Thermocouple model functions
for voltage, temperature, derivative and tolerance

D. A. Burkett
4 mar 04

Summary

Use these functions to calculate thermocouple temperature, voltage, tolerances and the derivative dEMF/dT for eleven thermocouple types: J, K, E, T, S, R, N, B, C, D and G. The relations for temperature and EMF come from the *Omega Complete Temperature Measurement Handbook and Encyclopedia*, 21st Century 2nd edition, 2000. I corrected a few errors in the *Handbook* functions and developed functions for thermocouple error and dEMF/dT. If you need these functions, you what to do with them, so I provide no background information.

All functions use temperatures in °C and thermal EMFs in volts. The reference junction temperature is 0°C. I use the abbreviation *TC* for thermocouple, and *EMF* for the thermal electromotive force (voltage) generated by the junction.

The model functions are corrected for the International Temperature Scale of 1990 (ITS-90). The source data can be found at *www.omega.com*. I refer to my TI-89/TI-92 Plus/Voyage 200 tip list, which can be found at *http://www.angelfire.com/realm/ti_tiplist/*.

Installing the model functions

Install the functions in any folder, unless you use the error function *tc_err*(). In that case the functions must be installed in folder *thermcpl*. *tc_err*() uses matrix *tc_errm* which also must be installed in folder *thermcpl*. *tc_err*() calls *util\casel*(), which is included in this package. *tc_err*() also calls the corresponding EMF function for the thermocouple type, for example, to find the errors for a type K TC, *tc_err*() will call *tck_e*().

All routines can be archived.

You need only install the functions for the thermocouples you use. For example, if you only use type J TCs, then you can get by with *tcj_e*(), *tcj_t*() to find EMF and temperature, and *tcj_d*() if you are interested in the derivative. The error function *tc_err*() works properly as long as you do not specify TC types for which the individual functions are not installed.

Using the model functions

Function names have the general form tc{*type*}_{*op*}(), where {*type*} is a single character to specify the TC type, and {*op*} is a single character to specify the function operation. {*type*} may be *j*, *k*, *e*, *t*, *s*, *r*, *n*, *b*, *c*, *d* or *g*. {*op*} may be *e*, *t* or *d*, to find the EMF, temperature or derivative, respectively. For example:

tck_e(100)	returns the output voltage for a type K TC at 100°C
tcj_t(.005)	returns the type J TC temperature at an EMF of 5mV
tcs_d(100)	returns the type S derivative (in V/°C) at 100°C

The functions test the input arguments against range limits. If the argument exceeds the limit, then *undef* is returned. Calling routines can compare the returned result to *undef* and take appropriate action. The range limits for temperature and EMF are shown in Table 1 below.

Most of the model functions consist of two or more functions to cover the entire range. There are discontinuities at the function boundaries, and I have made no effort to splice them smoothly. The discontinuities are usually smaller than the errors of the functions, so this should not be a problem for most uses. If you require a smooth transition from one function to another, you could try the fourth-order splice described in tip [6.56] of the TI-89 tip list. Table 2 below lists the range boundaries and the discontinuities at those boundaries.

Using the error function

The function *tc_err*() finds the error limits at a given temperature:

```
tc_err(type_string,temperature_°C,tol_string)
```

where

type_string	is a single character string which specifies the TC type, which may be upper- or lower-case, for example, "k" or "K"
temperature_°C	is the temperature in °C
tol_string	specifies the TC tolerance: "std" for standard wire, any other string for special tolerance wire.

tc_err() returns a list of the form {*temp_err*, *EMF_err*} where *temp_err* is the temperature error, and *EMF_err* is the equivalent thermoelectric voltage error. For example,

```
tc_err("k",1ØØ,"std")
```

find the errors for a standard type K TC at 100°C, and returns {2.2, 91.05E-6}, indicating that the tolerance is ±2.2°C, or ±91.05 uV. If you want to find the tolerances at a given voltage instead of temperature, use the corresponding function to convert voltage to temperature, for example

```
tc_err("j",thermcpl\tcj_t(.Ø27),"special")
```

returns the tolerances for a low-tolerance type J TC at an output of 27mV.

tc_err() can return *undef* or several error messages instead of tolerance list, as shown in this table:

tc_err() Error Messages

Result	Cause
{undef,undef}	*temperature_°C* is undef, or *temperature_°C* exceeds model limits
"tc_err fault, TC type"	*type_string* is not a string, or *type_string* is not a single character, or *type_string* is not a valid TC type
"tc_err fault, temp"	*temperature_°C* is not a number
"tc_err fault, tolerance"	*tol_string* is not a string

tc_err() calls function *util\casel*(), and uses matrix *thermcpl\tc_errm*. *util\casel*() is described in the *TI-89/TI-92 Plus/Voyage 200 Tip List* at tip [8.6]. The matrix is defined in a section below.

