

# RTKLIB ver.2.1 Manual

2008/03/23

## 目 次

1	はじめに.....	1
1.1	RTKLIB の概要.....	1
2	仕様.....	2
2.1	実行環境.....	2
2.2	機能.....	2
3	操作説明.....	4
3.1	RTKLIB のインストール.....	4
3.2	RTKLIB ライブラリの構築.....	4
3.3	コマンドラインアプリケーションプログラムの構築.....	5
3.4	GUI アプリケーションプログラムの構築.....	6
3.5	RTKLIB ライブラリの利用.....	7
3.6	RNX2RTKP による後処理基線解析.....	7
3.7	POS2KML による測位解の変換.....	8
3.8	GUI アプリケーションプログラムの操作.....	8
3.9	RTKLIB ライブラリを使った RTK-GPS プログラムの開発.....	9
A	RTKLIB Application Program Interface (API).....	11
A.1	New Matrix.....	14
A.2	Zero Matrix.....	15
A.3	New Identity Matrix.....	16
A.4	Inner Product.....	17
A.5	Euclid Norm.....	18
A.6	Multiply Matrix.....	19
A.7	Inverse of Matrix.....	20
A.8	Solve Linear Equation.....	21
A.9	Least Square Estimation.....	22
A.10	Kalman Filter State Update.....	23
A.11	Kalman Smoother.....	24

A.12	Print Matrix .....	25
A.13	Print Matrix to File .....	26
A.14	String to Number .....	27
A.15	String to Time .....	28
A.16	Calendar Day/Time to Time .....	29
A.17	Time to Calendar Day/Time .....	30
A.18	GPSTIME to Time.....	31
A.19	Time to GPSTIME.....	32
A.20	Add Time .....	33
A.21	Time Difference.....	34
A.22	Get Current time in UTC .....	35
A.23	GPSTIME to UTC .....	36
A.24	UTC to GPSTIME .....	37
A.25	Time to String .....	38
A.26	Time to Day of Year.....	39
A.27	ECEF to Geodetic Position.....	40
A.28	Geodetic to ECEF Position.....	41
A.29	ECEF to Local Coordinates .....	42
A.30	Local to ECEF Coordinates .....	43
A.31	Covariance in Local Coordinates.....	44
A.32	Geoid Height .....	45
A.33	Load Datum Transformation Parameter .....	46
A.34	Tokyo Datum to JGD2000 Datum.....	47
A.35	JGD2000 Datum to Tokyo Datum.....	48
A.36	Read Antenna Phase Center Parameters .....	49
A.37	Read Station Positions .....	50
A.38	Expand file path.....	51
A.39	Read RINEX Files .....	52
A.40	Read RINEX Files in Time Range/Interval .....	54
A.41	Output RINEX OBS Header.....	55
A.42	Output RINEX OBS Body .....	56
A.43	Output RINEX NAV Header .....	57
A.44	Output RINEX NAV Body .....	58
A.45	Uncompress File .....	59

A.47	Satellite Ephemeris to Satellite Position/Clock-bias.....	61
A.48	Satellite Positions/Clock-biases.....	62
A.49	Satellite Positions/Clock-biases by IODE .....	63
A.50	Satellite Positions/Velocities/Clock-biases/Clock-drifts.....	64
A.51	Geometric Distance .....	65
A.52	Satellite Azimuth/Elevation Angle .....	66
A.53	Compute DOPs .....	67
A.54	Antenna Model .....	68
A.55	Carrier Smoothing .....	69
A.56	Ionospheric Model.....	70
A.57	Ionospheric Mapping Function.....	71
A.58	Tropospheric Model.....	72
A.59	Tropospheric Mapping Function (NMF) .....	73
A.60	Single Point Positioning .....	74
A.61	RTK positioning .....	75
A.62	Post-processing Positioning.....	77
A.63	Read Positioning Options .....	78
A.64	Write Positioning Options .....	79
A.65	Read Solutions.....	80
A.66	Read Solutions in Time Range/Interval .....	81
A.67	Output Solution Header .....	82
A.68	Output Solution Header .....	83
A.69	Set Solution Output Options.....	84
A.70	Convert Solution File to Google Earth KML File .....	85
A.71	Read SBAS Message File.....	86
A.72	Read SBAS Message File in Time Range .....	87
A.73	Output SBAS Messages.....	88
A.74	Update SBAS Status .....	89
A.75	Decode SBAS Message .....	91
A.76	SBAS Fast Correction .....	92
A.77	SBAS Long Term Correction .....	93
A.78	SBAS Ionospheric Delay Correction.....	94
A.79	SBAS Tropospheric Delay Correction.....	95
A.80	SBAS Satellite Position .....	96

A.81	SBAS Point Positioning with Corrections .....	97
A.82	Decode GPS Navigation Data Frame .....	98
A.83	Decode u-blox Messages .....	99
A.84	Read u-blox Log File .....	100
A.85	Read u-blox Log File in Time Range/Interval .....	101
A.86	Convert u-blox Log File .....	102
A.87	Decode NovAtel message.....	103
A.88	Read NovAtel Log File.....	104
A.89	Read NovAtel Log File in Time Range/Interval.....	105
A.90	Convert NovAtel Log File .....	106
A.91	Execute Command.....	107
A.92	Debug Trace Functions .....	108
B	Command Line Application Programs .....	109
B.1	Baseline Analysis by Precise Relative Positioning.....	110
B.2	Convert Positions to Google Earth KML file .....	113
B.3	Generate Network RTK Correction .....	114
B.4	Generate Satellite Clock Variation Correction.....	115
B.5	Convert u-blox log file to RINEX OBS/NAV, SBAS messages .....	116
B.6	Convert NovAtel log file to RINEX OBS/NAV, SBAS messages.....	117
B.7	Single point positioning with SBAS DGPS correction.....	118
B.8	Dump SBAS messages .....	119
C	GUI-based Application Programs.....	120
C.1	RTKPOST .....	121
C.2	RTKPLOT .....	132
C.3	RTKCONV .....	140

## 1 はじめに

### 1.1 RTKLIB の概要

RTKLIB は RTK (Realtime Kinematic) - GPS 測位アルゴリズムの評価確認のため開発した、C 言語により記述された簡潔で可搬性の高い RTK-GPS 測位演算ライブラリです。RTKLIB は以下の機能を有しています。

- (1) 行列・ベクトル演算
- (2) 時刻・文字列処理
- (3) 座標系変換
- (4) RINEX 観測データ、航法メッセージファイル読み込み
- (5) OTF 整数 Ambiguity 決定
- (6) 航法演算処理
- (7) 対流圏モデル、電離層モデル計算
- (8) 単独測位演算、RTK-GPS 相対測位演算
- (9) SBAS DGPS 補正演算
- (10) 受信機バイナリデータ処理

RTKLIB には RTKLIB ライブラリを使用した各種アプリケーションプログラムが含まれています。

## 2 仕様

### 2.1 実行環境

RTKLIB の構築及び実行には ANSI 標準 C プログラムがコンパイル・リンクできる環境が必要になります。外部の行列演算ライブラリを使用する場合、BLAS/LAPACK または互換ライブラリがインストールされている必要があります (内蔵ルーチンを使うことも出来ます)。実行確認済み実行環境は以下の通りです。

- (1) Windows XP、Cygwin, gcc + BLAS/LAPACK
- (2) Windows XP、MS Visual C++ 2005 Express Edition (BLAS/LAPACK なし)
- (3) Windows XP、MS Visual Studio.NET 2002+ Intel C/C++ 8.0 + Intel MKL 7.2
- (4) Windows XP、Borland Turbo C++ 2006 (BLAS/LAPACK なし)
- (5) Mac OS X、gcc+BLAS/LAPACK

なお、付属する GUI アプリケーションプログラムの構築には Borland C++ Builder または Turbo C++2006 が必要になります。

### 2.2 機能

- (1) 行列・ベクトル演算
  - ・行列メモリ確保、ゼロ行列、単位行列、内積、ノルム、行列印刷
- (2) 時刻・文字列処理
  - ・文字列数値変換、文字列時刻変換、カレンダー時刻変換、GPS 時刻変換、GPST-UTC 変換
- (3) 座標系変換
  - ・ECEF-緯度経度高度変換、ECEF-局地座標系変換、ジオイド高
- (4) RINEX 観測データ、航法メッセージファイル読み込み
- (5) OTF 整数 Ambiguity 決定
  - ・LAMBDA/MLAMBDA 整数不定性決定
- (6) 航法演算処理
  - ・暦計算、衛星位置・時計バイアス計算、幾何学距離計算、衛星方位仰角計算
- (7) 対流圏モデル、電離層モデル計算
  - ・対流圏モデル：Saastamoinen モデル+標準大気、・電離層モデル：Klobucher モデル
- (8) 単独測位演算
- (9) RTK-GPS 相対測位演算

(10) SBAS DGPS 補正演算

(11) 受信機バイナリデータ処理

### 3 操作説明

#### 3.1 RTKLIB のインストール

(1) パッケージ **rtklib\_<ver>.tar.gz** または **rtklib\_<ver>.zip** を適当なディレクトリの下に解凍して下さい。(<ver>はバージョン番号)

(2) パッケージのディレクトリ構成は以下の通りです。

```
rtklib_<ver>
  /src          : ライブラリソースプログラム
  /lib          : ライブラリ構築環境
    /gcc        : GCC 用
  /app          : アプリケーションプログラム
    /convnov    : NovAtel コンバータ
    /convubx    : u-blox コンバータ
    /gencorr    : 補正情報生成
    /pos2kml    : Google Earth KML 変換
    /rnx2rtkp   : 後処理基線解析
    /rtkconv    : GUI 受信機ログコンバータ
    /rtkplot    : GUI 観測データ・測位解プロット
    /rtkpost    : GUI 後処理基線解析
    /sbasdump   : SBAS メッセージダンプ
    /sbaspos    : SBAS DGPS 補正測位
  /test        : 試験用データ、プログラム
  /util        : ユーティリティ
  /doc         : 文書ファイル
```

#### 3.2 RTKLIB ライブラリの構築

(1) GCC 環境における RTKLIB ライブラリの構築には以下のコマンドを入力してください。

**rtklib/lib/gcc** 下にオブジェクトファイル及びRTKライブラリ **librtk.a** が生成されます。

```
> cd rtklib_<ver>/lib/gcc
```



```
> make
```

- (2) 組み込みの行列演算ルーチンを使用し、BLAS/LAPACK を使用しない場合、C/C++のコンパイルオプションとしてマクロ **NOLAPACK** 定義を追加して下さい。
- (3) BLAS/LAPACK の代わりに Intel MKL を使用する場合、C/C++のコンパイルオプションとしてマクロ **MKL** 定義を追加して下さい。またリンクオプションとして Intel MKL を入力に追加してください。詳細は Intel MKL のマニュアルを参照下さい。
- (4) ライブラリをインストールするためには以下のコマンドを入力して下さい。デフォルトのインストール先は **/usr/local/bin** です。インストール先を変更する場合は **makefile** でインストール先ディレクトリを指定する変数 **BINDIR** の値を変更してください。

```
> su
passwd : ***
# make install
# exit
```

### 3.3 コマンドラインアプリケーションプログラムの構築

- (1) GCC 環境におけるコマンドラインアプリケーションプログラムの構築には以下のコマンドを入力して下さい。カレントディレクトリに実行プログラムが生成されます。

```
> cd rtklib_<ver>/app/<app>/gcc
> make
```

- (2) なお以上のコマンドのうち **<app>** はアプリケーションプログラムに合わせて以下に読み替えてください。

- **convnov** : NovAtel コンバータ
- **convubx** : u-blox コンバータ
- **gencorr** : 補正情報生成
- **pos2kml** : 後処理基線解析
- **pos2kml** : Google Earth KML 変換
- **sbasdump** : SBAS メッセージダンプ

・ **sbaspos** : SBAS DGPS 補正測位

- (3) ライブラリをインストールするためには以下のコマンドを入力して下さい。デフォルトのインストール先は **/usr/local/lib** です。インストール先を変更する場合は **makefile** でインストール先ディレクトリを指定する変数 **LIBNDIR** の値を変更してください。

```
> su
passwd : ***
# make install
# exit
```

### 3.4 GUI アプリケーションプログラムの構築

- (1) GUI アプリケーションプログラムの構築には Borland Turbo C++ または C++ Builder が必要です。Borland Turbo C++ または C++ Builder を起動してください。
- (2) Turbo C++ のメニュー「ファイル」－「プロジェクトを開く」でプロジェクトファイル **rtklib\_<ver>/app/<app>/<app>.bdsproj** を開いてください。なお **<app>** はアプリケーションプログラムに合わせて以下に読み代えてください。

・ **rtkpost** : 後処理基線解析プログラム  
 ・ **rtkplot** : 観測データ・測位解プロット  
 ・ **rtkconv** : RINEX コンバータ

- (3) Turbo C++ のメニュー「プロジェクト」－「再構築」を実行しプログラムを再構築してください。実行プログラムは **rtklib\_<ver>/app/<app>/<app>.bdsproj/Release\_Build** または **rtklib\_<ver>/app/<app>/<app>.bdsproj/Debug\_Build** の下に生成されます。

### 3.5 RTKLIB ライブラリの利用

- (1) RTKLIB の関数をアプリケーションプログラムから利用するためには、アプリケーションプログラム中で以下のヘッダファイルをインクルードして下さい。

```
rtklib_<ver>/src/rtklib.h
```

- (2) アプリケーションプログラムのリンクオプションとして RTK ライブラリ **rtklib/lib/gcc/librtk.a** を入力として追加して下さい。

- (3) アプリケーションプログラムから使用できる RTKLIB の関数仕様については付録 A を参照して下さい。

### 3.6 RNX2RTKP による後処理基線解析

- (1) 後処理基線解析にはローバ及び基準局の RINEX OBS 形式の GPS 観測データ、RINEX NAV 形式の航法メッセージファイルが必要になります。この例ではそれぞれのファイル名が以下であるとしています。

```
rov0010.07o ref0010.07o ref0010.07n
```

- (2) コンソール(コマンドプロンプト)で以下のコマンドを入力して下さい。

```
> rnx2rtkp rov001.07o ref0010.07o ref0010.07n > output.pos
```

- (3) プログラムの実行が終わったらエディタ等で出力ファイル **output.pos** の内容を確認して下さい。先頭%の行がコメント、それ以降にローバの測位解が 1 行に 1 エポック分、時間、座標、Q(品質)フラグ、有効衛星数、推定値標準偏差の形式で格納されます。このうち Q フラグの意味は以下の通りです。

**1: FIX 解、2: FLOAT 解、4: DGPS 解、5: 単独測位解**

なおデフォルトでは 2 周波キネマティック、仰角マスク 10 度、緯度・経度・高度出力となっています。またデフォルトでは基準局座標として単独測位解の平均値を使用します。

表 1 測位解出力例

```
% program      : rnx2rtkp ver.1.0
% inputs       : rov0010.07o ref0010.07o ref0010.07n
% obs start    : 2007/01/01 00:00:00.0 GPST (gpsweek1316 518400.0s)
% obs end      : 2007/01/01 23:59:30.0 GPST (gpsweek1316 604770.0s)
% mode/obsv    : kinematic/L1+L2
% elev mask    : 10.0 deg
% ref pos      : 35.132062716 139.624305669 72.3338
%
% (time=GPST, lat/lon/height=WGS84/ellipsoidal, Q=1:fix,2:float,4:dgps,5:single,ns=# of
sats)
% time         latitude(deg) longitude(deg) height(m) Q ns sdn(m) sde(m) sdu(m)
518400.000     35.160871612 139.613842087 66.8062 1 7 0.0072 0.0054 0.0164
518430.000     35.160871607 139.613842115 66.7987 1 7 0.0072 0.0054 0.0164
518460.000     35.160871593 139.613842110 66.7999 1 7 0.0072 0.0054 0.0163
518490.000     35.160871583 139.613842093 66.8118 1 7 0.0072 0.0053 0.0163
518520.000     35.160871627 139.613842143 66.8086 1 7 0.0072 0.0053 0.0163
518550.000     35.160871616 139.613842127 66.8061 1 7 0.0072 0.0053 0.0162
518580.000     35.160871596 139.613842108 66.8191 1 7 0.0072 0.0053 0.0162
518610.000     35.160871586 139.613842086 66.8229 1 7 0.0072 0.0053 0.0161
518640.000     35.160871615 139.613842077 66.8212 1 7 0.0072 0.0053 0.0161
518670.000     35.160871628 139.613842154 66.8044 1 7 0.0072 0.0053 0.0160
...
```

- (4) コマンドオプションを指定することにより、測位モード、周波数、仰角マスク、出力形式、基準局座標、等を変更することが出来ます。**RNX2KML** コマンドの詳細は付録 B を参照下さい。

### 3.7 POS2KML による測位解の変換

- (1) 3.6 で出力したローバの位置軌跡を付属ユーティリティプログラム **pos2kml** を使うことにより Google Earth 上で表示できる KML ファイルに変換することができます。例えば 3.3 の例で出力した測位解 **output.pos** を KML ファイルに変換するためには以下のコマンドを入力して下さい。デフォルトでは拡張子を **.kml** に変更したファイルに変換結果が出力されます。

```
> pos2kml output.pos
```

- (2) **POS2KML** コマンドの詳細については付録 B を参照下さい。

### 3.8 GUI アプリケーションプログラムの操作

RTKLIB v.2.1 には以下の GUI アプリケーションプログラムが含まれています。

- (1) **RTKPOST** : 後処理基線解析プログラム

- (2) **RTKPLOT** : 観測データ・測位解プロットプログラム
- (3) **RTKCONV** : RINEX 変換プログラム

各 GUI アプリケーションプログラムの操作については付録 C を参照下さい。

### 3.9 RTKLIB ライブラリを使った RTK-GPS プログラムの開発

- (1) RTKLIB ライブラリをユーザアプリケーションプログラムに組み込むことにより RTK-GPS プログラムを構築することができます。
- (2) RTK-GPS プログラムを構築する場合、1 エポック分の観測データ、航法メッセージ入力コールバック関数 **input()** 及び測位解出力コールバック関数 **output()** をユーザが記述する必要があります。各関数のプロトタイプについては付録 A **rtkpos()** 関数の説明を参照してください。
- (3) ユーザアプリケーションプログラム中で、コールバック関数 **input()** 及び **output()** のポインタを引数にして RTKLIB の **rtkpos()** 関数を呼び出して下さい。**rtkpos()** 内部でまず **input()** を呼び出し、1 エポック分の観測データ及び航法メッセージを入力して、RTK-GPS 測位演算処理を行い、その結果を **output()** を呼び出して出力し、再度 **input()** を呼び出して次のエポックの処理を行うという処理ループに入ります。**rtkpos()** 内部の処理ループから抜けるためにはコールバック関数 **input()** の戻り値として 0 以下の値を返してください。
- (4) RTKLIB を使用した RTK-GPS プログラムの例を表 2 に示します。**rtkpos()** 関数の詳細は付録 A を参照して下さい。

