Navigation Laboratories Inc.
GPS Constellation Simulators

The Tapestry Binary Data Base

The Tapestry System is a software suite developed by Navigation Laboratories Inc. that provides a modeling and control gateway for the LabPro SCS3500/3510 GPS Constellation Simulators.

To reduce scenario file size and control the content and format, essential simulation truth data is stored in binary files within the scenario folder. To access this data, we have provided tools within UTILITIES on the Function Bar.

Should you need to access the direct binary file; the following paragraphs describe those of interest to the general user. *Note the header files for these structures are in the* **c:\Tapestry\Tools folder**.

➡ A word about data packing.

Byte Alignment

Data alignment is compiler specific. All binaries files within the Tapestry System use **byte alignment**. To insure this, use the following PRAGMA pair.

```
→ # pragma pack(push,1)

          Typedef  tag {


          } name;

→ #pragma pack(pop)
```

Trajectory Truth File:

NAME: Trajectory1.scn (Vehicle 1) Trajectory2.scn (Vehicle 2)

```
typedef struct  NVDATA {
            long int Week;              // GPS week number
            double  Time;               // seconds into GPS week, resolution 0.01 seconds
            double  EcefPos[3];         // meters
            double  EcefVel[3];         // m/s
            double  EcefAcc[3];         // m/s/s
            double  EcefJerk[3];        // m/s/s/s
            double  Attitude[3];        // radians
            double  AngRate[3];         // r/s
            double  AngAccel[3];        // r/s/s
            double  AngJerk[3];         // r/s/s/s
};
```

ORGANIZATION: Successive [Time Tagged] records of type NVDATA

ΔT = 0.1 second or 0.01 second

struct NVDATA @ Simulation Start Time ← **must be ≥ 1 second**
struct NVDATA @ Simulation Start Time + ΔT

↓

struct NVDATA @ Simulation End Time

SV State Vector Truth File:

NAME: SvTruth1.dat (Vehicle 1) SvTruth2.dat (Vehicle 2)

```
struct SvXRec {
            short int          Svid;
            double             ECEFPos[3];          // Sat State Vector xyz @ time of Reception, corrected for Earth Rotation
            double             ECEFVel[3];          // xyz @ time of Reception, corrected for Earth Rotation
            double             ECEFAcc[3];          // xyz @ time of Reception, corrected for Earth Rotation
};
struct Header {
            int                GPSWeek;             // Simulation Start Week
            double             SimulationStartTime; // Start Time Sec into Week
            int                NumberOfChannels;    // Number of output channels
            int                OutputRate;          // 1Hz, 10Hz
};
```

ORGANIZATION: [ΔT = 1.0/OutputRate – Time tag is implied based upon header record]

Header Record (*First record in file, output only once – provides key to constructing time tags*)
struct SvXRec – SV#1 @ Time = Simulation Start Time
struct vXRec – SV#2 @ Time = Simulation Start Time

↓

struct SvXRec – SV#NumberOfChannels @ Time = Simulation Start Time

struct SvXRec – SV#1 @ Time = Simulation Start Time + ΔT
struct SvXRec – SV#2 @ Time = Simulation Start Time + ΔT

… ↓

struct SvXRec – SV#NumberOfChannels @ Time = Simulation Start Time + ΔT

records @ 2ΔT
records @ 3ΔT

↓

Range Truth File:

NAME: RangeTruth1.dat (Vehicle 1) RangeTruth2.dat (Vehicle 2)

```
struct SvDumpRec {
            short int   Svid;
            char                Status;                 // Unused
            double              PRange;                 // Corrupted Pseudorange (subtract Errors) [Time of Reception]
            double              PRangeRate;             // m/s corrupted
            float               PRangeAcc;              // m/s/s corrupted
            float               PRangeJerk;             // m/s/s/s corrupted
            double              Clock;                  // (m) lumped clock errors from SV –subtract off
            float               Elevation;              // Degrees
            float               Azimuth; // Degrees
            float               L1CAAtten;              // scale factor 0.1
            float               L1PAtten;               // scale factor 0.1
            float               L2PAtten;               // scale factor 0.1
            float               L2CAtten;               // scale factor 0.1
            float               L5Atten;  // scale factor 0.1
            float               SvSlantRangeAttenuation;     // Satellite-to-Vehicle Slant range effect
            float               SvPatternAttenuation;        // Atten due to GPS Beam Pattern
            float               AntMaskAttenuationL1;        // Atten due to Vehicle Antenna Mask
            float               Tropo;                  // Tropo (m) – subtract off
            float               L1Iono;                 // Ionosphere (m) – subtract off
            float               L2Iono;                 // Iono (m) – subtract off
            float               L5Iono;                 // Iono (m) – subtract off
            float               Sa;                     // Archiac Model error from RTCA DO-217
            float               Uere;                   // User Equiv Range Error. (m) – subtract off
            float               IntRamp; // Integrated ramp error(m)  – subtract off

            float               AntMaskAttenuationL2;// Vehicle Antenna Mask pattern attenuation
            float               AntMaskAttenuationL5;// Vehicle Antenna Mask pattern attenuation
            float               SwMpRange;              // SW generated multipath range error
            float               Tgd;                    // Group delay differentialSF1 - used by IEC
};

struct Header {
            int                 GPSWeek;                     // Simulation Start Week
            double              SimulationStartTime;         // Start Time Sec into Week
            int                 NumberOfChannels;            // Number of output channels
            int                 OutputRate;                  // 1Hz, 10Hz
};
```

ORGANIZATION: [ΔT = 1.0/OutputRate – Time tag is implied based upon header record]

Header Record (*First record in file, output only once – provides key to constructing time tags*)
struct SvDumpRec – SV#1 @ Time = Simulation Start Time
struct SvDumpRec – SV#2 @ Time = Simulation Start Time
↓
struct SvDumpRec – SV#NumberOfChannels @ Time = Simulation Start Time

struct SvDumpRec – SV#1 @ Time = Simulation Start Time + ΔT
struct SvDumpRec – SV#2 @ Time = Simulation Start Time + ΔT
… ↓
struct SvDumpRec – SV#NumberOfChannels @ Time = Simulation Start Time + ΔT

records @ 2ΔT
records @ 3ΔT
↓