# LRT Data Sharing Principles

The purpose of this document is to define a set of rules for an easy exchange of measurement data sets in the context of train-side positioning systems. The motivation behind that is to make it easier to evaluate and compare different positioning approaches.

**Table of Content**

**CONTACT**

info@lrt-initiative.org

**VERSION HISTORY**

| Date | Version | Change |
|---|---|---|
| 25.10.2019 | 0.1 | first notes |
| 09.12.2019 | 0.2 | initial draft |
| 16.12.2019 | 0.3.1 | feedback of TU BS incorporated |
| 19.12.2019 | 0.3.2 | feedback of TU BS incorporated |
| 28.01.2020 | 0.4.1 | - changed coordinate systems figure<br>- changed dataset structure figure |
| 31.01.2020 | 0.4.2 | - added format definitions for odometers and tachometers |
| 19.02.2020 | 0.4.3 | - added version history (modified version counting)<br>- introduced new dataset structure and description<br>- updated dataset structure figure<br>- added hint in section 1.2.4 how to deal with lever arms<br>- added lever-arm field to all format definitions |
| 26.02.2020 | 0.4.4 | - incorporated feedback of M. Roth from 2020-02-25 |
| 06.03.2020 | 0.4.5 | - incorporated feedback of LRT Telco from 2020-02-26<br>- added 1σ error-ellipse parameters for GNSS and reference data<br>- added extra description for parameters like, e.g., lever arms<br>- incorporated feedback of iVA from 2020-03-04<br>- incorporated feedback of L. Priebe from 2020-03-04<br>- dataset structure: using ‘session’ instead of ‘drive’ |
| 07.03.2020 | 1.0 | - published this document at lrt-initiative.org as working document |
| 26.03.2020 | 1.1 | - changed general descriptions for map format-definitions<br>- changed dataset structure for maps |

# 1 General Principles

## 1.1 Aspiration

1) It is tried to provide common measured quantities in a **standardized format**. Thereby, it is made possible to easily test and compare different train-side positioning approaches with different input data.
2) The provided data should be as **comprehensible** as possible. Therefore, all raw data, additional documentary material and if possible, examples demonstrating the usage of the data are provided.
3) **Keep it simple and small**. It is not aimed to provide a general solution which is applicable to any kind of test campaign. If it is in the sense of the previous two principles, it may be deviated from the rules presented in the context of this document.

## 1.2 Data Structure

### *1.2.1 Parameters*

Parameters describe static data which do not change throughout a measurement campaign.

- At least the lever-arm and the mounting position (attitude) of each sensor should be provided.
- If available, other parameters like, e.g., camera calibration data should be provided too.

### *1.2.2 Maps*

All types of maps (geometric, topologic, features, etc.) that may be available (see also section 3.3).

- General requirements for map data are still under discussion.

### *1.2.3 Raw Data*

On this level the pure sensor raw data is stored without any changes.

- It is desired to use existing standards (like e.g. RINEX in the case of GNSS data) to store the data. If no standard is available, the data can also be stored in a sensor specific output format as long as it is explained how to access the data.

### *1.2.4 Processed Data*

On this level harmonized sensor data is provided which is ready to use in different algorithms. It is called “processed data” because the data is normally generated by processing the raw data. Harmonized means:

- The data is synchronized
    - to the same time standard,
    - the same start and end times,
    - the same units,
    - and the same coordinate frames*.

      * That means that all sensor frames are rotated into a defined track, vehicle or navigation frame (see also section 2). The translational offset of each

sensor from the origin of its corresponding reference frame, i.e. the lever arm, is specified in an additional data field for each sensor (see also section 3). But the lever arm is not directly compensated in the processed data, this is up to the user.

- The data is labeled according to a minimum set of labels for each sensor class. If necessary, it is allowed to introduce additional data labels for each sensor.

#### *1.2.5 Reference Data*

On this level all kind of available reference data is provided, e.g. precise external measurements and sensor fusion results (online as well as post-processing results) etc.

- The reference data is harmonized to the processed data (if applicable). This possibly concerns the time standard, start and end times, units, coordinate frames and data labels being used.
- Reference data characterizing the performance of train-side positioning systems is labeled according to a minimum set of data labels to ensure a quick comparability of different positioning approaches on the same data.
- Unless otherwise stated, reference positioning solutions refer to the origin of the vehicle frame (see also section 2.2.2).

