Sumário

[var UTMScaleFactor = 0.9996; 2](#_Toc143182948)

[function DegToRad(deg) { 2](#_Toc143182949)

[function RadToDeg(rad) { 3](#_Toc143182950)

[function UTMCentralMeridian(zone) { 5](#_Toc143182951)

[function FootpointLatitude(y) { 6](#_Toc143182952)

[function MapLatLonToXY(phi, lambda, lambda0, xy) { 8](#_Toc143182953)

[function MapXYToLatLon(x, y, lambda0, philambda) { 11](#_Toc143182954)

[function LatLonToUTMXY(lat, lon, zone, xy) { 14](#_Toc143182955)

[function UTMXYToLatLon(x, y, zone, southhemi, latlon) { 15](#_Toc143182956)

[function btnToUTM\_OnClick() { 16](#_Toc143182957)

[function btnToGeographic\_OnClick() { 18](#_Toc143182958)

<script type="text/javascript">

<!--

var pi = 3.14159265358979;

/\* Ellipsoid model constants (actual values here are for WGS84) \*/

var sm\_a = 6378137.0;

var sm\_b = 6356752.314;

var sm\_EccSquared = 6.69437999013e-03;

# var UTMScaleFactor = 0.9996;

/\*

\* DegToRad

\*

\* Converts degrees to radians.

\*

\*/

# function DegToRad(deg) {

return (deg / 180.0 \* pi)

}

/\*

\* RadToDeg

\*

\* Converts radians to degrees.

\*

\*/

# function RadToDeg(rad) {

return (rad / pi \* 180.0)

}

/\*

\* ArcLengthOfMeridian

\*

\* Computes the ellipsoidal distance from the equator to a point at a

\* given latitude.

\*

\* Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,

\* GPS: Theory and Practice, 3rd ed. New York: Springer-Verlag Wien, 1994.

\*

\* Inputs:

\* phi - Latitude of the point, in radians.

\*

\* Globals:

\* sm\_a - Ellipsoid model major axis.

\* sm\_b - Ellipsoid model minor axis.

\*

\* Returns:

\* The ellipsoidal distance of the point from the equator, in meters.

\*

\*/

# function ArcLengthOfMeridian(phi) {

var alpha, beta, gamma, delta, epsilon, n;

var result;

/\* Precalculate n \*/

n = (sm\_a - sm\_b) / (sm\_a + sm\_b);

/\* Precalculate alpha \*/

alpha = ((sm\_a + sm\_b) / 2.0)

\* (1.0 + (Math.pow(n, 2.0) / 4.0) + (Math.pow(n, 4.0) / 64.0));

/\* Precalculate beta \*/

beta = (-3.0 \* n / 2.0) + (9.0 \* Math.pow(n, 3.0) / 16.0)

+ (-3.0 \* Math.pow(n, 5.0) / 32.0);

/\* Precalculate gamma \*/

gamma = (15.0 \* Math.pow(n, 2.0) / 16.0)

+ (-15.0 \* Math.pow(n, 4.0) / 32.0);

/\* Precalculate delta \*/

delta = (-35.0 \* Math.pow(n, 3.0) / 48.0)

+ (105.0 \* Math.pow(n, 5.0) / 256.0);

/\* Precalculate epsilon \*/

epsilon = (315.0 \* Math.pow(n, 4.0) / 512.0);

/\* Now calculate the sum of the series and return \*/

result = alpha

\* (phi + (beta \* Math.sin(2.0 \* phi))

+ (gamma \* Math.sin(4.0 \* phi))

+ (delta \* Math.sin(6.0 \* phi))

+ (epsilon \* Math.sin(8.0 \* phi)));

return result;

}

/\*

\* UTMCentralMeridian

\*

\* Determines the central meridian for the given UTM zone.

\*

\* Inputs:

\* zone - An integer value designating the UTM zone, range [1,60].

\*

\* Returns:

\* The central meridian for the given UTM zone, in radians, or zero

\* if the UTM zone parameter is outside the range [1,60].

\* Range of the central meridian is the radian equivalent of [-177,+177].

\*

\*/

# function UTMCentralMeridian(zone) {

var cmeridian;

cmeridian = [DegToRad](#_function_DegToRad(deg)_{)(-183.0 + (zone \* 6.0));

return cmeridian;

}

/\*

\* FootpointLatitude

\*

\* Computes the footpoint latitude for use in converting transverse

\* Mercator coordinates to ellipsoidal coordinates.

