Subject: Re: UTM Mapping support
Posted by aspinelli on Thu, 22 Aug 1996 07:00:00 GMT
View Forum Message <> Reply to Message

On 19 Aug 1996 22:37:51 GMT, you wrote:

> I have a need for inputting UTM based data sets into IDL's mapping
> utilities. The problem is that IDL does not directly support UTM
> in it's map utilites. Does anyone have any routines for converting
> from UTM to Lat/Lon and vica versa?

> Arno Granados
> Positive Systems Inc.
> Leaders in Digital Aerial Photography
> granados@possys.com

I have the following (precious!) code, which makes Greenwich
to UTM. Usage is

result = Gre2Utm(latitude_greenwich, longitude_greenwich, utm_fuse)

The fuse is the UTM fuse; Italy is contained in fuses 32 AND 33;
I do not know yours, but you have to find it...
Fuse may be deduced from latitude, but it is better left as a
parameter,
since sometimes you want a large map, with several fuses on it,
and you want to just choose one fuse (projection on different
fuses do not overlap at the margin).

Result is an array of 2 doubles containing UTM coordinates
(lat,lon) in ***metres***. If you want km, you multiply
by 0.001, of course :-)

I have no code in the oppsite direction, so at least half of your
problem is still unsolved. However, the optimistic view
is that half of the problem is solved!

Comments are in Italian, but they are anyway useless.

If you find the UTM-to-Greenwich side, please let me know!

Happy IDLing
Andrea

```
function gre2utm, latitudine, longitudine, fuso

r = dblarr( 2 )
```

```
kRadian = !DPI / 180d0

ke2  =  0.0067681702D0 ; // seconda eccentricita'
kC  =  6397376.633D0 ; // raggio polare

; coefficienti sviluppo trasformazione diretta
kA1  =  111092.0821D0 / kRadian ;
kA2  = -16100.59187D0
kA4  =  16.96942D0
kA6  = -0.02226D0


lambda0 = (3.0 + 6.0 * (fuso-31)) * kRadian
x0 = 500000;
lambda = longitudine* kRadian;
fi = latitudine * kRadian;
lambda1 = lambda - lambda0;
cosfi = cos( fi );
tanfi = tan( fi );
v = sqrt( 1 + ke2 * cosfi * cosfi );
csi = atan( tanfi / cos( lambda1 * v ) );
coscsi = cos( csi );
u = sqrt( 1 + ke2 * coscsi * coscsi );
xx = coscsi * tan( lambda1 ) / u;

x = kC * alog( xx + sqrt( xx*xx + 1 )) + x0
y = kA1*csi + kA2*sin( 2*csi ) + kA4*sin( 4*csi ) + kA6*sin( 6*csi );

r(0) = y
r(1) = x

return, r

end
```

Andrea Spinelli - Ismes Spa. 9, Via Pastrengo, 24068 Seriate BG, Italy
tel.: +39-35-307777 fax.: +39-35-302999 e-mail: aspinelli@ismes.it
"Truth hurts, but pimples much more"