## 1.3 Documentation and Examples

All data is complemented with documentation material to make all data as comprehensible as possible. The documentation should include:

- A brief description of the test campaign covering at least some details about:
  - the overall context of the campaign (motivation, date, etc.),
  - the test vehicle,
  - the test track,
  - and the measurement set-up (list of sensors and set-up that was used to record the data).
- A detailed description for each sensor that has been used. This could be realized by:
  - datasheets,
  - pictures,
  - and drawings (that document the mounting position)
- Programs and scripts that are necessary to:
  - make the raw data accessible,
  - shows how the data have been processed,
  - and show how to work with the data (examples).
  - If unknown sensor-internal calculation-results are used in the data, it should be documented in a proper way.
- Possibly, additional descriptions for the processed and reference data that e.g.:
  - explain which coordinate systems are valid for which data,
  - any other specialties concerning the data.

# 2 Reference Frames

## 2.1 Time Standard

- Time stamps are given in Unix time* with nanoseconds precision.

  * Note that Unix time is not a linear time scale. It is defined as the numbers of seconds since the epoch (1970-01-01 00:00:00 UTC) excluding leap seconds. Unfortunately, the expression “excluding leap seconds” is somehow ambiguous. Basically, it means that Unix time is the exact number of seconds since the epoch minus the number of leap seconds which have occurred until the date of interest. Therefore, each day since the epoch is accounted for exactly 86 400 seconds.
  It is also important to notice that the epoch is defined as 00:00:00 on January 1, 1970 given in UTC. This should not be mixed up with the same date given in TAI, which is ten seconds before the actual UTC epoch [see also Appendix A].

## 2.2 Coordinate Frames

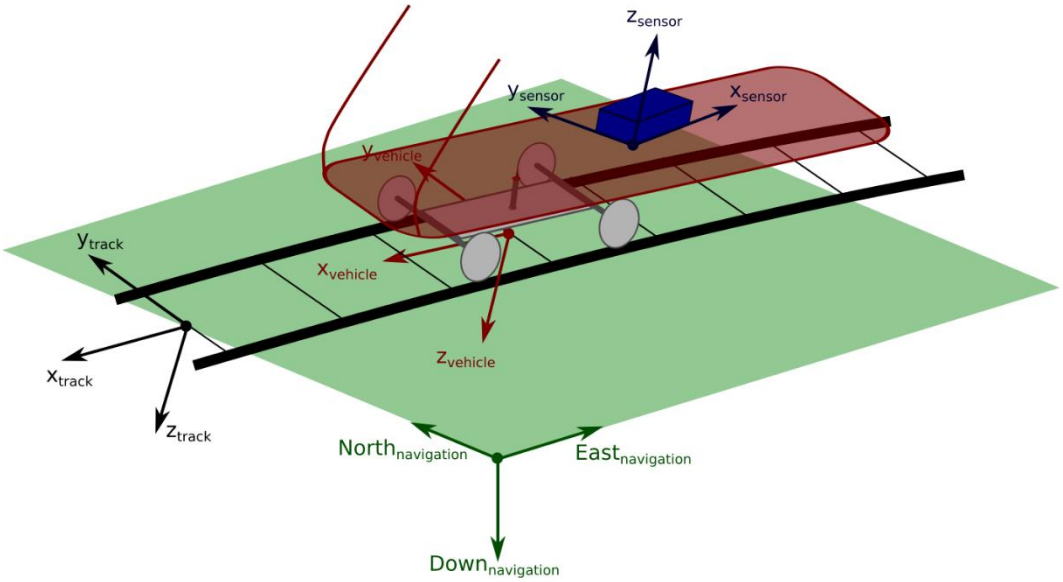


*Figure 1: Coordinate Systems*

### *2.2.1 Sensor Frame*

This is the specific frame each sensor provides its data relative to. As it is specific to each sensor, no general definitions can be made.

- For the raw data to be comprehensible, information about each sensor frame should be provided along with its raw data.

### *2.2.2 Vehicle Frame*

This is the onboard coordinate system of the vehicle which is carrying the sensors.

- It fulfills the following purposes:
  - All sensor positions are referred to this frame.
  - The kinematic states of the vehicle are referred to the origin of this frame.

    - Reference positioning solutions are referred to the origin of this frame, unless otherwise stated.
- It is defined by:
    - its origin at the center of the pin of the boogie located next to the front* of the vehicle,

      * The definition of the front of the vehicle is up to the user but it has to be documented.
    - the x-axis pointing to the front of the vehicle, the y-axis to the right and the z-axis down (see also *Figure 1*),
    - and the height being measured from the top of the railheads (note that this normally results in negative mounting heights for the most sensors as they are normally mounted above this level).

