1 Introduction

This is the documentation for the geodetices files of the paparazzi project (paparazzi.nongnu.org).
It should be a reference for the functions which are defined in the directory `(paparazzi)/sw/airborne/math`.

Martin: "*Still missing: hmsl and the gc_of_gd_lat_d conversion*"

Contents

2 ECEF

Earth-Centered-Earth-Fixed coordinates belong to the easiest coordinates to compute, because they have a fixed frame and they don't have to regard the non-spherical shape of the earth. Though they are not intuitiv.

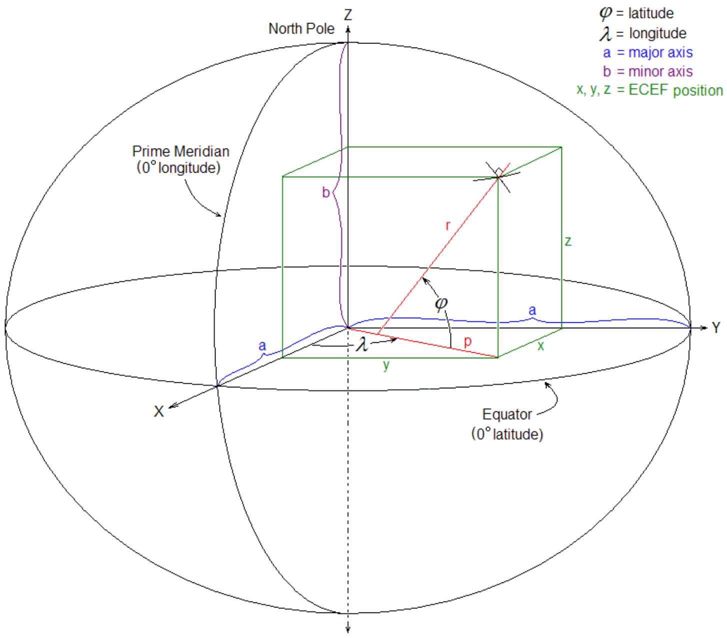

Fig. 1: ECEF coordinates

ECEF coordinates are defined using

$$p_{ECEF} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1}$$

available for the following simple types

- int32_t - `EcefCoor_i`
- float - `EcefCoor_f`
- double - `EcefCoor_d`

2.1 Transformations from ECEF

to LTP

Gets the LLA-coordinates and a transformation matrix from ECEF to ENU out of ECEF coordinates.

First, the LLA coordinates (WGS-84) are computed from the ECEF coordinates.

With the LLA-coordinates it is possible to construct a rotational matrix.

$$\mathbf{R}_{ecef2enu} = \begin{pmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{pmatrix} \tag{2}$$

Function `ltp_def_from_ecef_i(LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ltp_def_from_ecef_f(LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ltp_def_from_ecef_d(LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

to LLA

Generating LLA coordinates is made using the following calculations. These refer to [?] or [?](pages 31-33).

$$a = 6,378,137 \tag{3}$$
$$f = \frac{1}{298.257223563} \tag{4}$$

$$b = a \cdot (1 - f) \tag{5}$$
$$e^2 = \sqrt{2f - f^2} \tag{6}$$
$$e' = e\frac{a}{b} \tag{7}$$
$$E^2 = a^2 - b^2 \tag{8}$$
$$r = \sqrt{x^2 + y^2} \tag{9}$$
$$F = 54b^2z^2 \tag{10}$$
$$G = r^2 + (1 - e^2)z^2 - e^2E^2 \tag{11}$$
$$c = \frac{e^4Fr^2}{G^3} \tag{12}$$
$$s = \sqrt[3]{1 + c + \sqrt{c^2 + 2c}} \tag{13}$$
$$P = \frac{F}{3\left(s + \frac{1}{s} + 1\right)^2 G^2} \tag{14}$$
$$Q = \sqrt{1 + 2e^4P} \tag{15}$$
$$r_0 = -\frac{Pe^2r}{1 + Q} + \sqrt{\frac{1}{2}a^2\left(1 + \frac{1}{Q}\right) - \frac{P(1 - e^2)z^2}{Q(1 + Q)} - \frac{1}{2}Pr^2} \tag{16}$$
$$U = \sqrt{(r - e^2r_0)^2 + z^2} \tag{17}$$
$$V = \sqrt{(r - e^2r_0)^2 + (1 - e^2)z^2} \tag{18}$$
$$z_0 = \frac{b^2z}{aV} \tag{19}$$
$$\varphi = \arctan\left(\frac{z + (e')^2z_0}{r}\right) \tag{20}$$
$$\lambda = atan2(y, x) \tag{21}$$
$$h = U\left(\frac{b^2}{aV} - 1\right) \tag{22}$$