表2 RTKLIB による RTK-GPS プログラム例

```

#include "rtklib.h"

/* 観測データ、航法メッセージ入力コールバック関数 */
int input(obsd_t * obs, nav_t *nav, sbs_t *sbs)
{
    int n;

    /* 1エポック分のローバ、基準局観測データ及び航法メッセージ読み込み処理 */

    ...

    return n; /* 観測データ個数、n<=0: 終了 */
}

/* 測位解出力コールバック関数 */
void output(gtime_t t, sol_t sol, sol_t solf, double *Q, int stat, int ns)
{
    char str[64];
    time2str(t, str, 0);
    printf("%s : %12.3f %12.3f %12.3f %d¥n", str, solf.rr[0], solf.rr[1],
           solf.rr[2], stat);
}

/* RTKプログラムメイン */
int main(void)
{
    opt_t opt=opt_default;

    /* reference position (ECEF) (m) */
    opt.rb[0]=-3978241.958;
    opt.rb[1]= 3382840.234;
    opt.rb[2]= 3649900.853;

    /* RTK-GPS processing loop (exit if input() returns <=0) */
    rtkpos(input, NULL, output, &opt);

    return 0;
}

```

## A RTKLIB Application Program Interface (API)

Function	Description	Reference
Matrix and vector functions		
<b>mat()</b>	New matrix	A.1
<b>zeros()</b>	New zero matrix	A.2
<b>eye()</b>	New identity matrix	A.3
<b>dot()</b>	Inner Product	A.4
<b>norm()</b>	Euclid norm	A.5
<b>matmul()</b>	Multiply matrix	A.6
<b>matinv()</b>	Inverse of matrix	A.7
<b>solve()</b>	Solve linear equation	A.8
<b>lsq()</b>	Least square estimation	A.9
<b>filter()</b>	Kalman filter state update	A.10
<b>smoother()</b>	Kalman smoother	A.11
<b>matprint()</b>	Print matrix	A.12
<b>matfprint()</b>	Print matrix to file *	A.13
Time and string functions		
<b>str2num()</b>	String to number	A.14
<b>str2time()</b>	String to time	A.15
<b>epoch2time()</b>	Calendar day/time to time	A.16
<b>time2epoch()</b>	Time to calendar day/time	A.17
<b>gpst2time()</b>	GPSTIME to time	A.18
<b>time2gpst()</b>	Time to GPSTIME	A.19
<b>timeadd()</b>	Add time	A.20
<b>timediff()</b>	Time difference	A.21
<b>timeget()</b>	Get current time in UTC *	A.22
<b>gpst2utc()</b>	GPSTIME to UTC	A.23
<b>utc2gpst()</b>	UTC to GPSTIME	A.24
<b>time2str()</b>	Time to string	A.25
<b>time2doy()</b>	Time to Day of Year *	A.26
Coordinates transformations		
<b>ecef2pos()</b>	ECEF to geodetic position	A.27
<b>pos2ecef()</b>	Geodetic to ECEF position	A.28
<b>ecef2enu()</b>	ECEF to local coordinates	A.29
<b>enu2ecef()</b>	Local to ECEF coordinates	A.30
<b>covenu()</b>	Covariance in local coordinates	A.31
<b>geoidh()</b>	Geoid height	A.32
<b>loaddatum()</b>	Load datum transformation parameter *	A.33
<b>tokyo2jgd()</b>	Tokyo datum to JGD2000 datum *	A.34
<b>jgd2tokyo()</b>	JGD2000 datum to Tokyo datum *	A.35
File Input/Output functions		
<b>readpcv()</b>	Read antenna phase center parameters *	A.36
<b>readpos()</b>	Read station positions *	A.37
<b>expandpath()</b>	Expand file path *	A.38
<b>readrnex()</b>	Read RINEX files	A.39
<b>readrnxt()</b>	Read RINEX files in time range/interval *	A.40
<b>outrnxobs()</b>	Output RINEX OBS header *	A.41
<b>outrnxobsb()</b>	Output RINEX OBS body *	A.42
<b>outrnxnav()</b>	Output RINEX NAV header *	A.43
<b>outrnxnavb()</b>	Output RINEX NAV body *	A.44
<b>uncompress()</b>	Uncompress file *	A.45

Function	Description	Reference
	OTF Integer Ambiguity resolution	
<b>lambda()</b>	LAMBDA/MLAMBDA integer least-square estimation	A.46
	Navigation functions	
<b>eph2pos()</b>	Satelilte ephemeris to satellite position/clock-bias	A.47
<b>satpos()</b>	Satellite positions/clock-biases	A.48
<b>satposiode()</b>	Satellite positions/clock-biases by IODE *	A.49
<b>satpviode()</b>	Satellite positions/velocities/clock-biases/clock-drifts *	A.50
<b>geodist()</b>	Geometric distance	A.51
<b>satazel()</b>	Satellite azimuth/elevation angle	A.52
<b>dops()</b>	Compute DOPs *	A.53
<b>antmodel()</b>	Antenna model *	A.54
<b>csmooth()</b>	Carrier smoothing *	A.55
	Ionospheric/Tropospheric models	
<b>ionmodel()</b>	Ionospheric model	A.56
<b>ionmapf()</b>	Ionospheric mapping function *	A.57
<b>tropmodel()</b>	Tropospheric model	A.58
<b>tropmapf()</b>	Tropospheric mapping function (NMF) *	A.59
	Positioning solutions	
<b>pntpos()</b>	Single point positioning **	A.60
<b>rtkpos()</b>	RTK positioning **	A.61
<b>postpos()</b>	Post-processing positioning *	A.62
<b>readopt()</b>	Read positioning options *	A.63
<b>writeopt()</b>	Write positioning options *	A.64
<b>readsol()</b>	Read solutions *	A.65
<b>readsolt()</b>	Read solutions in time range/interval *	A.66
<b>outsolhead()</b>	Output solution header *	A.67
<b>outsol()</b>	Output solution body *	A.68
<b>setsolopt()</b>	Set solution output options *	A.69
<b>convkml()</b>	Convert solution file to Google Earth KML file *	A.70
	SBAS functions	
<b>sbsreadmsg()</b>	Read SBAS message file *	A.71
<b>sbsreadsmgt()</b>	Read SBAS message file in time range *	A.72
<b>sbsoutmsg()</b>	Output SBAS messages *	A.73
<b>sbsupdatestat()</b>	Update SBAS status *	A.74
<b>sbsdecodemsg()</b>	Decode SBAS message *	A.75
<b>sbsfastcorr()</b>	SBAS fast correction *	A.76
<b>sbslongcorr()</b>	SBAS long term correction *	A.77
<b>sbsioncorr()</b>	SBAS ionospheric delay correction *	A.78
<b>sbstropcorr()</b>	SBAS tropospheric delay correction *	A.79
<b>sbssatpos()</b>	SBAS satellite position *	A.80
<b>sbspntpos()</b>	SBAS point positioning with corrections *	A.81
	Receiver dependent functions	
<b>decodefrm()</b>	Decode GPS navigation data frame *	A.82
<b>decodeubx()</b>	Decode u-blox message *	A.83
<b>readubx()</b>	Read u-blox log file *	A.84
<b>readubxt()</b>	Read u-blox log file in time range/interval *	A.85
<b>convubx()</b>	Convert u-blox log file to RINEX OBS/NAV/SBAS file *	A.86
<b>decodenov()</b>	Decode NovAtel message *	A.87



Function	Description	Reference
<b>readnov( )</b>	Read NovAtel log file *	A.88
<b>readnovt( )</b>	Read NovAtel log file in time range/interval *	A.89
<b>convnov( )</b>	Convert NovAtel log file *	A.90
Misc functions		
<b>execcmd( )</b>	Execute command *	A.91
Debug Trace functions		
<b>traceopen( )</b>	Open trace file *	A.92
<b>traceclose( )</b>	Close trace file *	A.92
<b>trace( )</b>	Output trace *	A.92
<b>tracet( )</b>	Output trace with time tag *	A.92
<b>tracemat( )</b>	Output trace as matrix printing *	A.92
<b>traceobs( )</b>	Output trace as observation data printing *	A.92

\* Added function in RTKLIB ver.2.1

\*\* Modified in RTKLIB ver.2.1

## A.1 New Matrix

**mat()**

**[Prototype]**

```
#include "rtklib.h"
double *mat(int n, int m);
```

**[Args]**

int     n,m     I     number of rows and columns of new matrix

**[Return]**

Matrix pointer (if n<=0 or m<=0, return NULL)

**[Description]**

Allocate memory for new matrix.

**[Notes]**

If memory allocation error, print error message and exit program.

## A.2 Zero Matrix

### **zeros()**

#### **[Prototype]**

```
#include "rtklib.h"
double *zeros(int n, int m);
```

#### **[Args]**

int     n,m       I       number of rows and columns of matrix

#### **[Return]**

Matrix pointer (if n<=0 or m<=0, return NULL)

#### **[Description]**

Generate new zero matrix.

#### **[Notes]**

If memory allocation error, print error message and exit program.

### A.3 New Identity Matrix

#### **eye()**

##### **[Prototype]**

```
#include "rtklib.h"
double *eye(int n);
```

##### **[Args]**

int     n           I           number of rows and columns of matrix

##### **[Return]**

Matrix pointer (if  $n \leq 0$ , return NULL)

##### **[Description]**

Generate new identity matrix.

##### **[Notes]**

If memory allocation error, print error message and exit program.

## A.4 Inner Product

### `dot()`

#### [Prototype]

```
#include "rtklib.h"
```

```
double dot(const double *a, const double *b, int n);
```

#### [Args]

```
double a,b      I      vector a, b (n x 1)
int    n        I      size of vector a, b
```

#### [Return]

Inner product of vector  $a, b$  ( $a^T b$ )

#### [Description]

Inner product of vectors.

#### [Notes]

## A.5 Euclid Norm

**norm( )**

**[Prototype]**

```
#include "rtklib.h"
```

```
double norm(const double *a, int n);
```

**[Args]**

double *a	I	vector a (n x 1)
int n	I	size of vector a

**[Return]**

Euclid norm of vector  $\mathbf{a}$  ( $\|\mathbf{a}\|$ )

**[Description]**

Euclid norm of vector.

**[Notes]**

## A.6 Multiply Matrix

### **matmul()**

#### **[Prototype]**

```
#include "rtklib.h"

void matmul(const char *tr, int n, int k, int m, double alpha,
            const double *A, const double *B, double beta, double *C);
```

#### **[Args]**

char	*tr	I	transpose flags of matrix A, B ( "N": normal, "T": transpose)
int	n,k,m	I	size of (transposed) matrix A, B
double	alpha	I	alpha
double	*A,*B	I	(Transposed) Matrix A (n x m), B (m x k)
double	beta	I	beta
double	*C	IO	matrix C (n x k)

#### **[Return]**

None

#### **[Description]**

Multiply matrix by matrix. ( $C = \alpha A^* B^* + \beta C$ )

#### **[Notes]**

Lapper of BLAS DGEMM.

## A.7 Inverse of Matrix

### **matinv()**

#### **[Prototype]**

```
#include "rtklib.h"
int matinv(double *A, int n);
```

#### **[Args]**

double *A	IO	matrix A (n x n)
int n	I	size of matrix A

#### **[Return]**

Status (0:OK, 0>:Error)

#### **[Description]**

Inverse of matrix ( $A=A^{-1}$ ).

#### **[Notes]**



## A.8 Solve Linear Equation

### **solve()**

#### **[Prototype]**

```
#include "rtklib.h"

int solve(const char *tr, const double *A, const double *Y, int n, int m,
          double *X);
```

#### **[Args]**

char	*tr	I	transpose flag of matrix A ( "N": normal, "T": transpose)
double	*A	I	matrix A (n x n)
double	*Y	I	matrix Y (n x m)
int	n,m	I	size of matrix Y, X
double	*X	O	solution X of linear equation $AX=Y$ or $A^T X=Y$ (n x m)

#### **[Return]**

Status (0:OK, 0>:Error)

#### **[Description]**

Solve linear equation ( $AX=Y$  or  $A^T X=Y$ ).

#### **[Notes]**

Matirix stored by column-major order (FORTRAN convention).

X can be same as Y.

## A.9 Least Square Estimation

### lsq()

#### [Prototype]

```
#include "rtklib.h"

int lsq(const double *A, const double *y, int n, int m, double *x,
        double *Q);
```

#### [Args]

double *A	I	transpose of (weighted) design matrix A (n x m)
double *y	I	(weighted) measurements y (m x 1)
int n,m	I	number of parameters and measurements (n<=m)
double *x	O	estimated parameters x (n x 1)
double *Q	O	estimated parameters covariance matrix Q <sub>x</sub> (n x n)

#### [Return]

Status (0:OK, 0>:error)

#### [Description]

Least square estimation by solving normal equation ( $x=(AA^T)^{-1}Ay$ ).

#### [Notes]

For weighted least square, replace A and y to Aw and wy ( $w=W^{1/2}$ ).

Matirix stored by column-major order (FORTRAN convention).

## A.10 Kalman Filter State Update

### **filter()**

#### **[Prototype]**

```
#include "rtklib.h"

int filter(const double *x, const double *P, double *H, double *v,
          const double *R, int n, int m, double *xp, double *Pp);
```

#### **[Args]**

double *x	I	states vector $x$ ( $n \times 1$ )
double *P	I	covariance matrix of states $P$ ( $n \times n$ )
double *H	I	transpose of design matrix $H$ ( $n \times m$ )
double *v	I	innovation (measurement - model) $v$ ( $m \times 1$ )
double *R	I	covariance matrix of measurement error $R$ ( $m \times m$ )
int n,m	I	number of states and measurements
double *xp	O	updated states vector $x^+$ ( $n \times 1$ )
double *Pp	O	updated covariance matrix of states $P^+$ ( $n \times n$ )

#### **[Return]**

Status (0:OK,0>:Error)

#### **[Description]**

Kalman filter states update by measurements ( $K=PH (H^T PH+R)^{-1}$ ,  $x^+=x+Kv$ ,  $P^+=(I-KH^T)P$ ).

#### **[Notes]**

Matirix stored by column-major order (FORTRAN convention).

## A.11 Kalman Smoother

### **smoother()**

#### **[Prototype]**

```
#include "rtklib.h"

int smoother(const double *xf, const double *Qf, const double *xb,
             const double *Qb, int n, double *xs, double *Qs);
```

#### **[Args]**

double *xf,*Qf	I	forward solutions/covariances	(n x 1/n x n)
double *xb,*Qb	I	backward solutions/covariances	(n x 1/n x n)
int n	I	number of solutions	
double *xs,*Qs	O	smoothed solutions/covariances	(n x 1/n x n)

#### **[Return]**

Status (0:OK, 0>:error)

#### **[Description]**

Combine forward and backward filters by fixed-interval smoother (ref[1], 5.2)

$$(x_s = Q_s(Q_f^{-1}x_f + Q_b^{-1}x_b), Q_s = (Q_f^{-1} + Q_b^{-1})^{-1})$$

#### **[Notes]**

Reference : A.Gelb ed., Applied Optimal Estimation, The M.I.T Press, 1974

Matirix stored by column-major order (fortran convention).

## A.12 Print Matrix

### **matprint()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void matprint(const double *A, int n, int m, int p, int q);
```

#### **[Args]**

double *A	I	matrix A (n x m)
int n,m	I	number of rows and columns of A
int p,q	I	print format total columns and columns under decimal point.

#### **[Return]**

None

#### **[Description]**

Print matrix to STDOUT.

#### **[Notes]**

Matirix stored by column-major order (FORTRAN convention).

## A.13 Print Matrix to File

### **matfprint()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void matfprint(const double *A, int n, int m, int p, int q, FILE *fp);
```

#### **[Args]**

double *A	I	matrix A (n x m)
int n,m	I	number of rows and columns of A
int p,q	I	print format total columns and columns under decimal point.
FILE *fp	I	output file pointer

#### **[Return]**

None

#### **[Description]**

Print matrix to file

#### **[Notes]**

Matirix stored by column-major order (FORTRAN convention).

## A.14 String to Number

### **str2num()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double str2num(const char *s, int i, int n);
```

#### **[Args]**

char	*s	I	string ("... nnn.nnn ...")
int	i,n	I	substring position and width.

#### **[Return]**

Converted number (0.0: Error)

#### **[Description]**

Convert substring in string to number.

#### **[Notes]**

## A.15 String to Time

### **str2time()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
int str2time(const char *s, int i, int n, gtime_t *t);
```

#### **[Args]**

char	*s	I	string ("... yyyy mm dd hh mm ss ...")
int	i,n	I	substring position and width
gtime_t	*t	O	RTKLIB time struct

#### **[Return]**

Status (0:OK,0>:Error)

#### **[Description]**

Convert substring in string to RTKLIB time struct.

#### **[Notes]**



## A.16 Calendar Day/Time to Time

### **epoch2time()**

#### **[Prototype]**

```
#include "rtklib.h"
gtime_t epoch2time(const double *ep);
```

#### **[Args]**

double \*ep      I      day/time {year,month,day,hour,min,sec}

#### **[Return]**

RTKLIB time struct.

#### **[Description]**

Convert calendar day/time to RTKLIB time struct.

#### **[Notes]**

## A.17 Time to Calendar Day/Time

### **time2epoch()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void time2epoch(gtime_t t, double *ep);
```

#### **[Args]**

gtime_t t	I	RTKLIB time struct
double *ep	O	day/time {year,month,day,hour,min,sec}

#### **[Return]**

None

#### **[Description]**

Convert RTKLIB time struct to calendar day/time

#### **[Notes]**

## A.18 GPSTIME to Time

### **gpst2time()**

#### **[Prototype]**

```
#include "rtklib.h"
ptime_t gpst2time(int week, double sec);
```

#### **[Args]**

int	week	I	GPS week number
double	sec	I	GPSTIME (time of week) (sec)

#### **[Return]**

RTKLIB time struct.

#### **[Description]**

Convert GPS week and time of week to RTKLIB time struct.

#### **[Notes]**

## A.19 Time to GPSTIME

### **time2gpst()**

#### **[Prototype]**

```
#include "rtklib.h"
double time2gpst(gtime_t t, int *week);
```

#### **[Args]**

gtime_t t	I	RTKLIB time struct
int *week	O	GPS week number

#### **[Return]**

GPSTIME (time of week) (sec)

#### **[Description]**

Convert RTKLIB time struct to GPS week and time of week.

#### **[Notes]**

## A.20 Add Time

### **timeadd()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
gtime_t timeadd(gtime_t t, double sec);
```

#### **[Args]**

gtime_t t	I	RTKLIB time struct.
double sec	I	time to add (sec)

#### **[Return]**

RTKLIB time struct (t+sec)

#### **[Description]**

Add time to RTKLIB time struct.