### *2.2.3 Navigation Frame*

This frame is defined with respect to the surface of the earth to describe the navigation of the vehicle in the vicinity of the earth.

- The position* of the vehicle is given in this frame.

  * Unless otherwise stated, the position of the vehicle in the navigation frame refers to the origin of the vehicle-frame given in the navigation frame.
- The global navigation system is defined by geographic coordinates (latitude, longitude, altitude) with respect to the World Geodetic System 1984 (WGS84) as reference system.
- Positions are given in latitude (degree), longitude (degree) and altitude* (meter)

  * Altitudes may be provided with respect to the underlying WGS84 ellipsoid or the underlying WGS84 geoid (EGM96). Here the WGS84 ellipsoid should be used.
- If a local Cartesian navigation frame is used, the north, east, down (NED) mechanization is utilized (see also Figure 1). As transformation between the global and local navigation frame the Universal Transverse Mercator (UTM) transformation is used. The UTM zone must be specified in the description of the data.

### *2.2.4 Track Frame*

This frame is defined with respect to the current railway track the vehicle is currently traveling on. The track frame is a hybridized coordinate frame consisting of a fixed discrete tuple for referring to positions and a moving coordinate system to refer to the track's attitude at a specific point:

- Positions* are referred by a unique track ID and on-track distance value counted from the beginning of the track in its defined direction**

  * Unless otherwise stated, the position of the vehicle in the track frame refers to the origin of the vehicle-frame given in the track frame.

  ** The definition of the track direction is up to the user but it has to be documented.
- The attitude of the track is given in a right-handed coordinate system at a specific point (given by its on-track distance) of the track:
    - The origin of this coordinate system is located at the track's center line at a given on-track distance value.

- Its x-axis is aligned with the current tangent of the track's center line at a given on-track distance. The y-axis points to the right and the z-axis down, in which the xy-plane lies on the line directly connecting the rail heads to both sides (see also *Figure 1*).
- The track's attitude is defined by its heading angle (yaw), its bank angle (roll) and its slope angle (pitch) which are given relative to the navigation frame at a given on-track distance.

### *2.2.5 Inertial Frame*

This is a frame fixed in space with respect to the fixed stars. It is mainly used as a reference frame for inertial measurements.

- Its origin is at the center of the earth.
- Its z-axis is aligned with the earth's polar axis and the x and y-axis are aligned with respect to the fixed stars.

# 3 Format Definitions

## 3.1 General Rules

- The following format definitions mostly presume that it is possible to store the processed data and the reference data in a readable CSV file where it is easily possible to label the data. This may not be very practical for all kind of data, as CSV files grow very large quickly when the amount of data grows.
  At the moment there are no rules available how to deal with such storage consuming data. A possible way to deal with this kind of data, in accordance with the spirit of these principles, may be to provide the data in a binary format together with tools to access the data corresponding to the data labels defined below.
- It may not always be possible to provide all the data which is demanded by the below defined minimum data labels. In this case the data belonging to such a data label has to be virtually supplemented by adding “NaNs” or “empty fields” to the data.
- The following format definitions represent a minimum set that always should be provided to ensure the interchangeability of different data sets and an easy application of the data to the algorithms under test.

## 3.2 Parameters

| Data | Label | Unit | Frame |
|---|---|---|---|
| Lever arm (x-axis) | `LeverArmX_m` | *m* | vehicle |
| Lever arm (y-axis) | `LeverArmY_m` | *m* | vehicle |
| Lever arm (z-axis) | `LeverArmZ_m` | *m* | vehicle |
| Mounting roll angle | `MountingRoll_deg` | ° | sensor --> vehicle |
| Mounting pitch angle | `MountingPitch_deg` | ° | sensor --> vehicle |
| Mounting yaw angle | `MountingYaw_deg` | ° | sensor --> vehicle |

- For each sensor a separate parameter file with the format definitions described above should be provided.
- The mounting position (attitude) of a sensor is specified by its roll, pitch and yaw angle, so that the sensor data can be transformed from the sensor frame to the vehicle frame. Thereby, the following rotation order is assumed: 1. yaw, 2. pitch, 3. roll.

## 3.3 Maps

General format definitions for maps are still under discussion.

Maps play an important role in many positioning approaches. With accurate and complete maps, the track-constrained motion of rail vehicles can be exploited in positioning algorithms.