\*

\* Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,

\* GPS: Theory and Practice, 3rd ed. New York: Springer-Verlag Wien, 1994.

\*

\* Inputs:

\* y - The UTM northing coordinate, in meters.

\*

\* Returns:

\* The footpoint latitude, in radians.

\*

\*/

# function FootpointLatitude(y) {

var y\_, alpha\_, beta\_, gamma\_, delta\_, epsilon\_, n;

var result;

/\* Precalculate n (Eq. 10.18) \*/

n = (sm\_a - sm\_b) / (sm\_a + sm\_b);

/\* Precalculate alpha\_ (Eq. 10.22) \*/

/\* (Same as alpha in Eq. 10.17) \*/

alpha\_ = ((sm\_a + sm\_b) / 2.0)

\* (1 + (Math.pow(n, 2.0) / 4) + (Math.pow(n, 4.0) / 64));

/\* Precalculate y\_ (Eq. 10.23) \*/

y\_ = y / alpha\_;

/\* Precalculate beta\_ (Eq. 10.22) \*/

beta\_ = (3.0 \* n / 2.0) + (-27.0 \* Math.pow(n, 3.0) / 32.0)

+ (269.0 \* Math.pow(n, 5.0) / 512.0);

/\* Precalculate gamma\_ (Eq. 10.22) \*/

gamma\_ = (21.0 \* Math.pow(n, 2.0) / 16.0)

+ (-55.0 \* Math.pow(n, 4.0) / 32.0);

/\* Precalculate delta\_ (Eq. 10.22) \*/

delta\_ = (151.0 \* Math.pow(n, 3.0) / 96.0)

+ (-417.0 \* Math.pow(n, 5.0) / 128.0);

/\* Precalculate epsilon\_ (Eq. 10.22) \*/

epsilon\_ = (1097.0 \* Math.pow(n, 4.0) / 512.0);

/\* Now calculate the sum of the series (Eq. 10.21) \*/

result = y\_ + (beta\_ \* Math.sin(2.0 \* y\_))

+ (gamma\_ \* Math.sin(4.0 \* y\_))

+ (delta\_ \* Math.sin(6.0 \* y\_))

+ (epsilon\_ \* Math.sin(8.0 \* y\_));

return result;

}

/\*

\* MapLatLonToXY

\*

\* Converts a latitude/longitude pair to x and y coordinates in the

\* Transverse Mercator projection. Note that Transverse Mercator is not

\* the same as UTM; a scale factor is required to convert between them.

\*

\* Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,

\* GPS: Theory and Practice, 3rd ed. New York: Springer-Verlag Wien, 1994.

\*

\* Inputs:

\* phi - Latitude of the point, in radians.

\* lambda - Longitude of the point, in radians.

\* lambda0 - Longitude of the central meridian to be used, in radians.

\*

\* Outputs:

\* xy - A 2-element array containing the x and y coordinates

\* of the computed point.

\*

\* Returns:

\* The function does not return a value.

\*

\*/

# function MapLatLonToXY(phi, lambda, lambda0, xy) {

var N, nu2, ep2, t, t2, l;

var l3coef, l4coef, l5coef, l6coef, l7coef, l8coef;

var tmp;

/\* Precalculate ep2 \*/

ep2 = (Math.pow(sm\_a, 2.0) - Math.pow(sm\_b, 2.0)) / Math.pow(sm\_b, 2.0);

/\* Precalculate nu2 \*/

nu2 = ep2 \* Math.pow(Math.cos(phi), 2.0);

/\* Precalculate N \*/

N = Math.pow(sm\_a, 2.0) / (sm\_b \* Math.sqrt(1 + nu2));

/\* Precalculate t \*/

t = Math.tan(phi);

t2 = t \* t;

tmp = (t2 \* t2 \* t2) - Math.pow(t, 6.0);

/\* Precalculate l \*/

l = lambda - lambda0;