Function `lla_of_ecef_i(LlaCoor_i* out, EcefCoor_i* in)`
in File `pprz_geodetic_int.c`
Function `lla_of_ecef_f(LlaCoor_f* out, EcefCoor_f* in)`

in File `pprz_geodetic_float.c`
Function `lla_of_ecef_d(LlaCoor_d* lla, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

to NED/ENU

With a know transformation matrix (see section ??) it is quite easy to rotate a vector into the ENU frame:

$$\vec{v}_{ENU} = \mathbf{R}_{ecef2enu}\vec{v}_{ECEF} \tag{23}$$

For a transformation into the NED-frame you have to do an additional ENU/NED-transformation.
Function `enu_of_ecef_vect_i(EnuCoor_i* enu, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ned_of_ecef_vect_i(NedCoor_i* ned, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `enu_of_ecef_vect_f(EnuCoor_f* enu, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ned_of_ecef_vect_f(NedCoor_f* ned, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `enu_of_ecef_vect_d(EnuCoor_d* enu, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`
Function `ned_of_ecef_vect_d(NedCoor_d* ned, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

The transformation of a point is very similiar. Instead of a point you use a difference vector between the desired point p_d and the center of the local tangent plane p_0.

$$\vec{v}_{ECEF} = p_d - p_0 \tag{24}$$

Function `enu_of_ecef_point_i(EnuCoor_i* enu, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ned_of_ecef_point_i(NedCoor_i* ned, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `enu_of_ecef_point_f(EnuCoor_f* enu, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ned_of_ecef_point_f(NedCoor_f* ned, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `enu_of_ecef_point_d(EnuCoor_d* enu, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`
Function `ned_of_ecef_point_d(NedCoor_d* ned, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

2.2 Transformations to ECEF

from LLA

Calculating the ECEF coordinates out of LLA coordinates is a slightly easier task than the other way round. The following way refers to [?]. With the known constants

$$a = 6,378,137 \tag{25}$$

$$f = \frac{1}{298.257223563} \tag{26}$$

$$e^2 = \sqrt{2f - f^2} \tag{27}$$

the value

$$\chi = \sqrt{1 - e^2 \sin^2 \varphi} \tag{28}$$

can be precomputed and used in

$$x = \left(\frac{a}{\chi} + h\right) \cos\varphi \cos\lambda \tag{29}$$

$$y = \left(\frac{a}{\chi} + h\right) \cos\varphi \sin\lambda \tag{30}$$

$$z = \left(\frac{a}{\chi}(1 - e^2) + h\right) \sin\varphi \tag{31}$$

Function `ecef_of_lla_i(EcefCoor_i* out, LlaCoor_i* in)`
in File `pprz_geodetic_int.c`
Function `ecef_of_lla_f(EcefCoor_f* out, LlaCoor_f* in)`
in File `pprz_geodetic_float.c`
Function `ecef_of_lla_d(EcefCoor_d* ecef, LlaCoor_d* lla)`
in File `pprz_geodetic_double.c`

from NED/ENU

With a know transformation matrix (see section **??**) it is quite easy to rotate a vector from the ENU-frame to the ECEF-frame. Since

$$\mathbf{R}_{enu2ecef} = \mathbf{R}_{ecef2enu}^{-1} = \mathbf{R}_{ecef2enu}^{T} \tag{32}$$

a transformation is done as follows

$$\overrightarrow{v}_{ECEF} = \mathbf{R}_{enu2ecef} \overrightarrow{v}_{ENU} \tag{33}$$

For a transformation from the NED-frame you have to do an additional ENU/NED-transformation before.
Function `ecef_of_enu_vect_d(EcefCoor_d* ecef, LtpDef_d* def, EnuCoor_d* enu)`
in File `pprz_geodetic_double.c`
Function `ecef_of_ned_vect_d(EcefCoor_d* ecef, LtpDef_d* def, NedCoor_d* ned)`
in File `pprz_geodetic_double.c`

The transformation of a point is very similiar. After transforming into the ECEF-frame you add the position of the center p_0 to the result.