A minimal railway map comprises the geometries of all relevant tracks in the form of polygonal chains or “polylines”. A track must be understood as rail segment between two switches (or track ends). Connections between tracks must be modeled by shared starting or end points.

Given the above representation, the topology of the railway network can be algorithmically extracted from the track geometries. That is, incoming and outgoing connections to the adjacent tracks can be listed for each track. The topology can be represented using an adjacency list in which the values determine the connection type (start-to-start, start-to-end, end-to-start, or end-to-end connection). Shapefiles are one common option to represent track geometries. The reference frame of the track geometries must be specified. WGS84 or UTM data are common choices.

For certain applications it is sufficient to have the map topology only, with additional information about the lengths of the individual tracks instead of their absolute positions. One example are "fingerprinting" approaches that compare measured track-specific signatures with a database.

Of course, the topic of railway maps is a lot bigger than the aspects mentioned above. In addition to track geometries and the topology, it can be relevant to also include the positions of certain landmarks, signals, geometric parameters, user-specific parameters such as maintenance information, and speed limits. Because of the complexity of representing all that information, standards and exchange formats such as railML (https://www.railml.org/en/) are important.

## 3.4 Raw Data

- Raw data may be stored in sensor specific formats as long as it is explained how to access and use the data. For the sensor classes mentioned below existing standards are available and should be used.

### *3.4.1 GNSS*

- If available, GNSS observables are provided in the RINEX format [*https://kb.igs.org/hc/en-us/articles/201096516-IGS-Formats*]

## 3.5 Processed Data

### *3.5.1 GNSS*

| Data | Label | Unit | Frame |
|---|---|---|---|
| Time stamp (Unix) | `TimeUnix_s` | *s* | Unix Time |
| Latitude | `Latitude_deg` | ° | navigation |
| Longitude | `Longitude_deg` | ° | navigation |
| Altitude (ellipsoid) | `AltitudeEllipsoid_m` | *m* | navigation |
| Ground speed | `VelocityGround_ms` | *m/s* | navigation |
| Heading | `Heading_deg` | ° | navigation |
| Velocity (north) | `VelocityNorth_ms` | *m/s* | navigation |
| Velocity (east) | `VelocityEast_ms` | *m/s* | navigation |
| Velocity (down) | `VelocityDown_ms` | *m/s* | navigation |
| Error ellipse semi-major axis (1σ-error) * | `ErrorEllipseMajor_m` | *m* | navigation |

| | | | |
|---|---|---|---|
| Error ellipse semi-minor axis (1σ-error) * | `ErrorEllipseMinor_m` | *m* | navigation |
| Error ellipse orientation (from true north) * | `ErrorEllipseOrientation_deg` | ° | navigation |
| Latitude 1σ-error | `LatitudeSigma_m` | *m* | navigation |
| Longitude 1σ-error | `LongitudeSigma_m` | *m* | navigation |
| Altitude (ellipsoid) 1σ-error | `AltitudeEllipsoidSigma_m` | *m* | navigation |
| Ground speed 1σ-error | `VelocityGroundSigma_ms` | *m/s* | navigation |
| Heading 1σ-error | `HeadingSigma_deg` | ° | navigation |
| Velocity (north) 1σ-error | `VelocityNorthSigma_ms` | *m/s* | navigation |
| Velocity (east) 1σ-error | `VelocityEastSigma_ms` | *m/s* | navigation |
| Velocity (down) 1σ-error | `VelocityDownSigma_ms` | *m/s* | navigation |
| HDOP | `HDOP` | | |
| VDOP | `VDOP` | | |
| PDOP | `PDOP` | | |

* still under discussion

- Calculation of GNSS position:
  - Most GNSS receivers directly provide a GNSS positioning solution. If possible, the algorithm used by the receiver to calculate a GNSS position should be documented. If this is not possible, at least a short notice should be provided in the documentation.
  - If a GNSS receiver also provides the raw GNSS observables (see also section 3.2) it is possible to use a known or self-written algorithm to calculate a separate GNSS position from the GNSS raw data. A well-known tool to calculate GNSS positions is e.g. RTKLIB [*http://www.rtklib.com*]. For maximum traceability the used algorithm or the configuration of the tool used to calculate a GNSS position from the raw data should be provided in the documentation.