#### **[Notes]**

## A.21 Time Difference

### **timediff()**

#### **[Prototype]**

```
#include "rtklib.h"
double timediff(gtime_t t1, gtime_t t2);
```

#### **[Args]**

gtime\_t t1,t2 I          RTKLIB time structs

#### **[Return]**

Time difference (t1-t2) (sec)

#### **[Description]**

Difference between RTKLIB time struct.

#### **[Notes]**

## A.22 Get Current time in UTC

### **timeget()**

#### **[Prototype]**

```
#include "rtklib.h"  
gtime_t timeget(void);
```

#### **[Args]**

none

#### **[Return]**

Current time in UTC

#### **[Description]**

Get current time in UTC

#### **[Notes]**

## A.23 GPSTIME to UTC

### **gpst2utc()**

#### **[Prototype]**

```
#include "rtklib.h"
mtime_t gpst2utc(mtime_t t);
```

#### **[Args]**

mtime\_t t            I            time expressed in GPSTIME

#### **[Return]**

Time expressed in UTC.

#### **[Description]**

Convert time expressed in GPSTIME to UTC considering leap seconds

#### **[Notes]**



## A.24 UTC to GPSTIME

### **utc2gpst()**

#### **[Prototype]**

```
#include "rtklib.h"
mtime_t utc2gpst(mtime_t t);
```

#### **[Args]**

mtime\_t t            I            time expressed in UTC

#### **[Return]**

Time expressed in GPSTIME.

#### **[Description]**

Convert time expressed in UTC to GPSTIME considering leap seconds.

#### **[Notes]**

## A.25 Time to String

### **time2str()**

#### **[Prototype]**

```
#include "rtklib.h"
void time2str(gtime_t t, char *s, int n);
```

#### **[Args]**

gtime_t t	I	RTKLIB time struct
char *s	O	string in the form of "yyyy/mm/dd hh:mm:ss.s..."
int n	I	columns of sec under decimal point

#### **[Return]**

None

#### **[Description]**

Convert RTKLIB time struct to string.

#### **[Notes]**

## A.26 Time to Day of Year

### **time2doy()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double time2doy(gtime_t t);
```

#### **[Args]**

```
gtime_t t      I      RTKLIB time struct
```

#### **[Return]**

Day of Year (DOY) (days)

#### **[Description]**

Convert RTKLIB time struct to Day of Year (DOY)

#### **[Notes]**

## A.27 ECEF to Geodetic Position

### **ecef2pos()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void ecef2pos(const double *r, double *pos);
```

#### **[Args]**

double *r	I	ECEF position {x,y,z} (m)
-----------	---	---------------------------

double *pos	O	geodetic position {lat,lon,h} (rad,m)
-------------	---	---------------------------------------

#### **[Return]**

None

#### **[Description]**

Transform ECEF position to geodetic position (latitude, longitude and height).

#### **[Notes]**

WGS84, ellipsoidal height.

## A.28 Geodetic to ECEF Position

### **pos2ecef()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void pos2ecef(const double *pos, double *r);
```

#### **[Args]**

double *pos	I	geodetic position {lat,lon,h} (rad,m)
double *r	O	ECEF position {x,y,z} (m)

#### **[Return]**

None

#### **[Description]**

Transform geodetic position (latitude, longitude and height) to ECEF position.

#### **[Notes]**

WGS84, ellipsoidal height.

## A.29 ECEF to Local Coordinates

### **ecef2enu()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void ecef2enu(const double *pos, const double *r, double *e);
```

#### **[Args]**

double *pos	I	geodetic position {lat,lon} (rad)
double *r	I	vector in ECEF coordinates {x,y,z}
double *e	O	vector in local tangential coordinates {e,n,u}

#### **[Return]**

None

#### **[Description]**

Transform ECEF vector to local tangential coordinates.

#### **[Notes]**

## A.30 Local to ECEF Coordinates

### **enu2ecef()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void enu2ecef(const double *pos, const double *e, double *r);
```

#### **[Args]**

double *pos	I	geodetic position {lat,lon} (rad)
double *e	I	vector in local tangential coordinates {e,n,u}
double *r	O	vector in ECEF coordinates {x,y,z}

#### **[Return]**

None

#### **[Description]**

Transform local tangential coordinates vector to ECEF.

#### **[Notes]**

## A.31 Covariance in Local Coordinates

### **covenu( )**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void covenu(const double *pos, const double *P, double *Q);
```

#### **[Args]**

double *pos	I	geodetic position {lat,lon} (rad)
double *P	I	covariance in ECEF coordinates
double *Q	O	covariance in local tangential coordinates

#### **[Return]**

None

#### **[Description]**

Transform ECEF covariance to local tangential coordinates.

#### **[Notes]**



## A.32 Geoid Height

### **geoidh()**

#### **[Prototype]**

```
#include "rtklib.h"
double geoidh(const double *pos);
```

#### **[Args]**

double \*pos     I        geodetic position {lat,lon} (rad)

#### **[Return]**

Geoid height (m) (0.0: Error/out of range)

#### **[Description]**

Get geoid height from geoid model.

#### **[Notes]**

Geoid model is derived from EGM96. Only supports Japan area (longitude=120-155deg, latitude=20-50 deg).

### A.33 Load Datum Transformation Parameter

#### **loaddatump( )**

##### **[Prototype]**

```
#include "datum.h"  
int loaddatump(const char *file);
```

##### **[Args]**

char \*file     I       datum transformation parameter file path

##### **[Return]**

Status (0:ok,0>:error)

##### **[Description]**

Load datum transformation parameter.

##### **[Notes]**

Parameters file shall comply with GSI TKY2JGD.par.

## A.34 Tokyo Datum to JGD2000 Datum

### **tokyo2jgd( )**

#### **[Prototype]**

```
#include "datum.h"
int tokyo2jgd(double *pos);
```

#### **[Args]**

double *pos	I	position in Tokyo datum	{lat,lon,h}	(rad,m)
	O	position in JGD2000 datum	{lat,lon,h}	(rad,m)

#### **[Return]**

Status (0:ok,0>:error,out of range)

#### **[Description]**

Transform position in Tokyo datum to JGD2000 datum.

#### **[Notes]**

Before calling, call **load\_datump( )** to set parameter table.

## A.35 JGD2000 Datum to Tokyo Datum

### **jgd2tokyo( )**

#### **[Prototype]**

```
#include "datum.h"
int jgd2tokyo(double *pos);
```

#### **[Args]**

double *pos	I	position in JGD2000 datum {lat,lon,h} (rad,m)
	O	position in Tokyo datum {lat,lon,h} (rad,m)

#### **[Return]**

Status (0:ok,0>:error,out of range)

#### **[Description]**

Transform position in Tokyo datum to JGD2000 datum.

#### **[Notes]**

Before calling, call **loaddatum( )** to set parameter table.

## A.36 Read Antenna Phase Center Parameters

### **readpcv()**

#### **[Prototype]**

```
#include "rtklib.h"

int readpcv(const char *file, const char *ant, pcv_t *pcv);
```

#### **[Args]**

```
char   *file   I      antenna phase center parameter file
char   *ant    I      antenna model
pcv_t  *pcv     I      antenna phase center parameters
```

#### **[Return]**

Status (0:ok,0>:error)

#### **[Description]**

Read antenna phase center parameters from igs\_pcv format file.

#### **[Notes]**

Antenna phase center parameter struct is defined as follows.

```
typedef struct {          /* antenna phase center parameters */
    double ecc[3];        /* antenna delta-position e/n/u (m) */
    double off[NFREQ][3]; /* phase center offset e/n/u (m) */
    double var[NFREQ][19]; /* phase center variation (m) (el=0,5,...,90deg) */
} pcv_t;
```

## A.37 Read Station Positions

### **readpos()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void readpos(const char *file, const char *rcv, double *pos);
```

#### **[Args]**

char	*file	I	station position file containing lat(deg) lon(deg) height(m) name in a line
char	*rcvs	I	station name
double	*pos	O	station position {lat,lon,h} (rad/m) (all 0 if search error)

#### **[Return]**

none

#### **[Description]**

Read positions from station position file.

#### **[Notes]**

## A.38 Expand file path

### **expandpath( )**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
int expandpath(const char *path, char *paths[], int nmax);
```

#### **[Args]**

char	*path	I	file path to expand (capital insensitive)
char	*paths	O	expanded file paths
int	nmax	I	max number of expanded file paths

#### **[Return]**

Number of expanded file paths.

#### **[Description]**

Expand file path with wild-card (\*) in file.

#### **[Notes]**

## A.39 Read RINEX Files

### **readrnx()**

#### **[Prototype]**

```
#include "rinex.h"

int readrnx(char **files, int n, obs_t *obs, nav_t *nav);
```

#### **[Args]**

char	*files[]	I	RINEX files (wild-card * expanded)
int	n	I	number of RINEX files (0: Input from STDIN)
obs_t	*obs	O	observation data
nav_t	*nav	O	navigation messages

#### **[Return]**

Number of epochs (0: No data)

#### **[Description]**

Read RINEX observation data (OBS) and navigation message (NAV) files.

#### **[Notes]**

Observation data are sorted by time, receiver and satellite number. Navigation messages are sorted by TOE. Duplicated ephemerides are deleted. Observation data type `obs_t` and Navigation message type `nav_t` are defined as follows. Observation data receiver number `obs->data[i].rcv` is started as 1 and incremented observation data file by file. Supports RINEX 2.10 but only GPS.

```
typedef struct {          /* observation data record */
    gtime_t time;         /* receiver sampling time */
    int sat,rcv;          /* satellite/receiver number */
    double L[NFREQ];      /* observation data carrier-phase (cycle) */
    double P[NFREQ];      /* observation data pseudorange (m) */
    short LLI[NFREQ];     /* loss of lock indicator */
} obsd_t;

typedef struct {          /* observation data */
    int n;                /* number of observation data records */
    obsd_t *data;         /* observation data records */
} obs_t;

typedef struct {          /* satellite ephemeris and clock parameters */
    int sat;              /* satellite number */
    int iode,iodec;       /* IODE,IODC */
    int sva,svh;          /* sv accuracy, sv health */
}
```



```

    gtime_t toe,toc,ttr; /* Toe,Toc,T_trans */
    double A,e,i0,OMG0,omg,M0,deln,OMGd,idot, crs,crs,cuc,cus,cic,
           cis,toes; /* sv ephemeris parameters */
    double f0,f1,f2,tgd; /* sv clock parameters */
} eph_t;

typedef struct { /* navigation messages */
    int n; /* number of ephemeris and clock parameters */
    eph_t *eph; /* satellite ephemeris and clock parameters */
    double ion[8]; /* iono model params {a0,a1,a2,a3,b0,b1,b2,b3} */
    double utc[4]; /* delta-utc parameters */
} nav_t;

```

## A.40 Read RINEX Files in Time Range/Interval

### **readrnxt()**

#### **[Prototype]**

```
#include "rinex.h"

int readrnxt(char *files[], int n, gtime_t ts, gtime_t te, double tint,
             obs_t *obs, nav_t *nav);
```

#### **[Args]**

char *files[]	I	RINEX files (wild-card * expanded)
int n	I	number of RINEX files (0:input from stdin)
gtime_t ts,te	I	observation time start/end (ts>=te:all)
double tint	I	observation time interval (sec) (0:all)
obs_t *obs	O	observation data
nav_t *nav	O	navigation messages

#### **[Return]**

Number of epochs (0: No data)

#### **[Description]**

Read RINEX observation data (OBS) and navigation message (NAV) files.

#### **[Notes]**

See readobs().

## A.41 Output RINEX OBS Header

### outrnxobsh( )

#### [Prototype]

```
#include "rinex.h"

void outrnxobsh(FILE *fp, const char *pname, const char *runby,
                const char *mname, const char *obsv, const char *agency,
                const char *rname, const char *aname, const double *pos,
                const double *ant, char tobs[][3], int ntobs,
                gtime_t time, char **comments, int ncomm);
```

#### [Args]

FILE	*fp	I	output file pointer
char	*pname	I	program name
char	*runby	I	run-by name
char	*mname	I	antenna marker name
char	*obsv	I	observer name
char	*agency	I	agency name
char	*rname	I	receiver name (number,type,version)
char	*aname	I	antenna name (number,type,version)
double	*pos	I	approx position x/y/z (m)
double	*ant	I	antenna delta h/e/n (m)
char	tobs[][3]	I	observation types
int	ntobs	I	number of observation types
gtime_t	time	I	time of first observation record
char	**comment	I	comments
int	ncomm	I	number of comments

#### [Return]

None

#### [Description]

Output RINEX OBS file header.

## A.42 Output RINEX OBS Body

### **outrnxobsb()**

#### **[Prototype]**

```
#include "rinex.h"

void outrnxobsb(FILE *fp, const obsd_t *obs, int n, int epflag,
                char tobs[][3], int ntobs);
```

#### **[Args]**

FILE	*fp	I	output file pointer
obsd_t	*obs	I	observation data
int	n	I	number of observation data
int	epflag	I	epoch flag (0:ok,1:power failure,>1:event flag)
char	tobs[][3]	I	observation types
int	ntobs	I	number of observation types

#### **[Return]**

None

#### **[Description]**

Output RINEX OBS body

#### **[Notes]**

## A.43 Output RINEX NAV Header

### **outrnxnavh()**

#### **[Prototype]**

```
#include "rinex.h"

void outrnxnavh(FILE *fp, const char *pname, const char *runby,
                const double *ion, const double *dutc, int leaps,
                char *comments[], int ncomm);
```

#### **[Args]**

FILE	*fp	I	output file pointer
char	*pname	I	program name
char	*runby	I	run-by name
double	*ion	I	ionospheric parameters (NULL: no output)
double	*dutc	I	delta-utc parameters (NULL: no output)
int	leaps	I	leap seconds (s) (<0: no output)
char	**comment	I	comments
int	ncomm	I	number of comments

#### **[Return]**

None

#### **[Description]**

Output RINEX NAV header.

#### **[Notes]**

## A.44 Output RINEX NAV Body

### **outrnxnavb( )**

#### **[Prototype]**

```
#include "rinex.h"
void outrnxnavb(FILE *fp, const eph_t *eph);
```

#### **[Args]**

FILE	*fp	I	output file pointer
eph_t	*eph	I	ephemeris record

#### **[Return]**

None

#### **[Description]**

Output RINEX NAV body.

#### **[Notes]**

## A.45 Uncompress File

### **uncompress ( )**

#### **[Prototype]**

```
#include "rinex.h"
```

```
int uncompress(const char *file, char *uncfile);
```

#### **[Args]**

char *file	I	input file
------------	---	------------

char *uncfile	O	uncompressed file
---------------	---	-------------------

#### **[Return]**

Status (-1:error,0:not compressed file,1:uncompress completed)

#### **[Description]**

Uncompress (uncompress/unzip/uncompact hatanaka-compression) file.

#### **[Notes]**

Creates uncompressed file in temporary directory. `gzip` and `crx2rnx` command have to be installed in commands path.

## A.46 LAMBDA/MLAMBDA Integer Least-Square Estimation

### **lambda ( )**

#### **[Prototype]**

```
#include "rtklib.h"

int lambda(int n, int m, const double *a, const double *Q, double *F,
          double *s);
```

#### **[Args]**

int	n	I	number of float parameters
int	m	I	number of fixed solutions
double	*a	I	float parameters (n x 1)
double	*Q	I	covariance matrix of float parameters (n x n)
double	*F	O	fixed solutions (n x m)
double	*s	O	sum of squared residulas of fixed solutions (1 x m)

#### **[Return]**

Status (0:OK, other:Error)

#### **[Description]**

Integer least-square estimation.

Reduction is performed by LAMBDA (ref.[1]), and search by MLAMBDA (ref.[2]).

#### **[Notes]**

Matrix stored by column-major order (FORTRAN convension).

#### References

- [1] P.J.G.Teunissen, The least-square ambiguity decorrelation adjustment: a method for fast GPS ambiguity estimation, *J.Geodesy*, Vol.70, 65-82, 1995
- [2] X.-W.Chang, X.Yang, T.Zhou, MLAMBDA: A modified LAMBDA method for integer least-squares estimation, *J.Geodesy*, Vol.79, 552-565, 2005



## A.47 Satellite Ephemeris to Satellite Position/Clock-bias

### **eph2pos()**

#### **[Prototype]**

```
#include "rtklib.h"

void eph2pos(gtime_t t, const eph_t *eph, double pr, double *rs,
             double *dts);
```

#### **[Args]**

gtime_t t	I	time (GPSTIME)
eph_t *eph	I	satellite ephemeris and clock parameter
double pr	I	pseudorange measurement (m)
double *rs	O	satellite ECEF position at signal transmission {x,y,z} (m)
double *dts	O	satellite clock-bias at signal transmission (sec)

#### **[Return]**

None

#### **[Description]**

Compute satellite position and clock-bias at signal transmission from satellite ephemeris and clock parameters.

#### **[Notes]**

Satellite position is expressed in ECEF coordinates at signal reception time.

For type definition of eph\_t see readrx().

## A.48 Satellite Positions/Clock-biases

### **satpos()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void satpos(const obsd_t *obs, int n, const nav_t *nav, double *rs,
            double *dts);
```

#### **[Args]**

obsd_t	*obs	I	observation data records
int	n	I	number of observation data records
nav_t	*nav	I	navigation messages
double	*rs	O	satellite ECEF positions at signal transmission {x,y,z} (m)
double	*dts	O	satellite clock-biases at signal transmission (sec)

#### **[Return]**

None

#### **[Description]**

Compute satellite positions and clock-biases at signal transmission from navigation messages.

#### **[Notes]**

$rs[(0:2)+i*3], dts[i] = obs[i]$  satellite position/clock-bias.

If no ephemeris and clock parameter, set 0.0 to  $rs[]$ ,  $dts[]$ .

Satellite positions are expressed in ECEF coordinates at signal reception time.

For type definition of `obsd_t` and `nav_t`, see `readrnx()`.

## A.49 Satellite Positions/Clock-biases by IODE

### **satposiode()**

#### **[Prototype]**

```
#include "rtklib.h"

void satposiode(gtime_t time, const obsd_t *obs, int *iode, int n,
               const nav_t *nav, double rs[][3], double *dts);
```

#### **[Args]**

gtime_t time	I	time to search ephemeris
obsd_t *obs	I	observation data records
int *iode	I	IODEs
int n	I	number of observation data records
nav_t *nav	I	navigation messages (sorted by transmsion time)
double rs[][3]	O	satellilte positions (ecef) (m)
double *dts	O	satellilte clock-bias (sec)

#### **[Return]**

None

#### **[Description]**

Compute satellite positions and clock-biases from satellite ephemerides selected by IODE.

#### **[Notes]**

## A.50 Satellite Positions/Velocities/Clock-biases/Clock-drifts

### **satpviode()**

#### **[Prototype]**

```
#include "rtklib.h"

void satpviode(gtime_t time, const obsd_t *obs, int *iode, int n,
               const nav_t *nav, double rs[][6], double dts[][2]);
```

#### **[Args]**

gtime_t time	I	time to search ephemeris
obsd_t *obs	I	observation data records
int *iode	I	IODEs
int n	I	number of observation data records
nav_t *nav	I	navigation messages (sorted by transmsion time)
double rs[][6]	O	satellilte positions/velocities (ecef) (m,m/s)
double dts[][2]	O	satellilte clock-biases/clock-drifts (s,s/s)

#### **[Return]**

None

#### **[Description]**

Compute satellite positions, velocities, clock-biases and clock-drifts from satellite ephemerides selected by IODE.

#### **[Notes]**

## A.51 Geometric Distance

### **geodist()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double geodist(const double *rs, const double *rr, double *e);
```

#### **[Args]**

double *rs	I	satellite ECEF position at signal transmission {x,y,z} (m)
double *rr	I	receiver ECEF position {x,y,z} (m)
double *e	O	receiver-to-satellite unit vector {x,y,z}

#### **[Return]**

Geometric distance (m) (0>: Error/no satellite position)

#### **[Description]**

Compute geometric distance between satellite and receiver, and receiver-to-satellite unit vector.