/\* Precalculate coefficients for l\*\*n in the equations below

so a normal human being can read the expressions for easting

and northing

-- l\*\*1 and l\*\*2 have coefficients of 1.0 \*/

l3coef = 1.0 - t2 + nu2;

l4coef = 5.0 - t2 + 9 \* nu2 + 4.0 \* (nu2 \* nu2);

l5coef = 5.0 - 18.0 \* t2 + (t2 \* t2) + 14.0 \* nu2

- 58.0 \* t2 \* nu2;

l6coef = 61.0 - 58.0 \* t2 + (t2 \* t2) + 270.0 \* nu2

- 330.0 \* t2 \* nu2;

l7coef = 61.0 - 479.0 \* t2 + 179.0 \* (t2 \* t2) - (t2 \* t2 \* t2);

l8coef = 1385.0 - 3111.0 \* t2 + 543.0 \* (t2 \* t2) - (t2 \* t2 \* t2);

/\* Calculate easting (x) \*/

xy[0] = N \* Math.cos(phi) \* l

+ (N / 6.0 \* Math.pow(Math.cos(phi), 3.0) \* l3coef \* Math.pow(l, 3.0))

+ (N / 120.0 \* Math.pow(Math.cos(phi), 5.0) \* l5coef \* Math.pow(l, 5.0))

+ (N / 5040.0 \* Math.pow(Math.cos(phi), 7.0) \* l7coef \* Math.pow(l, 7.0));

/\* Calculate northing (y) \*/

xy[1] = ArcLengthOfMeridian(phi)

+ (t / 2.0 \* N \* Math.pow(Math.cos(phi), 2.0) \* Math.pow(l, 2.0))

+ (t / 24.0 \* N \* Math.pow(Math.cos(phi), 4.0) \* l4coef \* Math.pow(l, 4.0))

+ (t / 720.0 \* N \* Math.pow(Math.cos(phi), 6.0) \* l6coef \* Math.pow(l, 6.0))

+ (t / 40320.0 \* N \* Math.pow(Math.cos(phi), 8.0) \* l8coef \* Math.pow(l, 8.0));

return;

}

/\*

\* MapXYToLatLon

\*

\* Converts x and y coordinates in the Transverse Mercator projection to

\* a latitude/longitude pair. Note that Transverse Mercator is not

\* the same as UTM; a scale factor is required to convert between them.

\*

\* Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,

\* GPS: Theory and Practice, 3rd ed. New York: Springer-Verlag Wien, 1994.

\*

\* Inputs:

\* x - The easting of the point, in meters.

\* y - The northing of the point, in meters.

\* lambda0 - Longitude of the central meridian to be used, in radians.

\*

\* Outputs:

\* philambda - A 2-element containing the latitude and longitude

\* in radians.

\*

\* Returns:

\* The function does not return a value.

\*

\* Remarks:

\* The local variables Nf, nuf2, tf, and tf2 serve the same purpose as

\* N, nu2, t, and t2 in MapLatLonToXY, but they are computed with respect

\* to the footpoint latitude phif.

\*

\* x1frac, x2frac, x2poly, x3poly, etc. are to enhance readability and

\* to optimize computations.

\*

\*/

# function MapXYToLatLon(x, y, lambda0, philambda) {

var phif, Nf, Nfpow, nuf2, ep2, tf, tf2, tf4, cf;

var x1frac, x2frac, x3frac, x4frac, x5frac, x6frac, x7frac, x8frac;

var x2poly, x3poly, x4poly, x5poly, x6poly, x7poly, x8poly;

/\* Get the value of phif, the footpoint latitude. \*/

phif = [FootpointLatitude](#_function_FootpointLatitude(y)_{)(y);

/\* Precalculate ep2 \*/

ep2 = (Math.pow(sm\_a, 2.0) - Math.pow(sm\_b, 2.0))

/ Math.pow(sm\_b, 2.0);

/\* Precalculate cos (phif) \*/

cf = Math.cos(phif);

/\* Precalculate nuf2 \*/

nuf2 = ep2 \* Math.pow(cf, 2.0);

/\* Precalculate Nf and initialize Nfpow \*/

Nf = Math.pow(sm\_a, 2.0) / (sm\_b \* Math.sqrt(1 + nuf2));

Nfpow = Nf;

/\* Precalculate tf \*/

tf = Math.tan(phif);

tf2 = tf \* tf;

tf4 = tf2 \* tf2;

