. apparsecs

*NB. applies the verb in string (x) to (y)*  
apply=: 128!:2

*NB. arc cosine*

```
arccos=: _2&o.
```

*NB. arc sine*

```
arcsin=: _1&o.
```

*NB. arc tangent*

```
arctan=: _3&o.
```



*NB. signal with optional message*

```
assert=: 0 0" _ $ 13!:8^:((0: e. ])^(12"_))
```

```
atan2=: 3 : 0
```

*NB.\*atan2 v-- arctangent of (Y % X) in degrees.*

*NB.*

*NB. FORTRAN (ATN2) variation of the standard (arctan) (\_3&o.) for  
NB. ratios. Based on a PASCAL function from Astronomy on the  
NB. Personal Computer by Montenbruck and Pfleger ISBN  
NB. 0-387-52754-0 pg. 9.*

*NB.*

*NB. Result is between \_180 <: atn2 <: 180 degrees*

*NB.*

*NB. monad: fl =. atan2 flYX*

*NB.*

*NB. atan2 1 ,: 1 NB. 45 degrees*

*NB. atan2 1 ,: %: 3 NB. 30 degrees*

*NB.*

*NB. NB. random ratios comparing two atan2 verbs*

*NB. r=: ?. 2 500\$50*

*NB. r=: r \* (\$r) \$ (?.~ \*/\$r) { (\*/\$r)\$\_1 1*

*NB. (atan2b |.r) -: atan2 r*

*NB.*

*NB. NB. surprisingly (atan2) is faster than (atan2b)*

*NB. NB. (j 9.41 2023) but (atan2b) consumes less memory*

*NB. NB. 1000 ts"1 'atan2b r',: 'atan2 |.r'*

---

```

NB. vector J                      NB. scalar PASCAL
rad=. 0.0174532925199432955      NB. CONST RAD=0.0174532925199433;
r=. 0 #~ {: $y

b0=. */0=y                      NB. IF (X=0.0) AND (Y=0.0) THEN ATN2:= 0.0
ir=. i. #r=. 0 (I. b0)} r

if. +./b1=. -.b0 do.
  t=. |(I. b1) {"1 y            NB. AX=: ABS(X); AY=: ABS(Y)
  it=. (I. b1) { ir

  b2=. (1{t) > 0{t                NB. IF (AX>AY) THEN PHI=: ARCTAN(AY/AX)/RAD
  s=. (I. b2) {"1 t
  r=. (rad %~ arctan %/s) (b2#it)} r

  s=. (I. -.b2) {"1 t            NB. ELSE PHI=: 90.0-ARCTAN(AX/AY)/RAD;
  r=. (90 - rad %~ arctan %/|.s) (it #~ -.b2)} r
end.

x10=. I. b1 *. (1{y) < 0         NB. IF (X<0.0) THEN PHI=: 180.0-PHI;
r=. (180 - x10{r) (x10)} r
y10=. I. b1 *. (0{y) < 0         NB. IF (Y<0.0) THEN PHI=: -PHI;
r=. (-y10{r) (y10)} r

r
)

NB. boxes open nouns

```

```
boxopen=: <^(L. = 0:)
```

```
cold_iau_named_stars=: 3 : 0
```

```
NB.*cold_iau_named_stars v-- convert IAU btcl to column dictionary.
```

```
NB.
```

```
NB. monad: bt =. cold_iau_named_stars btcl
```

```
NB.
```

```
NB. iau=. ; {: , > {: 4 get 'iau_named_stars_2022_txt'
```

```
NB. ciau=. cold_iau_named_stars parse_iau_named_stars iau
```

```
NB.
```

```
NB. NB. define columns
```

```
NB. (0 {"1 ciau)=: 1 {"1 ciau
```

```
c=. 0{"1 t=. |: y
```

```
p0=. c i. ;:'Vmag RA_J2000 Dec_J2000'
```

```
d=. _999&".&> p0 { t=. }."1 t
```

```
'invalid mag, ra, dec' assert -. _999 e. d
```

```
p1=. c i. ;:'IAU_Name Designation Bayer_Name'
```

```
c ,. (<"1 ] p1 { t) , <"1 d
```

```
)
```

```
NB. cosine radians
```

```
cos=: 2&o.
```

```
NB. cosine degrees
```

```
cosd=: cos@rfd
```

```
NB. decimal degrees from degrees, minutes, seconds - inverse (dmsfrdd)
ddfrdms=: (60"_ #. ])% 3600"_

deltaT0=: 3 : 0

NB.*deltaT0 v-- dynamical time  $\Delta T$  in seconds.
NB.
NB. Returns the difference in seconds between UT and TD based on
NB. polynomial expressions by Espenak and Meesus. This
NB. calculation is useful for the years -1999 to 3000: a five
NB. thousand year period.
NB.
NB. see: https://eclipse.gsfc.nasa.gov/SEhelp/deltatpoly2004.html
NB.
NB. also in (futs): nasa_polynomial_expressions_for_delta_t_md
NB.
NB. monad: flSecs =. deltaT0 flYd
NB.
NB. ymd=. |: (3 {. 6!:0 ' '), _1812 3 12 , _12 12 11 , 2137 12 13, 1700 1 1 ,: 35 7 6
NB. |: ymd , deltaT0 deltaTdy ymd

NB. (ry) time intervals are (l,u]

NB. before -500:
NB.  $\Delta T = -20 + 32 * u^2$ ; where:  $u = (y-1820)/100$ 
ry=. ,: _1999 _500
t1=. {{ _20 + 32 * U^2 [ U=. (y - 1820) % 100 }}
```

NB. between -500 and +500:

NB.  $\Delta T = 10583.6 - 1014.41 * u + 33.78311 * u^2 - 5.952053 * u^3$

NB.  $- 0.1798452 * u^4 + 0.022174192 * u^5 + 0.0090316521 * u^6$ ; where:  $u = y/100$

NB. **NOTE:** for the year -500 set value of 17190 to 17203.7

ry=. ry , \_500 500

t2=. {{ 10583.6 - (1014.41\*U) + (33.78311\*U^2) - (5.952053\*U^3) - (0.1798452\*U^4) + (0.022174192\*U^5) + 0.  
>..>0090316521\*U^6 [ U=. y % 100 ]}}