#### **[Notes]**

Satellite position and receiver-to-satellite unit vector are expressed in ECEF coordinates at signal reception time.

## A.52 Satellite Azimuth/Elevation Angle

### **satazel()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void satazel(const double *pos, const double *e, double *azel);
```

#### **[Args]**

double *pos	I	receiver geodetic position {lat,lon,h} (rad, m)
double *e	I	receiver-to-satellite unit vector {x,y,z}
double *azel	O	satellite azimuth/elevation angle {az,el} (rad)

#### **[Return]**

None

#### **[Description]**

Compute satellite azimuth/elevation angle from receiver.

#### **[Notes]**

Receiver-to-satellite unit vector are expressed in ECEF coordinates at signal reception time.

## A.53 Compute DOPs

### **dops ( )**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void dops(int ns, const double azel[][2], double elmin, double *dop);
```

#### **[Args]**

int	ns	I	number of satellites
double	*azel	I	satellite azimuth/elevation angle (rad)
double	elmin	I	elevation cutoff angle (rad)
double	*dop	O	DOPs {GDOP,PDOP,HDOP,VDOP}

#### **[Return]**

None

#### **[Description]**

Compute DOP (dilution of precision).

#### **[Notes]**

## A.54 Antenna Model

### **antmodel()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void antmodel(const pcv_t *pcv, const double *azel, double *dant);
```

#### **[Args]**

pcv_t *pcv	I	antenna phase center parameters
double *azel	I	azimuth/elevation angle {az,el} (rad)
double *dant	O	range offsets for each frequency (m)

#### **[Return]**

None

#### **[Description]**

Compute antenna offset by antenna phase center parameters.

#### **[Notes]**

See readpcv().



## A.55 Carrier Smoothing

### **csmooth()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
void csmooth(obs_t *obs, int ns);
```

#### **[Args]**

obs_t	*obs	IO	raw observation data/smoothed observation data
int	ns	I	smoothing window size (epochs)

#### **[Return]**

None

#### **[Description]**

Carrier smoothing by Hatch filter.

#### **[Notes]**

## A.56 Ionospheric Model

### **ionmodel()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double ionmodel(gtime_t t, const double *ion, const double *pos,
                const double *azel);
```

#### **[Args]**

gtime_t t	I	RTKLIB time struct
double *ion	I	ionospheric model parameters {a0,a1,a2,a3,b0,b1,b2,b3}
double *pos	I	receiver geodetic position {lat,lon,h} (rad, m)
double *azel	I	satellite azimuth/elevation angle {az,el} (rad)

#### **[Return]**

Ionospheric delay (L1) (m)

#### **[Description]**

Compute ionospheric delay by broadcast ionosphere model (Klobuchar model).

#### **[Notes]**

## A.57 Ionospheric Mapping Function

### **ionmapf()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double ionmapf(const double *pos, const double *azel);
```

#### **[Args]**

```
double *pos    I      receiver position {lat,lon,h} (rad,m)
```

```
double *azel   I      azimuth/elevation angle {az,el} (rad)
```

#### **[Return]**

Ionospheric mapping function.

#### **[Description]**

Compute ionospheric delay mapping function by single layer model.

#### **[Notes]**

## A.58 Tropospheric Model

### **tropmodel()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
double tropmodel(const double *pos, const double *azel);
```

#### **[Args]**

double *pos	I	receiver geodetic position {lat,lon,h} (rad,m)
double *azel	I	satellite azimuth/elevation angle {az,el} (rad)

#### **[Return]**

Tropospheric delay (m)

#### **[Description]**

Compute tropospheric delay by Standard Atmosphere and Saastamoinen model.

#### **[Notes]**

## A.59 Tropospheric Mapping Function (NMF)

### **tropmapf()**

#### **[Prototype]**

```
#include "rtklib.h"

double tropmapf(gtime_t time, const double pos[], const double azel[],
                double *mapfw);
```

#### **[Args]**

gtime_t t	I	time
double pos[]	I	receiver position {lat,lon,h} (rad,m)
double azel[]	I	azimuth/elevation angle {az,el} (rad)
double *mapfw	IO	wet mapping function (NULL: not output)

#### **[Return]**

Dry mapping function.

#### **[Description]**

Compute tropospheric mapping function by NMF.

#### **[Notes]**

Reference: A.E.Niell, Global mapping functions for the atmosphere delay at radio wavelengths, J.Geophys.Res. 1996

## A.60 Single Point Positioning

### **pntpos( )**

#### **[Prototype]**

```
#include "rtklib.h"

int pntpos(const obsd_t *obs, int n, const nav_t *nav, double elmin,
           double snrmin, double *rr, double *Qr, double *dtr,
           double azel[][2]);
```

#### **[Args]**

obsd_t	*obs	I	observation data records
int	n	I	number of observation data records
nav_t	*nav	I	navigation messages
double	elmin	I	elevation cutoff angle (rad)
double	snrmin	I	SNR mask (dBHz)
double	*rr	IO	initial/estimated receiver position (ecef) (m)
double	*Qr	O	estimated position covariance (3 x 3)
double	*dtr	O	estimated receiver clock-bias (sec)
double	azel[][2]	O	satellite azimuth/elevation angle (rad)

#### **[Return]**

Number of valid satellites (0>:error)

#### **[Description]**

Compute receiver position and clock-bias by Single Point Positioning.

#### **[Notes]**

`rs[(0:2)+i*3],dts[i] = obs[i]` Satellite position/clock-bias

`azel[(0:1)+i*2] = obs[i]` Satellite azimuth/elevation angle

Satellite positions are expressed in ECEF coordinates at signal reception time.

For type definition of `obs_t` see `readrnx( )`.

## A.61 RTK positioning

### **rtkpos()**

#### **[Prototype]**

```
#include "rtklib.h"

void rtkpos(infunc_t input, incfunc_t inputc, outfunc_t output,
           const opt_t *opt);
```

#### **[Args]**

infunc_t input	I	input callback function
incfunc_t inputc	IO	input network correction callback function (NULL: no correction)
outfunc_t output	I	output callback function
opt_t *opt	I	positioning options

#### **[Return]**

None

#### **[Description]**

Input observation data and navigation message, compute rover position by relative positioning and output results. if callback input() return 0, exit the function.

#### **[Notes]**

Input, Input correction, output and output correction callback functions shall comply the following prototypes.

```
int input(obsd_t *obs, nav_t *nav);
    obsd_t **obs    0      observation data records for an epoch
                        obs[i].rcv=1:rover,2:reference
                        sorted by receiver and satellite
    nav_t  **nav     0      navigation messages
    sbs_t  **sbs     0      sbas corrections
    return                                number of observation data records

void output(gtime_t t, sol_t sol, sol_t solf, int stat, int ns);
    gtime_t t        I      time
    sol_t sol        I      float positioning solution
    sol_t solf       I      fixed positioning solution
    double *rb       I      reference (base) station position (ecef) (m)
    int  stat        I      status (1:fix,2:float,4:dgps,5:single)
    int  ns          I      number of valid satellites
    return                                none
```

```

int inputc(gtime_t t, const double *obsd_t *obs, nav_t *nav);
    gtime_t t          I          time
    double *rr          I          receiver position (ecef) (m)
    obsd_t **obsd       O          network corrections for an epoch
    return              O          number of network correction

```

The type `opt_t`, `sol_t` and `obsd_t` are defined as follows. For type definition of `obsd_t` and `nav_t`, see `readrnx()`.

```

typedef struct {
    int mode;
    int nf;
    double elmin;
    double snrmin;
    int modear;
    int maxout;
    int minlock;
    int estiono;
    int esttrop;
    int codesmooth;
    int intpref;
    int sbascorr;
    double err[4];
    double std[3];
    double prn[3];
    double sclkstb;
    double thresar;
    double elmaskar;
    double thresslip;
    double maxtdiff;
    double maxinno;
    double ru[3];
    double rb[3];
    pcv_t pcvu;
    pcv_t pcvr;
} opt_t;

typedef struct {
    int n;
    double *rr;
    double *Qr;
} sol_t;

typedef struct {
    gtime_t time;
    int sat, iode;
    double cc[NFREQ];
} obsd_t;
/* rtkpos processing options */
/* positioning mode */
/* (0:single,1:dgps,2:kinematic,3:static, */
/* 4:moving-baseline,5:fixed) */
/* number of frequencies (1:L1,2:L1+L2,3:L1+L2+L5) */
/* elevation mask angle (rad) */
/* snr mask (dBHz) */
/* AR mode (0:continuous,1:instantaneous) */
/* obs outage count to reset bias */
/* min lock count to fix ambiguity */
/* ionosphere estimation */
/* troposphere estimation */
/* code smoothing window size (0:none) */
/* interpolate reference obs */
/* sbas correction options */
/* measurement error factor */
/* [0]:code/phase, [1-3]:error factor a/b/c of phase */
/* initial-state std [0]bias,[1]ionos [2]tropos */
/* process-noise std [0]bias,[1]ionos [2]tropos */
/* satellite clock stability (sec/sec) */
/* AR validation threshold */
/* elevation mask of AR for rising satellite (deg) */
/* slip threshold of geometry-free phase (m) */
/* max difference of time (sec) */
/* reject threshold of innovation (m) */
/* rover position for fixed mode(5) (ecef) (m) */
/* reference position for rel mode(1-3) (ecef) (m) */
/* antenna phase center parameters of rover */
/* antenna phase center parameters of reference */
/* estimation result type */
/* number of solutions */
/* estimated solutions */
/* estimated solutions covariance */
/* network correction */
/* correction time */
/* satellite number/ephemeris iode */
/* carrier-phase correction (m) */

```



## A.62 Post-processing Positioning

### **postpos()**

#### **[Prototype]**

```
#include "rtklib.h"

int postpos(gtime_t ts_, gtime_t te_, double tint_, const opt_t *opt_,
            char *sep_, int timeu_, int dirs_, int times_, int timef_,
            int posf_, int degf_, char **infile, int nfile, char *outfile,
            char **corrfile_, int ncorr_);
```

#### **[Args]**

gtime_t ts,te	I	processing start/end time
double tint	I	processing interval (s) (0:all)
opt_t *opt	I	processing options
char *sep	I	output field separator
int timeu	I	digits under decimal point
int dirs	I	proc direction (0:forward,1:backward,2:combined)
int times	I	time system
int timef	I	time format
int posf	I	solution format
int degf	I	latitude/longitude format
char *infile[]	I	input files
int nfile	I	number of input files
char *outfile	I	output file (":stdout)
char **corrfile	I	correction files
int ncorr	I	number of correction file

#### **[Return]**

Status (0:ok,0>:error,1:aborted).

#### **[Description]**

Post-processing positioning.

#### **[Notes]**

## A.63 Read Positioning Options

### **readopt()**

#### **[Prototype]**

```
#include "rtklib.h"
```

```
int readopt(const char *file, opt_t *opt);
```

#### **[Args]**

char	*file	I	processing option file
opt_t	*opt	O	processing option

#### **[Return]**

Status (0:ok,0>:error).

#### **[Description]**

Read positioning options from file.

#### **[Notes]**

See `rtkpos()`.

## A.64 Write Positioning Options

### **writeopt()**

#### **[Prototype]**

```
#include "rtklib.h"
int writeopt(const char *file, opt_t *opt);
```

#### **[Args]**

char	*file	I	processing option file
opt_t	*opt	I	processing option

#### **[Return]**

Status (0:ok,0>:error).

#### **[Description]**

Save positioning options to file.

#### **[Notes]**

See `rtkpos()`.

## A.65 Read Solutions

### **readsol()**

#### **[Prototype]**

```
#include "solution.h"

int readsol(char *files[], int nfile, solp_t *solp, double rpos[][3],
            int *nrpos, char **inputs, int *ninp);
```

#### **[Args]**

char	*files[]	I	solution files
int	nfile	I	number of solution files
solp_t	*solp	O	position solutions
double	rpos[][]	IO	ref station pos {lat,lon,height} (NULL: no input)
int	*nrpos	IO	number of ref station pos (NULL: no input)
char	**inputs	IO	input files (NULL: no input)
int	*ninp	IO	number of input files (NULL: no input)

#### **[Return]**

Number of solution data.

#### **[Description]**

Read position solution from file as text position format.

#### **[Notes]**

## A.66 Read Solutions in Time Range/Interval

### **readsolt()**

#### **[Prototype]**

```
#include "solution.h"

int readsolt(char *files[], int nfile, gtime_t ts, gtime_t te,
             double tint, int qflag, solp_t *solp, double rpos[][3],
             int *nrpos, char **inputs, int *ninp);
```

#### **[Args]**

char	*files[]	I	solution files
int	nfile	I	number of solution files
gtime_t	ts	I	start time
gtime_t	te	I	end time
double	tint	I	time interval
int	qflag	I	quality flag (0:all)
solp_t	*solp	O	position solutions
double	rpos[][]	IO	reference station position {lat,lon,height} (NULL: no input)
int	*nrpos	IO	number of ref station pos (NULL: no input)
char	**inputs	IO	input files (NULL: no input)
int	*ninp	IO	number of input files (NULL: no input)

#### **[Return]**

Number of solution data.

#### **[Description]**

Read solution from file as text position format screened by time span, time interval and quality flag.

#### **[Notes]**

## A.67 Output Solution Header

### **outsolhead()**

#### **[Prototype]**

```
#include "solution.h"
```

```
void outsolhead(FILE *fp);
```

#### **[Args]**

```
FILE    *fp    I        output file pointer
```

#### **[Return]**

None

#### **[Description]**

Output solution heade to file.

#### **[Notes]**

Before calling that set solution options with `setsolopt()`.

## A.68 Output Solution Header

### **outsolhead()**

#### **[Prototype]**

```
#include "solution.h"

void outsol(FILE *fp, gtime_t time, const double *rr, const double *Qr,
            int stat, int ns);
```

#### **[Args]**

FILE	*fp	I	output file pointer
gtime_t	time	I	time
double	*rr	I	solution position (m) (ecef)
double	*Qr	I	solution covariance matrix (m)
int	stat	I	solution status flag
int	ns	I	number of satellites

#### **[Return]**

None

#### **[Description]**

Output solution body to file.

#### **[Notes]**

Before calling that set solution options with `setsolopt()`.

## A.69 Set Solution Output Options

### **setsolopt()**

#### **[Prototype]**

```
#include "solution.h"

void setsolopt(char *sep_, int timeu_, int times_, int timef_, int posf_,
               int degf_);
```

#### **[Args]**

char	*sep	I	output field separator for text position
int	timeu	I	digits under decimal point
int	times	I	time system (0:gpst,1:utc)
int	timef	I	time format (0:sssss.s,1:yyyy/mm/dd hh:mm:ss.s)
int	posf	I	solution format (0:lat,lon,hgt,1:x,y,z-ecef,2:nmea)
int	degf	I	latitude/longitude format (0:ddd.ddd,1:ddd mm ss)

#### **[Return]**

None

#### **[Description]**

Set output options of text position format.

#### **[Notes]**



## A.70 Convert Solution File to Google Earth KML File

### **convkml()**

#### **[Prototype]**

```
#include "convkml.h"
```

```
void convkml(const char *infile, const char *outfile, gtime_t ts,
             gtime_t te, double *offset, int tcolor, int pcolor,
             int outalt, int outtime, double tint, int qflg);
```

#### **[Args]**

char *infile	I	input solution file
char *outfile	I	output Google Earth KML file
gtime_t ts,te	I	solution start, end time
double *offset	I	offsets to add to solution {east,north,up} (m)
int tcolot	I	track line color (1:white,2:green,3:orange,4:red,5:yellow)
int pcolot	I	track point color (same as above)
int outalt	I	altitude output flag (0:off,1:on)
int outtime	I	time output flag (0:off,1:on)
double tint	I	time interval (s)
int qflag	I	output quality flag (0:all)

#### **[Return]**

None

#### **[Description]**

Convert solution file to Google Earth KML file.

#### **[Notes]**

## A.71 Read SBAS Message File

### **sbsreadmsg()**

#### **[Prototype]**

```
#include "sbas.h"

int sbsreadmsg(const char *file, int sat, sbsmsg_t **sbsmsg);
```

#### **[Args]**

```
char  *file      I      SBAS message file
int    sat        I      SBAS satellite prn number (0:all)
sbsmsg_t **sbsmsg O  SBAS message array
```

#### **[Return]**

Number of sbas messages.

#### **[Description]**

Read sbas message file.

#### **[Notes]**

SBAS message file shall be one of:

- (1) NovAtel OEM-4/5 RAWWAASFRAMEA messages
- (2) NovAtel OEM-3 FRMA messages
- (3) RTKLIB SBAS messages form

SBAS message struct sbsmsg\_t is defined as follows.