Thermocouple temperature ranges and errors

This table specifies the TC ranges and tolerances as they relate to the model functions.

Table 1
TC ranges and tolerances

TC Type	**Temperature range °C**	**EMF range mV**	**Standard wire Tolerance**	**Special wire Tolerance**
J	-210 1200	-8.095 69.553	T < 293.33: 2.2°C T > 293.333: 0.75%	T < 275°C: 1.1°C T > 275°C: 0.4%
K	-270 1372	-6.458 54.886	T < -110: 2% -110 < T < 293.33: 2.2°C T > 293.33: 0.75%	T < 275°C: 1.1°C T > 275°C: 0.4%
E	-270 1000	-9.835 76.373	T < -170: 1% -170 < T < 340: 1.7°C T > 340: 1.7%	T < -250: 0.4% -250 < T < 250: 1°C T > 250: 0.4%
T	-270 400	-6.258 20.872	T < -66.67: 1.5% -66.67 < T < 133.33: 1°C T > 133.33: 0.75%	T < -125: 0.4% -125 < T < 125: 0.5°C T > 125: 0.4%
S	-50 1768.1	-.23555 18.6935	T < 600: 1.5°C T > 600: 0.25%	T < 600: 0.6°C T > 600: 0.1%
R	-50 1768.1	-0.22647 21.1027	T < 600: 1.5°C T > 600: 0.25%	T < 600: 0.6°C T > 600: 0.1%
N	-270 1300	-4.345 47.513	T < -110: 2% -110 < T < 293.33: 2.2°C T > 293.33: 0.75%	T < 275: 1.1°C T > 275: 0.4%
B	0 1820	-0.0257938 13.82028 (Note 1)	T < 800°C: not specified T > 800: 0.5°C	Not available
C	-17.778 2315.556	-0.0234471 37.06598	T < 425: 4.5°C T > 425: 1%	Not available
D	-17.778 2320	-0.163494 39.5472	Not specified	Not available
G	-17.778 2315.556	-0.0158 38.5644	Not specified	Not available

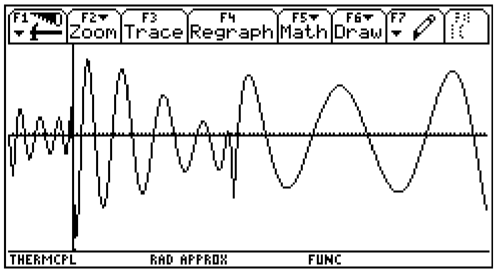
Note (1): Type B voltage as a function of temperature is non-monotonic, with a minimum of -2.584972uV at a temperature of 21.020262°C. In consequence, function *tcb_t*() has a lower range limit of about -2.579382uV, at 22°C.

Model function temperature errors, breakpoints and discontinuities

Table 2 below summarizes the error limits for the model functions T = f(EMF). The model functions for EMF = f(T) are considered 'exact', but the inverse functions are approximations. The error limits in the table are conservative, and the errors over some temperature ranges can be much less. The error for a particular temperature range can be found by plotting the error expression

```
tcy_t(tcy_e(x)) - x
```

where *y* is the thermocouple type. This plot shows the error for a type K thermocouple from -200 to 1300 °C. The y-axis range is -0.05 to 0.04 °C.



The table also summarizes the model function range breakpoints and the discontinuities at those breakpoints. In most cases the temperature discontinuities are on the order of the temperature errors, which isn't too impressive, but the functions are still useful. It does make the point, though, that these functions are more appropriate for general engineering use than precision thermometry.