NB. between +500 and +1600:

NB.  $\Delta T = 1574.2 - 556.01 * u + 71.23472 * u^2 + 0.319781 * u^3$

NB.  $- 0.8503463 * u^4 - 0.005050998 * u^5 + 0.0083572073 * u^6$ ; where:  $u = (y-1000)/100$

ry=. ry , 500 1600

t3=. {{ 1574.2 - (556.01\*U) + (71.23472\*U^2) + (0.319781\*U^3) - (0.8503463\*U^4) - (0.005050998\*U^5) + 0.00  
>..>83572073\*U^6 [ U=. (y-1000) % 100 ]}}

NB. between +1600 and +1700:

NB.  $\Delta T = 120 - 0.9808 * t - 0.01532 * t^2 + t^3 / 7129$ ; where:  $t = y - 1600$

ry=. ry , 1600 1700

t4=. {{ 120 - (0.9808\*t) - (0.01532\*t^2) + (t^3)%7129 [ t=. y - 1600 ]}}

NB. between +1700 and +1800:

NB.  $\Delta T = 8.83 + 0.1603 * t - 0.0059285 * t^2 + 0.00013336 * t^3 - t^4 / 1174000$ ; where:  $t = y - 1700$

ry=. ry , 1700 1800

t5=. {{ 8.83 + (0.1603\*t) - (0.0059285\*t^2) + (0.00013336\*t^3) - (t^4)%1174000 [ t=. y - 1700 ]}}

NB. between +1800 and +1860:

NB.  $\Delta T = 13.72 - 0.332447 * t + 0.0068612 * t^2 + 0.0041116 * t^3 - 0.00037436 * t^4$

```

NB.      + 0.0000121272 * t^5 - 0.0000001699 * t^6 + 0.000000000875 * t^7; where: t = y - 1800
ry=. ry , 1800 1860
t6=. {{ 13.72 - (0.332447*t) + (0.0068612*t^2) + (0.0041116*t^3) - (0.00037436*t^4) + (0.0000121272*t^5) -
>..> (0.0000001699*t^6) + 0.000000000875*t^7 [ t=. y - 1800 ]}

NB. between 1860 and 1900:
NB.    ΔT = 7.62 + 0.5737 * t - 0.251754 * t^2 + 0.01680668 * t^3
NB.      - 0.0004473624 * t^4 + t^5 / 233174; where: t = y - 1860
ry=. ry , 1860 1900
t7=. {{ 7.62 + (0.5737*t) - (0.251754*t^2) + (0.01680668*t^3) - (0.0004473624*t^4) + (t^5)%233174 [ t=. y
>..>- 1860 ]}

NB. between 1900 and 1920:
NB.    ΔT = -2.79 + 1.494119 * t - 0.0598939 * t^2 + 0.0061966 * t^3 - 0.000197 * t^4; where: t = y - 1900
ry=. ry , 1900 1920
t8=. {{ -2.79 + (1.494119*t) - (0.0598939*t^2) + (0.0061966*t^3) - 0.000197*t^4 [ t=. y - 1900 ]}

NB. between 1920 and 1941:
NB.    ΔT = 21.20 + 0.84493*t - 0.076100 * t^2 + 0.0020936 * t^3; where: t = y - 1920
ry=. ry , 1920 1941
t9=. {{ 21.20 + (0.84493*t) - (0.076100*t^2) + 0.0020936*t^3 [ t=. y - 1920 ]}

NB. between 1941 and 1961:
NB.    ΔT = 29.07 + 0.407*t - t^2/233 + t^3 / 2547; where: t = y - 1950
ry=. ry , 1941 1961
t10=. {{ 29.07 + 0.407*t - ((t^2)%233) + (t^3)%2547 [ t=. y - 1950 ]}

```

*NB. between 1961 and 1986:*

*NB.  $\Delta T = 45.45 + 1.067*t - t^2/260 - t^3 / 718$  ; where:  $t = y - 1975$*

*ry=. ry , 1961 1986*

*t11=. {{ 45.45 + (1.067\*t) - ((t^2)%260) - (t^3)%718 [ t=. y - 1975 ]}}*

*NB. between 1986 and 2005:*

*NB.  $\Delta T = 63.86 + 0.3345 * t - 0.060374 * t^2 + 0.0017275 * t^3 + 0.000651814 * t^4$*

*NB.  $+ 0.00002373599 * t^5$ ; where:  $t = y - 2000$*

*ry=. ry , 1986 2005*

*t12=. {{ 63.86 + (0.3345\*t) - (0.060374\*t^2) + (0.0017275\*t^3) + (0.000651814\*t^4) + 0.00002373599\*t^5 [ t  
>..>=. y - 2000 ]}}*

*NB. between 2005 and 2050:*

*NB.  $\Delta T = 62.92 + 0.32217 * t + 0.005589 * t^2$ ; where:  $t = y - 2000$*

*ry=. ry , 2005 2050*

*t13=. {{ 62.92 + (0.32217\*t) + 0.005589\*t^2 [ t=. y - 2000 ]}}*

*NB. between 2050 and 2150:*

*NB.  $\Delta T = -20 + 32 * ((y-1820)/100)^2 - 0.5628 * (2150 - y)$*

*ry=. ry , 2050 2150*

*t14=. {{ \_20 + (32 \* ((y-1820)%100)^2) - 0.5628 \* 2150 - y ]}}*

*NB. after 2150:*

*NB.  $\Delta T = -20 + 32 * u^2$ ; where:  $u = (y-1820)/100$*

*ry=. ry , 2150 3000*

*t15=. {{ \_20 + 32 \* U^2 [ U=. (y-1820)%100 ]}}*

*NB. NOTE: the t(i) verbs match the intervals*

```
ti=. (rb=. /:~ ~. ,ry) I. y
'year range _1999 to 3000 exceeded' assert -. (0,#rb) e. ti
```

*NB. t(i) gerund*

```
tg=. t1`t2`t3`t4`t5`t6`t7`t8`t9`t10`t11`t12`t13`t14`t15
```