```
typedef struct {          /* sbas message */
    int week,tow;         /* receiption time */
    int sat;              /* sbas satellite prn number */
    unsigned char msg[29]; /* sbas message (226bit) padded by 0 */
} sbsmsg_t;
```

## A.72 Read SBAS Message File in Time Range

### **sbsreadmsgt()**

#### **[Prototype]**

```
#include "sbas.h"

int sbsreadmsgt(const char *file, int sat, gtime_t ts, gtime_t te,
                sbsmsg_t **sbsmsg);
```

#### **[Args]**

char	*file	I	SBAS message file
int	sat	I	SBAS satellite prn number (0:all)
gtime_t	ts	I	start time
gtime_t	te	I	end time
sbsmsg_t	**sbsmsg	O	SBAS message array

#### **[Return]**

Number of sbas messages.

#### **[Description]**

Read sbas message file in time range.

#### **[Notes]**

See sbsreadmsg().

## A.73 Output SBAS Messages

### **sbsoutmsg( )**

#### **[Prototype]**

```
#include "sbas.h"
void sbsoutmsg(FILE *fp, sbsmsg_t *sbsmsg);
```

#### **[Args]**

FILE	*fp	I	output file pointer
sbsmsg_t	*sbsmsg	I	output SBAS messages

#### **[Return]**

None

#### **[Description]**

Output sbas message to file as the RTKLIB form.

#### **[Notes]**

See sbsreadmsg( ).

## A.74 Update SBAS Status

### sbsupdatestat()

#### [Prototype]

```
#include "sbas.h"
```

```
void sbsupdatestat(const sbsmsg_t *msg, sbsstat_t *stat);
```

#### [Args]

```
sbsmsg_t  *msg  I      SBAS message
sbsstat_t *stat IO    SBAS status struct
```

#### [Return]

None

#### [Description]

Update sbas status struct with a sbas message.

#### [Notes]

SBAS status struct sbsstat\_t is defined as follows.

```
typedef struct {
    int sat;           /* sbas navigation message type */
    gtime_t t0;        /* satellite prn number */
    int ura;           /* reference epoch time */
    double pos[3];     /* URA+1 (user range accuracy) */
    double vel[3];     /* satellite position (m) (ecef) */
    double acc[3];     /* satellite velocity (m/s) (ecef) */
    double af0,af1;    /* satellite acceleration (m/s^2) (ecef) */
    double af2,af3;    /* satellite clock-offset/drift (s,s/s) */
} sbsnav_t;

typedef struct {
    gtime_t t0;        /* fast correction type */
    double prc;        /* time of applicability (tof) */
    double rrc;        /* pseudorange correction (PRC) (m) */
    double dt;         /* range-rate correction (RRC) (m/s) */
    int iodf;          /* range-rate correction delta-time (s) */
    short udrei;       /* IODF (issue of date fast corr) */
    short ai;          /* UDREI+1 */
    short ai;          /* degradation factor indicator */
} sbsfcrr_t;

typedef struct {
    gtime_t t0;        /* long term satellite error correction type */
    int iode;          /* correction time */
    double dpos[3];    /* IODE (issue of date ephemeris) */
    double dvel[3];    /* delta position (m) (ecef) */
    double daf0,daf1;  /* delta velocity (m/s) (ecef) */
    double daf2,daf3;  /* delta clock-offset/drift (s,s/s) */
}
```

```

} sbslcorr_t;

typedef struct {          /* satellite correction type */
    int prn;              /* satellite prn number */
    sbsfcorr_t fcorr;     /* fast correction */
    sbslcorr_t lcorr;     /* long term correction */
} sbssatp_t;

typedef struct {          /* satellite corrections type */
    int iodp;             /* IODP (issue of date mask) */
    int nprn;             /* number of satellites */
    int tlat;             /* system latency (s) */
    sbssatp_t sat[MAXNPRN]; /* satellite correction */
} sbssat_t;

typedef struct {          /* ionospheric correction type */
    gtime_t t0;           /* correction time */
    short lat,lon;        /* latitude/longitude (deg) */
    short give;           /* GIVE+1 */
    float delay;          /* vertical delay estimate (m) */
} sbsigp_t;

typedef struct {          /* ionospheric corrections type */
    int iodi;             /* IODI (issue of date ionos corr) */
    int nigp;             /* number of igps */
    sbsigp_t igp[MAXNIGP]; /* ionospheric correction */
} sbsion_t;

typedef struct {          /* igp band type */
    short x;              /* longitude/latitude (deg) */
    const short *y;       /* latitudes/longitudes (deg) */
    unsigned char bits;    /* igp mask start bit */
    unsigned char bite;    /* igp mask end bit */
} sbsigpband_t;

typedef struct {          /* sbas status type */
    sbssat_t sbssat;      /* satellite corrections */
    sbsion_t sbsion[MAXBAND+1]; /* ionos corrections */
    sbsnav_t sbsnav[MAXNGEO]; /* sbas navigation messages */
} sbsstat_t;

```

## A.75 Decode SBAS Message

### **sbsdecodemsg()**

#### **[Prototype]**

```
#include "sbas.h"

int sbsdecodemsg(gtime_t time, int sat, const unsigned int *words,
                 sbsmsg_t *sbsmsg);
```

#### **[Args]**

```
gtime_t time    I      reception time
int      sat     I      SBAS satellite prn number
unsigned int *word I message frame words (24bit x 10)
sbsmsg_t *sbsmsg O      SBAS message
```

#### **[Return]**

Status (1:ok,0:CRC error)

#### **[Description]**

Decode sbas message frame.

#### **[Notes]**

## A.76 SBAS Fast Correction

### **sbsfastcorr()**

#### **[Prototype]**

```
#include "sbas.h"

int sbsfastcorr(gtime_t time, const sbssat_t *sbssat, int sat,
               double *prc, double *var);
```

#### **[Args]**

gtime_t time	I	correction time
sbssat_t *sbssat	I	SBAS satellite correction paramters
int sat	I	satellite PRN number
double *prc	O	pseudorange correction (m)
double *var	O	variance of correction (m <sup>2</sup> )

#### **[Return]**

Status (1:ok,0:no correction).

#### **[Description]**

Compute SBAS fast correction.

#### **[Notes]**



## A.77 SBAS Long Term Correction

### **sbslongcorr()**

#### **[Prototype]**

```
#include "sbas.h"

int sbslongcorr(gtime_t time, const sbssat_t *sbssat, int sat,
               double *drs, double *ddts, int *iode);
```

#### **[Args]**

gtime_t time	I	correction time
sbssat_t *sbssat	I	SBAS satellite correction paramters
int sat	I	satellite PRN number
double *drs	O	satellite position correction (m) (ecef)
double *ddts	O	satellite clock correction (s) (ecef)
int *iode	O	IODE

#### **[Return]**

Status (1:ok,0:no correction).

#### **[Description]**

Compute SBAS long term satellite error correction.

#### **[Notes]**

## A.78 SBAS Ionospheric Delay Correction

### **sbsioncorr()**

#### **[Prototype]**

```
#include "sbas.h"

int sbsioncorr(gtime_t time, const sbsion_t *ion, int sat,
               const double *pos, const double *azel, double *delay,
               double *var);
```

#### **[Args]**

gtime_t	time	I	time
sbsion_t	*ion	I	SBAS ionospheric parameters
double	*pos	I	receiver position {lat,lon,height} (rad/m)
double	*azel	I	satellite azimuth/elavation angle (rad)
double	*delay	O	slant ionospheric delay (L1) (m)
double	*var	O	variance of ionospheric delay (m^2)

#### **[Return]**

Status (1:ok,0:no correction).

#### **[Description]**

Compute SBAS ionospheric delay correction.

#### **[Notes]**

## A.79 SBAS Tropospheric Delay Correction

### **sbstropcorr( )**

#### **[Prototype]**

```
#include "sbas.h"
```

```
double sbstropcorr(gtime_t time, const double *pos, const double *azel,
                  double *var);
```

#### **[Args]**

gtime_t	time	I	time
double	*pos	I	receiver position {lat,lon,height} (rad/m)
double	*azel	I	satellite azimuth/elavation angle (rad)
double	*var	O	variance of troposphric error (m^2)

#### **[Return]**

Tropospheric delay (m).

#### **[Description]**

Compute SBAS tropospheric delay correction.

#### **[Notes]**

## A.80 SBAS Satellite Position

### **sbssatpos()**

#### **[Prototype]**

```
#include "sbas.h"

int sbssatpos(gtime_t time, const sbsnav_t *nav, double pr, double *rs,
              double *dts);
```

#### **[Args]**

gtime_t	time	I	time
sbsnav_t	*nav	I	SBAS ephemeris parameters
double	pr	I	pseudorange observables (m)
double	*rs	O	satellite position at transmission (m) (ecef)
double	*dts	O	satellite clock-bias (s)

#### **[Return]**

URA+1 (0:no correction).

#### **[Description]**

Compute SBAS satellite position.

#### **[Notes]**

## A.81 SBAS Point Positioning with Corrections

### **sbspntpos()**

#### **[Prototype]**

```
#include "sbas.h"

int sbspntpos(const obsd_t *obs, int n, const nav_t *nav, sbsstat_t *stat,
              double elmin, double snrmin, int opt, double *rr,
              double *Qr, double *dtr, double azel[][2]);
```

#### **[Args]**

obsd_t	*obs	I	observation data records
int	n	I	number of observation data records
nav_t	*nav	I	navigation messages
sbsstat_t	*stat	I	SBAS status struct
double	elmin	I	elevation mask (rad)
double	snrmin	I	SNR mask (dbHz)
int	opt	I	positioning options
double	*rr	IO	initial/estimated receiver position (ecef) (m)
double	*Qr	O	estimated position covariance (3 x 3)
double	*dtr	O	estimated receiver clock-bias (sec)
double	azel[][2]	O	azimuth/elevation angle (rad)

#### **[Return]**

Number of valid satellites(<0:error).

#### **[Description]**

Single point positioning with sbas dgps corrections.

#### **[Notes]**

## A.82 Decode GPS Navigation Data Frame

### **decodefrm()**

#### **[Prototype]**

```
#include "ublox.h"

int decodefrm(const unsigned int *words, eph_t *eph, double *ion,
              double *utc, int *leaps);
```

#### **[Args]**

```
unsigned int *words  I navigation data frame(24bitx10) without parity
eph_t *eph          0      ephemeris message
double *ion         0      ionospheric parameters
double *utc         0      delta-utc parameters
int *leaps          0      leap seconds (s)
```

#### **[Return]**

Subframe ID (1-3: ephemeris frame, 9:ion/utc parameters).

#### **[Description]**

Decode gps navigation data frame.

#### **[Notes]**

## A.83 Decode u-blox Messages

### **decodeubx( )**

#### **[Prototype]**

```
#include "ublox.h"

int decodeubx(const unsigned char *buff, int len, obs_t *obs, nav_t *nav,
              sbsmsg_t *sbsmsg, int *sat);
```

#### **[Args]**

unsigned char *	buff	I	message buffer
int	len	I	buffer length (bytes)
obs_t *	obs	O	observation data
nav_t *	nav	O	navigation message
sbsmsg_t *	sbsmsg	O	SBAS message
int	*sat	O	satellite PRN number for ephemeris or SBAS message

#### **[Return]**

Input status (0:no valid data, 1:obs data, 2:ephemeris, 3:SBAS message, 9:ion/utc parameters)

#### **[Description]**

Decode and input a u-blox receiver message. Support RXMRAW and RXMSFRB.

#### **[Notes]**

## A.84 Read u-blox Log File

### **readubx()**

#### **[Prototype]**

```
#include "ublox.h"
```

```
int readubx(const char *file, obs_t *obss, nav_t *navs, sbs_t *sbss);
```

#### **[Args]**

char	*file	I	input log file	
obs_t	*obss	IO	observation data	(NULL: no output)
nav_t	*navs	IO	navigation messages	(NULL: no output)
sbs_t	*sbss	IO	SBAS messages	(NULL: no output)

#### **[Return]**

None

#### **[Description]**

Read u-blox log file as obs data, nav message and sbas messages. Log file shall contain RXMRAW and RXMSFRB.

#### **[Notes]**



## A.85 Read u-blox Log File in Time Range/Interval

### **readubxt()**

#### **[Prototype]**

```
#include "ublox.h"

int readubxt(const char *file, gtime_t ts, gtime_t te, double tint,
             obs_t *obss, nav_t *navs, sbs_t *sbss);
```

#### **[Args]**

char *file	I	input log file	
gtime_t ts	I	time start	
gtime_t te	I	time end	
double tint	I	time interval	
obs_t *obss	IO	observation data	(NULL: no output)
nav_t *navs	IO	navigation messages	(NULL: no output)
sbs_t *sbss	IO	SBAS messages	(NULL: no output)

#### **[Return]**

None

#### **[Description]**

Read u-blox log file as obs data, nav message and sbas messages in time range/interval. Log file shall contain RXMRAW and RXMSFRB.

#### **[Notes]**

## A.86 Convert u-blox Log File

### **convubx()**

#### **[Prototype]**

```
#include "ublox.h"

void convubx(FILE *fp, FILE *ofp, FILE *nfp, FILE *sfp, gtime_t ts,
             gtime_t te, double tint, const char *file);
```

#### **[Args]**

FILE	*fp	I	input log file pointer
FILE	*ofp	I	output RINEX OBS file pointer (NULL:no output)
FILE	*nfp	I	output RINEX NAV file pointer (NULL:no output)
FILE	*sfp	I	output SBAS message file pointer (NULL:no output)
gtime_t	ts	I	time start
gtime_t	te	I	time end
double	tint	I	time interval
const char	*file	I	input file

#### **[Return]**

None

#### **[Description]**

Convert u-blox log file to RINEX OBS/NAV and SBAS message file. Log file shall contain RXMRAW and RXMSFRB.

#### **[Notes]**

## A.87 Decode NovAtel message

### **decodenov( )**

#### **[Prototype]**

```
#include "novatel.h"

int decodenov(const unsigned char *buff, int len, obs_t *obs, nav_t *nav,
              sbsmsg_t *sbsmsg, int *sat);
```

#### **[Args]**

unsigned char *	buff	I	message buffer
int	len	I	buffer length (bytes)
obs_t *	obs	O	observation data
nav_t *	nav	O	navigation message
sbsmsg_t *	sbsmsg	O	SBAS message
int	*sat	O	satellite PRN number for ephemeris or SBAS message

#### **[Return]**

Input status (0:no valid data, 1:obs data, 2:ephemeris, 3:SBAS message, 9:ion/utc parameters)

#### **[Description]**

Decode and input a NovAtel message. support RANGECPMB, RAWEPHB, IONUTCB and RAWWAASFRMB.

#### **[Notes]**

## A.88 Read NovAtel Log File

### **readnov()**

#### **[Prototype]**

```
#include "novatel.h"
```

```
int readnov(const char *file, obs_t *obss, nav_t *navs, sbs_t *sbss);
```

#### **[Args]**

char	*file	I	input log file	
obs_t	*obss	IO	observation data	(NULL: no output)
nav_t	*navs	IO	navigation messages	(NULL: no output)
sbs_t	*sbss	IO	SBAS messages	(NULL: no output)

#### **[Return]**

None

#### **[Description]**

Read NovAtel log file. Input log file shall contain RANGECPB, RAWEPHB, IONUTCB and RAWWAASFRMB.

#### **[Notes]**

## A.89 Read NovAtel Log File in Time Range/Interval

### **readnov()**

#### **[Prototype]**

```
#include "novatel.h"

int readnovt(const char *file, gtime_t ts, gtime_t te, double tint,
             obs_t *obss, nav_t *navs, sbs_t *sbss);
```

#### **[Args]**

char *file	I	input log file	
gtime_t ts	I	time start	
gtime_t te	I	time end	
double tint	I	time interval	
obs_t *obss	IO	observation data	(NULL: no output)
nav_t *navs	IO	navigation messages	(NULL: no output)
sbs_t *sbss	IO	SBAS messages	(NULL: no output)

#### **[Return]**

None

#### **[Description]**

Read NovAtel log file in time range/interval. Input log file shall contain RANGECMPB, RAWEPHB, IONUTCb and RAWWAASFRMB.

#### **[Notes]**

## A.90 Convert NovAtel Log File

### **convnov( )**

#### **[Prototype]**

```
#include "novatel.h"

void convnov(FILE *fp, FILE *ofp, FILE *nfp, FILE *sfp, gtime_t ts,
             gtime_t te, double tint, const char *file);
```

#### **[Args]**

FILE	*fp	I	input log file pointer
FILE	*ofp	I	output rinex obs file pointer (NULL:no output)
FILE	*nfp	I	output rinex nav file pointer (NULL:no output)
FILE	*sfp	I	output sbas message file pointer (NULL:no output)
gtime_t	ts	I	time start
gtime_t	te	I	time end
double	tint	I	time interval
const char	*file	I	input file

#### **[Return]**

None

#### **[Description]**

Convert NovAtel log file to RINEX OBS/NAV and SBAS message file. Log file shall contain RANGECPMB, RAWEPHB, IONUTCB and RAWWAASFRMB.

#### **[Notes]**

## A.91 Execute Command

### **execcmd()**

#### **[Prototype]**

```
#include "rtklib.h"
int execcmd(const char *cmd);
```

#### **[Args]**

char \*cmd            I    command line

#### **[Return]**

Execution status (0:ok,0>:error)

#### **[Description]**

Execute command line by operating system shell.

#### **[Notes]**

## A.92 Debug Trace Functions

**traceopen()**  
**traceclose()**  
**trace()**  
**tracet()**  
**tracemat()**  
**traceobs()**

### [Prototype]

```
#include "rtklib.h"

void traceopen(const char *file);
void traceclose(void);
void trace(const char *format, ...);
void tracet(const char *format, ...);
void tracemat(const double *A, int n, int m, int p, int q);
void traceobs(const obsd_t *obs, int n);
```

### [Args]

See rtkcmn.c

### [Return]

None

### [Description]

Output debug traces to file or standard output.

### [Notes]



## B Command Line Application Programs

Command	Description	Reference
<b>rnx2rtkp</b>	Baseline analysis by precise relative positioning	B.1
<b>pos2kml</b>	Convert positions to Google Earth KML file	B.2
<b>gennetcorr</b>	Generate network RTK corrections	B.3
<b>genclkcorr</b>	Generate satellite clock variation corrections	B.4
<b>convubx</b>	Convert u-blox log file to RINEX OBS/NAV, SBAS messages	B.5
<b>convnov</b>	Convert NovAtel log file to RINEX OBS/NAV, SBAS messages	B.6
<b>sbaspos</b>	Single point positioning with SBAS DGPS correction	B.7
<b>sbasdump</b>	Dump SBAS messages	B.8

## B.1 Baseline Analysis by Precise Relative Positioning

### **rnx2rtkp**

#### **SYNOPSIS**

```
rnx2rtkp [option ...] file file [...]
```

#### **DESCRIPTION**

Read RINEX OBS/NAV files, compute receiver (rover) positions and output position solutions. The first RINEX OBS file shall contain receiver (rover) observations. For the relative mode, the second RINEX OBS file shall contain reference (base) receiver observations. At least one RINEX NAV file shall be included in input files. Command options are as follows. ([]:default)

#### **OPTIONS**