### *3.5.2 IMU*

| **Data** | **Label** | **Unit** | **Frame** |
|---|---|---|---|
| Time stamp (Unix) | `TimeUnix_s` | *s* | Unix Time |
| Acceleration (x-axis) | `AccX_mss` | *m/s^2* | vehicle |
| Acceleration (y-axis) | `AccY_mss` | *m/s^2* | vehicle |
| Acceleration (z-axis) | `AccZ_mss` | *m/s^2* | vehicle |
| Turn rate (x-axis, “roll”) | `TurnRateX_degs` | *°/s* | vehicle |

| | | | |
|---|---|---|---|
| Turn rate (y-axis, “pitch”) | TurnRateY_degs | °/s | vehicle |
| Turn rate (z-axis, “yaw”) | TurnRateZ_degs | °/s | vehicle |

### *3.5.3 Generic speed and distance sensors*

| Data | Label | Unit | Frame |
|---|---|---|---|
| Time stamp (Unix) | TimeUnix_s | *s* | Unix Time |
| Distance since start | DistanceVehicle_m | *m* | navigation |
| Distance 1σ-error | DistanceVehiceSigma_m | *m* | navigation |
| Ground speed | VelocityGround_ms | *m/s* | navigation |
| Ground speed 1σ-error | VelocityGroundSigma_ms | *m/s* | navigation |
| Vehicle Direction<br>(1 = forward, 0 = unknown, -1 = backward) | DirectionVehicle | *integer* | vehicle |
| Standstill Flag<br>(1 = standstill, 0 = unknown) | StandstillFlag | *boolean* | vehicle |
| Motion Flag<br>(1 = motion, 0 = unknown) | MotionFlag | *boolean* | vehicle |

## 3.6 Reference Data

### *3.6.1 Positioning and Kinematic States*

| Data | Label | Unit | Frame |
|---|---|---|---|
| Time stamp (Unix) | TimeUnix_s | *s* | Unix Time |
| Latitude | Latitude_deg | ° | navigation |
| Longitude | Longitude_deg | ° | navigation |
| Altitude (ellipsoid) | AltitudeEllipsoid_m | *m* | navigation |
| Distance since start | distanceVehicle_m | *m* | navigation |
| Vehicle speed<br>(positive = forward,<br>negative = backward) | VelocityVehicle_ms | *m/s* | navigation |
| Ground speed | VelocityGround_ms | *m/s* | navigation |
| Heading | Heading_deg | ° | navigation |
| Velocity (north) | VelocityNorth_ms | *m/s* | navigation |
| Velocity (east) | VelocityEast_ms | *m/s* | navigation |
| Velocity (down) | VelocityDown_ms | *m/s* | navigation |

| | | | |
|---|---|---|---|
| Error ellipse semi-major axis (1σ-error) * | `ErrorEllipseMajor_m` | *m* | navigation |
| Error ellipse semi-minor axis (1σ-error) * | `ErrorEllipseMinor_m` | *m* | navigation |
| Error ellipse orientation (from true north) * | `ErrorEllipseOrientation_deg` | ° | navigation |
| Latitude 1σ-error | `LatitudeSigma_m` | *m* | navigation |
| Longitude 1σ-error | `LongitudeSigma_m` | *m* | navigation |
| Altitude (ellipsoid) 1σ-error | `AltitudeEllipsoidSigma_m` | *m* | navigation |
| Vehicle speed 1σ-error | `VelocityVehicleSigma_ms` | *m/s* | navigation |
| Ground speed 1σ-error | `VelocityGroundSigma_ms` | *m/s* | navigation |
| Heading 1σ-error | `HeadingSigma_deg` | ° | navigation |
| Velocity (north) 1σ-error | `VelocityNorthSigma_ms` | *m/s* | navigation |
| Velocity (east) 1σ-error | `VelocityEastSigma_ms` | *m/s* | navigation |
| Velocity (down) 1σ-error | `VelocityDownSigma_ms` | *m/s* | navigation |
| Acceleration (x-axis) | `AccX_mss` | *m/s^2* | vehicle |
| Acceleration (y-axis) | `AccY_mss` | *m/s^2* | vehicle |
| Acceleration (z-axis) | `AccZ_mss` | *m/s^2* | vehicle |
| Turn rate (x-axis, “roll rate”) | `TurnRateX_degs` | *°/s* | vehicle |
| Turn rate (y-axis, “pitch rate”) | `TurnRateY_degs` | *°/s* | vehicle |
| Turn rate (z-axis, “yaw rate”) | `TurnRateZ_degs` | *°/s* | vehicle |
| Roll angle | `Roll_deg` | ° | vehicle |
| Pitch angle | `Pitch_deg` | ° | vehicle |
| Yaw angle | `Yaw_deg` | ° | vehicle |
| Roll angle 1σ-error | `RollSigma_deg` | ° | vehicle |
| Pitch angle 1σ-error | `PitchSigma_deg` | ° | vehicle |
| Yaw angle 1σ-error | `YawSigma_deg` | ° | vehicle |

* still under discussion

# 4 Folder Structure

All data of a test campaign should be stored in a single zip file with the structure like depicted in Figure 2.