*NB. apply t(i) verbs to appropriate intervals*

```
(;ti </. i.#y) { ;(tg {~ <: ~.ti) apply&.> ti </. y
)
```

*NB. delta  $\Delta T$  decimal year: deltaTdy 2023 3 12 ,. 1959 12 11*

```
deltaTdy=: (0 { ])+ 12 %~ 0.5 -- 1 { ]
```

*NB. degrees from radian*

```
dfr=: *&57.2957795130823229
```

*NB. degrees, minutes, seconds from decimal degrees - inverse (ddfrdms)*

```
dmsfrdd=: <. (,.) 60 60 #: 3600 * 1 | ,
```

*NB. fractional day from hms: 0.80625 = fdfrhms 19 21 0*

```
fdfrhms=: 24 %~ (60"_ #. ]) % 3600"
```

*NB. fractional centuries from epoch J2000 Meeus pg. 83: gT0jd julfrcal 1957 10 4.81*

```
gT0jd=: 36525 %~ 2451545. -- ]
```



*NB. fractional centuries from epoch J2000 Meeus pg. 83: gT0ymd 1957 10 4.81*  
gT0ymd=: 36525 %~ 2451545. -- julfrcal

*NB. hours, minutes from decimal seconds: hmfrds dsfrhms 20 27 43.23*  
hmfrds=: [: 24 60&#: 60 %~ ]

iau\_tonight=: 3 : 0

*NB.\*iau\_tonight v-- named IAU stars rising/setting tonight.*

*NB.*

*NB. monad: bt =. iau\_tonight uuIgnore*

*NB.*

*NB. iau\_tonight 0*

*NB.*

*NB. dyad: bt =. bLYmd\_LB\_UO\_LMAG\_LHORZ iau\_tonight uuIgnore*

*NB.*

*NB. NB. date of Uluru star party diner*

*NB. YMD=. 2022 10 19*

*NB. ULURU=. 131.01941 \_25.34301*

*NB. UTC=. \_9.5*

*NB. LMAG=. 6.0*

*NB. LHORZ=. 5*

*NB. (YMD;ULURU;UTC;LMAG;LHORZ) iau\_tonight 0*

*((3 {. 6!:0 ' ');OBSLOCATION;UTCOFFSET;LIMITMAG;LIMITHORZ) iau\_tonight y*

*:*

*NB. date, location, UTC offset, magnitude, horizon*

```
'YMD LB UO LMAG LHORZ'=. x

NB. IAU star data
({."1 IAU)=. {:"1 IAU [ 'IAU NAV'=. loadstars 0

NB. !(*)=. Vmag IAU_Name
NB. brighter than limiting magnitude
stars=. (LMAG > Vmag) # IAU_Name
Rsiau=. (YMD;UO;LB) riseset stars

NB. retain rising setting - circumpolar NIMP
Rsiau=. Rsiau #~ -. ; 1 {"1 Rsiau

NB. name ,. transit altitude, hour minutes
Rsiau=. (0 {"1 Rsiau) ,. (0 2 3)&{&.> 1&{&.> 2 {"1 Rsiau

NB. retain above local horizon
Rsiau=. Rsiau #~ LHORZ < 0&{&> 1 {"1 Rsiau

NB. sort by transit time
Rsiau {~ /: }."1 > 1 {"1 Rsiau
)

intr3p=: 4 : 0

NB.*intr3p v-- interpolate three values - meeus pg 25.
NB.
NB. dyad: fln intr3p fl
```

```
NB.
NB.  NB. meeus pg. 24
NB.  yi=. 0.884226 0.877366 0.870531
NB.  0.05 intr3p yi

NB.  y = y2 + (n/2)(a + b nc)
NB.  a b c are differences

'only 3 values' assert 3=#y

d=. 1 2{alldifs y
'a b'=. >0{d [ c=. ,/ >1{d
(1{y) + (x%2) * a + b + x*c
)

julfrcal=: 3 : 0

NB.*julfrcal v-- Julian dates from calendar dates.
NB.
NB.  Astronomical Julian date. Similiar to (tojulian) but handles
NB.  the fact that Julian days start at noon rather than midnight
NB.  for calendar days.
NB.
NB.  monad: fl =. julfrcal ilYYYYMMDD / ftYYYYMMDD
NB.
NB.  julfrcal 2001 9 11
NB.  julfrcal 1776 1941 1867 , 7 12 7 ,: 4 7 1
NB.
```

NB. Meeus (Astronomical Algorithms) test cases (pg. 61)  
 NB. NB. **NOTE:** the fractional day representation of time  
 NB. 2436116.31 = julfrcal 1957 10 4.81 NB. 7.a Sputnik 1  
 NB. 1842713.0 = julfrcal 333 1 27.5 NB. 7.b  
 NB.  
 NB. NB. zero date is roughly the age of the oldest bristlecone pines (coincidence?)  
 NB. julfrcal -4711 10 29.5

NB. vector J	NB. scalar BASIC
'y m d'=. y	NB. INPUT "Y,M,D ";Y,M,D
g=. 1582 <: y	NB. G=1: IF Y<1582 THEN G=0
f=. (d - d1) - 0.5 [ d1=. <. d	NB. D1=INT(D): F=D-D1-0.5
j=. - <. 7 * 4 %~ <.y + 12 %~ m+9	NB. J=-INT(7*(INT((M+9)/12)+Y)/4)
	NB. IF G=0 THEN 805
s=. * m-9 [ a=.   m-9	NB. S=SGN(M-9): A=ABS(M-9)
j3=. <. y + s * <. a%7	NB. J3=INT(Y+S*INT(A/7))
j3=. - <. 3r4 * >: <. j3 % 100	NB. J3=-INT((INT(J3/100)+1)*3/4)
j=. j + (<.275 * m%9) + d1 + g*j3	NB. 805 J=J+INT(275*M/9)+D1+G*J3
j=. j + 1721027 + (2*g) + 367*y	NB. J=J+1721027+2*G+367*Y
b=. f >: 0	NB. IF F>=0 THEN 825
f=. f + b [ j=. j - b	NB. F=F+1: J=J-1
f + j	
)	