$$\overrightarrow{p}_{ECEF} = p + p_0 \tag{34}$$

Function `ecef_of_enu_point_d(EcefCoor_d* ecef, LtpDef_d* def, EnuCoor_d* enu)`
in File `pprz_geodetic_double.c`
Function `ecef_of_ned_point_d(EcefCoor_d* ecef, LtpDef_d* def, NedCoor_d* ned)`
in File `pprz_geodetic_double.c`

3 LLA

LLA coordinates (longitude, lattitude, attitude) are the more intuitive way to express a position on the earth's surface. The values "longitude" and "lattitude" are angles and therefore in degrees or radians and the "attitude" is in meters above the surface of the earth's approximated shape.

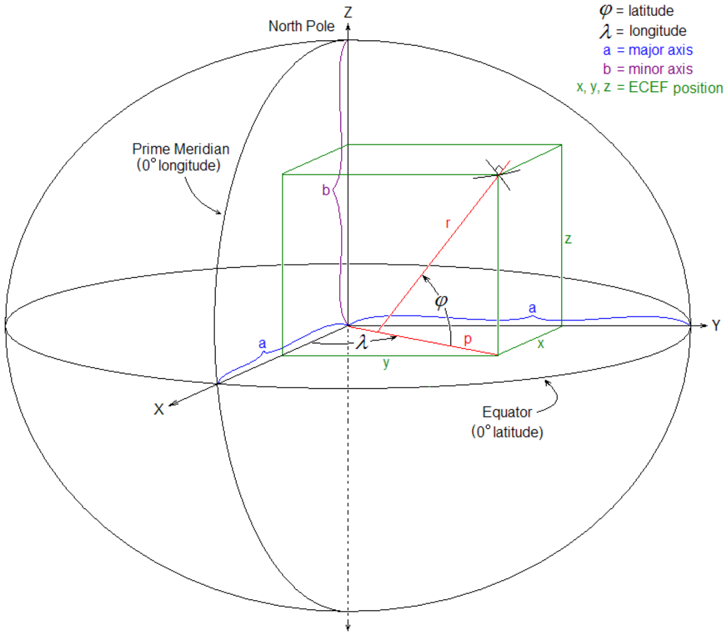

Fig. 2: LLA coordinates

LLA coordinates are defined using

$$p_{LLA} = \begin{pmatrix} \varphi \\ \lambda \\ h \end{pmatrix} \tag{35}$$

available for the following simple types

- int32_t - `LlaCoor_i`
- float - `LlaCoor_f`
- double - `LlaCoor_d`

3.1 Simple Operations

Assigning

It is either possible to assign every single value of LLA-coordinate

$$pos = \begin{pmatrix} \varphi \\ \lambda \\ h \end{pmatrix} \tag{36}$$

Function `LLA_ASSIGN(pos, lat, lon, alt)`
in File `pprz_geodetic.h`

or to copy one coordinate to another

```
pos1 = pos2
```

Function `LLA_COPY(pos1, pos2)`
in File `pprz_geodetic.h`

3.2 Trasnformation from LLA

to ECEF

Calculating the ECEF coordinates out of LLA coordinates is a slightly easier task than the other way round. The following way refers to [?]. With the known constants

$$a = 6,378,137 \tag{37}$$

$$f = \frac{1}{298.257223563} \tag{38}$$

$$e^2 = \sqrt{2f - f^2} \tag{39}$$

the value

$$\chi = \sqrt{1 - e^2 \sin^2 \varphi} \tag{40}$$

can be precomputed and used in

$$x = \left(\frac{a}{\chi} + h\right) \cos\varphi \cos\lambda \tag{41}$$

$$y = \left(\frac{a}{\chi} + h\right) \cos\varphi \sin\lambda \tag{42}$$

$$z = \left(\frac{a}{\chi}(1 - e^2) + h\right) \sin\varphi \tag{43}$$

Function `ecef_of_lla_i(EcefCoor_i* out, LlaCoor_i* in)`
in File `pprz_geodetic_int.c`
Function `ecef_of_lla_f(EcefCoor_f* out, LlaCoor_f* in)`
in File `pprz_geodetic_float.c`
Function `ecef_of_lla_d(EcefCoor_d* ecef, LlaCoor_d* lla)`
in File `pprz_geodetic_double.c`