```
-h          print help
-o output  output file [stdout]
-ts ds ts  start day/time (ds=y/m/d ts=h:m:s) [obs start time]
-te de te  end day/time   (de=y/m/d te=h:m:s) [obs end time]
-ti tint  time interval (sec) [all]
-p mode    mode (0:single,1:dgps,2:kinematic,3:static) [2]
-m mask    elevation mask angle (deg) [10]
-f freq    number of frequencies for relative mode (1:L1,2:L1+L2) [2]
-v thres   validation threshold for integer ambiguity (0.0:no AR) [3.0]
-b         backward solutions [off]
-c         forward/backward combined solutions [off]
-i         instantaneous integer ambiguity resolution [off]
-e         output x/y/z-ecef position [latitude/longitude/height]
-n         output NMEA-0183 GGA sentence [off]
-g         output latitude/longitude in the form of ddd mm ss.ss' [ddd.ddd]
-t         output time in the form of yyyy/mm/dd hh:mm:ss.ss [sssss.ss]
-u         output time in utc [gpst]
-d col     columns of time under decimal point [3]
-s sep     field separator [' ']
-r x y z   reference (base) receiver ecef pos (m) [average of single pos]
-cf file   correction file [no correction]
```

**EXAMPLES**

Example 1. Kinematic Positioning, L1+L2, output Latitude/Longitude/Height to STDOUT.

**command**

```
> rnx2rtkp 07590920.05o 30400920.05o 30400920.05n
```

**result**

```
% program   : rnx2rtkp ver.1.0
% inputs    : 07590920.05o 30400920.05o 30400920.05n
% obs start : 2005/04/02 00:00:00.0 GPST (gpsweek1316 518400.0s)
% obs end   : 2005/04/02 23:59:30.0 GPST (gpsweek1316 604770.0s)
% mode/obsv : kinematic/L1+L2
% elev mask : 10.0 deg
% ref pos   : 35.132062716 139.624305669 72.3338
%
% (time=GPST, lat/lon/hight=WGS84/ellipsoidal, Q=1:fix,2:float,4:dgps,5:single,
ns=# of sats)
% time      latitude(deg) longitude(deg) hight(m)  Q ns sdn(m) sde(m) sdu(m)
518400.000  35.160871612 139.613842087 66.8062 1 7 0.0072 0.0054 0.0164
518430.000  35.160871607 139.613842115 66.7987 1 7 0.0072 0.0054 0.0164
518460.000  35.160871593 139.613842110 66.7999 1 7 0.0072 0.0054 0.0163
518490.000  35.160871583 139.613842093 66.8118 1 7 0.0072 0.0053 0.0163
...
```

Example 2. Single Point Positioning, El Mask=15deg, output NMEA GGA to file out .pos

**command**

```
> rnx2rtkp -p 0 -m 15 -n -o out.pos 07590920.05o 30400920.05n
```

**result**

```
$GPGGA,235947.00,35 9.6524150,N,13936.8296671,E,1,07,,34.318,M,36.181,M,,,42
$GPGGA,000017.00,35 9.6525341,N,13936.8298278,E,1,07,,33.808,M,36.181,M,,,47
$GPGGA,000047.00,35 9.6524354,N,13936.8299014,E,1,07,,33.717,M,36.181,M,,,4F
$GPGGA,000117.00,35 9.6522549,N,13936.8298201,E,1,07,,34.418,M,36.181,M,,,4B
$GPGGA,000147.00,35 9.6522543,N,13936.8298643,E,1,07,,33.317,M,36.181,M,,,49
$GPGGA,000217.00,35 9.6523911,N,13936.8296580,E,1,07,,34.406,M,36.181,M,,,47
$GPGGA,000247.00,35 9.6524503,N,13936.8299040,E,1,07,,34.230,M,36.181,M,,,4F
$GPGGA,000317.00,35 9.6523647,N,13936.8300515,E,1,07,,33.780,M,36.181,M,,,42
...
```

Example 3. Static Positioning, L1, time form yyyy/mm/dd hh:mm:ss , output X/Y/Z-ECEF positions

**command**

```
> rnx2rtkp -p 3 -f 1 -t -e 07590920.05o 30400920.05o 30400920.05n
```

**result**

```
% program      : rnx2rtkp ver.1.0
% inputs       : 07590920.05o 30400920.05o 30400920.05n
% obs start    : 2005/04/02 00:00:00.0 GPST (gpsweek1316 518400.0s)
% obs end      : 2005/04/02 23:59:30.0 GPST (gpsweek1316 604770.0s)
% mode/obsv    : static/L1
% elev mask    : 10.0 deg
% ref pos      : -3978240.6491   3382839.2297   3649900.4598
%
% (time=GPST, x/y/z-ecef=WGS84, Q=1:fix,2:float,4:dgps,5:single, ns=# of sats)
% time          x-ecef(m)      y-ecef(m)      z-ecef(m)   Q  ns   sdx(m)
sdy(m)   sdz(m)
2005/04/02 00:00:00.000 -3976217.9351  3382371.4458  3652511.3843  2  7  1.2124
1.3748    1.0970
2005/04/02 00:00:30.000 -3976217.8724  3382370.5977  3652510.7455  1  7  0.0108
0.0120    0.0094
2005/04/02 00:01:00.000 -3976217.8733  3382370.5987  3652510.7454  1  7  0.0088
0.0098    0.0077
...
```

Example 4. Kinematic Positioning, Instantaneous AR, validation threshold=2, comma separator

**command**

```
> rnx2rtkp -i -v 2 -s , 07590920.05o 30400920.05o 30400920.05n
```

**result**

```
% program      : rnx2rtkp ver.1.0
% inputs       : 07590920.05o 30400920.05o 30400920.05n
% obs start    : 2005/04/02 00:00:00.0 GPST (gpsweek1316 518400.0s)
% obs end      : 2005/04/02 23:59:30.0 GPST (gpsweek1316 604770.0s)
% mode/obsv    : kinematic/L1+L2/instantaneous AR
% elev mask    : 10.0 deg
% ref pos      : 35.132062716, 139.624305669,   72.3338
%
% (time=GPST, lat/lon/hight=WGS84/ellipsoidal, Q=1:fix,2:float,4:dgps,5:single,
ns=# of sats)
% time          latitude(deg) longitude(deg) hight(m)   Q  ns   sdn(m)   sde(m)   sdu(m)
518400.000,   35.160871612, 139.613842087,   66.8062,   1,  7,   0.0072,   0.0054,
0.0164
518430.000,   35.160871607, 139.613842115,   66.7987,   1,  7,   0.0072,   0.0054,
0.0164
518460.000,   35.160871593, 139.613842110,   66.7999,   1,  7,   0.0072,   0.0054,
0.0163
518490.000,   35.160871583, 139.613842093,   66.8118,   1,  7,   0.0072,   0.0053,
0.0163
518520.000,   35.160871627, 139.613842143,   66.8086,   1,  7,   0.0072,   0.0053,
0.0163
...
```

## B.2 Convert Positions to Google Earth KML file

### **pos2kml**

#### **SYNOPSIS**

```
pos2kml [option ...] file [...]
```

#### **DESCRIPTION**

Read position file(s) and convert it to Google Earth KML file. Each line in the input file shall contain fields of time, position fields (Latitude/Longitude/Height or X/Y/Z-ECEF), and Quality flag (option). The line started with '%', '#', ';' is treated as comment. Command options are as follows. ([]:default)

#### **OPTIONS**

```
-h          print help
-o file     output file [infile + .kml]
-c color    track color
            (0:off,1:white,2:green,3:orange,4:red,5:yellow) [5]
-p color    point color
            (0:off,1:white,2:green,3:orange,4:red,5:by qflag) [5]
-a          output altitude information [off]
-ag         output geodetic altitude [off]
-tg         output time stamp of gpst [off]
-tu         output time stamp of utc [gpst]
-i tint     output time interval (s) (0:all) [0]
-q qflg     output q-flags (0:all) [0]
-f n e h    add north/east/height offset to position (m) [0 0 0]
-e          input x/y/z-ecef position [latitude/longitude/height]
-n          input NMEA-0183 GGA sentence [off]
-g          input latitude/longitude in the form of ddd mm ss.ss
            [ddd.ddd]
-s sep      field separator [' ']
```

## B.3 Generate Network RTK Correction

### **gennetcorr**

#### **SYNOPSIS**

`gennetcorr [option ...] file [...]`

#### **DESCRIPTION**

#### **OPTIONS**

## B.4 Generate Satellite Clock Variation Correction

### **genclcorr**

#### **SYNOPSIS**

```
genclcorr [-i tint][-p file][-c] obsfile [...] navfile
```

#### **DESCRIPTION**

Generate satellite clock variation correction.

#### **OPTIONS**

```
-i tint      Time interval (s)
-p file      station position file
-c
```

## B.5 Convert u-blox log file to RINEX OBS/NAV, SBAS messages

### **convubx**

#### **SYNOPSIS**

```
convubx [-ts d t] [-te d t] [-d dec] [-o ofile] [-n nfile] [-s sfile] file
```

#### **DESCRIPTION**

Read and convert u-blox ubx raw file to RINEX OBS/NAV and SBAS message file. u-blox ubx raw file shall contain RXM-RAW and RXM-SFRB messages.

#### **OPTIONS**

```
-ts d t      start day/time (d t=yyyy/mm/dd hh:mm:ss) [all]
-te d t      end day/time   (d t=yyyy/mm/dd hh:mm:ss) [all]
-d dec       obs data interval (s) [all]
-o ofile     output rinex obs file    [<infile>.obs]
-n nfile     output rinex nav file    [<infile>.nav]
-s sfile     output sbas message file [<infile>.sbs]
```



## B.6 Convert NovAtel log file to RINEX OBS/NAV, SBAS messages

### **convnov**

#### **SYNOPSIS**

```
convnov [-ts d t] [-te d t] [-d dec] [-o ofile] [-n nfile] [-s sfile] file
```

#### **DESCRIPTION**

Read and convert NovAtel raw file to RINEX OBS/NAV and SBAS message file. NovAtel raw file shall contain RANGECPMB, RAWEPHMB, and RAWWAASFRMB.

#### **OPTIONS**