loadstars=: 3 : 0

NB.\*loadstars v-- loads riseset star data.  
 NB.

```
NB. monad: bLIAU_Nav =. loadstars uuIgnore
NB.
NB. NB. needs (futs,utils) dictionaries
NB. od;: 'futs utils' [ 3 od'' [ require 'general/jod'
NB. loadstars 0
NB.
NB. dyad: bLIAU_Nav=. pa loadstars uuIgnore
NB.
NB. 1 loadstars 0 NB. define columns
NB. loadstars~ 1 NB. shorter idiom

0 loadstars y
:
NB. load IAU stars !(*)=. get
ciau=. ; {: , > {: MACRO_ajod_ get 'iau_named_stars_2022_txt'
ciau=. cold_iau_named_stars parse_iau_named_stars ciau

NB. load navigation stars
cnavs=. parsetd (; {: , > {: MACRO_ajod_ get 'Navigation_Stars_txt') -. CR
cnavs=. (0 { cnavs) ,. <"1 |: }. cnavs
'star column overlap' assert 0 = #(0 {"1 cnavs) ([ -. -. ) 0 {"1 ciau

NB. define columns - override mixed assignments (<:)=:
if. x -: 1 do.
  (0 {"1 ciau)=: 1 {"1 ciau
  (0 {"1 cnavs)=: 1 {"1 cnavs
end.
```

```
(<ciau),<cnavs
)

location_uluru=: 3 : 0

NB.*location_uluru v-- set parameters for Uluru location.
NB.
NB. monad: location_uluru uuIgnore
NB.
NB. location_uluru 0 NB. set location
NB. iau_tonight 0 NB. uses set location with current date
NB.
NB. NB. uses location with set date
NB. (location_uluru 0) iau_tonight 0

NB. date of Uluru star party diner
ymd=. 2022 10 19

NB. lat,lon with standard signs
OBSLOCATION_riseset=: 131.01941 _25.34301

UTCOFFSET_riseset=: _9.5 NB. time zone
LIMITMAG_riseset=: 6.0 NB. stellar magnitude
LIMITHORZ_riseset=: 5 NB. degrees above horizon

ymd;OBSLOCATION;UTCOFFSET;LIMITMAG;LIMITHORZ
)
```

```
location_yellowstone=: 3 : 0
```

```
NB.*location_yellowstone v-- set parameters for Old Faithful location.
```

```
NB.
```

```
NB. monad: location_yellowstone uuIgnore
```

```
NB.
```

```
NB. location_yellowstone 0 NB. set location
```

```
NB. iau_tonight 0 NB. uses set location with current date
```

```
NB.
```

```
NB. NB. uses location with set date
```

```
NB. (location_yellowstone 0) iau_tonight 0
```

```
NB. date of mom's death
```

```
ymd=. 2013 5 7
```

```
NB. longitude, latitude with standard signs
```

```
OBSLOCATION_riseset=: _110.82792 44.46057
```

```
UTCOFFSET_riseset=: 6.0 NB. MST time zone
```

```
LIMITMAG_riseset=: 6.0 NB. stellar magnitude
```

```
LIMITHORZ_riseset=: 10 NB. degrees above horizon
```

```
ymd;OBSLOCATION;UTCOFFSET;LIMITMAG;LIMITHORZ
```

```
)
```

```
meanobliquityT0=: 3 : 0
```

```
NB.*meanobliquityT0 v-- mean obliquity of the ecliptic IAU in degrees.
```

*NB.*

*NB. monad: fl =. meanobliquityT0 flT*

*NB. units are decimal arc seconds*

*ea=. +/3600 60 1 \* OBLIQUITYDMS2000*

*NB. meeus (21.2) pg. 135*

*3600 %~ ea - (46.8150\*y) - (0.00059\*y^2) + 0.001813\*y^3  
)*

*meanobliquityT1=: 3 : 0*

*NB.\*meanobliquityT1 v-- mean obliquity of the ecliptic Laskar in  
NB. degrees.*

*NB.*

*NB. Mean obliquity using Laskar's polynomial. This expression is  
NB. more accurate than (meanobliquityT0): see Meeus (21.2) pg.  
NB. 135.*

*NB.*

*NB. monad: fl =. meanobliquityT1 flT*

*NB. units are decimal arc seconds*

*ea=. +/3600 60 1 \* OBLIQUITYDMS2000*

*NB. time units 10000 Julian years*

*U=. y % 100*

*e0=. (39.05\*U^6) + (7.12\*U^7) + (27.87\*U^8) + (5.79\*U^9) + 2.45\*U^10*



```
3600 %~ ea - (4680.93*U) - (1.55*U^2) + (1999.25*U^3) - (51.38*U^4) - (249.67*U^5) - e0  
)
```

```
meanobliquityjd0=: 3 : 0
```

```
NB.*meanobliquityjd0 v-- mean obliquity ecliptic for Julian date (y) degrees.
```

```
NB.
```

```
NB. monad: fl =. meanobliquityjd0 flJD
```

```
NB.
```

```
NB. NB. meeus pg. 136
```

```
NB. e0=. ,dmsfrdd meanobliquityjd0 2446895.5
```

```
NB.
```

```
NB. NB. matches to 3 decimals
```

```
NB. 23 26 27.407 -: 0.001 round e0
```

```
NB.
```

```
NB. dyad: fl =. pa meanobliquityjd0 flJD
```

```
NB.
```

```
NB. NB. Laskar algorithm
```

```
NB. el=. ,dmsfrdd 1 meanobliquityjd0 2446895.5
```

```
0 meanobliquityjd0 y
```

```
:
```

```
meanobliquityT0`meanobliquityT1@.(x) gT0jd y
```

```
)
```

```
meansid0=: 4 : 0
```

*NB.\*meansid0 v-- mean sidereal time at Greenwich for T (x) JD (y).*

*NB.*

*NB. dyad: flDegs =. flT meansid flJD*

*NB. meeus (11.4) pg 84*

*280.46061837 + (360.98564736629 \* y - 2451545.0) + (0.000387933 \* x^2) - 38710000 %~ x^3  
)*

*meansidjd0=: 3 : 0*

*NB.\*meansidjd0 v-- mean sidereal time at Greenwich for julian day (y) in degrees.*