3.3 Transforming to LLA

from ECEF

Generating LLA coordinates is made using the following calculations. These refer to [?] or [?](pages 31-33).

$$a = 6,378,137 \tag{44}$$

$$f = \frac{1}{298.257223563} \tag{45}$$

$$
\begin{aligned}
b &= a \cdot (1-f) && (46)\\
e^2 &= \sqrt{2f - f^2} && (47)\\
e' &= e\frac{a}{b} && (48)\\
E^2 &= a^2 - b^2 && (49)\\
r &= \sqrt{x^2 + y^2} && (50)\\
F &= 54b^2z^2 && (51)\\
G &= r^2 + (1-e^2)z^2 - e^2E^2 && (52)\\
c &= \frac{e^4Fr^2}{G^3} && (53)\\
s &= \sqrt[3]{1 + c + \sqrt{c^2 + 2c}} && (54)\\
P &= \frac{F}{3\left(s + \frac{1}{s} + 1\right)^2 G^2} && (55)\\
Q &= \sqrt{1 + 2e^4P} && (56)\\
r_0 &= -\frac{Pe^2r}{1+Q} + \sqrt{\frac{1}{2}a^2\left(1 + \frac{1}{Q}\right) - \frac{P(1-e^2)z^2}{Q(1+Q)} - \frac{1}{2}Pr^2} && (57)\\
U &= \sqrt{(r - e^2r_0)^2 + z^2} && (58)\\
V &= \sqrt{(r - e^2r_0)^2 + (1-e^2)z^2} && (59)\\
z_0 &= \frac{b^2z}{aV} && (60)\\
\varphi &= \arctan\left(\frac{z + (e')^2z_0}{r}\right) && (61)\\
\lambda &= atan2(y, x) && (62)\\
h &= U\left(\frac{b^2}{aV} - 1\right) && (63)
\end{aligned}
$$

Function `lla_of_ecef_i(LlaCoor_i* out, EcefCoor_i* in)`
in File `pprz_geodetic_int.c`
Function `lla_of_ecef_f(LlaCoor_f* out, EcefCoor_f* in)`
in File `pprz_geodetic_float.c`
Function `lla_of_ecef_d(LlaCoor_d* lla, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

4 LTP

The Local Tangent Plane (LTP) is an approximation of the earth at a fixed position. It is defined using an ECEF position, a LLA position and a rotational matrix to convert between them. *Please note that the matrix transforms from ECEF to ENU!*

```
LtpDef   = {      ecef      lla      ltp_of_ecef     }
```

It is available for the following simple types:

sinmple type	struct name	p_{ECEF}	p_{LLA}	$\mathbf{R}_{ltp\_of\_ecef}$
int32_t	LtpDef_i	EcefCoor_i ecef	LlaCoor_i lla	INT32Mat33 ltp_of_ecef
float	LtpDef_f	EcefCoor_f ecef	LlaCoor_f lla	FloatMat33 ltp_of_ecef
double	LtpDef_d	EcefCoor_d ecef	LlaCoor_d lla	DoubleMat33 ltp_of_ecef

The fixed-point struct has hmsl (height above mean sea level) as an additional parameter.

4.1 Transformations to LTP

from ECEF

Gets the LLA-coordinates and a transformation matrix from ECEF to ENU out of ECEF coordinates.

First, the LLA coordinates (WGS-84) are computed from the ECEF coordinates.

With the LLA-coordinates it is possible to construct a rotational matrix.

$$\mathbf{R}_{ecef2enu} = \begin{pmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{pmatrix} \tag{64}$$

Function `ltp_def_from_ecef_i(LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ltp_def_from_ecef_f(LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ltp_def_from_ecef_d(LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

5 NED / ENU

North-East-Down or East-North-Up coordinates are fixed in the local tangent plane.