/\* Precalculate fractional coefficients for x\*\*n in the equations

below to simplify the expressions for latitude and longitude. \*/

x1frac = 1.0 / (Nfpow \* cf);

Nfpow \*= Nf; /\* now equals Nf\*\*2) \*/

x2frac = tf / (2.0 \* Nfpow);

Nfpow \*= Nf; /\* now equals Nf\*\*3) \*/

x3frac = 1.0 / (6.0 \* Nfpow \* cf);

Nfpow \*= Nf; /\* now equals Nf\*\*4) \*/

x4frac = tf / (24.0 \* Nfpow);

Nfpow \*= Nf; /\* now equals Nf\*\*5) \*/

x5frac = 1.0 / (120.0 \* Nfpow \* cf);

Nfpow \*= Nf; /\* now equals Nf\*\*6) \*/

x6frac = tf / (720.0 \* Nfpow);

Nfpow \*= Nf; /\* now equals Nf\*\*7) \*/

x7frac = 1.0 / (5040.0 \* Nfpow \* cf);

Nfpow \*= Nf; /\* now equals Nf\*\*8) \*/

x8frac = tf / (40320.0 \* Nfpow);

/\* Precalculate polynomial coefficients for x\*\*n.

-- x\*\*1 does not have a polynomial coefficient. \*/

x2poly = -1.0 - nuf2;

x3poly = -1.0 - 2 \* tf2 - nuf2;

x4poly = 5.0 + 3.0 \* tf2 + 6.0 \* nuf2 - 6.0 \* tf2 \* nuf2

- 3.0 \* (nuf2 \* nuf2) - 9.0 \* tf2 \* (nuf2 \* nuf2);

x5poly = 5.0 + 28.0 \* tf2 + 24.0 \* tf4 + 6.0 \* nuf2 + 8.0 \* tf2 \* nuf2;

x6poly = -61.0 - 90.0 \* tf2 - 45.0 \* tf4 - 107.0 \* nuf2

+ 162.0 \* tf2 \* nuf2;

x7poly = -61.0 - 662.0 \* tf2 - 1320.0 \* tf4 - 720.0 \* (tf4 \* tf2);

x8poly = 1385.0 + 3633.0 \* tf2 + 4095.0 \* tf4 + 1575 \* (tf4 \* tf2);

/\* Calculate latitude \*/

philambda[0] = phif + x2frac \* x2poly \* (x \* x)

+ x4frac \* x4poly \* Math.pow(x, 4.0)

+ x6frac \* x6poly \* Math.pow(x, 6.0)

+ x8frac \* x8poly \* Math.pow(x, 8.0);

/\* Calculate longitude \*/

philambda[1] = lambda0 + x1frac \* x

+ x3frac \* x3poly \* Math.pow(x, 3.0)

+ x5frac \* x5poly \* Math.pow(x, 5.0)

+ x7frac \* x7poly \* Math.pow(x, 7.0);

return;

}

/\*

\* LatLonToUTMXY

\*

\* Converts a latitude/longitude pair to x and y coordinates in the

\* Universal Transverse Mercator projection.

\*

\* Inputs:

\* lat - Latitude of the point, in radians.

\* lon - Longitude of the point, in radians.

\* zone - UTM zone to be used for calculating values for x and y.

\* If zone is less than 1 or greater than 60, the routine

\* will determine the appropriate zone from the value of lon.

\*

\* Outputs:

\* xy - A 2-element array where the UTM x and y values will be stored.

\*

\* Returns:

\* The UTM zone used for calculating the values of x and y.

\*

\*/

# function LatLonToUTMXY(lat, lon, zone, xy) {

MapLatLonToXY(lat, lon, UTMCentralMeridian(zone), xy);

/\* Adjust easting and northing for UTM system. \*/

xy[0] = xy[0] \* UTMScaleFactor + 500000.0;

xy[1] = xy[1] \* UTMScaleFactor;

if (xy[1] < 0.0)

xy[1] = xy[1] + 10000000.0;

return zone;

}

/\*

\* UTMXYToLatLon

\*

\* Converts x and y coordinates in the Universal Transverse Mercator

\* projection to a latitude/longitude pair.

\*

\* Inputs:

\* x - The easting of the point, in meters.

\* y - The northing of the point, in meters.

\* zone - The UTM zone in which the point lies.