*NB.*

*NB. monad: fl =. meansidjd0 flJD*

*NB.*

*NB. NB. julian day for April 10, 1987 19h:24m:00s UT*

*NB. JD=. julfrcal 1987 4,10 + fdfrhms 19 21 0*

*NB. meansidjd0 JD*

*(gT0jd y) meansid0 y  
)*

*NB. normalize negative degree sidereal time: nnth0 -1677831.2621266*

*nnth0=: ] + 360 \* [: | [: (<.) 360 %~ ]*

*NB. normalize positive degree sidereal time: npth0 1677831.2621266*

*npth0=: ] - 360 \* [: (<.) 360 %~ ]*

```
NB. normalize degree sidereal time: nth0 _35555 77777
nth0=: npth0`nnth0@.(0&>:@[])

nututation_longitude_dPsi=: 3 : 0

NB.*nututation_longitude_dPsi v-- nutation in ecliptical longitude in degrees (1980 iau theory).
NB.
NB. NOTE: the pseudo-code is vector ready and easily converted to J.
NB.
NB. verbatim: algorithm from Jay Tanner https://neoprogrammics.com/nutations/
NB.
NB. see: nututation_in_longitude_dPsi_md
NB.
NB. monad: flDeg =. nututation_longitude_dPsi flJD
NB.
NB. ymd=. |: 2023 3 12 , 1959 12 11 , 2135 12 13, 1700 1 1 ,: 1935 7 6
NB. JD=. julfrcal ymd NB. no delT adj.
NB. 2460015.5 = 0{JD
NB. nututation_longitude_dPsi JD
NB.
NB. NB. see (futs) test: (risetset_tanner_smoke) for examples

T=. (y - 2451545) % 36525 NB. T = (JD - 2451545) / 36525
T2=. T*T NB. T2 = T*T
T3=. T*T2 NB. T3 = T*T2

NB. DegToRad = 3.1415926535897932 / 180
DegToRad=. 3.1415926535897932 % 180
```

```

NB. w1 = 297.85036 + 445267.11148*T - 0.0019142*T2 + (T3 / 189474)
w1=. 297.85036 + (445267.11148*T) - (0.0019142*T2) + (T3 % 189474)
w1=. DegToRad*(w1)          NB. w1 = DegToRad*(w1)

```

```

NB. w2 = 357.52772 + 35999.05034*T - 0.0001603*T2 - (T3 / 300000)
w2=. 357.52772 + (35999.05034*T) - (0.0001603*T2) - (T3 % 300000)
w2=. DegToRad*(w2)          NB. w2 = DegToRad*(w2)

```

```

NB. w3 = 134.96298 + 477198.867398*T + 0.0086972*T2 + (T3 / 56250)
w3=. 134.96298 + (477198.867398*T) + (0.0086972*T2) + (T3 % 56250)
w3=. DegToRad*(w3)          NB. w3 = DegToRad*(w3)

```

```

NB. w4 = 93.27191 + 483202.017538*T - 0.0036825*T2 + (T3 / 327270)
w4=. 93.27191 + (483202.017538*T) - (0.0036825*T2) + (T3 % 327270)
w4=. DegToRad*(w4)          NB. w4 = DegToRad*(w4)

```

```

NB. w5 = 125.04452 - 1934.136261*T + 0.0020708*T2 + (T3 / 450000)
w5=. 125.04452 - (1934.136261*T) + (0.0020708*T2) + (T3 % 450000)
w5=. DegToRad*(w5)          NB. w5 = DegToRad*(w5)

```

```

w=. (sin w5)*((_174.2*T) - 171996)          NB. w = sin(w5)*(-174.2*T - 171996)
w=. w + (sin 2 * w4 + w5 - w1)*((_1.6*T) - 13187) NB. w = w + sin(2*(w4 + w5 - w1))*(-1.6*T - 13187)
w=. w + (sin 2 * w4 + w5)*(_2274 - 0.2*T)      NB. w = w + sin(2*(w4 + w5))*(-2274 - 0.2*T)
w=. w + (sin 2 * w5)*((0.2*T) + 2062)          NB. w = w + sin(2 * w5)*(0.2*T + 2062)
w=. w + (sin w2)*(1426 - 3.4*T)                NB. w = w + sin(w2)*(1426 - 3.4*T)
w=. w + (sin w3)*((0.1*T) + 712)               NB. w = w + sin(w3)*(0.1*T + 712)

```

*NB.*  $w = w + \sin(2*(w_4 + w_5 - w_1) + w_2)*(1.2*T - 517)$   
 $w = w + (\sin(2 * w_4 + w_5 - w_1) + w_2)*((1.2*T) - 517)$

$w = w + (\sin(2*w_4) + w_5)*((_0.4*T) - 386)$       *NB.*  $w = w + \sin(2 * w_4 + w_5)*(-0.4*T - 386)$

*NB.*  $w = w + \sin(2*(w_4 + w_5 - w_1) - w_2)*(217 - 0.5*T)$   
 $w = w + (\sin(2 * w_4 + w_5 - w_1) - w_2)*(217 - 0.5*T)$