NED and ENU coordinates are defined using

$$p_{NED} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad p_{ENU} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{65}$$

available for the following simple types

- int32_t - `NedCoor_i` and `EnuCoor_i`
- float - `NedCoor_f` and `EnuCoor_f`
- double - `NedCoor_d` and `EnuCoor_d`

5.1 Transformation between NED and ENU

The transformation between NED and ENU is rather simple. It can be expressed using a rotational matrix:

$$p_{NED} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} p_{ENU} \qquad p_{ENU} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} p_{NED} \tag{66}$$

or directly switching the values

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{NED} = \begin{pmatrix} y \\ x \\ -z \end{pmatrix}_{ENU} \tag{67}$$

Function `ENU_OF_TO_NED(po, pi)`
in File `pprz_geodetic.h`
Function `INT32_VECT2_ENU_OF_NED(o, i)`
in File `pprz_geodetic_int.h`
Function `INT32_VECT2_NED_OF_ENU(o, i)`
in File `pprz_geodetic_int.h`
Function `INT32_VECT3_ENU_OF_NED(o, i)`
in File `pprz_geodetic_int.h`
Function `INT32_VECT3_NED_OF_ENU(o, i)`
in File `pprz_geodetic_int.h`

5.2 Transformation from NED/ENU

to ECEF

With a know transformation matrix (see section **??**) it is quite easy to rotate a vector from the ENU-frame to the ECEF-frame. Since

$$\mathbf{R}_{enu2ecef} = \mathbf{R}_{ecef2enu}^{-1} = \mathbf{R}_{ecef2enu}^{T} \tag{68}$$

a transformation is done as follows

$$\vec{v}_{ECEF} = \mathbf{R}_{enu2ecef} \vec{v}_{ENU} \tag{69}$$

For a transformation from the NED-frame you have to do an additional ENU/NED-transformation before.

Function `ecef_of_enu_vect_d(EcefCoor_d* ecef, LtpDef_d* def, EnuCoor_d* enu)`
in File `pprz_geodetic_double.c`
Function `ecef_of_ned_vect_d(EcefCoor_d* ecef, LtpDef_d* def, NedCoor_d* ned)`
in File `pprz_geodetic_double.c`

The transformation of a point is very similiar. After transforming into the ECEF-frame you add the position of the center p_0 to the result.

$$\overrightarrow{p}_{ECEF} = p + p_0 \tag{70}$$

Function `ecef_of_enu_point_d(EcefCoor_d* ecef, LtpDef_d* def, EnuCoor_d* enu)`
in File `pprz_geodetic_double.c`
Function `ecef_of_ned_point_d(EcefCoor_d* ecef, LtpDef_d* def, NedCoor_d* ned)`
in File `pprz_geodetic_double.c`

5.3 Transformation to NED/ENU

from ECEF

With a know transformation matrix (see section **??**) it is quite easy to rotate a vector into the ENU frame:

$$\overrightarrow{v}_{ENU} = \mathbf{R}_{ecef2enu}\overrightarrow{v}_{ECEF} \tag{71}$$

For a transformation into the NED-frame you have to do an additional ENU/NED-transformation.
Function `enu_of_ecef_vect_i(EnuCoor_i* enu, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ned_of_ecef_vect_i(NedCoor_i* ned, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `enu_of_ecef_vect_f(EnuCoor_f* enu, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ned_of_ecef_vect_f(NedCoor_f* ned, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `enu_of_ecef_vect_d(EnuCoor_d* enu, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`
Function `ned_of_ecef_vect_d(NedCoor_d* ned, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

The transformation of a point is very similiar. Instead of a point you use a difference vector between the desired point p_d and the center of the local tangent plane p_0.

$$\overrightarrow{v}_{ECEF} = p_d - p_0 \tag{72}$$

Function `enu_of_ecef_point_i(EnuCoor_i* enu, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `ned_of_ecef_point_i(NedCoor_i* ned, LtpDef_i* def, EcefCoor_i* ecef)`
in File `pprz_geodetic_int.c`
Function `enu_of_ecef_point_f(EnuCoor_f* enu, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `ned_of_ecef_point_f(NedCoor_f* ned, LtpDef_f* def, EcefCoor_f* ecef)`
in File `pprz_geodetic_float.c`
Function `enu_of_ecef_point_d(EnuCoor_d* enu, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`
Function `ned_of_ecef_point_d(NedCoor_d* ned, LtpDef_d* def, EcefCoor_d* ecef)`
in File `pprz_geodetic_double.c`

from LLA

This functions transforms a point from the LLA coordinates in the ECEF-frame and then into the NED-/ENU-frame. Function `enu_of_lla_point_i(EnuCoor_i* enu, LtpDef_i* def, LlaCoor_i* lla)`
in File `pprz_geodetic_int.c`
Function `ned_of_lla_point_i(NedCoor_i* ned, LtpDef_i* def, LlaCoor_i* lla)`
in File `pprz_geodetic_int.c`