\* southhemi - True if the point is in the southern hemisphere;

\* false otherwise.

\*

\* Outputs:

\* latlon - A 2-element array containing the latitude and

\* longitude of the point, in radians.

\*

\* Returns:

\* The function does not return a value.

\*

\*/

# function UTMXYToLatLon(x, y, zone, southhemi, latlon) {

var cmeridian;

x -= 500000.0;

x /= [UTMScaleFactor](#_var_UTMScaleFactor_=);

/\* If in southern hemisphere, adjust y accordingly. \*/

if (southhemi)

y -= 10000000.0;

y /= [UTMScaleFactor](#_var_UTMScaleFactor_=);

cmeridian = [UTMCentralMeridian](#_function_UTMCentralMeridian(zone)_{)(zone);

[MapXYToLatLon](#_function_MapXYToLatLon(x,_y,)(x, y, cmeridian, latlon);

return;

}

/\*

\* btnToUTM\_OnClick BOTAO <<

\*

\* Called when the btnToUTM button is clicked.

\*

\*/

# function btnToUTM\_OnClick() {

var xy = new Array(2);

if (isNaN(parseFloat(document.frmConverter.txtLongitude.value))) {

alert("Por favor, digite uma longitude válida.");

return false;

}

lon = parseFloat(document.frmConverter.txtLongitude.value);

if ((lon < -180.0) || (180.0 < lon)) {

alert("A longitude que você digitou não é válida. " +

"Por favor digite um valor entre [-180, 180].");

return false;

}

if (isNaN(parseFloat(document.frmConverter.txtLatitude.value))) {

alert("Por favor, digite uma latitude válida.");

return false;

}

lat = parseFloat(document.frmConverter.txtLatitude.value);

if ((lat < -90.0) || (90.0 < lat)) {

alert("A latitude que você digitou não é válida. " +

"Por favor, digite um valor entre [-90, 90].");

return false;

}

// Compute the UTM zone.

zone = Math.floor((lon + 180.0) / 6) + 1;

zone = LatLonToUTMXY(DegToRad(lat), DegToRad(lon), zone, xy);

/\* Set the output controls. \*/

document.frmConverter.txtX.value = xy[0];

document.frmConverter.txtY.value = xy[1];

document.frmConverter.txtZone.value = zone;

if (lat < 0)

// Set the S button.

document.frmConverter.rbtnHemisphere[1].checked = true;

else

// Set the N button.

document.frmConverter.rbtnHemisphere[0].checked = true;

return true;

}

/\*

\* btnToGeographic\_OnClick BOTAO >>

\*

\* Called when the btnToGeographic button is clicked.

\*

\*/

# function btnToGeographic\_OnClick() {

latlon = new Array(2);

var x, y, zone, southhemi;

if (isNaN(parseFloat(document.frmConverter.txtX.value))) {

alert("Por favor, digite um valor válido para a coordenada X.");

return false;

}

x = parseFloat(document.frmConverter.txtX.value);

if (isNaN(parseFloat(document.frmConverter.txtY.value))) {

alert("Por favor, digite um valor válido para a coordenada Y.");

return false;

}

y = parseFloat(document.frmConverter.txtY.value);

if (isNaN(parseInt(document.frmConverter.txtZone.value))) {

alert("Por favor, digite um valor válido para a fuso UTM.");

return false;

}

zone = parseFloat(document.frmConverter.txtZone.value);

if ((zone < 1) || (60 < zone)) {

alert("A fuso UTM digitada não é válida. " +

"Por favor, digite um valor entre [1, 60], normalmente 23.");

return false;

}

if (document.frmConverter.rbtnHemisphere[1].checked)

southhemi = true;

else

southhemi = false;

[UTMXYToLatLon](#_function_UTMXYToLatLon(x,_y,)(x, y, zone, southhemi, latlon);

document.frmConverter.txtLatitude.value = [RadToDeg](#_function_RadToDeg(rad)_{)(latlon[0]);

document.frmConverter.txtLongitude.value = [RadToDeg](#_function_RadToDeg(rad)_{)(latlon[1]);

if (window.opener && window.opener.document) {

var txtLat = window.opener.document.getElementById(get('latTxt'));

var txtLon = window.opener.document.getElementById(get('lonTxt'));

if(txtLat && txtLon){

txtLat.value = RadToDeg(latlon[0]).toFixed(8).replace('.', ',');

txtLon.value = RadToDeg(latlon[1]).toFixed(8).replace('.', ',');