$w = w + (\sin(2*w_4 - w_1) + w_5)*(129 + 0.1*T)$

$w = w + (\sin w_3 + w_5)*((0.1*T) + 63)$

$w = w + (\sin w_5 - w_3)*((_0.1*T) - 58)$

$w = w + (\sin 2*w_2)*(17 - 0.1*T)$

$w = w + (\sin 2 * w_2 + w_4 + w_5 - w_1)*((0.1*T) - 16)$

$w = w - 301*(\sin(2 * w_4 + w_5) + w_3)$

$w = w - 158*(\sin w_3 - 2*w_1)$

$w = w + 123*(\sin(2 * w_4 + w_5) - w_3)$

$w = w + 63*(\sin 2*w_1)$

$w = w - 59*(\sin(2 * w_1 + w_4 + w_5) - w_3)$

$w = w - 51*(\sin(2*w_4) + w_3 + w_5)$

$w = w + 48*\sin(2 * w_3 - w_1)$

$w = w + 46*(\sin(2 * w_4 - w_3) + w_5)$

$w = w - 38*(\sin 2 * w_1 + w_4 + w_5)$

$w = w - 31*(\sin 2 * w_3 + w_4 + w_5)$

$w = w + 29*(\sin 2*w_3)$

$w = w + 29*(\sin(2 * w_4 + w_5 - w_1) + w_3)$

$w = w + 26*(\sin 2*w_4)$

*NB.*  $w = w + \sin(2*(w_4 - w_1) + w_5)*(129 + 0.1*T)$

*NB.*  $w = w + \sin(w_3 + w_5)*(0.1*T + 63)$

*NB.*  $w = w + \sin(w_5 - w_3)*(-0.1*T - 58)$

*NB.*  $w = w + \sin(2*w_2)*(17 - 0.1*T)$

*NB.*  $w = w + \sin(2*(w_2 + w_4 + w_5 - w_1))*(0.1*T - 16)$

*NB.*  $w = w - 301*\sin(2*(w_4 + w_5) + w_3)$

*NB.*  $w = w - 158*\sin(w_3 - 2*w_1)$

*NB.*  $w = w + 123*\sin(2*(w_4 + w_5) - w_3)$

*NB.*  $w = w + 63*\sin(2*w_1)$

*NB.*  $w = w - 59*\sin(2*(w_1 + w_4 + w_5) - w_3)$

*NB.*  $w = w - 51*\sin(2 * w_4 + w_3 + w_5)$

*NB.*  $w = w + 48*\sin(2*(w_3 - w_1))$

*NB.*  $w = w + 46*\sin(2*(w_4 - w_3) + w_5)$

*NB.*  $w = w - 38*\sin(2*(w_1 + w_4 + w_5))$

*NB.*  $w = w - 31*\sin(2*(w_3 + w_4 + w_5))$

*NB.*  $w = w + 29*\sin(2*w_3)$

*NB.*  $w = w + 29*\sin(2*(w_4 + w_5 - w_1) + w_3)$

*NB.*  $w = w + 26*\sin(2*w_4)$

```

w=. w - 22*(sin 2* w4 - w1)
w=. w + 21*(sin (2*w4) + w5 - w3)
w=. w + 16*(sin (2*w1) - w3 + w5)
w=. w - 15*(sin w2 + w5)
w=. w - 13*(sin w3 + w5 - 2*w1)
w=. w - 12*(sin w5 - w2)
w=. w + 11*(sin 2 * w3 - w4)
w=. w - 10*(sin (2 * w4 + w1) + w5 - w3)
w=. w - 8*(sin (2 * w4 + w1 + w5) + w3)
w=. w + 7*(sin (2 * w4 + w5) + w2)
w=. w - 7*(sin w3 - (2*w1) + w2)
w=. w - 7*(sin (2 * w4 + w5) - w2)
w=. w - 7*(sin (2*w1) + (2*w4) + w5)
w=. w + 6*(sin (2*w1) + w3)
w=. w + 6*(sin 2 * w3 + w4 + w5 - w1)
w=. w + 6*(sin (2 * w4 - w1) + w3 + w5)
w=. w - 6*(sin (2 * w1 - w3) + w5)
w=. w - 6*(sin (2*w1) + w5)
w=. w + 5*(sin w3 - w2)
w=. w - 5*(sin (2* w4 - w1) + w5 - w2)
w=. w - 5*(sin w5 - 2*w1)
w=. w - 5*(sin (2 * w3 + w4) + w5)
w=. w + 4*(sin (2 * w3 - w1) + w5)
w=. w + 4*(sin (2 * w4 - w1) + w2 + w5)
w=. w + 4*(sin w3 - 2*w4)
w=. w - 4*(sin w3 - w1)
w=. w - 4*(sin w2 - 2*w1)

```

```

NB. w = w - 22*sin(2*(w4 - w1))
NB. w = w + 21*sin(2*w4 + w5 - w3)
NB. w = w + 16*sin(2*w1 - w3 + w5)
NB. w = w - 15*sin(w2 + w5)
NB. w = w - 13*sin(w3 + w5 - 2*w1)
NB. w = w - 12*sin(w5 - w2)
NB. w = w + 11*sin(2*(w3 - w4))
NB. w = w - 10*sin(2*(w4 + w1) + w5 - w3)
NB. w = w - 8*sin(2*(w4 + w1 + w5) + w3)
NB. w = w + 7*sin(2*(w4 + w5) + w2)
NB. w = w - 7*sin(w3 - 2*w1 + w2)
NB. w = w - 7*sin(2*(w4 + w5) - w2)
NB. w = w - 7*sin(2*w1 + 2*w4 + w5)
NB. w = w + 6*sin(2*w1 + w3)
NB. w = w + 6*sin(2*(w3 + w4 + w5 - w1))
NB. w = w + 6*sin(2*(w4 - w1) + w3 + w5)
NB. w = w - 6*sin(2*(w1 - w3) + w5)
NB. w = w - 6*sin(2*w1 + w5)
NB. w = w + 5*sin(w3 - w2)
NB. w = w - 5*sin(2*(w4 - w1) + w5 - w2)
NB. w = w - 5*sin(w5 - 2*w1)
NB. w = w - 5*sin(2*(w3 + w4) + w5)
NB. w = w + 4*sin(2*(w3 - w1) + w5)
NB. w = w + 4*sin(2*(w4 - w1) + w2 + w5)
NB. w = w + 4*sin(w3 - 2*w4)
NB. w = w - 4*sin(w3 - w1)
NB. w = w - 4*sin(w2 - 2*w1)