Table 2
TC temperature errors, breakpoints and discontinuities

TC Type	T=f(EMF) error (°C)	T = f(EMF) breakpoints (°C)	Discontinuity at EMF breakpoint (V)	EMF = f(T) breakpoints (mV)	Discontinuity at temperature breakpoint (°C)
J	±0.05	760	7.49 E-11	0	0
				42.919	0.0675
K	±0.06	0	1.974 E-12	-5.891	0.0405
				0	0
				20.644	0.0331
E	±0.03	0	0	-8.825	0.0218
				0	0
T	±0.04	0	0	-5.603	0.0354
				0	0
S	±0.02	1064.18	9.942 E-12	1.874	0.001883
		1664.5	3.354 E-9	11.950	0.00967
				17.536	0.00130
R	±0.02	1064.18	1.7 E-14	1.923	0.00875
		1664.5	1.72 E-12	13.228	0.00456
				19.739	0.000833
N	±0.06	0	0	-3.998	0.0352
				0	0
				20.613	0.0118
B	±0.03	630.615	4.402 E-7	0.2913	0.0255
				2.4306	0.0264
C	±0.013	None	n/a	14.02	0.00204
D	±0.03	783	4.377 E-8	2.887	0.0127
				3.8277	0.00744
G	±0.04	None	n/a	0.03088	0.0113
				3.7202	0.000373

Comments on the model equations

I have implemented the functions for TC EMF as given by the Omega *Handbook*. The *Handbook* also gives functions for TC temperature as a function of EMF, but for some TC types the functions do not cover the same temperature range as the EMF functions. For those TCs I found approximating functions to provide guesses for *nSolve*() to solve for the temperature. This means that solutions will be slower for those TC types, at low temperatures.

The derivative functions *tcx_d*() use the 'exact' EMF = f(T) models, so the derivatives are dEMF/dT V/°C. To find dT/dEMF just use 1/(dEMF/dT). The derivative error limits can be estimated with

$$\left(\frac{dEMF}{dT}\right)_{min} = \frac{d}{dT}f(T - T_e) \qquad \left(\frac{dEMF}{dT}\right)_{max} = \frac{d}{dT}f(T + T_e)$$

where T_e is the temperature error at T. As an example, find the derivative limits for a standard type K TC at 500 °C. First find the temperature error with

```
tc_err("k",500,"std")[1]
```

which returns 3.75 so the temperature error is ±3.75 °C. Then find the derivative limits with

```
tck_d(500-3.75)     returns 4.2622 38 E-5
tck_d(500+3.75)     returns 4.2633 53 E-5
```

The nominal derivative at 500°C is 4.2628 33 E-5, so the worst-case limit is ±5.95 E-9 V/°C.

The equations for the minimum and maximum derivatives at a temperature arise from the fact that the nominal model function EMF = f(T) is bound above and below by functions $f_1(T)$ and $f_2(T)$, where $f_1(T) = f(T) + E_e$ and $f_2(T) = f(T) - E_e$. We want $f_1'(T)$ and $f_2'(T)$. E_e is the EMF error which we assume is the same for f_1 and f_2. Now, $E_e = f(T+T_e) - f(T)$, where T_e is the temperature error. So $f_1(T) = f(T+T_e)$, and $f_1'(T) = f'(T+T_e)$.

The temperature tolerance error is found from applying the tolerance in Table 1. The equivalent EMF errors are found by calculating the EMFs at the temperature tolerance extrema and returning the largest EMF error. For example, suppose that the temperature tolerance at temperature T is Td, then the temperature limits are T1 = T - Td and T2 = T + Td. If the EMFs corresponding to T, T1 and T2 are E, E1 and E2, respectively, then the EMF errors are | E - E1| and | E - E2 |.

Some TC types (S, R, B and C) exhibit a peak in the EMF error near the maximum temperature. While it is physically unlikely that the EMF reaches a maximum, I have not modified the calculation prevent it. The table below shows the errors and associated temperatures.