}

}

return true;

}

function get(name) {

if(name=(new RegExp('[?&]'+encodeURIComponent(name)+'=([^&]\*)')).exec(location.search))

return decodeURIComponent(name[1]);

}

// CONVERSOR GMS PARA GEOGRAFICA

//BOTÃO >>

function btnGMSParaGeo\_OnClick() {

var graulat = parseInt(document.frmConverter.LatGrau.value);

var minutolat = parseInt(document.frmConverter.LatMin.value);

var segundolat = parseFloat(document.frmConverter.LatSeg.value);

var graulon = parseInt(document.frmConverter.LonGrau.value);

var minutolon = parseInt(document.frmConverter.LonMin.value);

var segundolon = parseFloat(document.frmConverter.LonSeg.value);

var consistido = false;

if (!isNaN(graulat) && !isNaN(minutolat) && !isNaN(segundolat) && !isNaN(graulon) && !isNaN(minutolon) && !isNaN(segundolon)) {

if (!EstaForaDoIntervalo(graulat, 0, 90) && !EstaForaDoIntervalo(minutolat, 0, 60) && !EstaForaDoIntervalo(segundolat, 0, 60) && !EstaForaDoIntervalo(graulon, 0, 180) && !EstaForaDoIntervalo(minutolat, 0, 60) && !EstaForaDoIntervalo(segundolat, 0, 60)) {

consistido = true;

//Acerta o Sinal da Latitude e Longitude

var latitudeconvertida = CauculaLatitude(graulat, minutolat, segundolat, document.frmConverter.LatQuad.selectedIndex == 0 ? -1 : 1);

frmConverter.txtLatitudeN.value = latitudeconvertida.toFixed(8);

var longitudeconvertida = CauculaLongitude(graulon, minutolon, segundolon, document.frmConverter.LonQuad.selectedIndex == 0 ? -1 : 1);

frmConverter.txtLongitudeN.value = longitudeconvertida.toFixed(8);

//Coloca os valores no formulário chamador

if (window.opener && window.opener.document) {

var txtLat = window.opener.document.getElementById(get('latTxt'));

var txtLon = window.opener.document.getElementById(get('lonTxt'));

if (txtLat && txtLon) {

txtLat.value = latitudeconvertida.toFixed(8).replace('.',',');

txtLon.value = longitudeconvertida.toFixed(8).replace('.',',');

}

}

}

}

if (consistido == false) {

EmiteOAlertaGeral();

LimparCamposCoordenadas();

}

}

//BOTÃO <<

function btnGeoParaGMS\_OnClick() {

var latitude = parseFloat(document.frmConverter.txtLatitudeN.value);

var longitude = parseFloat(document.frmConverter.txtLongitudeN.value);

var consistido = false;

if(!isNaN(latitude) && !isNaN(longitude))

{

if (!EstaForaDoIntervalo(latitude, -90, 90) && !EstaForaDoIntervalo(longitude, -180,180))

{

consistido = true;

var latsinal=1;

var lonsinal=1;

if (latitude < 0) {latsinal = -1; }

if (longitude < 0) {lonsinal = -1; }

var latitudeabs = Math.abs(Math.round(latitude \* 1000000.));

var longitudeabs = Math.abs(Math.round(longitude \* 1000000.));

document.frmConverter.LatGrau.value = Math.floor(latitudeabs / 1000000);

document.frmConverter.LatMin.value = Math.floor(((latitudeabs / 1000000) - Math.floor(latitudeabs / 1000000)) \* 60);

document.frmConverter.LatSeg.value = (Math.floor(((((latitudeabs / 1000000) - Math.floor(latitudeabs / 1000000)) \* 60) - Math.floor(((latitudeabs / 1000000) - Math.floor(latitudeabs / 1000000)) \* 60)) \* 100000) \* 60 / 100000).toFixed(8);

document.frmConverter.LonGrau.value = Math.floor(longitudeabs / 1000000);

document.frmConverter.LonMin.value = Math.floor(((longitudeabs / 1000000) - Math.floor(longitudeabs / 1000000)) \* 60);