```

```

w=. w - 4*(sin w1)
w=. w + 3*(sin (2*w4) + w3)
w=. w - 3*(sin 2 * w4 + w5 - w3)
w=. w - 3*(sin w3 - w1 - w2)
w=. w - 3*(sin w2 + w3)
w=. w - 3*(sin (2 * w4 + w5) + w3 - w2)
w=. w - 3*(sin (2 * w1 + w4 + w5) - w2 - w3)
w=. w - 3*(sin (2 * w4 + w5) + 3*w3)
w=. w - 3*(sin (2 * w1 + w4 + w5) - w2)

NB. w = w - 4*sin(w1)
NB. w = w + 3*sin(2*w4 + w3)
NB. w = w - 3*sin(2*(w4 + w5 - w3))
NB. w = w - 3*sin(w3 - w1 - w2)
NB. w = w - 3*sin(w2 + w3)
NB. w = w - 3*sin(2*(w4 + w5) + w3 - w2)
NB. w = w - 3*sin(2*(w1 + w4 + w5) - w2 - w3)
NB. w = w - 3*sin(2*(w4 + w5) + 3*w3)
NB. w = w - 3*sin(2*(w1 + w4 + w5) - w2)

dPsiDeg=. w % 36000000.0 NB. dPsiDeg = w / 36000000.0
)

parse_iau_named_stars=: 3 : 0

NB.*parse_iau_named_stars v-- IAU named star list to btcl header
NB. table.
NB.
NB. Original star name data was downloaded from:
NB.
NB. https://www.iau.org/public/themes/naming_stars/
NB.
NB. and slightly adjusted in Excel. The data stored in (futs) is
NB. a Unicode UTF-8 CSV export.
NB.
NB. monad: btcl =. parse_iau_named_stars clTxt
NB.
NB. NB. get stars from (futs)

```

```
NB.  NB. od ;:'futs utils'
NB.  iau=. ; {: , > {: 4 get 'iau_named_stars_2022_txt'
NB.  parse_iau_named_stars iau

NB. remove any byte order mark
t=. parsecsv y }~ (BOM -: (#BOM){.y){0,#BOM

NB. extract relevant columns
c=. ;:'IAU_Name Designation Bayer_Name Vmag RA_J2000 Dec_J2000'
t=. t {"1~ (0 { t) i. c

NB. scrub objects with questionable magnitude
t #~ _ ~: _999&".&> (c i. <'Vmag') {"1 t
)

parscsv=: 3 : 0

NB.*parscsv v-- parses comma delimited files. (x) is the field
NB. delimiter. Lines are delimited with either CRLF or LF
NB.
NB. monad: btcl =. parsecsv cl
NB. dyad:  btcl =. ca parsecsv cl
NB.
NB.  ', ' parsecsv read 'c:\comma\delimited\text.csv'

', ' parsecsv y
:
```



```
'separator cannot be the " character' assert -. x -: ''
```

```
NB. CRLF delimited *.csv text to char table
```

```
y=. x ,. ] ;. _2 y -. CR
```

```
NB. bit mask of unquoted " field delimiters
```

```
b=. -. }. ~:/\ ' ' e.~ ' ' , , y
```

```
b=. ($y) $ b *. , x = y
```

```
NB. use masks to cut lines
```

```
b <;. _1"1 y
```

```
)
```

```
NB. parse TAB delimited table text - see long document
```

```
parsetd=: [: <;. _2&> (a.{~9) ,&.>~ [: <;. _2 [: (] , ((10{a.)"_ = {:) }. (10{a.)"_ ) (13{a.) -.> ]
```

```
NB. radians from degrees
```

```
rfd=: *&0.0174532925199432955
```

```
riset=: 4 : 0
```

```
NB.*riset v-- rise, transit, set times of IAU named stars.
```

```
NB.
```

```
NB. dyad: bt=. bLYMD_UO_LB_AOBJ riset blclStarNames
```

```
NB.
```

```
NB. LB=. _116.375956 43.646775 NB. Meridian
```

```
NB. YMD=. 2023 3 27
```

```
NB.    UO=. 6
NB.    (YMD;UO;LB) riseset 'Algol'
NB.    (YMD;UO;LB) riseset 'Algol';'Rigel';'Spica'
NB.
NB.    NB. add objects not IAU names - need name, ra, dec
NB.    AOB=. (<;:'Venus'),(<41.73129),<18.44092
NB.    AOB=. ,&.> (;:'OBJ_Name OBJ_RA_J2000 OBJ_Dec_J2000') ,. AOB
NB.    (YMD;UO;LB;<AOB) riseset 'Venus'

NB. local time, UT offset (0=Greenwich), Latitude Longitude
'ymfd uo LB AOB'=. 4 {. x

NB. convert LB to meeus convention
LB=. _1 1 * LB

NB. local time to UT
UT=. ymfd + 0 0,uo%24

NB. look up RA, Dec
'IAU Navigation'=. loadstars 0
NB. IAU stars !(*)=. IAU_Name RA_J2000 Dec_J2000
({."1 IAU)=. {:"1 IAU
Stars=. boxopen y

if. #AOB do.
  NB. insert additional objects
  ({."1 AOB)=. {:"1 AOB
```

---

```

NB. !(*)=. OBJ_Dec_J2000 OBJ_Name OBJ_RA_J2000
IAU_Name=. OBJ_Name , IAU_Name
RA_J2000=. OBJ_RA_J2000 , RA_J2000
Dec_J2000=. OBJ_Dec_J2000 , Dec_J2000
end.

if. 0 e. b=. Stars e. IAU_Name do.
  smoutput 'not in IAU named stars -> '; Stars #~ -.b
else.
  ix=. IAU_Name i. Stars
  RA=. <ix{RA_J2000 [ Dec=. <ix{Dec_J2000
  riseset_calc UT;LB;(<Stars),RA,Dec
end.
)

riseset_calc=: 3 : 0

NB.*riseset_calc v-- rise, transit, set times of stars.
NB.
NB. Main rise/set calculations. Argument (y) set in (riseset).
NB.
NB. monad: bt =. riseset_calc bLYMD_LB_OBJ_RA_Dec

'ymd LB obj ra dec'=. ,&.> y

NB. (L) longitude, west positive
NB. (B) latitude, north positive
'L B'=. LB