To determine the conditions for an error peak, let EMF = f(T) and define the temperature tolerance as a multiplier *n* such that Te = nT so that Te is the maximum-tolerance temperature at T. For example, if the tolerance specification is 2%, then n = 1.02. Define the EMF error as

$$EMFerr = e(T) = f(T) - f(T_e)$$

Take the derivative and set to zero to find the maximum:

$$\frac{d}{dT}e(T) = \frac{d}{dT}f(T) - \frac{d}{dT}f(nT) = 0 \qquad \text{or} \qquad \frac{d}{dT}f(T) = \frac{d}{dT}f(nT)$$

If this condition is met for some T within the operating temperature range, then there will be an error peak. The analysis is somewhat different for the type B TC, as the error is specified as a flat 0.5°C, instead of a percentage tolerance, but the basic idea is the same.

Peak EMF tolerance errors and temperatures

TC Type	Max T	EMF error at max T	EMF error peak T	EMF error at peak T
S, standard	1768.1 °C	45.8033 uV	1685.184 °C	48.8775 uV
S, special	1768.1 °C	18.2671 uV	1685.184 °C	19.5206 uV
R, standard	1768.1 °C	54.4105 uV	1689.912 °C	57.3355 uV
R, special	1768.1 °C	21.7072 uV	1688.645 °C	22.9172 uV
B	1820 °C	5.7098 uV	1634.6054 °C	5.8520 uV
C	2315.556 °C	215.1205 uV	1900.0924 °C	246.6944 uV

Matrix tc_errm description

tc_errm is an 11-row, 8-column matrix which holds the tolerances and temperature limits for all the TC types. *tc_err*() uses this matrix in place of a lot of hard-coded constants to simplify the code. Not all elements are relevant for all TC types.

tc_err matrix contents

TC Type	EAstd T>0	ERstd T>0	EAstd T<0	ERstd T<0	EAspc	ERspc	Tmin	Tmax
J	2.2	0.0075	2.2	0.0075	1.1	0.004	-210	1,200
K	2.2	0.0075	2.2	0.02	1.1	0.004	-270	1,372
E	1.7	0.005	1.7	0.01	1	0.004	-270	1,000
T	1	0.0075	1	0.015	0.5	0.004	-270	400
S	1.5	0.0025	1.5	0.0025	0.6	0.001	-50	1,768.1
R	1.5	0.0025	1.5	0.0025	0.6	0.001	-50	1,768.1
N	2.2	0.0075	2.2	0.02	1.1	0.004	-270	1,300
B	0	0	0	0	0	0	0	1,820
C	0	0	0	0	0	0	-17.78	2,315.56
D	0	0	0	0	0	0	-17.78	2,320
G	0	0	0	0	0	0	-17.78	2,315.56

The column descriptions are:

EAstd T>0	Absolute error in °C for standard wire, T > 0°C
ERstd T>0	Relative error for standard wire, T > 0°C
EAstd T<0	Absolute error in °C for standard wire, T < 0°C
ERstd T<0	Relative error for standard wire, T < 0°C
EAspc	Absolute error in °C for special limits-of-error wire
ERspc	Relative error for special limits-of-error wire

Tmin	Minimum model temperature
Tmax	Maximum model temperature

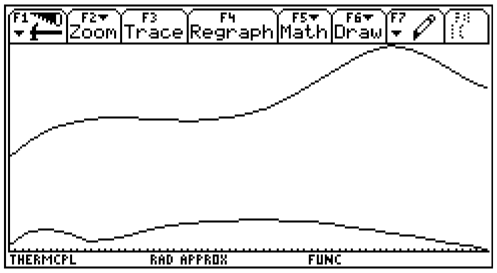
Some examples

Example 1: Compare the linearity of a type J to type K thermocouple over the range 0 to 1000°C.

A perfectly linear function would have a constant derivative, so we plot the derivatives over the desired temperature range. From the command line or in the Y= editor, set

```
tcj_d(x)→y1(x)
tck_d(x)→y2(x)
```

Set *xmin* to 0, *xmax* to 1000, then ZoomFit to get this plot:



The upper trace is the type J TC, the lower trace is the type K. The type J is more sensitive (more mV output for each °C), but the type K is closer to linear.

Example 2: A temperature measurement system is subject to noise with a peak amplitude of 30 uV at the type E TC input connector. Find the equivalent worst-case temperature error over a 200 to 400° measurement range.