```
-ts d t    start day/time (d t=yyyy/mm/dd hh:mm:ss) [all]
-te d t    end day/time   (d t=yyyy/mm/dd hh:mm:ss) [all]
-d dec     obs data interval (s) [all]
-o ofile   output rinex obs file      [<infile>.obs]
-n nfile   output rinex nav file      [<infile>.nav]
-s sfile   output sbas message file  [<infile>.sbs]
```

## B.7 Single point positioning with SBAS DGPS correction

### **sbaspos**

#### **SYNOPSIS**

sbaspos [option ...] file [...]

#### **DESCRIPTION**

Single point positioning with SBAS DGPS corrections. Files shall include receiver RINEX OBS file, NAV file and SBAS message log file (.sbs). Command options are as follows. ([]:default)

#### **OPTIONS**

-o output output file [stdout]  
-b SBAS satellite prn number [129]  
-m mask elevation mask angle (deg) [10]  
-c mask snr mask (dbHz) [0]  
-p single point positioing without SBAS DGPS corrections [off]  
-l apply SBAS long term corrections [all]  
-i apply SBAS ionospheric corrections [all]  
-f apply SBAS fast corrections [all]  
-r apply SBAS ranging [all]  
-s apply doppler smoothing [off]  
-t file output trace to file [off]

## B.8 Dump SBAS messages

### **sbasdump**

#### **SYNOPSIS**

sbasdump [option ...] file

#### **DESCRIPTION**

Dump sbas messages. specify sbas log as file. options are as follows

#### **OPTIONS**

-h	print help
-b	sbas satellite prn number [129]
-s	corrected satellite prn number [all]
-f	dump fast correction messages [off]
-i	dump ionospheric correction messages [off]
-l	dump long term correction messages [off]
-n	dump geo navigation message [off]
-g	dump ionospheric grid points [off]
-t	dump integrity messages [off]

## C GUI-based Application Programs

## C.1 RTKPOST

### **rtkpost**

#### **SYNOPSIS**

rtkpost

#### **DESCRIPTION**

Read and convert NovAtel raw file to RINEX OBS/NAV and SBAS message file. NovAtel raw file shall contain RANGECMPB, RAWEPHMB, and RAWWAASFRMB.

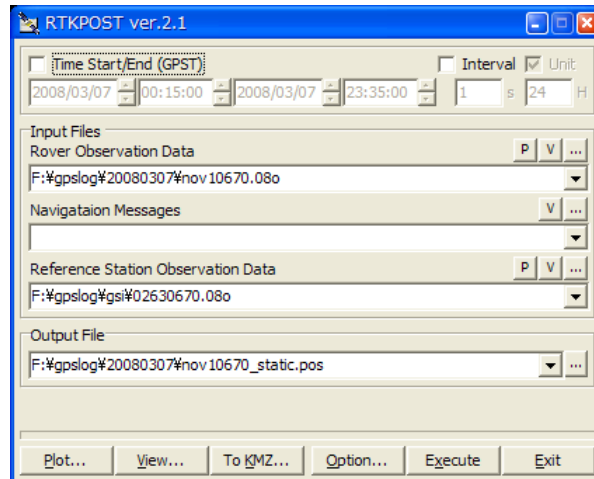
#### **OPTIONS**

none

## USER INTERFACE

### (1) Main Window

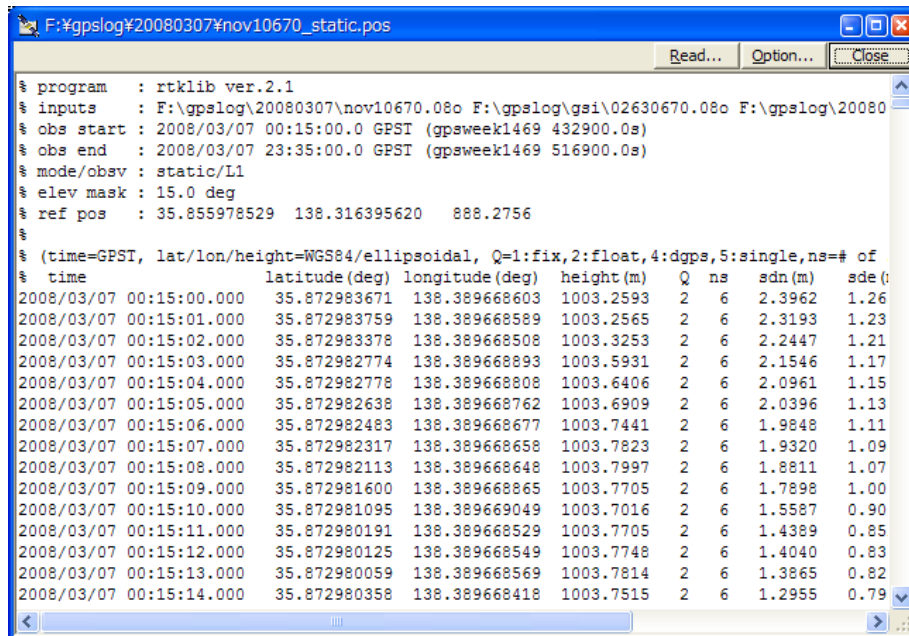
RTKPOST メイン画面。入力・出力ファイルを指定し基線解析処理を実行する。



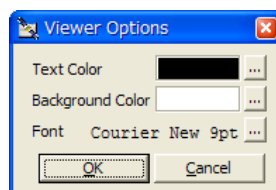
No	Item	Descriptions	Notes
1	Time Start/End	処理時刻範囲を指定する。チェックボックスを OFF にした場合観測データの全時刻を処理対象とする。	
2	Interval	処理時刻間隔を指定する。チェックボックスを OFF にした場合観測データの全時刻を処理対象とする。	
3	Rover Observation Data	ユーザ受信機観測データ RINEX OBS ファイルパスを指定する。直接入力するか、ボタン[...]を押して表示されるファイル選択ダイアログで指定する。ボタン[P]を押すことにより RTKPLOT で観測データを表示することが出来る。ボタン[V]を押すことによりテキストビューアで観測データ内容を表示することが出来る。	
4	Navigation Messages	ナビゲーションメッセージ RINEX NAV ファイルパスを指定する。空白とした場合、ユーザ受信機観測データファイルの末尾を以下の様に置き換えたファイルを指定したものとする。 *.yyo→*.yyn, *.obs→*.nav	
5	Reference Station Observataion Data	基準局受信機観測データ RINEX OBS ファイルパスを指定する。単独測位の場合無視される。	
6	Output File	出力測位解ファイルパスを指定する。	
7	Plot...	出力測位解ファイルを RTKPLOT によりプロットする。	
8	View...	出力測位解ファイルをテキストビューアで表示する。	
9	To KMZ...	Google Earth コンバータ画面を表示する。	
10	Option...	処理オプションダイアログを表示する。	
11	Execute/Abort	処理を開始する。また実行途中で中止する。	
12	Exit	プログラムを終了する。	

## (2) View Window

テキストビューア。テキストファイルの表示を行う。



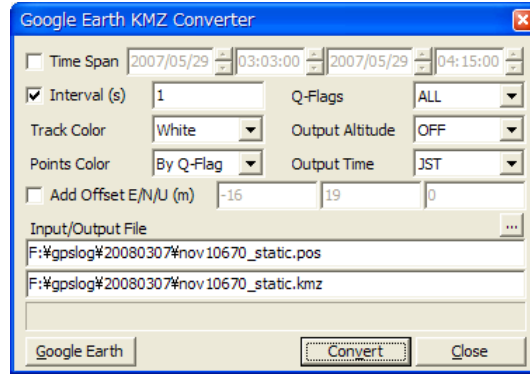
No	Item	Descriptions	Notes
1	Read...	ファイル選択ダイアログを表示し表示するテキストファイルを指定する。	
2	Option...	テキストビューアオプションダイアログを表示する。	
3	Close	テキストビューアを終了する。	
4	(テキスト)	指定したテキストファイルを表示する。	



No	Item	Descriptions	Notes
1	Text Color	テキスト表示色を指定する。[...]ボタンを押して表示されるカラー選択ダイアログでカラーを選択する。	
2	Background Color	テキスト背景色を指定する。[...]ボタンを押して表示されるカラー選択ダイアログでカラーを選択する。	
3	Font	テキスト表示フォントを指定する。[...]ボタンを押して表示されるフォント選択ダイアログでフォントを先駆する。	
4	OK	修正を有効としてダイアログを閉じる。	
5	Cancel	修正を無効としてダイアログを閉じる。	

## (3) Google Earth Converter Window

測位解を Google Earth KML/KMZ ファイルに変換する。

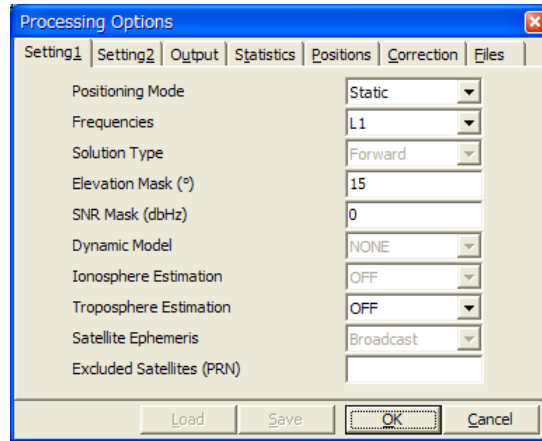


No	Item	Descriptions	Notes
1	Time Span	変換時間範囲を指定する。チェックボックスを OFF にした場合は変換元測位解の全時刻を指定したものとする。	
2	Interval (s)	変換時間間隔 (s) を指定する。チェックボックスを OFF にした場合は変換元測位解の全時刻を指定したものとする。	
3	Track Color	測位解軌跡のラインカラーを指定する。	
4	Points Color	測位解ポイントのカラーを指定する。	
5	Q-Flags	変換する測位解の品質(Q)フラグを指定する。ALL を指定した場合全測位解を変換する。	
6	Output Altitude	測位解の高度情報を出力に含めるか否かを指定する。	
7	Output Time	測位解の時刻情報を出力に含めるか否かを指定する。また出力する場合の時刻系 (UTC, JST) を指定する。	
8	Add Offset E/N/U	測位解にオフセットを加算して出力するか指定する。加算する場合、東西・南北・上下成分を m で指定する。	
9	Input File	変換元測位解ファイルのパスを指定する。	
10	Output File	変換後 Google Earth KML/KMZ ファイルパスを指定する。	
11	Google Earth	変換後 Google Earth KML/KMZ ファイルを引数にして Google Earth を起動する。	
12	Convert	Google Earth 変換を実行する。	
13	Close	Google Earth 変換画面を閉じる。	

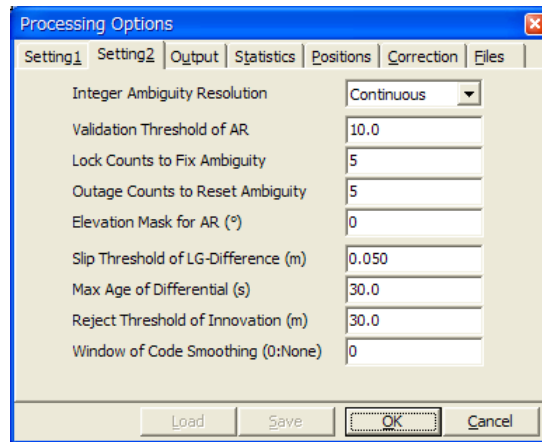


## (4) Processing Options

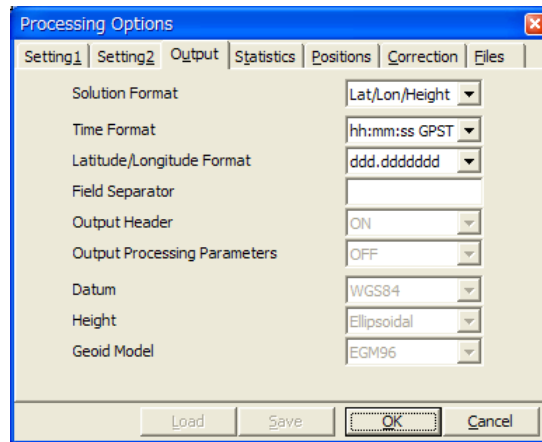
後処理基線解析のオプションを指定する。



No	Item	Descriptions	Notes
1	Positioning Mode	測位モードを指定する。 <ul style="list-style-type: none"> <li>• Single : 単独測位</li> <li>• DGPS : コード DGPS 測位</li> <li>• Static : スタティック測位</li> <li>• Kinematic : キネマティック測位</li> <li>• Moving-Base : 移動ベースライン</li> </ul>	
2	Frequencies	使用する観測データ周波数を指定する。 <ul style="list-style-type: none"> <li>• L1 : L1</li> <li>• L1+L2 : L1+L2</li> </ul>	
3	Solution Type	測位解種別を指定する。 <ul style="list-style-type: none"> <li>• Forward : フォワード解</li> <li>• Backward : バックワード解</li> <li>• Combined : フォワード+バックワードスムーズ解</li> </ul>	
4	Elevation Mask	仰角マスク(度)を指定する。	
5	SNR Mask	SNR マスク(dbHz)を指定する。	
6	Dynamic Model	運動モデルを指定する。 <ul style="list-style-type: none"> <li>• NONE : モデルなし</li> </ul>	
7	Ionosphere Estimation	電離層遅延パラメータの推定を行なうか否かを指定する。	
8	Troposphere Estimation	対流圏遅延パラメータの推定を行なうか否かを指定する。	
9	Satellite Ephemeris	衛星エフェメリスを指定する。 <ul style="list-style-type: none"> <li>• Broadcast : 放送暦</li> <li>• Precise : 精密暦</li> </ul>	
10	Exclude Satellites	除外する衛星の PRN 番号を指定する。衛星番号を空白で指定する。	



No	Item	Descriptions	Notes
1	Integer Ambiguity Resolution	整数バイアス決定手法を指定する。 ・ OFF : 整数バイアスを解決しない ・ Continuous : 時間連続のバイアスを一定値と扱う ・ Instantaneous : 瞬時整数バイアス決定を行う	
2	Validation Threshold of AR	整数バイアス検定 (ratio-test) のスレッシュホールド値を指定する。	
3	Lock Counts to Fix Ambiguity	整数バイアス決定の最低ロックエポック数を指定する。	
4	Outage Counts to Reset Ambiguity	データ欠損が指定エポック数連続した場合、整数バイアスをリセットする。	
5	Elevation Mask for AR	整数バイアス決定における仰角マスク (度) を指定する。	
6	Slip Threshold of LG-Difference	指定(m)以上 LG 線形結合飛びが認められた場合サイクルスリップとして扱う。	
7	Max Age of Differential	ローバ/基準局の時刻差分の最大値 (s) を指定する。	
8	Reject Threshold of Innovation	指定(m)以上のイノベーションは異常データとして削除する。	
9	Window of Code Smoothing	コードキャリアスムージングウィンドウを指定する。0を指定した場合スムージングを行わない。	

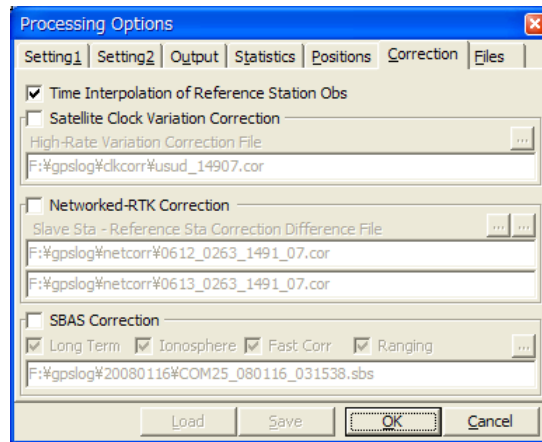


No	Item	Descriptions	Notes
1	Solution Format	測位解出力形式を指定する。 ・ Lat/Lon/Height : 緯度・経度・高度 ・ ECEF-X/Y/Z : ECEF 座標 X/Y/Z ・ NMEA0183 : NMEA RMC/GGA	
2	Time Format	測位解時刻形式を指定する。 ・ ssssssss.sss GPST : GPS 時刻 (TOW) ・ hh:mm:ss GPST : GPS 時刻 (年月日時分秒) ・ hh:mm:ss UTC : UTC 時刻 (年月日時分秒)	