```

```
obj=. obj ,"0 1 a:,a:  NB. result table

NB. dynamical time  $\Delta T$  in fractional days NOTE:  $\Delta T$  is not
NB. going to change a lot over the interpolation period
if. 0=nc<'DeltaTsOverride_risese_ ' do.
  dTfd=. DeltaTsOverride_risese_ % DAYSECS
else.
  dTfd=. (,/deltaT0 deltaTdy ymd) % DAYSECS
end.

NB. apparent sidereal time Greenwich at 0h in degrees
th0=. ,/ddfrdms 15 * apparsidjd0 julfrcal ymd

NB. TD times  $\Delta T + UT = TD$ 
TD=. (2 {. ymd),"1 0 (_1 0 1 + {:ymd) + dTfd

NB. apparent ra,dec for _1 0 1 days around rise/set
rdi=. |: TD apparRADEC"1 _ ra ,: dec
h0=. STDALTITUDE

NB. approximate times (14.1) meeus pg. 98
cosH0=. ((sind h0) - (sind B)*sind (<a::;1;1){rdi) % (cosd B)*cosd (<a::;1;1){rdi

NB. 1 indicates above or below horizon
bhrz=. 1 < |cosH0
obj=. (<"0 bhrz) (<a::;1){ obj
```

```
obj=. (<'above or below horizon') (<(I. bhrz);2)} obj
ix=. I. -.bhrz  NB. objects that rise and set

NB. m(i) are fractional day times (1/) puts mi in [0,1]
H0=. dfr arccos ix{cosH0
m0=. 1|360 %~ ((<ix;0;1){rdi) + L - th0
m1=. 1|m0 - H0 % 360
m2=. 1|m0 + H0 % 360

NB. rise, transit, setting
m=. m1 ,. m0 ,. m2

NB. sidereal time at Greenwich - meeus pg. 99
th=. nth0 th0 + 360.985647*m

NB. adjusted ra,dec
rda=. nu intr3p"1 ix{rdi [ nu=. dTfd + m

NB. local hour angles
rax=. <a.;0 [ decx=. <a.;1
H=. (th - L) - rax{rda

NB. body's altitude (12.6) meeus pg. 89
sih=. ((sind B)*sind decx{rda) + (cosd B)*(cosd decx{rda)*cosd H

NB. degree altitudes positive
h=. |dfr arcsin sih
```

*NB. corrections for transits (trx), rise/sets (rsx)*

```
dltm=. ($m)$0
trx=. <a;;1 [ rsx=. <a;;0 2
dltm=. (-(trx{H}%360) trx} dltm
drs=. rsx { (h - h0) % 360 * (cosd decx{rda}*(cosd B)*sind H
dltm=. drs rsx} dltm
m=. m + dltm
```

*NB. objects, above/below, altitudes, fractional day UT, UT hours/minutes*

```
(<"2 (,."1 ] 0.5 round h) ,"1 (,."1 m) ,"1 ] 1 round hmfrds DAYSECS*m) (<ix;2)} obj
)
```

*NB. round (y) to nearest (x) (e.g. 1000 round 12345)*

```
round=: [ * [: (<.) 0.5 + %~
```

*NB. sine radians*

```
sin=: 1&o.
```

*NB. sin degrees*

```
sind=: sin@rfd
```

*NB. session manager output*

```
smoutput=: 0 0 $ 1!:2&2
```

```
zetzthT0=: 3 : 0
```

*NB.\*zetzthT0 v-- epoch adjustment terms for J2000 RA DEC in degrees.*

*NB.*

*NB. monad: fT =. zetzthT0 ftYYYYMMDD*

*NB.*

*NB. zetzthT0 2028 11 13.19*

*NB.*

*NB. zetzthT0 2023 4 23 , 1988 3 20 ,: 1987 4,10 + fdfrhms 19 21 0*

*t=. gT0ymd y*

*'t2 t3'=. t (^"1 0) 2 3 NB. t^2 and t^3*

*NB. meeus (20.3) pg. 126*

*zet=. (2306.2181\*t) + (0.30188\*t2) + 0.017988\*t3*

*z=. (2306.2181\*t) + (1.09468\*t2) + 0.018203\*t3*

*th=. (2004.3109\*t) + (0.42665\*t2) + 0.041833\*t3*

*NB. insure degree result rank matches (y) rank*

*3600 %~ zet , z (,`,:.)@.(2=#\$y) th*

*)*

*NB.POST\_riseset post processor.*

*smoutput IFACE=: (0 : 0)*

*NB. (riseset) interface word(s): 20230329j102721*

*NB. -----*

*NB. iau\_tonight NB. named IAU stars visible tonight*

*NB. loadstars NB. loads riseset star data*

```
NB. riseset      NB. rise, transit, set times of stars
)

NB. smoutput 'NB. vmd: ' , , '0,p<; >q<; >0,0' (8!:2) VMDriseset

cocurrent 'base'
coinsert  'riseset'

NB. high print precision
(9!:11) 16

NB. test stars
NB. smoutput loadstars 0
```



## Index

(...)=:, 21

alldifs, 7

apparRADEC, 7

apparsecs, 8

apparsidjd0, 8

apply, 8

arccos, 8

arcsin, 8

arctan, 8

assert, 9

atan2, 9

BOM, 5

boxopen, 11

cold\_iau\_named\_stars, 11

cos, 11

cosd, 11

CR, 5

DAYSECS, 5

ddfrdms, 12

deltaT0, 12

deltaTdy, 16

dfr, 16

dmsfrdd, 16

fdfrhms, 16

gT0jd, 16

gT0ymd, 17

hmfrds, 17

iau\_tonight, 17

IFACE, 39

IFACEWORDSriset, 6

intr3p, 18

julfrcal, 19

LIMITHORZ, 6

LIMITHORZ\_riset, 22, 23

LIMITMAG, 6

LIMITMAG\_riset, 22, 23

loadstars, 20

location\_uluru, 22

location\_yellowstone, 23

meanobliquityjd0, 25

meanobliquityT0, 23

meanobliquityT1, 24

meansid0, 25

meansidjd0, 26

nnth0, 26

npth0, 26

nth0, 27

nutatation\_longitude\_dPsi, 27

OBLIQUITYDMS2000, 6

OBSLOCATION, 6

OBSLOCATION\_riset, 22, 23

parse\_iau\_named\_stars, 31

parsecsv, 32

parsetd, 33

rfd, 33

riset, 33

riset\_calc, 35

ROOTWORDSriset, 6

round, 38

sin, 38

sind, 38

smoutput, 38

STDALTITUDE, 6

UTCOFFSET, 6

UTCOFFSET\_riset, 22, 23

VMDriset, 6

zetzthT0, 38