Again we use the derivative function to find the sensitivity. *tct_d*(200) returns 53.15uV/°C, *tct_d*(400) returns 61.8 uV/°C, so the worst case is at 400°C. Then, (30 uV)/(61.8 uV/°C) gives a temperature uncertainty of about 0.5°C.

Example 3: Find a 2nd-order approximation T=f(EMF) for a type K transfer function over a temperature range of -10 to 100°C. Find the worst-case approximation error.

Try fitting 20 points:

```
seq(x,x,-1Ø,1ØØ,(1ØØ-¯1Ø)/19)→tlist
seq(tck_e(tlist[k]),k,1,2Ø)→elist
quadreg elist,tlist
regeq(elist)-tlist→resid
```

The minimum and maximum values of the fit residuals list *resid* are -0.161 and 0.158°C, so the worst-case approximation error is 0.161°C. The fit function coefficients are in the system variable *regcoef*.

Source code - tc_err()

Annotated source code for *tc_err*() is shown below. The code is completely straightforward with the possible exception of building the function calls for the EMF routines, to calculate the EMF error.

```
tc_err(τtype,τc,τol)
Func
©("TC type",T°C,"std" or "special")
©Return {temp_err°C,emf_err volts}
©Calls util\casel(), uses thermcpl\tc_errm
©29mar04/dburkett@infinet.com

local τerr,εmferr,τi,εa,εr,ε1,ε2,εt,τmin,τmax

© τerr          temperature error
© εmferr        EMF error
© τi            index into tc_errm matrix
© εa            absolute temperature error
© εr            relative temperature error
© ε1            EMF at τmin
© ε2            EMF at τmax
© εt            EMF at τc
© τmin          Temperature minimum error bound
© τmax          Temperature maximum error bound


© Validate input arguments. Test temperature to undef, as tcx_t() functions may return
undef.  In this case, return {undef,undef} to prevent faults when plotting, or to
propogate undef through subsequent calculations. Otherwise, ensure that TC type is a
single-character string, temperature is a number, and tolerance type is a string.

if τc=undef : return {undef,undef}
if gettype(τtype)≠"STR"
 return "tc_err fault, TC type"
if dim(τtype)≠1
 return "tc_err fault, TC type"
if gettype(τc)≠"NUM"
 return "tc_err fault, temp"
if gettype(τol)≠"STR"
 return "tc_err fault, tolerance"

© Convert tolerance and TC type to lower-case for subsequent comparisons. Convert the TC
type to index τi, and return error string if TC type not valid.

util\casel(τol)→τol
util\casel(τtype)→τype
instring("jketsrnbcdg",τtype)→τi
if τi=0
 return "tc_err fault, TC type"

© Find the minimum and maximum temperatures for the TC type and return {undef,undef} if
© temperature exceeds limits. Returning undef prevents faults when plotting errors.
thermcpl\tc_errm[τi,7]→τmin
thermcpl\tc_errm[τi,8]→τmax

if τc<τmin or τc>τmax
 return {undef,undef}

© Types J, K, E, T, S, R, and N can all be handled in the same way. Find the relative and
absolute error specifications εr and εa from the tc_errm matrix based on the wire
tolerance and temperature. Temperture error τerr is the larger of the absolute or
relative error. Find the EMF limits ε1 and ε2 from the temperature limits, then the
maximum EMF error.

if τi≤7 then
  if τol="std" then
    if τi<0 then
      thermcpl\tc_errm[τi,3]→εa
      thermcpl\tc_errm[τi,4]→εr
    else
      thermcpl\tc_errm[τi,1]→εa
      thermcpl\tc_errm[τi,2]→εr
    endif
  else
    thermcpl\tc_errm[τi,5]→εa
    thermcpl\tc_errm[τi,6]→εr
  endif

  max({εa,εr*τc})→τerr

  max({τc-τerr,τmin})→τmin
  min({τc+τerr,τmax})→τmax

  expr("thermcpl\tc"&τtype&"_e(τmin)")→ε1
  expr("thermcpl\tc"&τtype&"_e(τmax)")→ε2
  expr("thermcpl\tc"&τtype&"_e(τc)")→εt
  max(abs({ε1-εt,ε2-εt}))→εmferr

  return {τerr,εmferr}

endif

© Type B is a special case since the temperature error is a constant Ø.5C above 8ØØC.
if τtype="b" then
  if τc<8ØØ
    return {undef,undef}

  .5→τerr

  max({τc-τerr,τmin})→τmin
  min({τc+τerr,τmax})→τmax

  thermcpl\tcb_e(τmin)→ε1
  thermcpl\tcb_e(τmax)→ε2
  thermcpl\tcb_e(τc)→εt

  max(abs({ε1-εt,ε2-εt}))→εmferr

  return {τerr,εmferr}

endif

© Type C is a special case because of the error boundary point
if τtype="c" then

  when(τc<425,4.5,.Ø1*τc)→τerr

  max({τc-τerr,τmin})→τmin
  min({τc+τerr,τmax})→τmax

  thermcpl\tcc_e(τmin)→ε1
  thermcpl\tcc_e(τmax)→ε2
  thermcpl\tcc_e(τc)→εt

  max(abs({ε1-εt,ε2-εt}))→εmferr

  return {τerr,εmferr}

endif

© Error not specified for types D and G
if τtype="d" or τtype="g"
 return {undef,undef}

EndFunc
```