3	Latitude/Longitude Format	測位解緯度経度形式を指定する。 ・ ddd.ddd : 度 ・ ddd mm ss.sss : 度分秒	
4	Field Separator	測位解のフィールドセパレータを指定する。	
5	Output Header	測位解ヘッダ出力を行なうか否かを指定する。 ・ ON	*
6	Output Processing Parameters	測位解ヘッダに処理パラメータを出力するか否かを指定する。 ・ OFF	*
7	Datum	測位解測地系を指定する。 ・ WGS84 : WGS84 測地系 ・ Tokyo : Tokyo 測地系	*
8	Height	測位解高度形式を指定する。 ・ Ellipsoidal : 楕円体高 ・ Geodetic : 測地高度	*
9	Geoid Model	測地高度を出力する場合に使用するジオイドモデルを指定する。 ・ EGM96 : EGM96 モデル	*

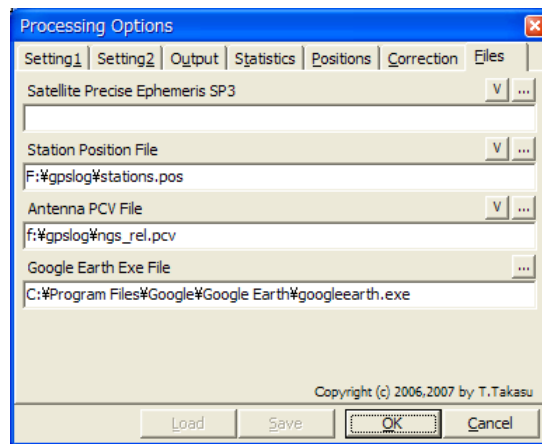
\* ver.2.1 では固定。変更は出来ない。

No	Item	Descriptions	Notes
1	Measurement Errors (1-sigma)	観測誤差を指定する。	
2	Code/Carrier-Phase Error Rate	コード観測値と搬送波位相観測値の観測誤差の大きさの比を指定する。	
3	Carrier-Phase Error	搬送波位相観測値観測誤差標準偏差(m)を指定する。	
4	Carrier-Phase Error/sinEl	搬送波位相観測値観測誤差の仰角依存項(m)を指定する。	
5	Carrier-Phase Error/Baseline	搬送波位相観測値観測誤差の基線長依存項(m/km)を指定する。	
6	Process Noises (1-sigma/sqrt(s))	プロセスノイズの値を指定する。	
7	Carrier-Phase Bias	搬送波位相バイアスのプロセスノイズ(cycle/sqrt(s))を指定する。	
8	Vertical Ionospheric Delay	基線長 10kmあたりの垂直電離層遅延のプロセスノイズ(m/sqrt(s))を指定する。	
9	Zenith Tropospheric Delay	天頂対流圏遅延のプロセスノイズ(m/sqrt(s))を指定する。	
10	Satellite Clock Stability	衛星時計安定度(s/s)を指定する。	

No	Item	Descriptions	Notes
1	Lat/Lon/Height	位置を緯度経度高度で指定する。	
2	X/Y/Z-ECEF	位置を ECEF 座標 X/Y/Z 成分で指定する。	
3	Rover	ローバの位置・アンテナパラメータを指定する。	
4	Fixed Position	ローバ位置を固定する。	
5	Get From Pos File	ローバ位置を位置ファイルから取得する。	
6	Antenna Type	ローバアンテナ型名を指定する。	
7	Delta-E/N/U	ローバアンテナ位置の東西・南北・上下オフセット(m)を指定する。	
8	Reference Station	基準局の位置・アンテナパラメータを指定する。	
9	Fixed Position	基準局位置を固定する。	
10	Get From Pos File	基準局位置を位置ファイルから取得する。	
11	Antenna Type	基準局アンテナ型名を指定する。	
12	Delta-E/N/U	基準局アンテナ位置の東西・南北・上下オフセット(m)を指定する。	



No	Item	Descriptions	Notes
1	Time Interpolation of Reference Station Obs	基準局観測データの時間補間を行なうか否か指定する。	
2	Satellite Clock Variation Correction	基準局観測データの時間補間において衛星時計変動補正を行なう。	
3	High-Rate Variation Correction File	衛星時計変動補正に使用する衛星時計変動補正ファイルを指定する。 <b>genclkcorr</b> コマンドで生成したファイルを指定する。	
4	Network-RTK Correction	ネットワーク RTK 型補正を行なうか否か指定する。	
5	Slave-Reference Sta Correction Difference File	ネットワーク型 RTK 補正に使用する基準局間補正值差ファイルを指定する。 <b>gennetcorr</b> コマンドで生成したファイルを最大 2 個指定する。	
6	SBAS Correction	SBAS DGPS 補正を行なうか否か指定する。	
7	Long Term	SBAS 長期衛星誤差補正を行なうか否か指定する。	
8	Ionosphere	SBAS 電離層遅延補正を行なうか否か指定する。	
9	Fast Corr	SBAS 高速補正を行なうか否か指定する。	
10	Ranging	SBAS 測距信号を利用するか否か指定する。	
11	(SBAS Message File)	SBAS メッセージファイルを指定する。	



No	Item	Descriptions	Notes
1	Satellite Precise Ephemeris SP3	衛星精密暦ファイルを指定する。	
2	Station Position File	局位置ファイルを指定する。	
3	Antenna PCV File	アンテナ位相中心変動補正パラメータファイルを指定する。	
4	Google Earth Exe File	Google Earth の実行プログラムを指定する。	

## C.2 RTKPLOT

### **rtkplot**

#### **SYNOPSIS**

rtkplot

#### **DESCRIPTION**

Read and convert NovAtel raw file to RINEX OBS/NAV and SBAS message file. NovAtel raw file shall contain RANGECMPB, RAWEPHMB, and RAWWAASFRMB.

#### **OPTIONS**

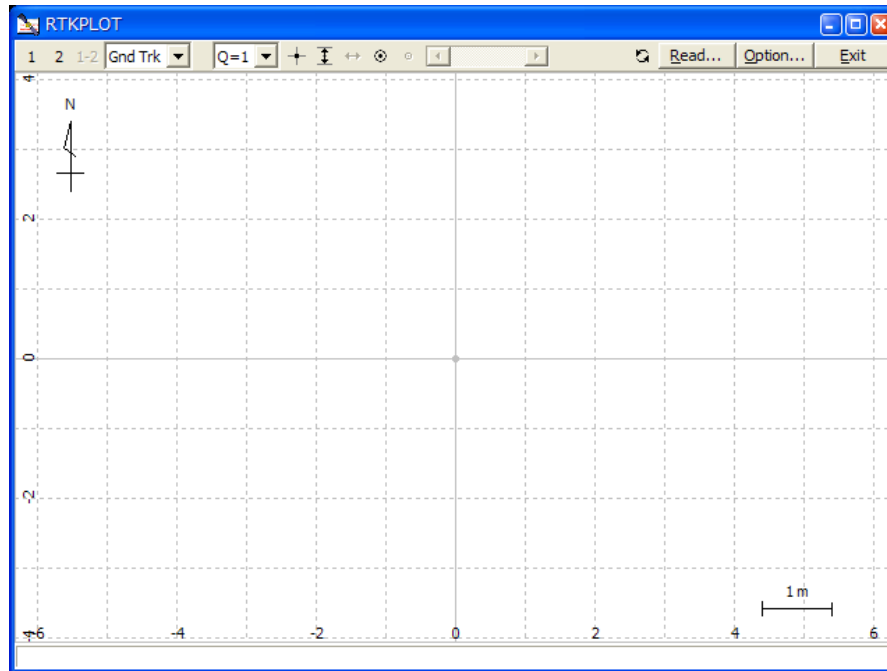
none



## USER INTERFACE

### (1) Main Window

観測データまたは測位解を読み込みその内容をプロットする。

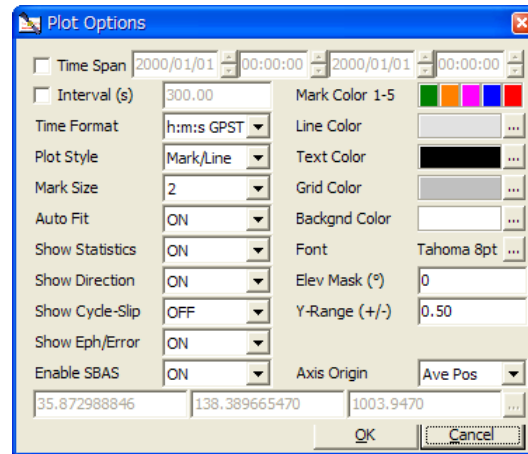


No	Item	Descriptions	Notes
1	1	測位解表示の場合、データ No1 の表示を切り替える。 観測データ表示の場合、データ表示を切り替える。	
2	2	測位解表示の場合、データ No2 の表示を切り替える。	
3	1-2	測位解表示の場合、データ No1 とデータ No2 の差分表示を切り替える。	
4	(PlotType)	表示種別の切り替えを行う。 観測データの場合 <ul style="list-style-type: none"> <li>• Raw Obs : 生観測データ取得状況</li> <li>• Sky Plot : スカイプロット</li> <li>• NSAT/DOP : 衛星数および DOP</li> </ul> 測位解の場合 <ul style="list-style-type: none"> <li>• GndTrack : 地上軌跡</li> <li>• Position : 位置 ENU 成分</li> <li>• Velocity : 速度 ENU 成分</li> <li>• Accel : 加速度 ENU 成分</li> </ul>	

No	Item	Descriptions	Notes
5	(ContentType)	表示内容の切り替えを行なう。 観測データの場合 ・ ALL : 全て ・ L1/L2/P1/P2 : L1/L2/P1/P2 測位解の場合 ・ ALL : 全て ・ Q=1,2,... : 品質フラグ	
6	(Origin Center)	原点を中心位置に移動する。	
7	(Fit XY)	データに合わせて縦軸範囲を調整する。	
8	(Fit Time)	データに合わせて時間軸範囲を調整する。	
9	(Track Point)	現在時刻位置を追跡する。	
10	(Track Center)	現在時刻位置を中心に移動する。	
11	(Track Time)	現在時刻位置を移動する。	
12	(Reload)	ファイルを再度読み直す。	
13	Read...	ファイル選択ダイアログを表示し、ファイルを読み込む。	
14	Option...	プロットオプションダイアログを表示し、プロット設定を行なう。	
15	Exit	プログラムを終了する。	
16	(Plot Area)	観測データまたは測位解をプロットする。 カーソルをエリアに入れた状態で、 左ボタンを押してドラッグすることにより中心位置を移動することが出来る。 また右ボタンを押しながら上下・左右にドラッグすることにより拡大・縮小することが出来る。	

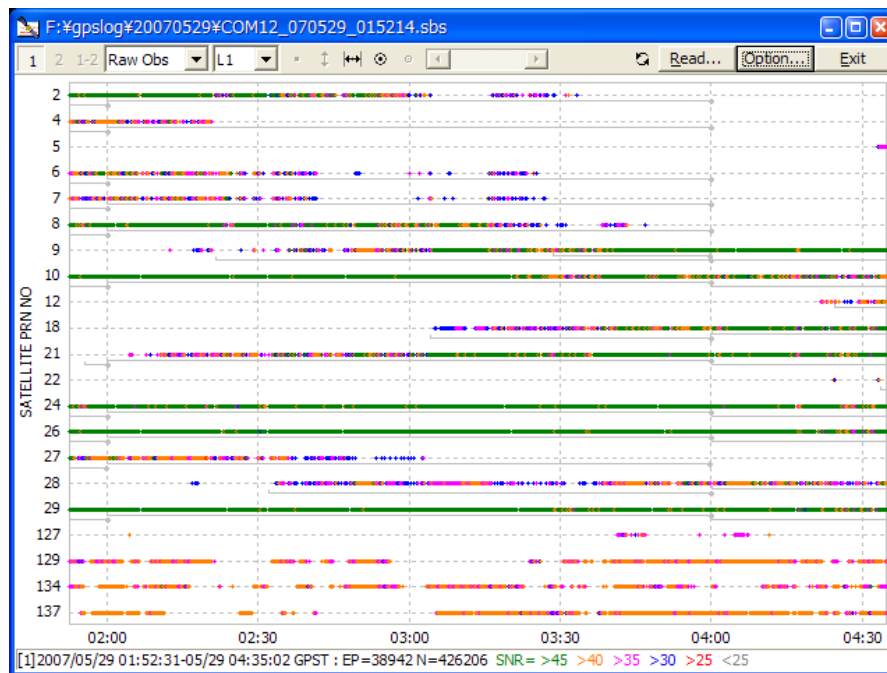
## (2) Plot Options Dialog

プロットオプションを設定する。

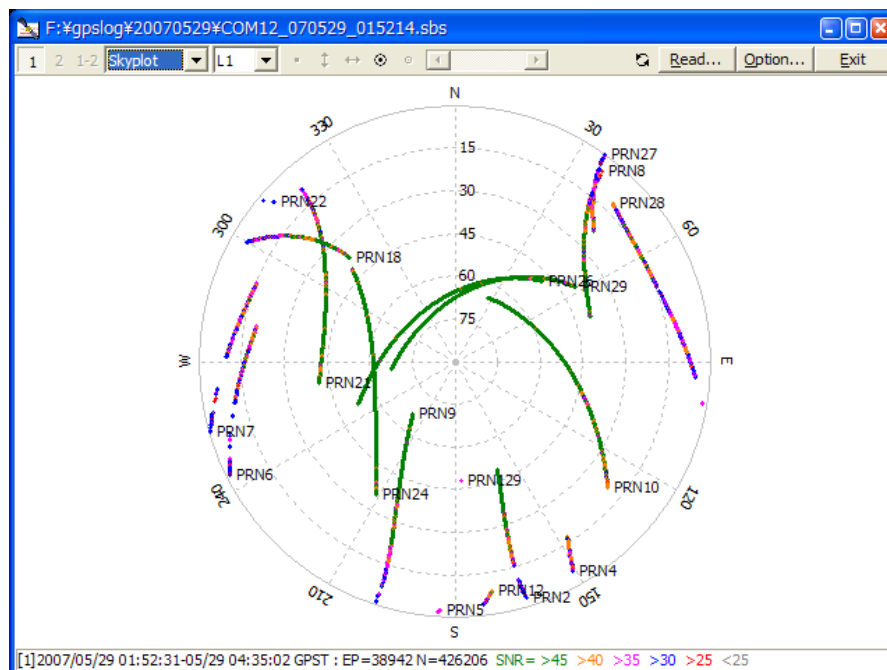


No	Item	Descriptions	Notes
1	Time Span	プロット時刻範囲を指定する。	
2	Interval	プロット時刻間隔 (s) を指定する。	
3	Time Format	時刻形式を指定する。	
4	Plot Style	プロットスタイルを指定する。	
5	Mark Size	マーカーサイズを指定する。	
6	Auto Fit	縮尺自動調整を行なうか否かを指定する。	
7	Show Statistics	統計情報を表示するか否かを指定する。	
8	Show Direction	測位解軌跡表示時に方向矢印を表示するか否かを指定する。	
9	Show Cycle-Slip	観測データ表示時にサイクルスリップ位置を表示するか否かを指定する。	
10	Show Eph/Error	測位解表示時にエラーバーを表示するか否かを指定する。 観測データ表示時にエフェメリスステータスを表示するか否かを指定する。	
11	Enable SBAS	観測データ表示時に SBAS 衛星を有効とするか否かを指定する。	
12	Mark Color 1-5	マーカーカラーを指定する。	
13	Line Color	ラインカラーを指定する。	
14	Text Color	テキストカラーを指定する。	
15	Grid Color	グリッドカラーを指定する。	
16	Backgnd Color	背景カラーを指定する。	
17	Font	フォントを指定する。	
18	Elev Mask	観測データ表示時の仰角マスク (度) を指定する。	
19	Y-Range	縦軸範囲を指定する。	
20	Axis Origin	測位解表示時の原点位置を指定する。	
21	OK	設定変更を有効としてダイアログを閉じる。	
22	Cancel	設定変更を無効としてダイアログを閉じる。	

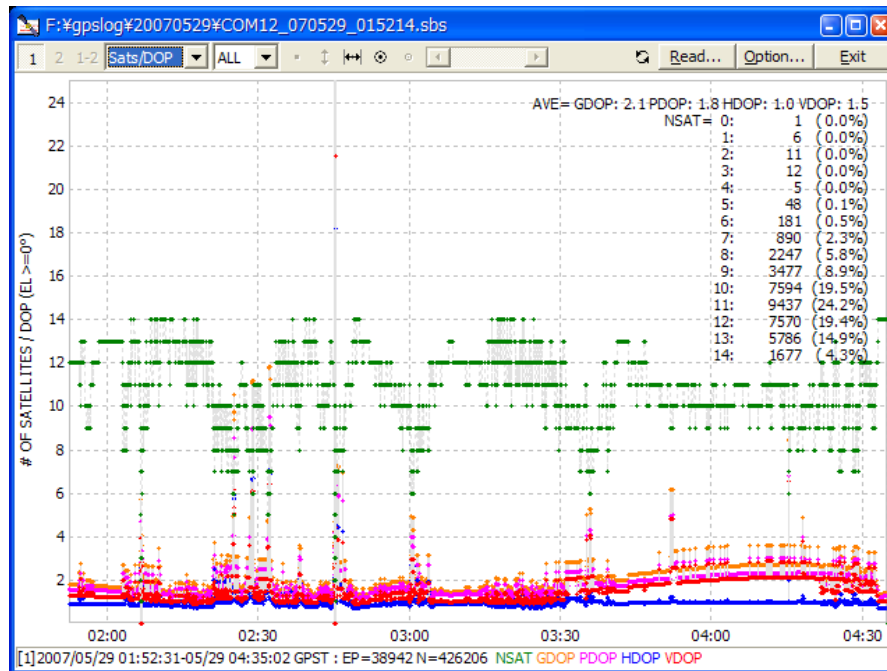
# EXAMPLES



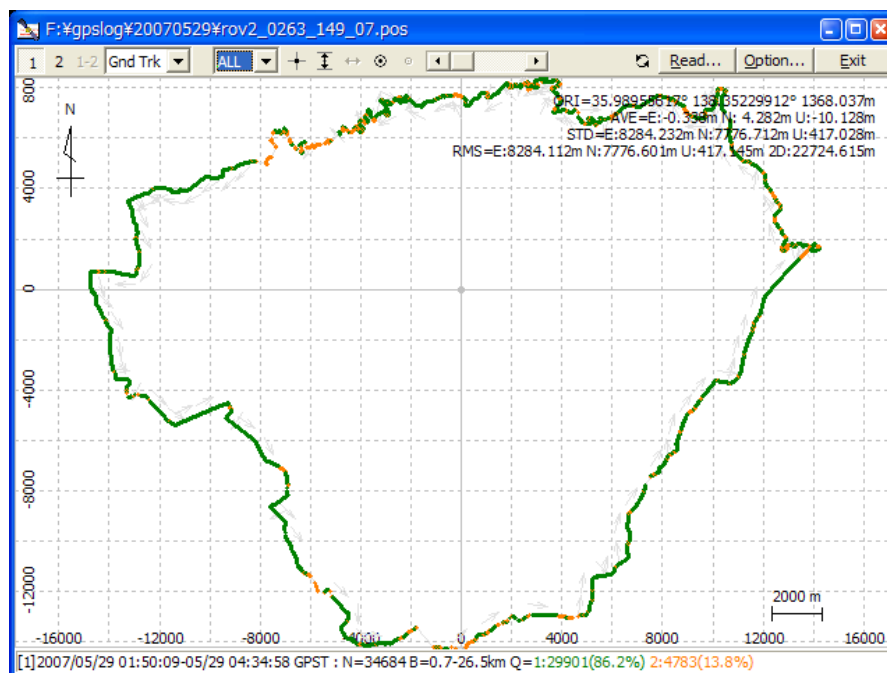
生観測データ取得状況プロット



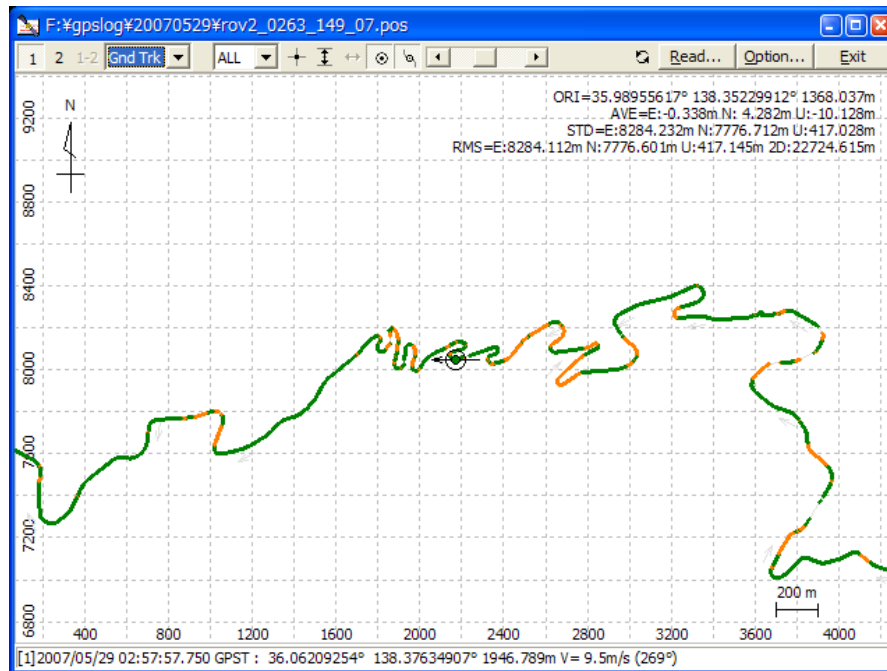
観測データスカイプロット



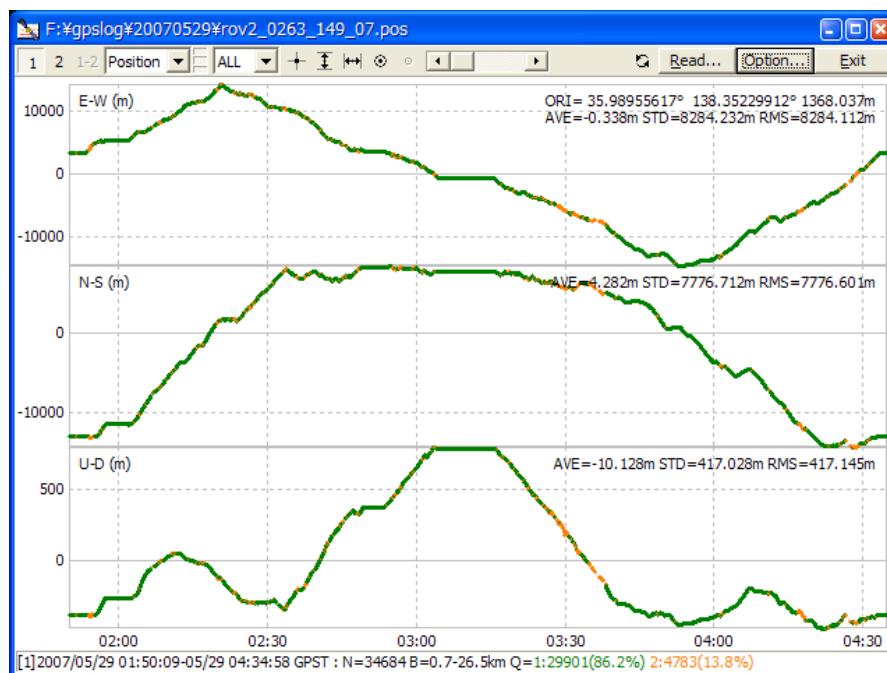
観測データ衛星数/DOPプロット



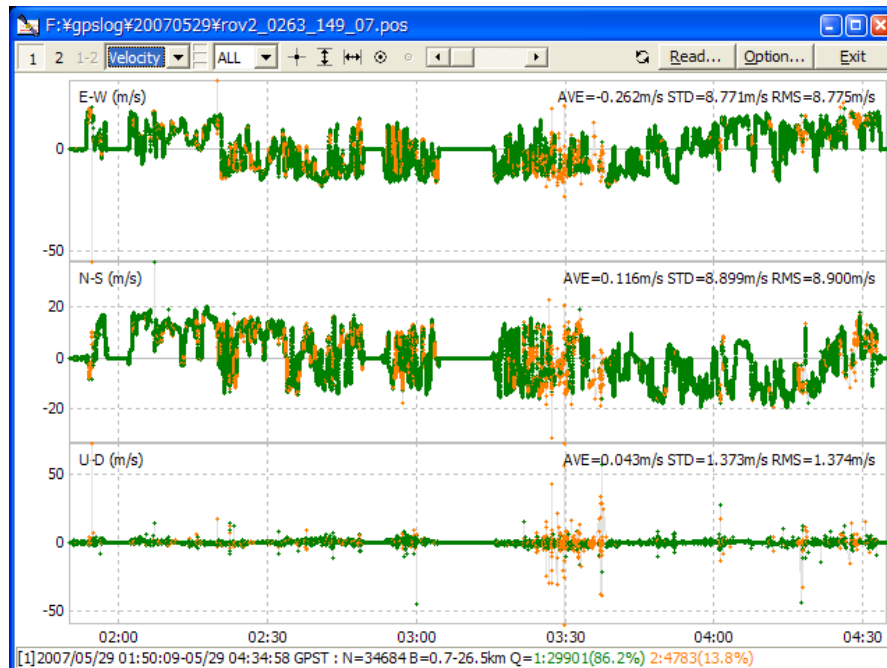
測位解軌跡プロット



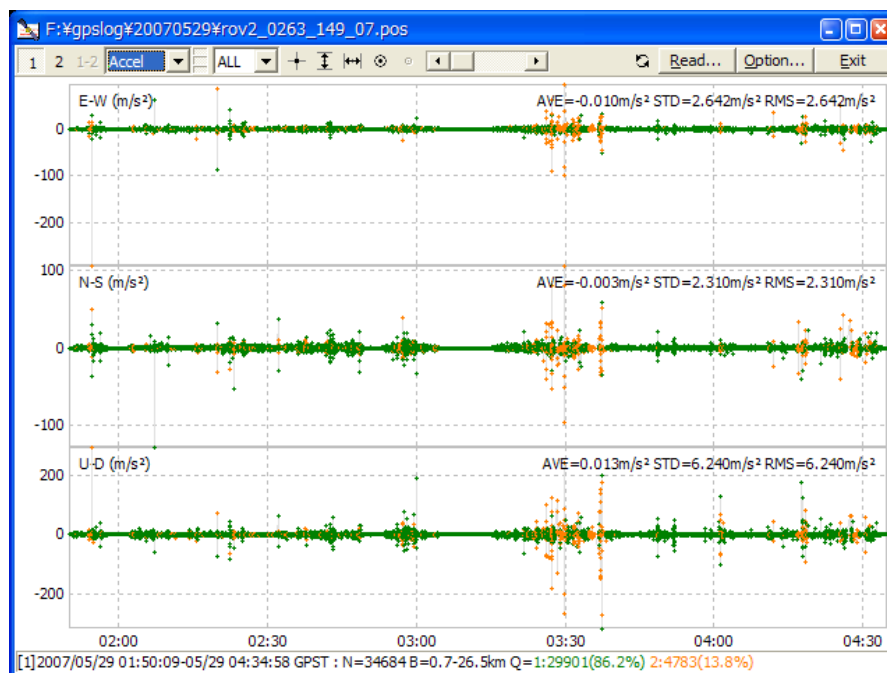
測位解軌跡プロット（現在位置中心表示）



測位解位置プロット



測位解速度プロット



測位解加速度プロット

## C.3 RTKCONV

### **rtkconv**

#### **SYNOPSIS**

rtkconv

#### **DESCRIPTION**

Read and convert NovAtel raw file to RINEX OBS/NAV and SBAS message file. NovAtel raw file shall contain RANGECMPB, RAWEPHMB, and RAWWAASFRMB.

#### **OPTIONS**

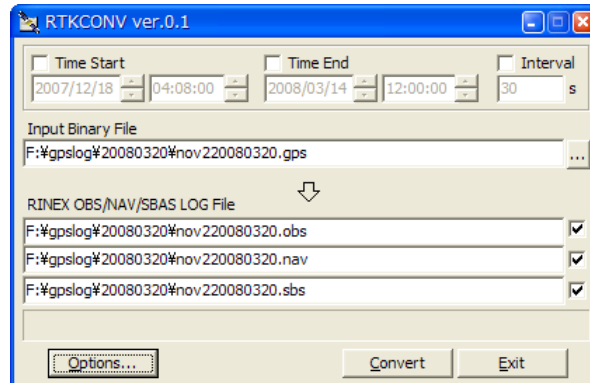
none



## USER INTERFACE

### (1) Main Window

受信機ログデータを指定し RINEX OBS/NAV、SBAS メッセージファイルに変換する。



No	Item	Descriptions	Notes
1	Time Start	変換開始時刻を指定する。	
2	Time End	変換終了時刻を指定する。	
3	Interval	変換時刻間隔 (s) を指定する。	
4	Input Binary File	変換元受信機ログファイルパスを指定する。 <ul style="list-style-type: none"> <li>• *.ubx : u-blox ログファイル</li> <li>• *.gps : NovAtel ログファイル</li> </ul>	
5	RINEX OBS File	出力先 RINEX OBS ファイルパスを指定する。 チェックボックスを OFF にした場合出力されない。	
6	RINEX NAV File	出力先 RINEX NAV ファイルパスを指定する。 チェックボックスを OFF にした場合出力されない。	
7	SBAS LOG File	出力先 SBAS メッセージファイルパスを指定する。 チェックボックスを OFF にした場合出力されない。	
8	(Message Area)	変換状況ステータスを表示する。	
9	Options	変換オプションを指定する。	*
10	Convert/Abort	受信機ログデータの RINEX OBS/NAV、SBAS メッセージファイル変換を実行する。また途中で変換を中止する。	
11	Exit	プログラムを終了する。	

\* RTKLIB ver.2.1 では無効