document.frmConverter.LonSeg.value = (Math.floor(((((longitudeabs / 1000000) - Math.floor(longitudeabs / 1000000)) \* 60) - Math.floor(((longitudeabs / 1000000) - Math.floor(longitudeabs / 1000000)) \* 60)) \* 100000) \* 60 / 100000).toFixed(8);

document.frmConverter.LatQuad.selectedIndex = 0;

if (latsinal == 1)

document.frmConverter.LatQuad.selectedIndex = 1;

document.frmConverter.LonQuad.selectedIndex = 0;

if (lonsinal == 1)

document.frmConverter.LonQuad.selectedIndex = 1;

}

}

if (consistido == false) {

//Limpar Campos Coordenadas

document.frmConverter.txtLatitudeN.value = "";

document.frmConverter.txtLongitudeN.value = "";

var aviso = "As coordenadas que você digitou não são válidas\n";

aviso = aviso + "Digite um valor entre -90 e +90 para latitude\n";

aviso = aviso + "e um valor entre -180 e +180 para longitude.\n";

alert(aviso);

}

};

function CauculaLatitude(graulat, minutolat, seglat, latsinal)

{

var latitude = (graulat + (minutolat / 60.) + (seglat / 3600.)) \* latsinal;

return (latitude);

}

function CauculaLongitude(graulon, minutolon, seglon, lonsinal)

{

var longitude = (graulon + (minutolon / 60.) + (seglon / 3600.)) \* lonsinal;

return (longitude);

}

function ConsisteLatGrau() {

const minimo = 0;

const maximo = 90;

var coordenada = "Latitude";

aviso = "A " + coordenada + " que você digitou não é válida.\n";

ehnumerico = !isNaN(parseFloat(document.frmConverter.LatGrau.value));

if (ehnumerico) {

valordigitado = parseFloat(document.frmConverter.LatGrau.value);

ConsisteValores(valordigitado, minimo, maximo, coordenada, aviso);

}

if (!ehnumerico && document.frmConverter.LatGrau.value != "") {

EmiteOAlerta(coordenada, minimo, maximo, aviso);

}

}

function ConsisteLonGrau()

{

const minimo = 0;

const maximo = 180;

coordenada = "Longitude";

aviso = "A " + coordenada + " que você digitou não é válida.\n";

ehnumerico = !isNaN(parseFloat(document.frmConverter.LonGrau.value));

if (ehnumerico)

{

var valordigitado = parseFloat(document.frmConverter.LonGrau.value);

ConsisteValores(valordigitado, minimo, maximo, coordenada, aviso);

}

if (!ehnumerico && document.frmConverter.LonGrau.value != "") {

EmiteOAlerta(coordenada, minimo, maximo, aviso);

}

}

function ConsisteLatMin() {

var aviso = "";

var coordenada = "Minuto";

var ehnumerico = !isNaN(parseFloat(document.frmConverter.LatMin.value));

if (ehnumerico) {

var valordigitado = parseFloat(document.frmConverter.LatMin.value);

ConsisteMinESeg(valordigitado, coordenada);

}

if (!ehnumerico && document.frmConverter.LatMin.value != "") {

EmiteOAlerta(coordenada, 0, 60, aviso);

}

}

//----------------------------------------------------------------------

function ConsisteLatSeg() {

var aviso = "";

const coordenada = "Segundo";

ehnumerico = !isNaN(parseFloat(document.frmConverter.LatSeg.value));

if (ehnumerico) {

valordigitado = parseFloat(document.frmConverter.LatSeg.value);

ConsisteMinESeg(valordigitado, coordenada);

}

if (!ehnumerico && document.frmConverter.LatSeg.value != "") {

EmiteOAlerta(coordenada, 0, 60, aviso);

}

}

//----------------------------------------------------------------------

function ConsisteLonMin() {

var aviso = "";

const coordenada = "Minuto";

ehnumerico = !isNaN(parseFloat(document.frmConverter.LonMin.value));

if (ehnumerico) {

valordigitado = parseFloat(document.frmConverter.LonMin.value);

ConsisteMinESeg(valordigitado, coordenada);

}

if (!ehnumerico && document.frmConverter.LonMin.value != "") {

EmiteOAlerta(coordenada, 0, 60, aviso);