Source code - a typical set of model functions

I won't list the source code for all the model functions, as that would be repetitive with little benefit. However, the three functions for the type K TC show the typical structure of all the functions.

First, the function to find the TC EMF, given temperature, consists of no more than a temperature range limit test and a test for an *undef* input argument, followed by evaluation the appropriate polynomial to find the EMF. As the Omega reference functions usually return the EMF in uV, I scale the result to return volts.

```
tck_e(τc)
Func
©(temp °C) Type K TC EMF
©13febØ4/dburkett@infinet.com

if τc=undef : return undef
if τc<⁻27Ø or τc>1372 :  return undef

when(τc<Ø,polyeval({⁻1.6322697486ᴇ⁻2Ø,⁻1.9889266878ᴇ⁻17,⁻1.Ø4516Ø9365ᴇ⁻14,⁻3.1Ø88872894ᴇ⁻
12,⁻5.741Ø327428ᴇ⁻1Ø,⁻6.75Ø9Ø59173ᴇ⁻8,⁻4.99Ø4828777ᴇ⁻6,⁻3.28589Ø6784ᴇ⁻4,2.3622373598ᴇ⁻2,3
9.45Ø128Ø25,Ø},τc)*1Ø^⁻6,(polyeval({⁻1.21Ø4721275ᴇ⁻23,9.7151147152ᴇ⁻2Ø,⁻3.2Ø2Ø72ØØØ3ᴇ⁻16,
5.6Ø75Ø59Ø59ᴇ⁻13,⁻5.6Ø72844889ᴇ⁻1Ø,3.184Ø945719ᴇ⁻7,⁻9.9457592874ᴇ⁻5,1.855877ØØ32ᴇ⁻2,38.92
12Ø4975,⁻17.6ØØ413686},τc)+118.5976*e^(⁻1.183432ᴇ⁻4*(τc-126.9686)^2))*1Ø^⁻6)

EndFunc
```