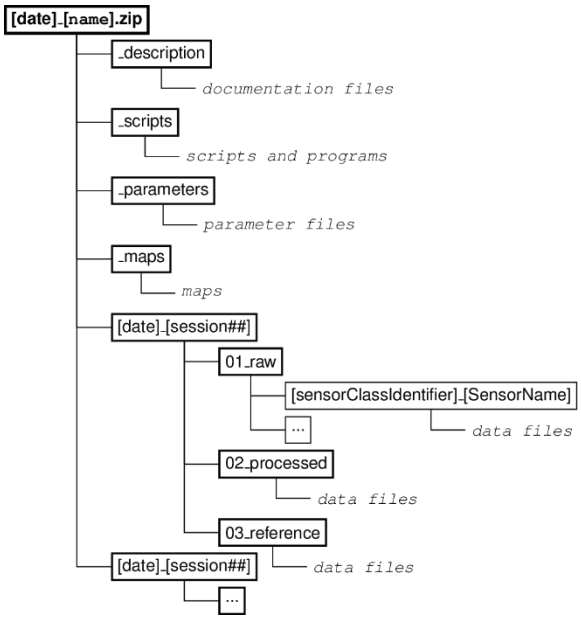


*Figure 2: Exemplary folder structure. The expressions in square brackets represent placeholders.*

- All data is stored in a single zip file named according to **`[date]_[name]`**. The name of the dataset can be chosen freely. A valid file name for data recorded on 22nd of February 2019, e.g., would be “2019-02-22_MyDataset1.zip”.
- The recorded data is arranged by sessions in corresponding **`[date]_[session##]`** folders. A session is characterized by a continuous stream of recordings. Valid folder names, e.g., would be “2019-02-22_session01”, “2019-02-22_session02”, etc.
- For each session the raw, processed and reference data (see also section 1.2) is stored in the corresponding folders **`01_raw`**, **`02_processed`**, and **`03_reference`**.
- All documentary material is stored in the **`_description`** folder.
- All programs and scripts used to process the data or to demonstrate its usage go to the folder **`_scripts`**.
- All static map data is stored in the folder **`_maps`**.
- All constant sensor parameters (like e.g. lever arms) are stored in the folder **`_parameters`** (separately for each sensor).

# Appendix

## A Definition of Seconds Since the Epoch

Reference:
"Standard for Information Technology - Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7", in IEEE Std 1003.1, 2016 Edition, pp.1-3957, 30 Sept. 2016, doi: 10.1109/IEEESTD.2016.7582338

> **Seconds Since the Epoch**
>
> A value that approximates the number of seconds that have elapsed since the Epoch. A Coordinated Universal Time name (specified in terms of seconds (tm_sec), minutes (tm_min), hours (tm_hour), days since January 1 of the year (tm_yday), and calendar year minus 1900 (tm_year)) is related to a time represented as seconds since the Epoch, according to the expression below. If the year is <1970 or the value is negative, the relationship is undefined.
>
> If the year is ≥1970 and the value is non-negative, the value is related to a Coordinated Universal Time name according to the C-language expression, where tm_sec, tm_min, tm_hour, tm_yday, and tm_year are all integer types:
>
> *tm_sec + tm_min*60 + tm_hour*3600 + tm_yday*86400 + (tm_year−70)*31536000 + ((tm_year−69)/4)*86400 − ((tm_year−1)/100)*86400 + ((tm_year+299)/400)*86400*
>
> The relationship between the actual time of day and the current value for seconds since the Epoch is unspecified.
>
> How any changes to the value of seconds since the Epoch are made to align to a desired relationship with the current actual time is implementation-defined. As represented in seconds since the Epoch, each and every day shall be accounted for by exactly 86 400 seconds.
>
> Note:
> The last three terms of the expression add in a day for each year that follows a leap year starting with the first leap year since the Epoch. The first term adds a day every 4 years starting in 1973, the second subtracts a day back out every 100 years starting in 2001, and the third adds a day back in every 400 years starting in 2001. The divisions in the formula are integer divisions; that is, the remainder is discarded leaving only the integer quotient.