}

}

//------------------------------------------------------------------------

function ConsisteLonSeg() {

var aviso = "";

const coordenada = "Segundo";

var ehnumerico = !isNaN(parseFloat(document.frmConverter.LonSeg.value));

if (ehnumerico) {

valordigitado = parseFloat(document.frmConverter.LonSeg.value);

ConsisteMinESeg(valordigitado,coordenada);

}

if (!ehnumerico && document.frmConverter.LonSeg.value != "") {

EmiteOAlerta(coordenada, 0, 60, aviso);

}

}

//-------------------------------------------------------------------------

function ConsisteMinESeg(valordigitado,coordenada) {

minimo = 0;

maximo = 60;

aviso = "";

ConsisteValores(valordigitado, minimo, maximo, coordenada, aviso)

}

function ConsisteValores(valordigitado,minimo,maximo,coordenada,aviso)

{

estaforadorange = EstaForaDoIntervalo(valordigitado, minimo, maximo);

if (estaforadorange)

{

EmiteOAlerta(coordenada, minimo, maximo,aviso);

}

}

function EmiteOAlerta(coordenada, minimo, maximo,aviso)

{

if (aviso == "") {

avisopadrao = "O " + coordenada + " que você digitou não é válido. \n";

aviso = avisopadrao;

}

agrad = "Por favor digite um valor entre [" + minimo + "," + maximo + "].";

alert(aviso.concat(agrad));

}

function EmiteOAlertaGeral()

{

aviso = "As coordenadas que você digitou não são válidas\n";

aviso = aviso + "Digite um valor entre -90 e +90 para latitude\n";

aviso = aviso + "um valor entre -180 e +180 para longitude\n";

aviso = aviso + "e valores entre 0 e 60 para minutos e segundos\n";

alert(aviso);

}

function EmiteOAlertaSimples() {

aviso = "As coordenadas que você digitou não são válidas\n";

aviso = aviso + "Digite um valor entre -90 e +90 para latitude\n";

aviso = aviso + "e um valor entre -180 e +180 para longitude\n";

alert(aviso);

}

function EstaForaDoIntervalo(valor, minimo, maximo)

{

estaforadorange = ((valor < minimo || valor > maximo) ? true : false);

return estaforadorange;

}

function LimparCamposCoordenadas()

{

document.frmConverter.LatGrau.value = "";

document.frmConverter.LatMin.value = "";

document.frmConverter.LatSeg.value = "";

document.frmConverter.LonGrau.value = "";

document.frmConverter.LonMin.value = "";

document.frmConverter.LonSeg.value = "";

}

//Se Hemisfério Ocidental Longitude Negativa

//Para longitudes negativas setar hemisfério ocidental

//

//Mesmo para valores positivos o checkbox será mantido por default nos Hemisférios Ocidente e Sul

//Se o usuario digitar valores positivos e atualizar o checkbox para os hemisférios

//Oriente e/ou Norte, o checkbox deve ser atualizado para estas coordenadas e convertido o valor

//para estas coordenadas.

//

function AcertaOCheckBoxDaLatitude(graulat)

{

var sinaldalatitude = 1;

//Verifica se o grau digitado pelo usuario é negativo e acerta o checkbox e o sinal de acordo

if (graulat < 0) {

sinaldalatitude = -1;

document.frmConverter.rbtnNorteSul[1].checked = true;

}

//Verifica se o checkbox está marcado como sul e acerta o sinal da longitude para negativo

var sulflegado = document.frmConverter.rbtnNorteSul[1].checked;

if (sulflegado) { sinaldalatitude = -1; };

return sinaldalatitude;

}

function AcertaOCheckBoxDaLongitude(graulon) {

var sinaldalongitude = -1;

//Verifica se o grau digitado pelo usuario é negativo

if (graulon < 0) {

sinaldalongitude = -1;

}

return sinaldalongitude;

}

function MostraTabUTMparaGeo() {

document.getElementById('tbUTMparaGeo').style.display = 'block';

document.getElementById('tbGMSparaGeo').style.display = 'none';

}

function MostraTabGMSparaGeo() {

document.getElementById('tbGMSparaGeo').style.display = 'block';

document.getElementById('tbUTMparaGeo').style.display = 'none';

}

// -->

</script>