Some of the functions to find temperature given EMF are more involved because the Omega reference does not give model functions at low EMFs. I get around this by using *nSolve*() with an approximating function, as shown below.

```
tck_t(εmf)
Func
©(EMF V) Type K TC temperature °C
©13febØ4/dburkett@infinet.com

local τemf,εmfuv

© Define a function to return EMF=f(t), which is just taken from tck_e(), for the
appropriate temperature range.

define τemf(τ1)=func
polyeval({⁻1.6322697486ᴇ⁻2Ø,⁻1.9889266878ᴇ⁻17,⁻1.Ø4516Ø9365ᴇ⁻14,⁻3.1Ø88872894ᴇ⁻12,⁻5.741Ø
327428ᴇ⁻1Ø,⁻6.75Ø9Ø59173ᴇ⁻8,⁻4.99Ø4828777ᴇ⁻6,⁻3.28589Ø6784ᴇ⁻4,2.3622373598ᴇ⁻2,39.45Ø128Ø2
5,Ø},τ1)*1Ø^⁻6
endfunc

© Test the EMF for undef, return same if so. Convert EMV to uV. Return undef if EMF is
out of range.

if εmf=undef : return undef
εmf*1Ø^6→εmfuv
if εmfuv<⁻6458 or εmfuv>54886
 return undef

© Omega reference does not give corresponding function for T = f(EMF) for EMF < 5891 uV,
so use nSolve() to find T. A rational polynomial estimating function gives a close guess
for nSolve(), to speed execution time and ensure the correct root is returned.

if εmfuv<⁻5891 then
return
nsolve(τemf(τ)=εmf,τ=polyeval({⁻1Ø589.5276,⁻69.Ø3562Ø3},εmf)/polyeval({1698Ø.82Ø2Ø,264.13
52642,1},εmf))
© For remaining EMF ranges just implement polynomials given by Omega reference.
```

```
elseif εmfuv<Ø then
return
polyeval({⁻5.192Ø577ᴇ⁻28,⁻1.Ø45Ø598ᴇ⁻23,⁻8.6632643ᴇ⁻2Ø,⁻3.7342377ᴇ⁻16,⁻8.977354ᴇ⁻13,⁻1.Ø8
33638ᴇ⁻9,⁻1.1662878ᴇ⁻6,2.5173462ᴇ⁻2,Ø},εmfuv)

elseif εmfuv<2Ø644 then
return
polyeval({⁻1.Ø52755ᴇ⁻35,1.Ø57734ᴇ⁻3Ø,⁻4.413Ø3Øᴇ⁻26,9.8Ø4Ø36ᴇ⁻22,⁻1.228Ø34ᴇ⁻17,8.31527ᴇ⁻14
,⁻2.5Ø3131ᴇ⁻1Ø,7.86Ø1Ø6ᴇ⁻8,2.5Ø8355ᴇ⁻2,Ø},εmfuv)

else
return
polyeval({⁻3.11Ø81Øᴇ⁻26,8.8Ø2193ᴇ⁻21,⁻9.65Ø715ᴇ⁻16,5.464731ᴇ⁻11,⁻1.646Ø31ᴇ⁻6,4.83Ø222ᴇ⁻2,
⁻131.8Ø58},εmfuv)

endif

EndFunc
```

The derivative functions are straightforward conversions of the EMF = f(T) functions. The derivatives of the estimating polynomials are coded, using the same range boundaries as the original f(T). Again, the functions return *undef* for an *undef* input, and return *undef* if the input temperature is out of range.

```
tck_d(τc)
Func
©(temp °C) Type K TC dEMF/dT
©14febØ4/dburkett@infinet.com

if τc=undef : return undef
if τc<⁻27Ø or τc>1372
 return undef

1Ø^⁻6*when(τc<Ø,polyeval({⁻1.6322697486ᴇ⁻19,⁻1.79ØØ34Ø19Ø2ᴇ⁻16,⁻8.361287492ᴇ⁻14,⁻2.176221
1Ø258ᴇ⁻11,⁻3.4446196456801ᴇ⁻9,⁻3.37545295865ᴇ⁻7,⁻1.996193151Ø8Ø1ᴇ⁻5,⁻9.857672Ø352ᴇ⁻4,.Ø47
244747196,39.45Ø128Ø25},τc),polyeval({⁻1.Ø8942491475ᴇ⁻22,7.772Ø9177216ᴇ⁻19,⁻2.24145Ø4ØØ21
ᴇ⁻15,3.3645Ø354354ᴇ⁻12,⁻2.8Ø364224445ᴇ⁻9,1.27363782876ᴇ⁻6,⁻.ØØØ298372778622,.Ø37117540Ø64
,38.9212Ø4975},τc)-.Ø28Ø7Ø4389927Ø4*(τc-126.9686)*(.99988166380228)^((τc-126.9686)^2))

EndFunc
```

Source code - casel() utility

casel() is used by *tc_err()* to convert input arguments to lower-case for subsequent testing. It must be installed in the *\util* folder.

```
casel(s)
Func
© (string) convert to lower case
© 7mayØ2/dburkett@infinet.com

local ä,ï,ÿ

""→ÿ

for ä,1,dim(s)
 ord(mid(s,ä,1))→ï
 ÿ&char(when(ï≥65 and ï≤9Ø or ï≥192 and ï≤214 or ï≥216 and ï≤223,ï+32,ï))→ÿ
endfor

return ÿ

EndFunc
```